



*Personal Computer
Hardware Reference
Library*

Technical Reference

6139821



*Personal Computer
Hardware Reference
Library*

Technical Reference

Revised Edition (March 1986)

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

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

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

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 2. Coprocessor

Section 3. Power Supplies

Section 4. Keyboards

Section 5. System BIOS

Section 6. Instruction Set

SECTION 1

SECTION 2

SECTION 3

SECTION 4

SECTION 5

SECTION 6

Notes:



Section 7. Characters, Keystrokes, and Colors

SECTION 7

Glossary

GLOSSARY

Bibliography

BIBLIOGRAPHY

Index

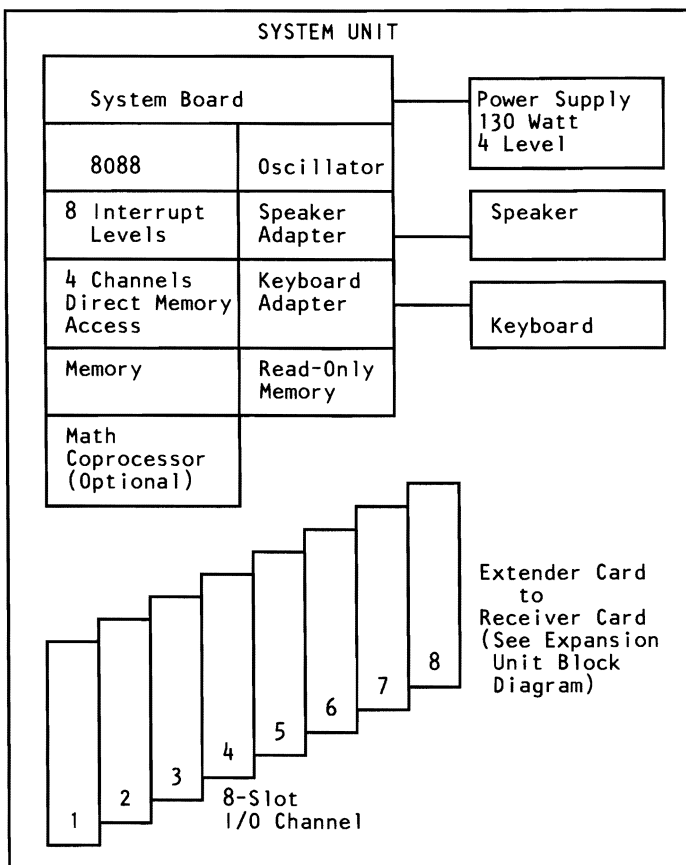
INDEX

Notes:



System Block Diagram (XT)

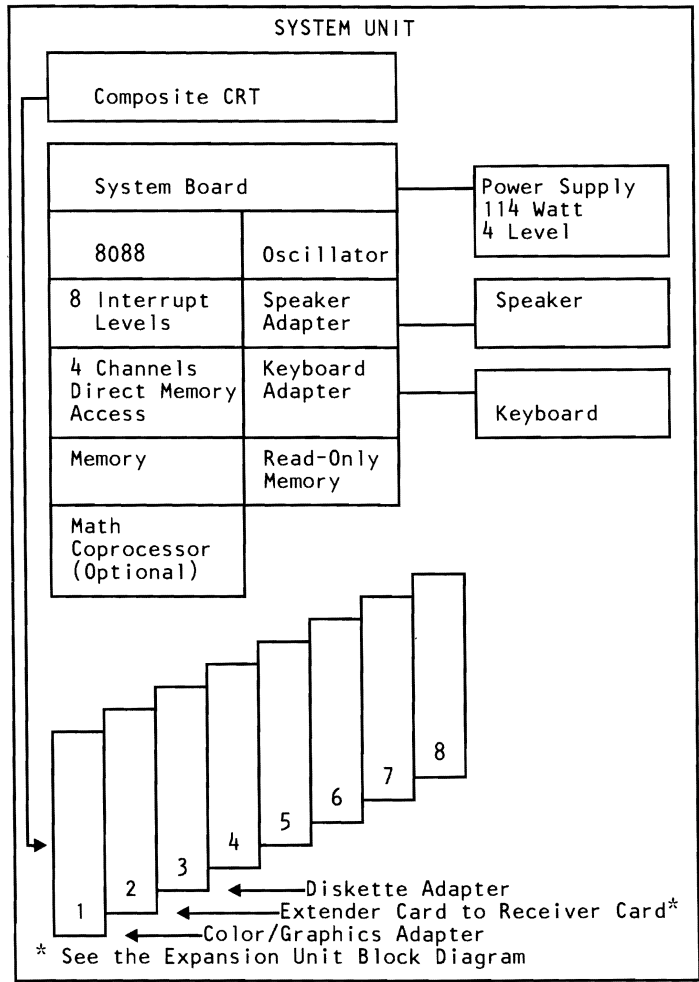
The following is a system block diagram of the IBM Personal Computer XT.



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)

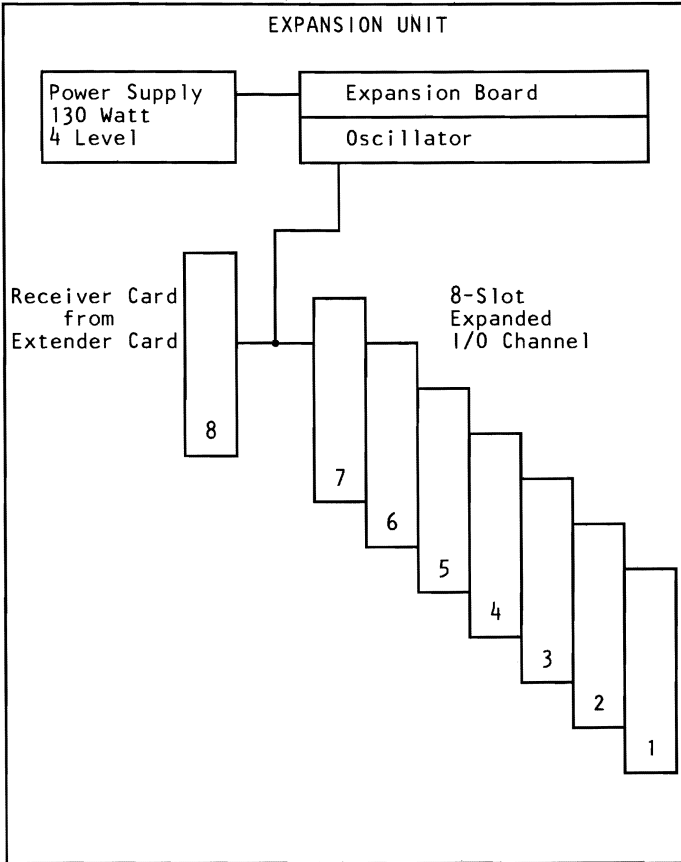
The following is a system block diagram of the IBM Portable Personal Computer.



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.

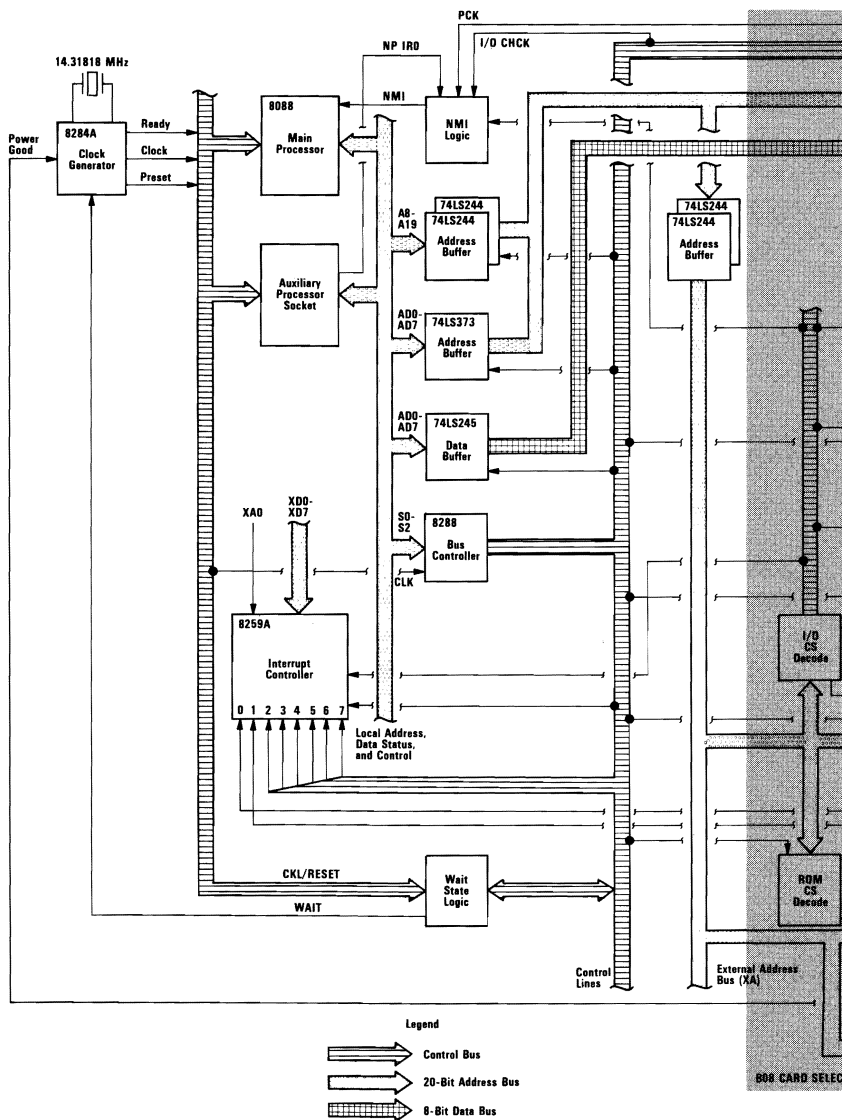
Microprocessor

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 (μ s).

Data Flow Diagrams

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



System Memory Map

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

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 1.05 μ s.

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.



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

$$\begin{array}{lcl} \text{\% Bandwidth used} & 4 \text{ cycles} \times 128 & 512 \\ \text{for Refresh} & = \frac{\text{-----}}{1.93\text{ms}/210\text{ns}} & = \frac{\text{-----}}{9190} = 5.6\% \end{array}$$

256K X 1

$$\begin{array}{lcl} \text{\% Bandwidth used} & 4 \text{ cycles} \times 256 & 1024 \\ \text{for Refresh} & = \frac{\text{-----}}{3.86\text{ms}/210\text{ns}} & = \frac{\text{-----}}{19048} = 5.6\% \end{array}$$

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: $+5 \text{ Vdc} \pm 5\%$, $-5 \text{ Vdc} \pm 10\%$, $+12 \text{ Vdc} \pm 5\%$, and $-12 \text{ Vdc} \pm 10\%$. 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 $1.05\mu\text{s}/\text{byte}$.
- DMA transfers require five clocks for a cycle time of $1.05\mu\text{s}/\text{byte}$.

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.

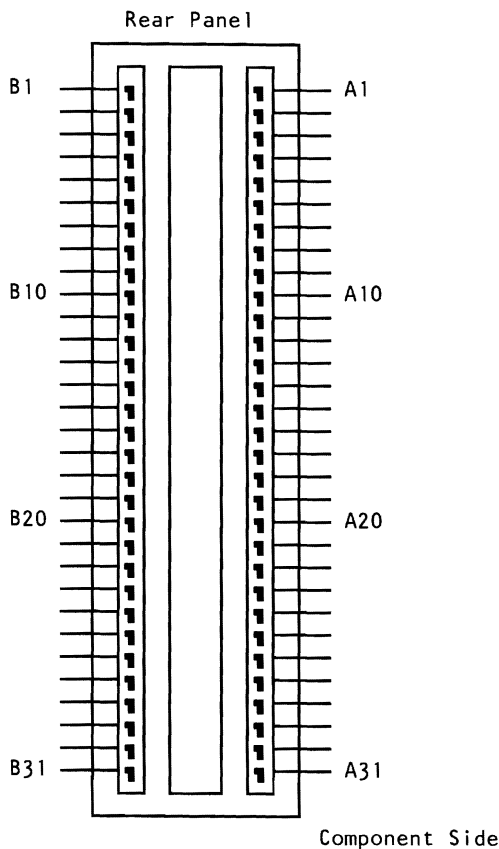
A 'channel check' line exists for reporting error conditions to the microprocessor. Activating this line results in a non-maskable interrupt (NMI) to the 8088 microprocessor. Memory expansion options use this line to report parity errors.

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	0
A12	SA19	I/O
A13	SA18	I/O
A14	SA17	I/O
A15	SA16	I/O
A16	SA15	I/O
A17	SA14	I/O
A18	SA13	I/O
A19	SA12	I/O
A20	SA11	I/O
A21	SA10	I/O
A22	SA9	I/O
A23	SA8	I/O
A24	SA7	I/O
A25	SA6	I/O
A26	SA5	I/O
A27	SA4	I/O
A28	SA3	I/O
A29	SA2	I/O
A30	SA1	I/O
A31	SA0	I/O

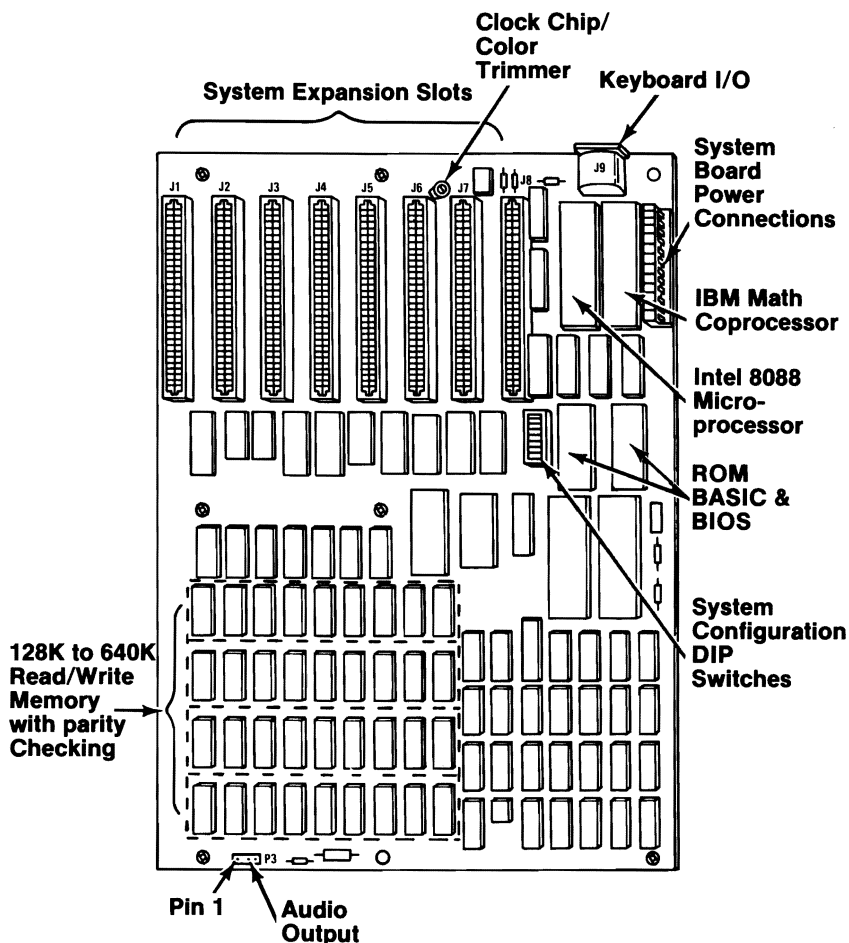
I/O Channel (A-Side, J1 through J8)

I/O Pin	Signal Name	I/O
B1	GND	Ground
B2	RESET DRV	0
B3	+5 Vdc	Power
B4	IRQ 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	0
B12	-MEMR	0
B13	-IOW	I/O
B14	-IOR	I/O
B15	-DACK3	0
B16	DRQ3	I
B17	-DACK1	0
B18	DRQ1	I
B19	-DACK0	I/O
B20	CLK	0
B21	IRQ7	I
B22	IRQ6	I
B23	IRQ5	I
B24	IRQ4	I
B25	IRQ3	I
B26	-DACK2	0
B27	T/C	0
B28	ALE	0
B29	+5Vdc	Power
B30	OSC	0
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
280-2DF	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.

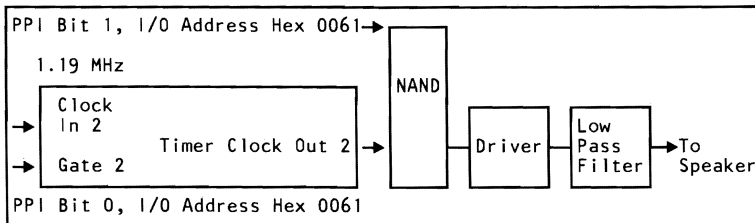
Channel I/O Address Map

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.

P3	Pin	Function
	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	PA0 1 2 3 4 5 6 7	+ Keyboard Scan Code	0 1 2 3 4 5 6 7	Or	Diagnostic Outputs	0 1 2 3 4 5 6 7																
0060																						
	PB0 1 2 3 4 5 6 7	+ Timer 2 Gate Speaker + Speaker Data Spare Read High Switches or Read Low Switches - Enable RAM Parity Check - Enable I/O Channel Check - Hold Keyboard Clock Low - (Enable Keyboard or + (Clear Keyboard)																				
0061																						
	PC0 1 2 3 4 5 6 7	Loop on POST + CoProcessor Installed + Planar RAM Size 0 + Planar RAM Size 1 Spare + Timer Channel 2 Out + I/O Channel Check + RAM Parity Check	Sw-1 Sw-2 Sw-3 Sw-4	Or	Display 0 Display 1 # Drives # Drives 1	**Sw-5 **Sw-6 ***Sw-7 ***Sw-8																
0062																						
0063	Command/Mode Register		Hex 99																			
	Mode Register Value		<table border="1"> <tr> <td>7</td> <td>6</td> <td>5</td> <td>4</td> <td>3</td> <td>2</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> <td>1</td> <td>1</td> <td>0</td> <td>0</td> <td>1</td> </tr> </table>				7	6	5	4	3	2	1	0	1	0	0	1	1	0	0	1
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

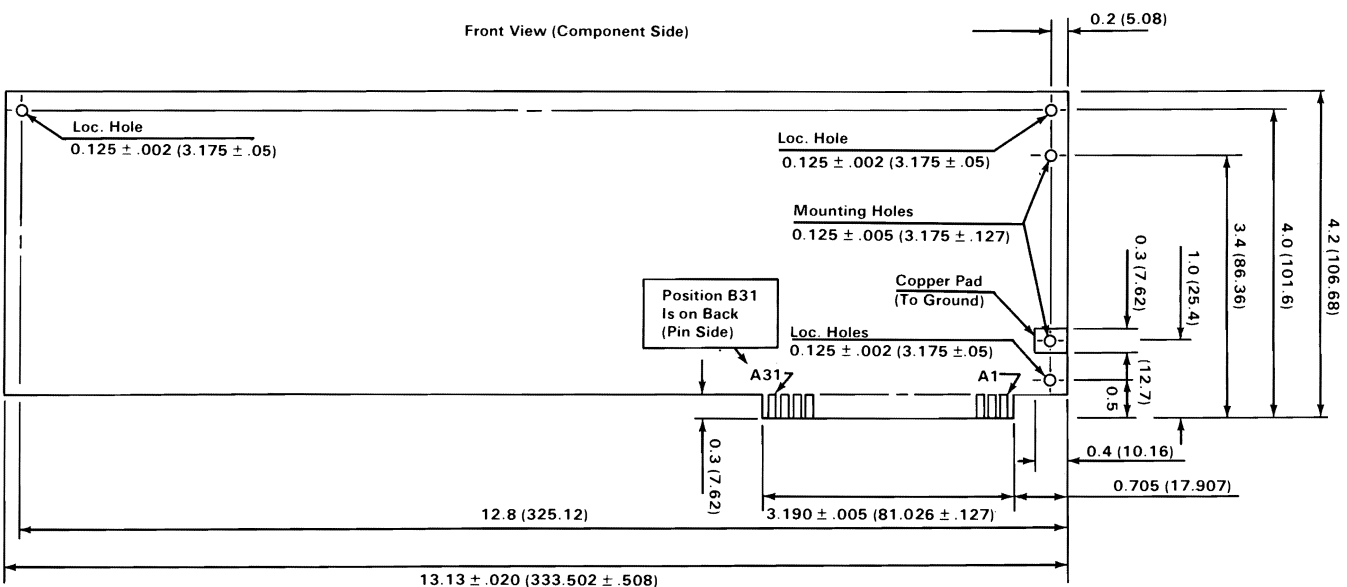
- 43 decibels average-noise rating (without printer)

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:

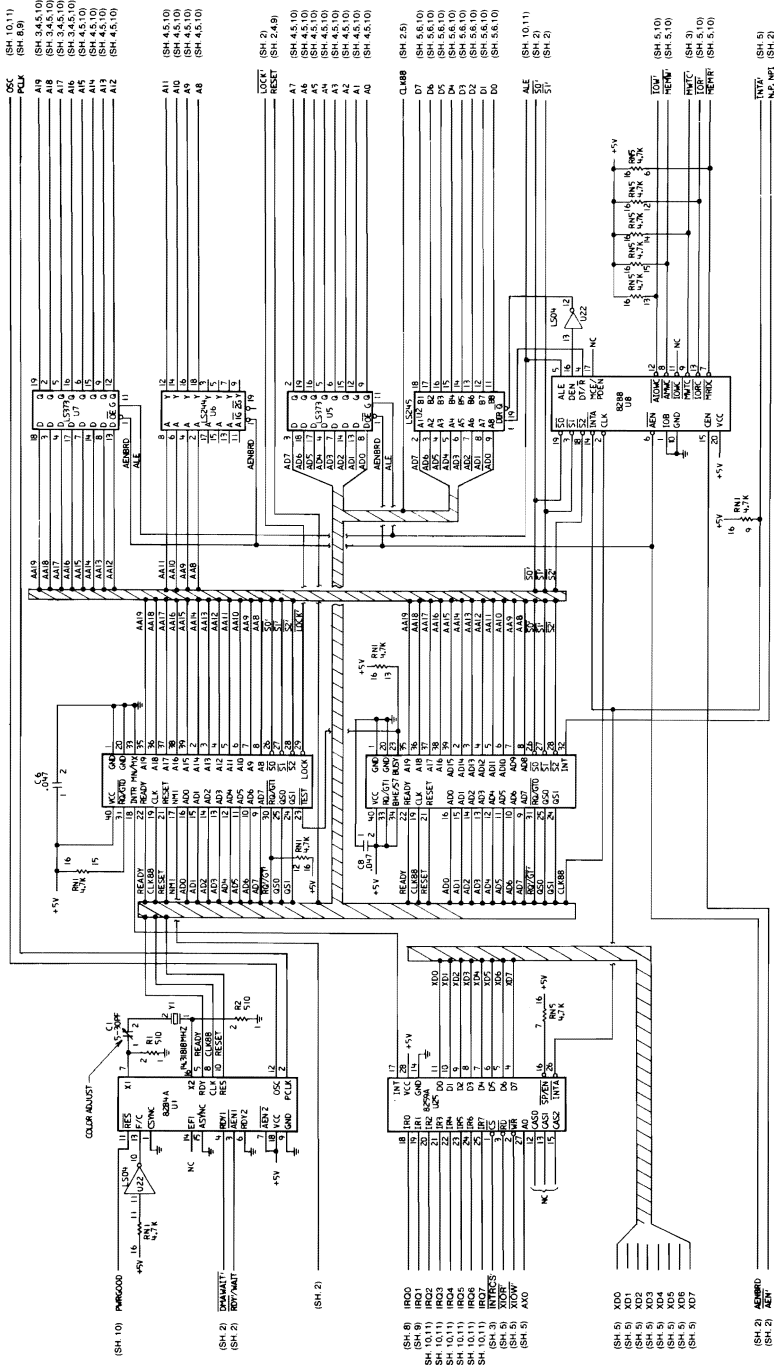
Connector	Pin	Assignments
J22	1	Keyboard Clock
	2	Keyboard Data
	3	Reserved
	4	Ground
	5	+5 Vdc

Keyboard Connector (J22)

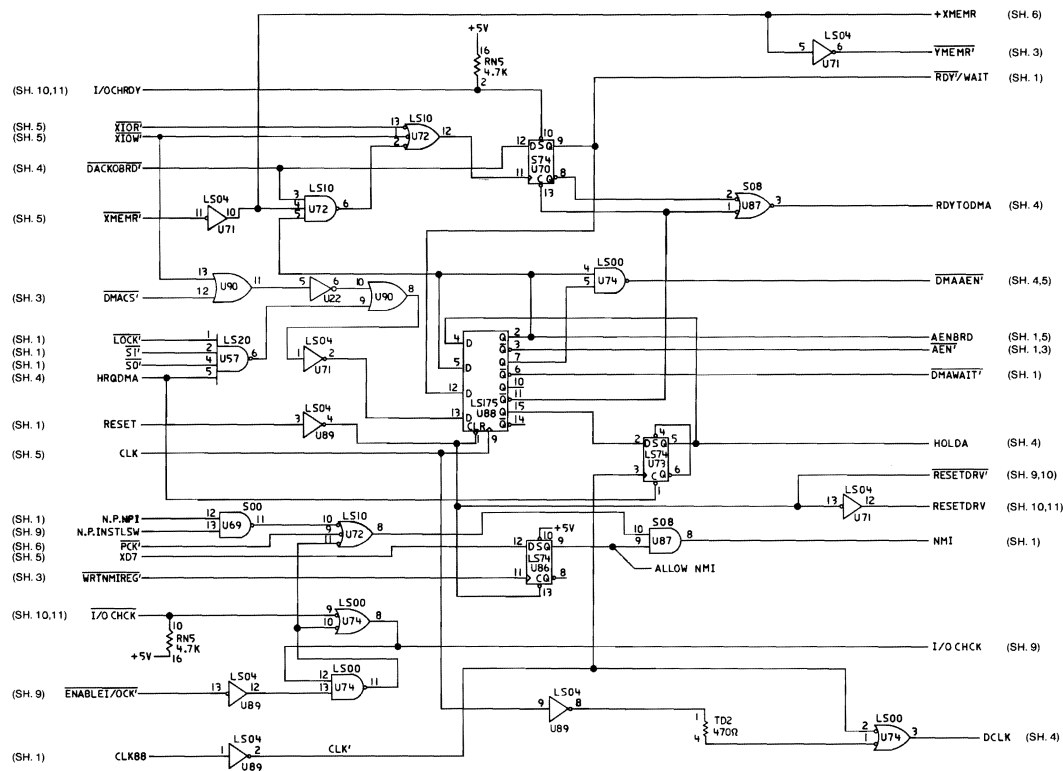
Logic Diagrams - 64/256K

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

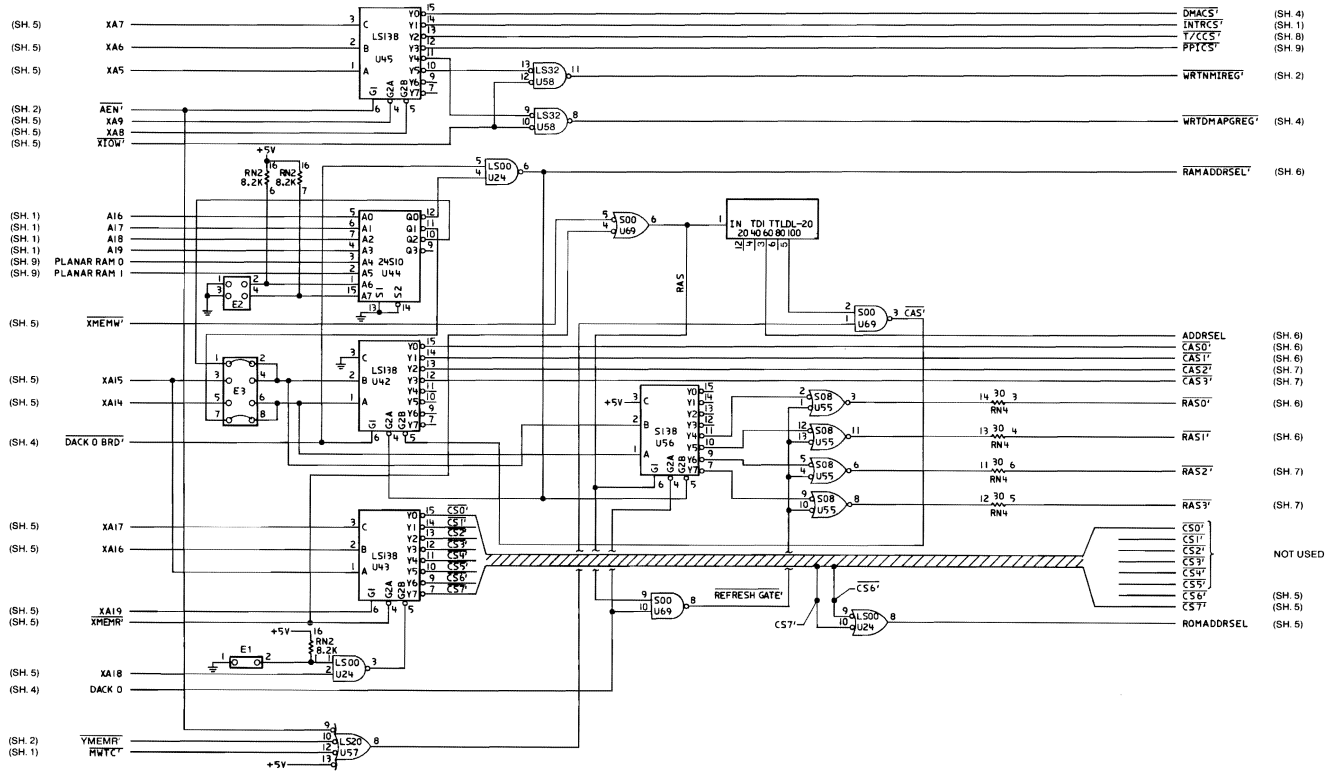


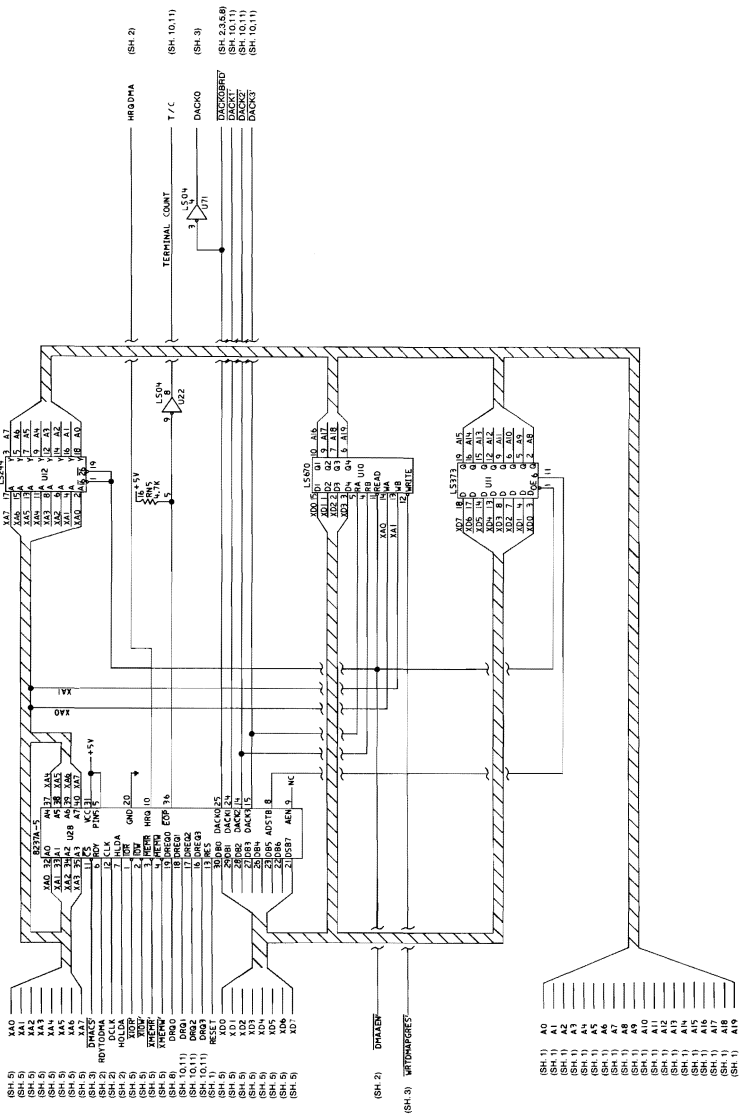


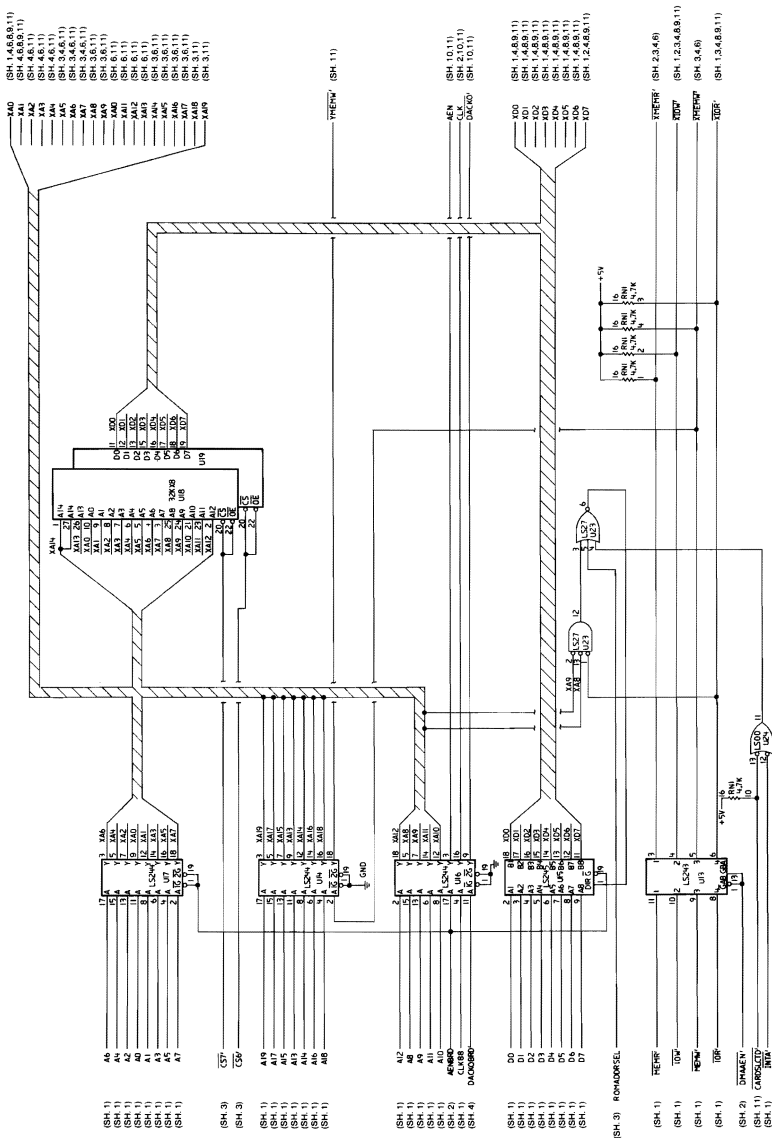
64/256K System Board (Sheet 1 of 11)



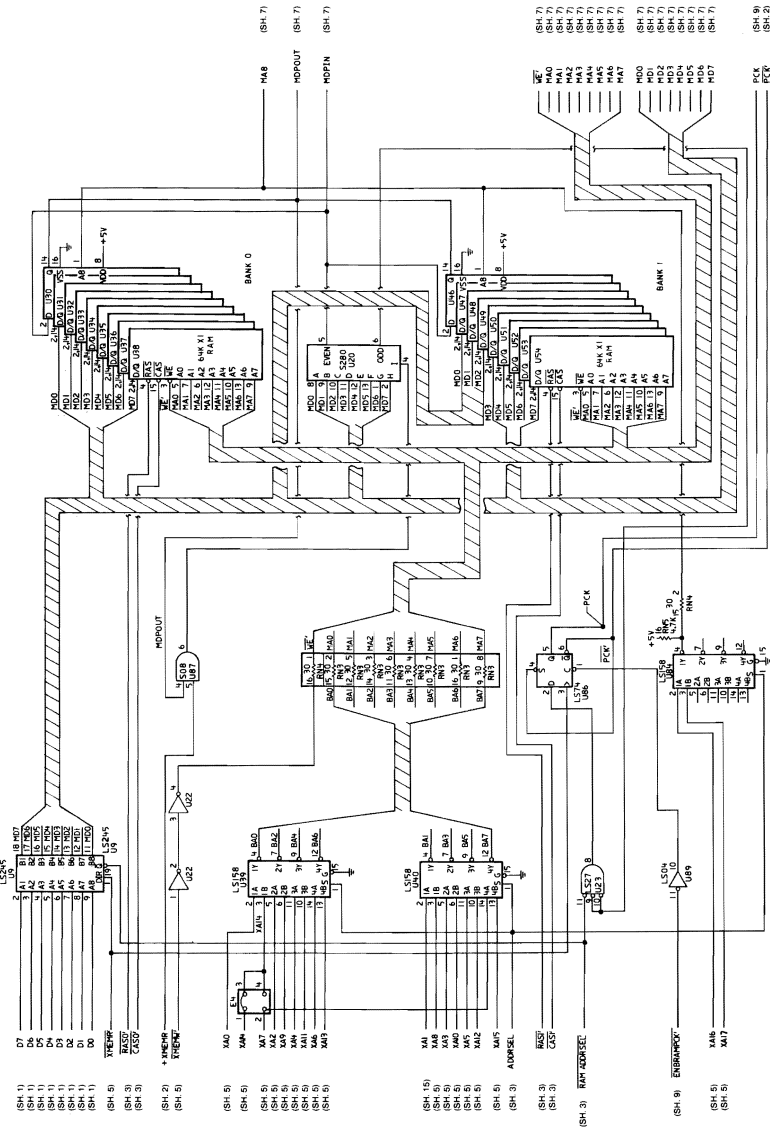
64/256K System Board (Sheet 2 of 11)

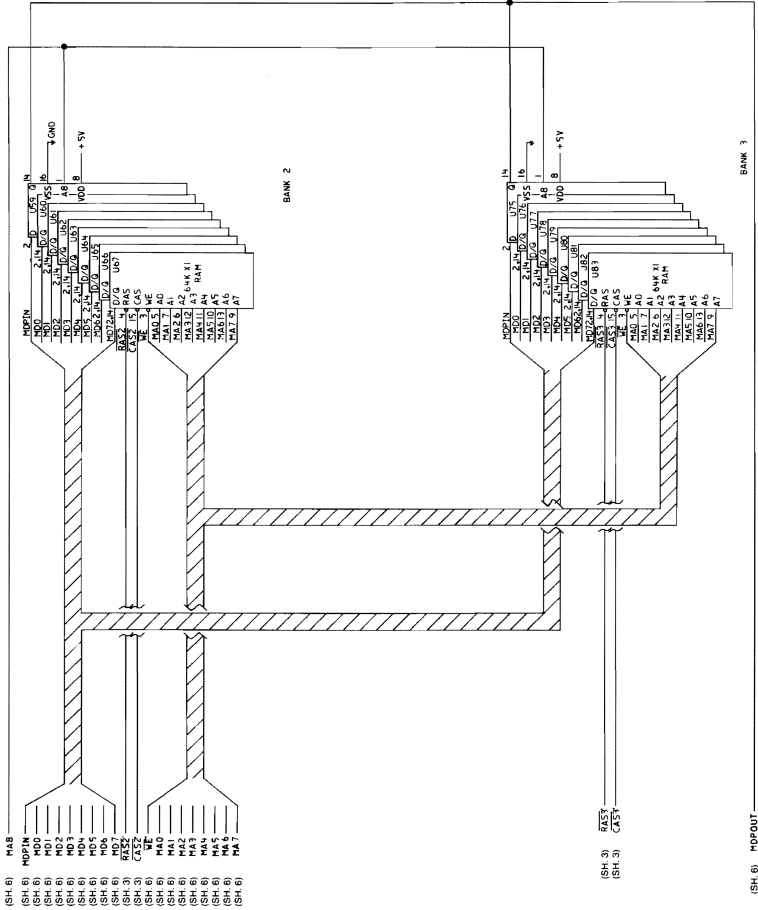




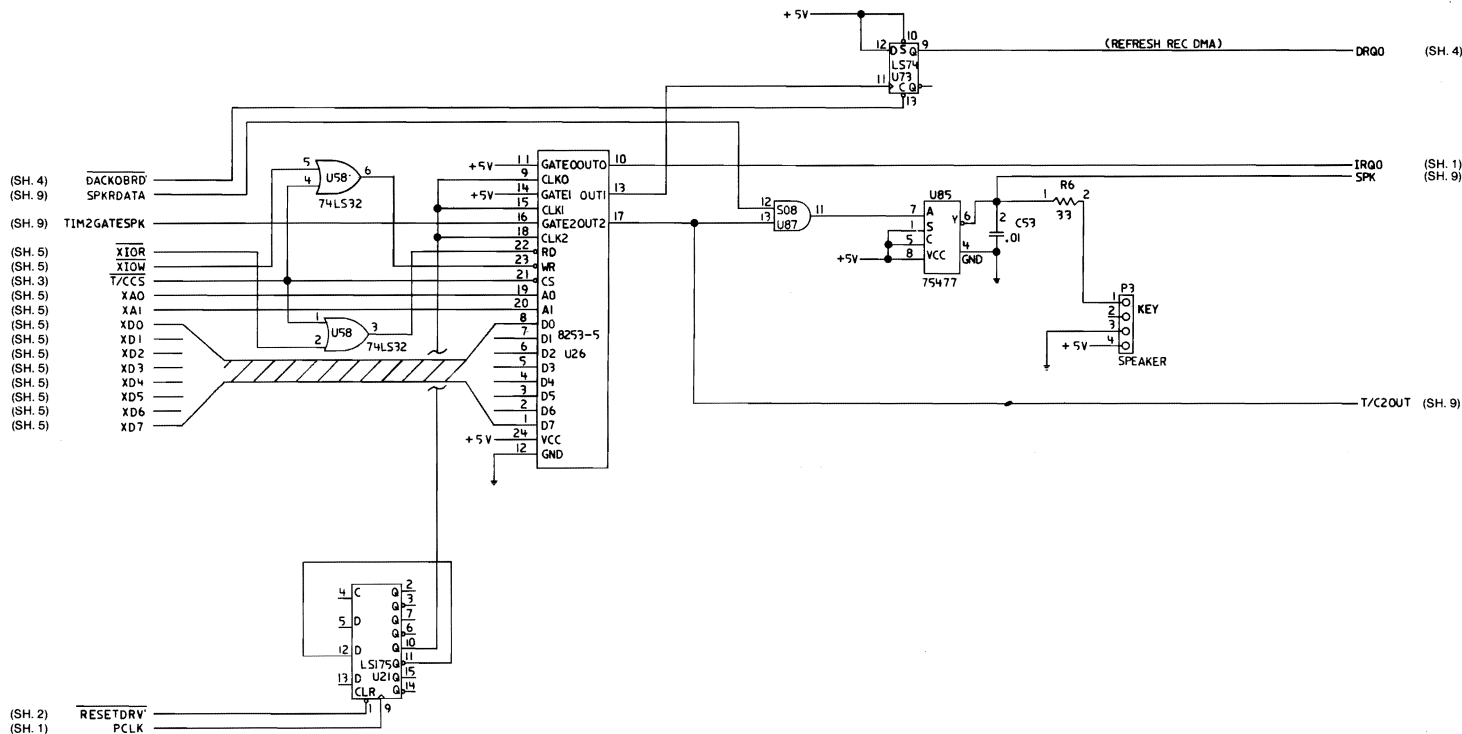


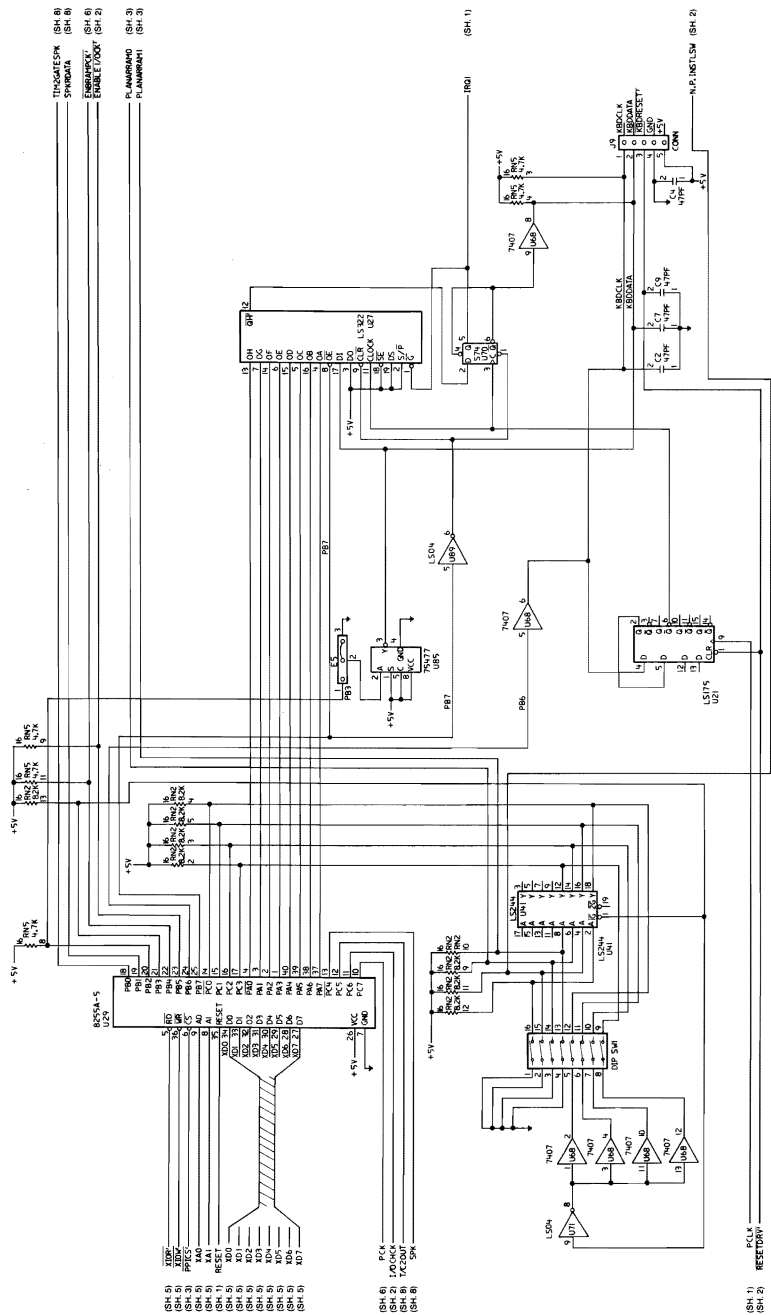
64/256K System Board (Sheet 5 of 11)



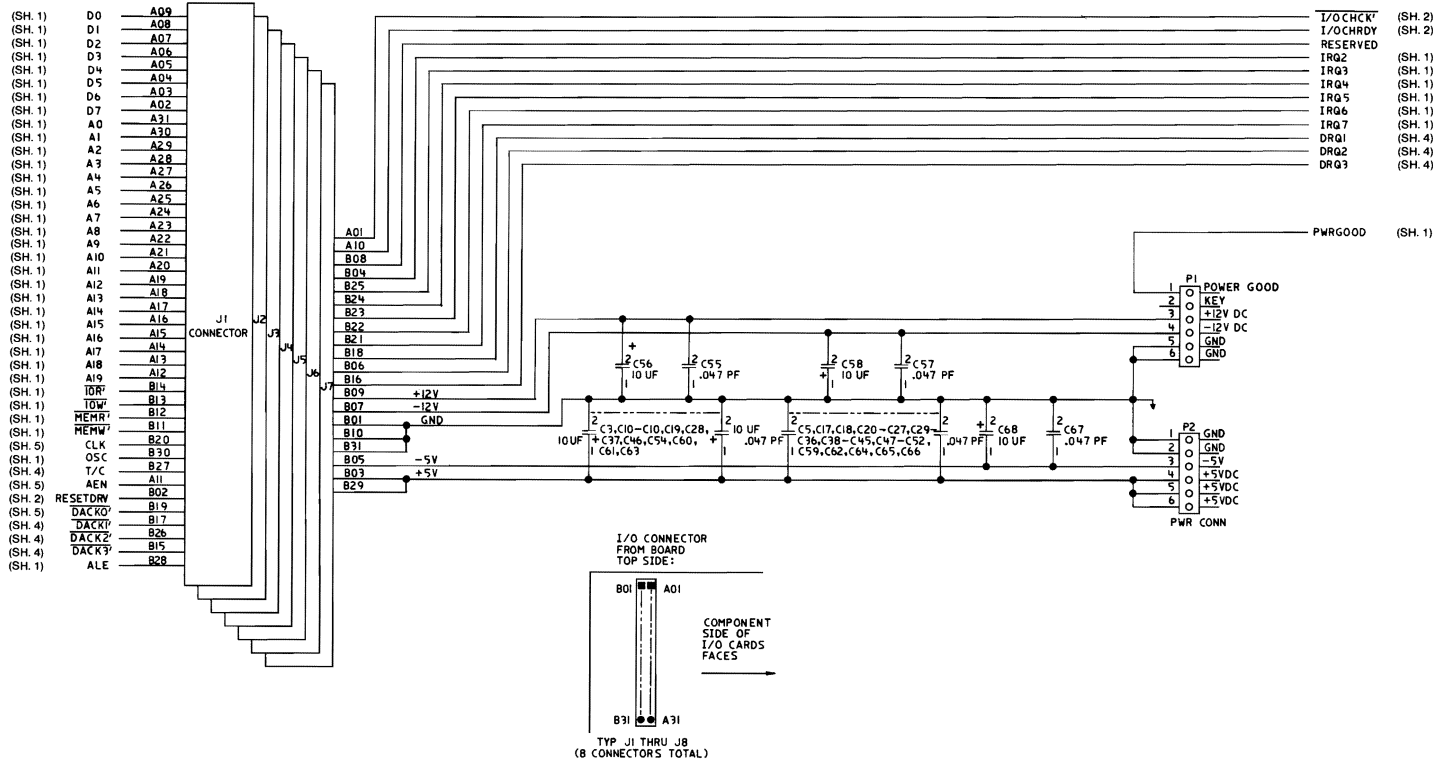


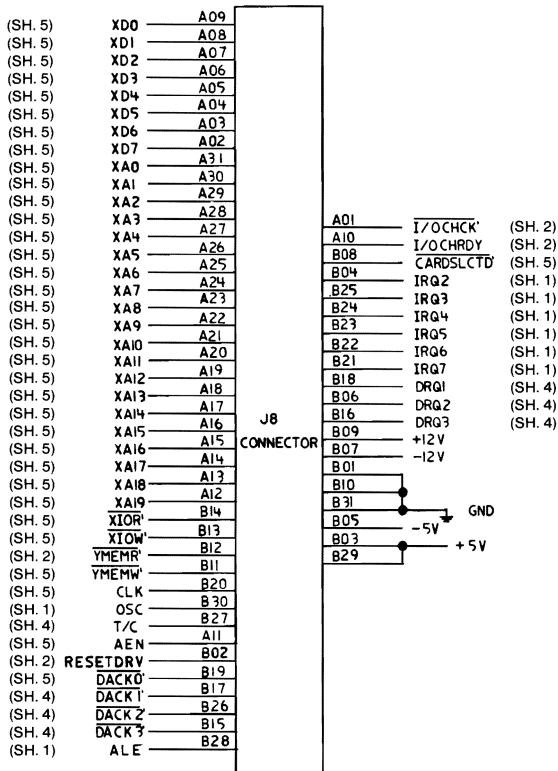
64/256K System Board (Sheet 7 of 11)





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





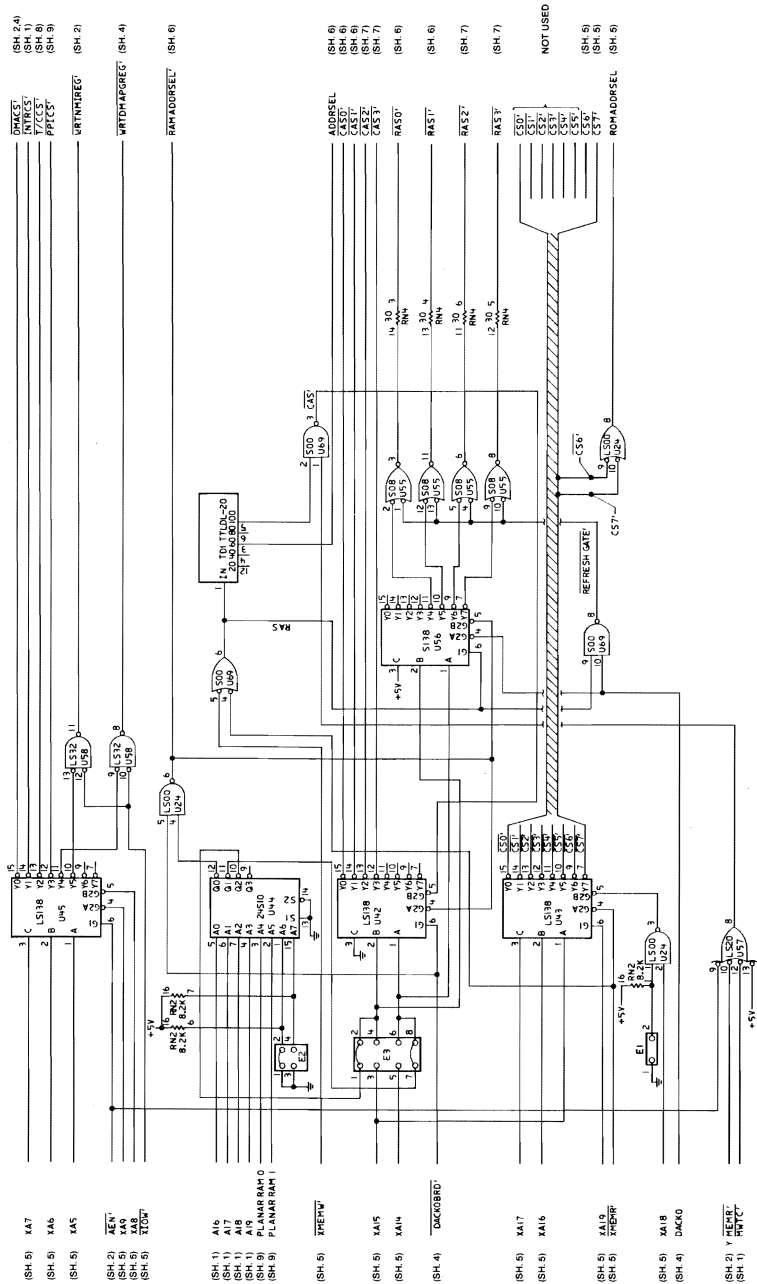
64/256K System Board (Sheet 11 of 11)

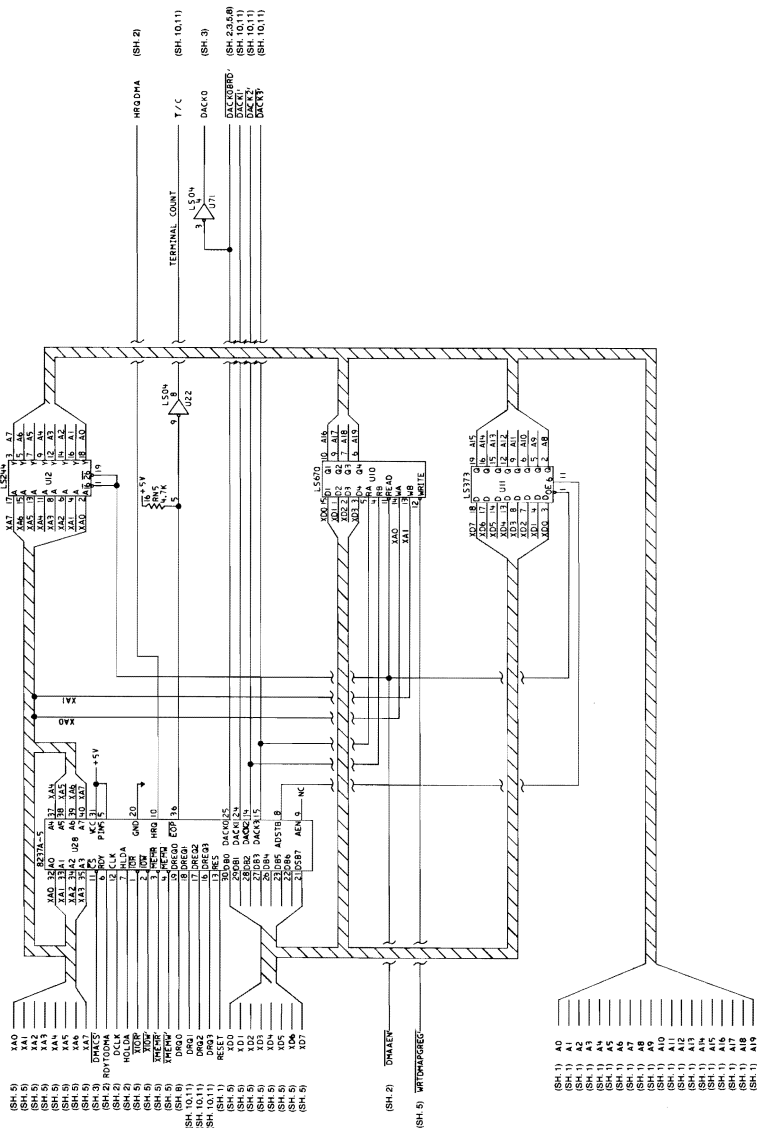
Logic Diagrams - 256/640K

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

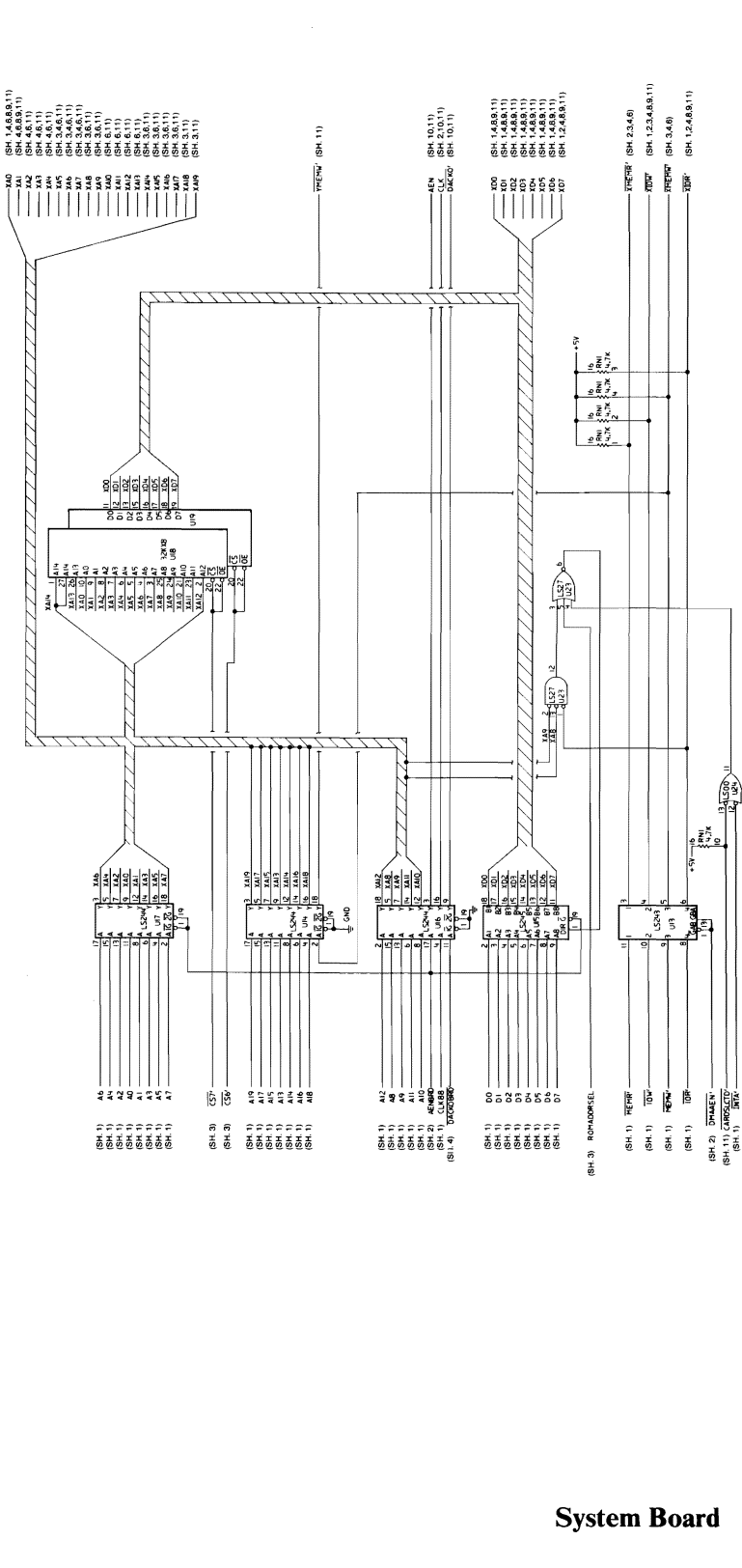




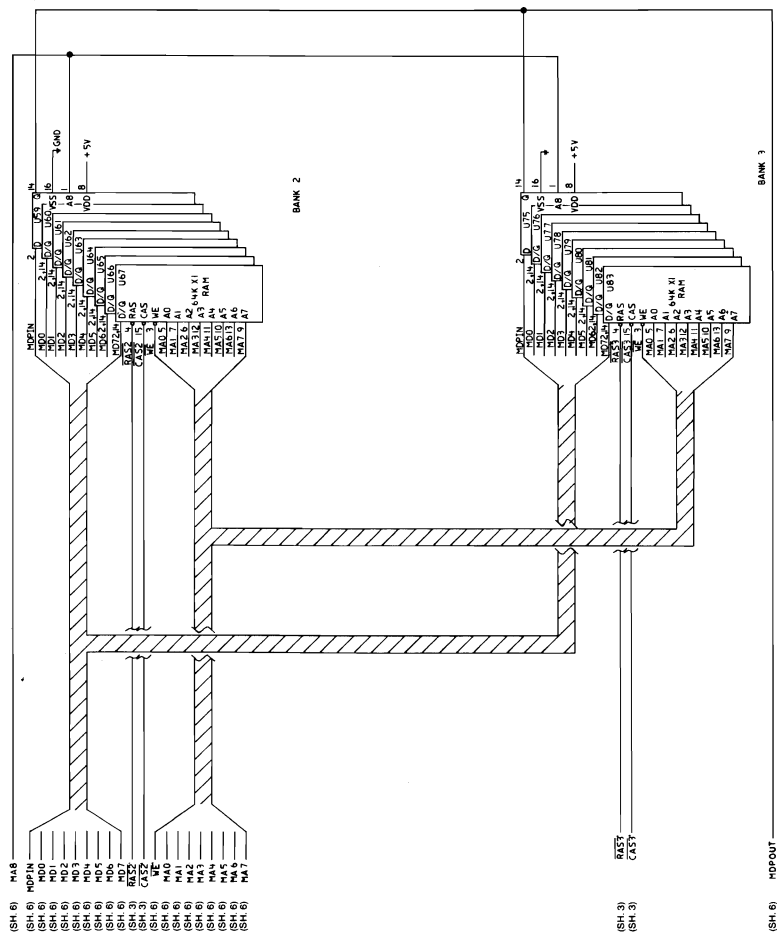




51

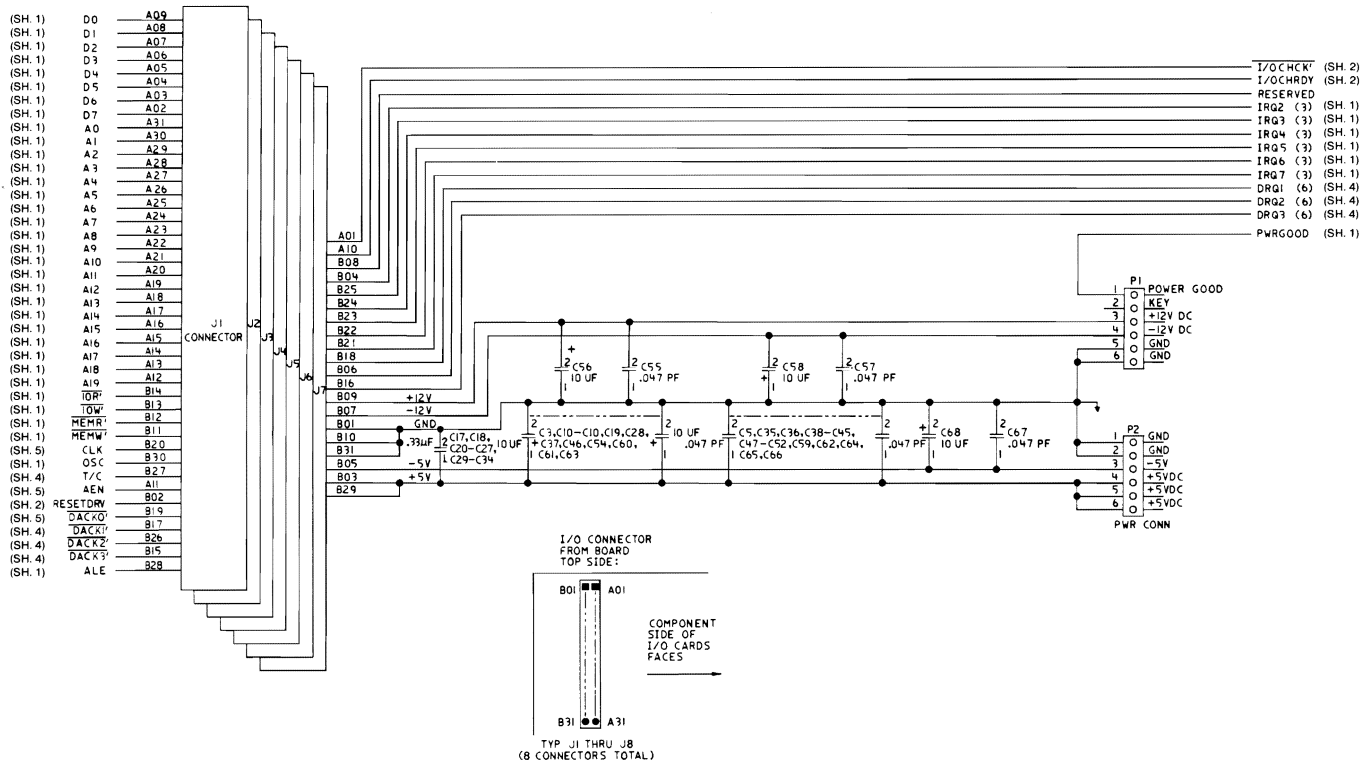


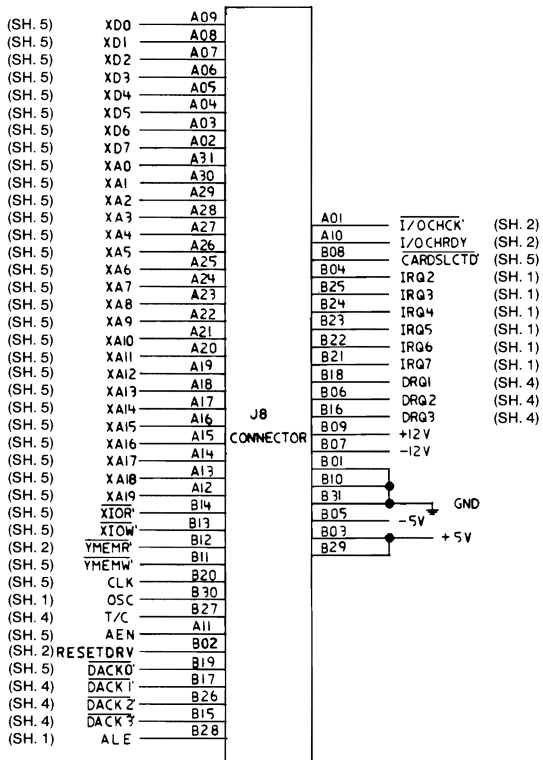
256/640K System Board (Sheet 7 of 11)











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

Notes:



SECTION 2. COPROCESSOR

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

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	$-2 \times 10^9 \leq X \leq +2 \times 10^9$
Long Integer	64	18	$-9 \times 10^{18} \leq X \leq +9 \times 10^{18}$
Packed Decimal	80	18	$-9..99 \leq X \leq +9..99$ (18 digits)
Short Real *	32	6-7	$8.43 \times 10^{-37} \leq X \leq 3.37 \times 10^{38}$
Long Real *	64	15-16	$4.19 \times 10^{-307} \leq X \leq 1.67 \times 10^{308}$
Temporary Real	80	19	$3.4 \times 10^{-4932} \leq X \leq 1.2 \times 10^{4932}$
* 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).

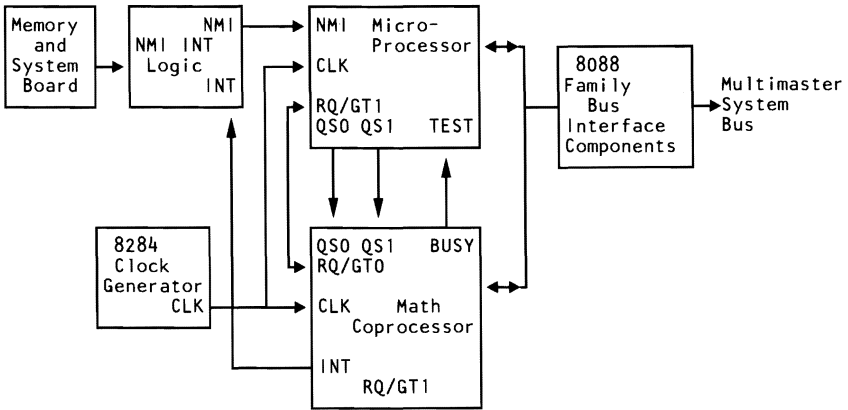
When an incorrect instruction is sent to the coprocessor (for example, divide by 0 or load a full register), the coprocessor can signal the microprocessor with an interrupt. There are three conditions that will disable the coprocessor interrupt to the microprocessor:

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

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 $\pm$ 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)		Regulation Tolerance
	Min	Max	
+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)		Voltage Limits	
	Min	Max	Min	Max
120 220/240	0.0 0.0	1.0 0.5	90 180	137 259

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.

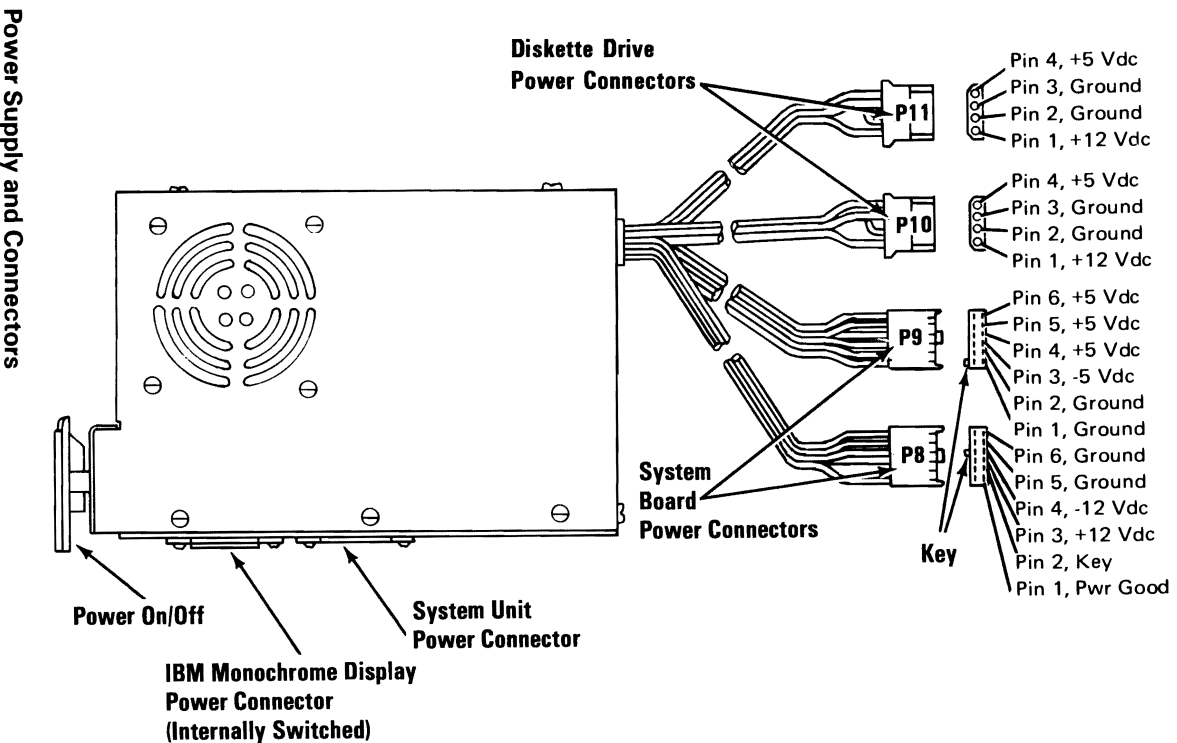
The 'power good' signal indicates that there is adequate power to continue processing. If the power goes below the specified levels, the 'power good' signal triggers a system shutdown.

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.



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.

Voltage and Current Requirements

Voltage @ 50/60 Hz $\pm$ 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, $\pm$ 3 hertz.

Nominal Output(Vdc)	Load Current (A)		Regulation Tolerance
	Min	Max	
+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.

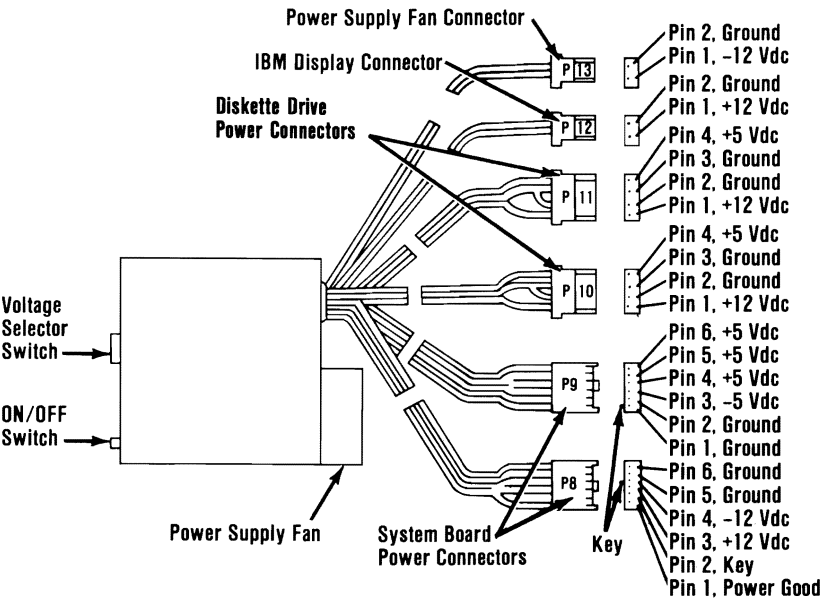
This signal is the logical **AND** of the dc output-voltage sense signal and the ac input-voltage fail signal. This signal is **TTL-compatible** up-level for normal operation or down-level for fault conditions. The ac 'fail' signal causes 'power good' to go to a down-level when any output voltage falls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

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

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.

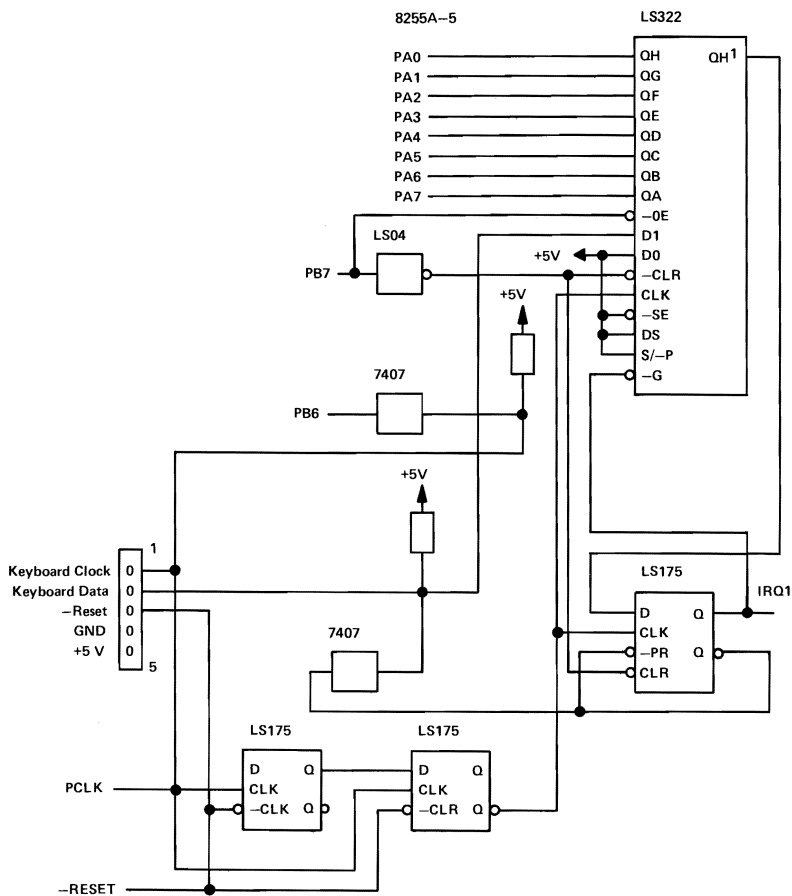
The keyboard has 83 keys arranged in three major groupings. The central portion of the keyboard is a standard typewriter keyboard layout. On the left side are 10 function keys. These keys are user-defined by the software. On the right is a 15-key keypad. These keys are also defined by the software, but 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

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

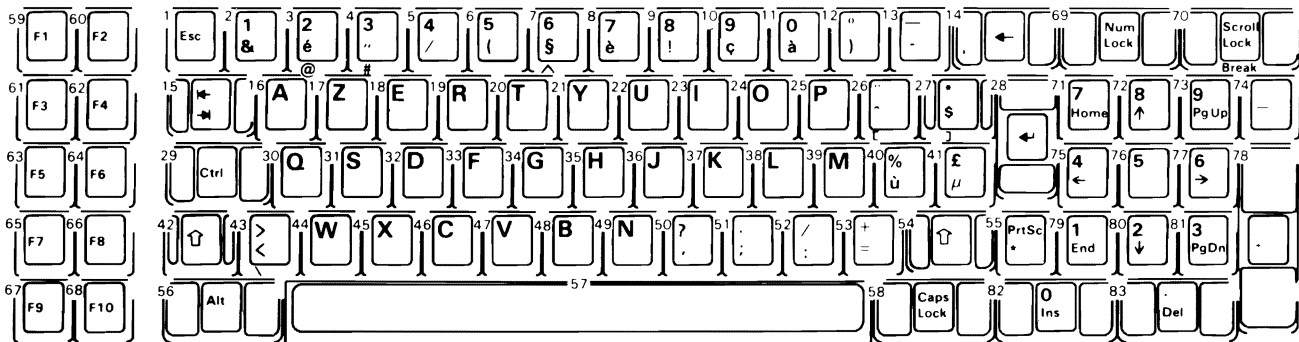
The keyboard routine does its own buffering. The keyboard buffer is large enough that few typists will ever fill it. However, if a key is pressed when the buffer is full, the key will be ignored and the “bell” will sound.

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

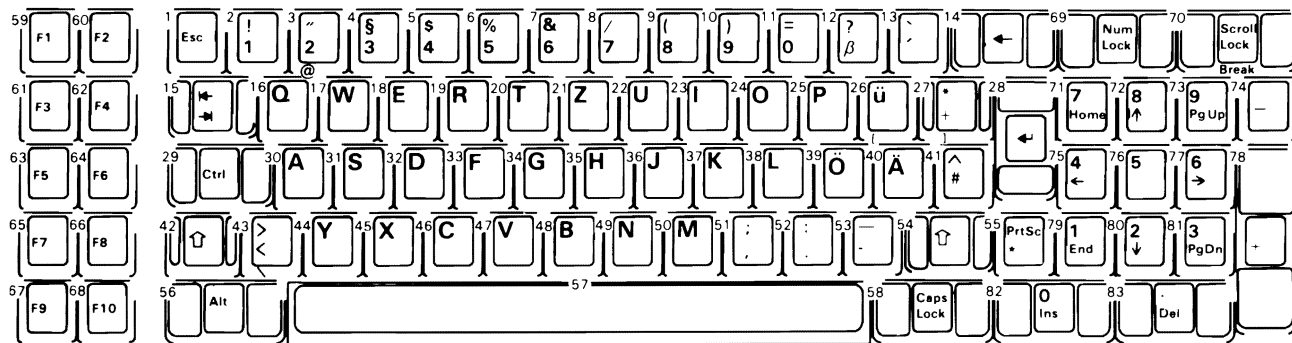
Keyboard Layouts

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

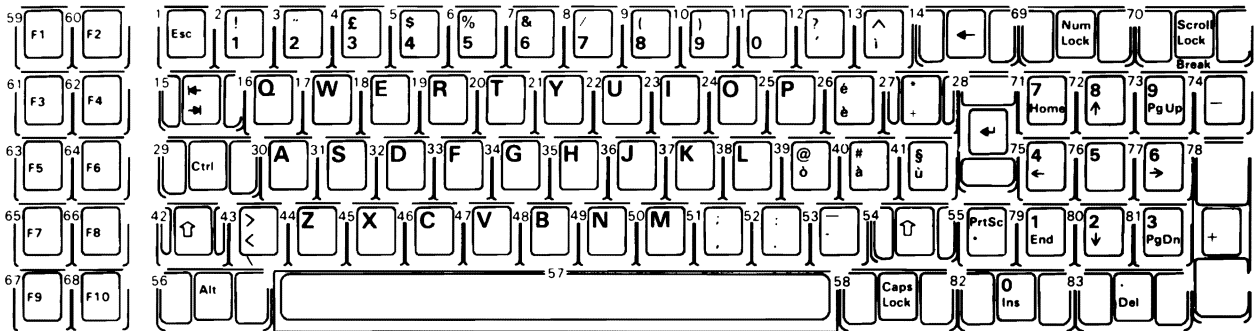
- French
- German
- Italian
- Spanish
- UK English
- US English



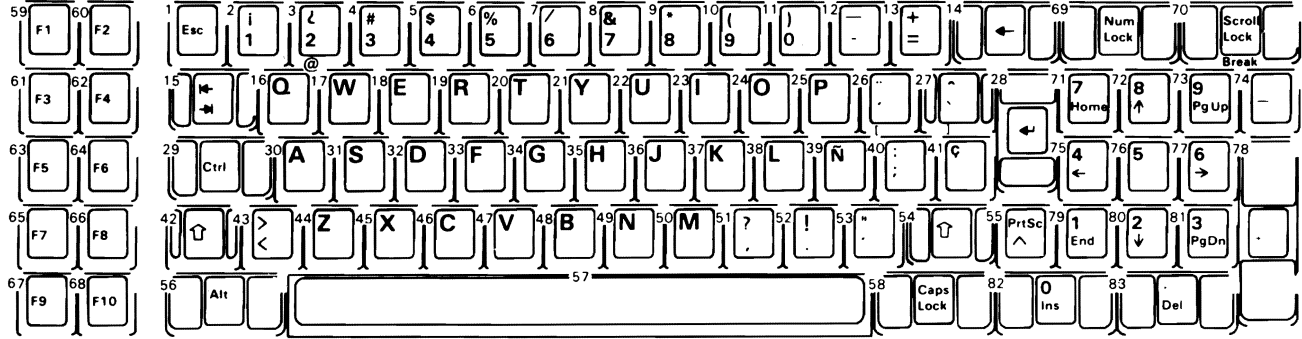
Note: Nomenclature is on both the top and front face of keybuttons as shown. The number to the upper left designates the button position.



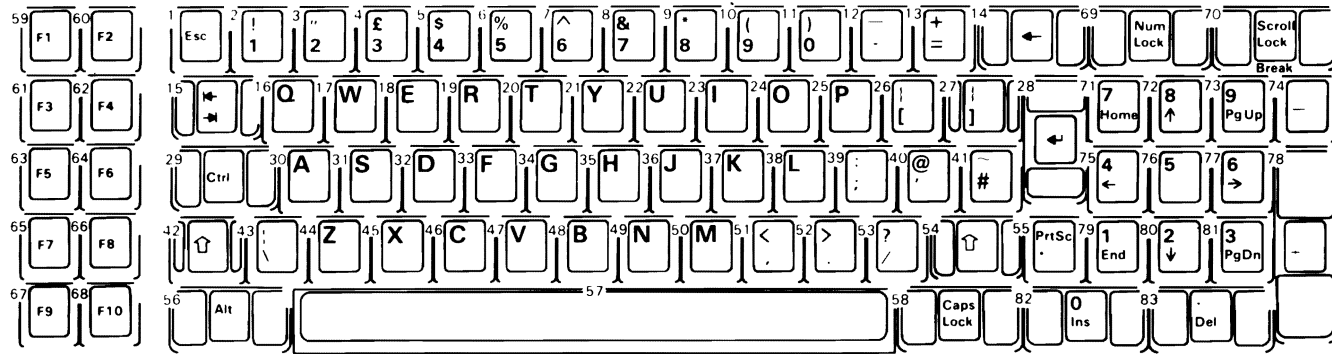
Note: Nomenclature is on both the top and front face of keybuttons as shown. The number to the upper left designates the button position.



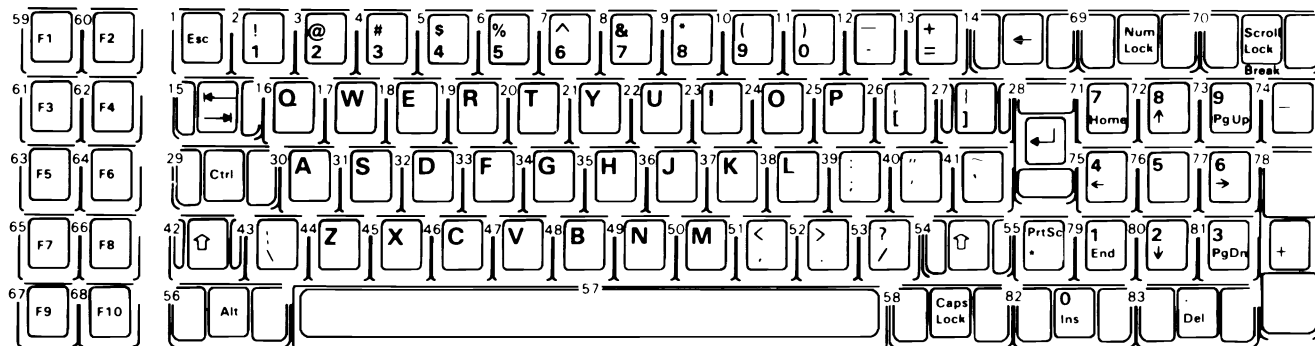
Note: Nomenclature is on both the top and front face of keybuttons as shown. The number to the upper left designates the button position.



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

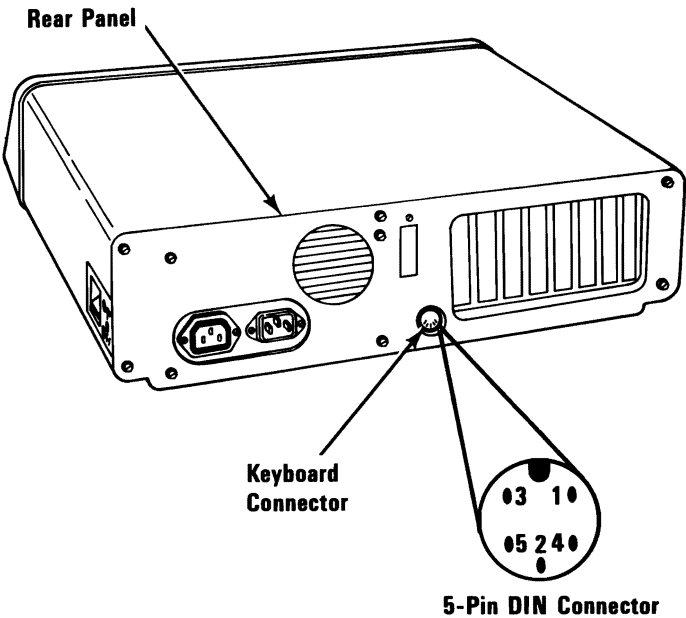


Note: Nomenclature is on both the top and front face of keybuttons as shown. The number to the upper left designates the button position.



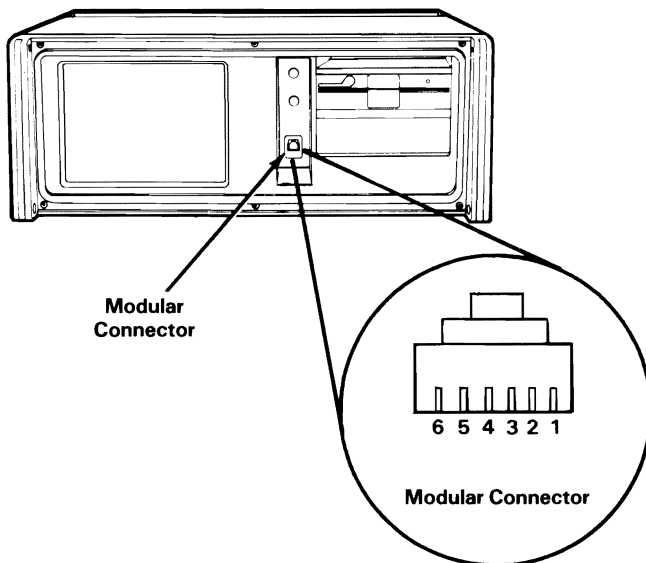
Note: Nomenclature is on both the top and front face of keybuttons as shown. The number to the upper left designates the button position.

Connector Specifications



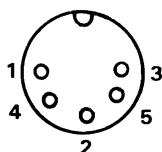
Pin	TTL Signal	Signal Level
1	+ Keyboard Clock	+ 5 Vdc
2	+ Keyboard Data	+ 5 Vdc
3	- Keyboard Reset (Not used by keyboard)	
	Power Supply Voltages	Voltage
4	Ground	0
5	+ 5 Volts	+ 5 Vdc

Keyboard Interface Connector Specifications

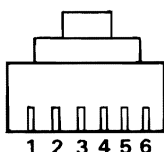


Keyboard Cable Connections

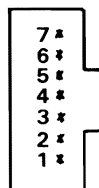
DIN Connector



Modular Connector



Keyboard Connector



	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.

SECTION 4



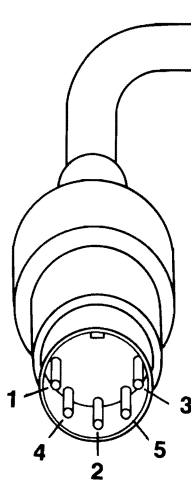
101/102-Key Keyboard

Description

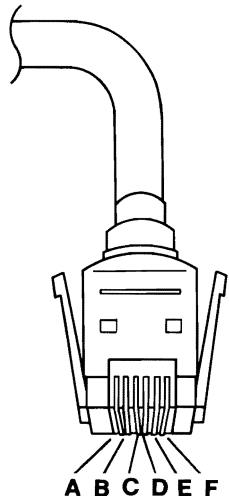
The keyboard has 101 keys (102 in countries outside the U. S.). At system power-on, the keyboard monitors the signals on the 'clock' and 'data' lines and establishes its line protocol.

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
Shield	F	Not used	
	Shield	Frame Ground	

Sequencing Key-Code Scanning

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

Keyboard Buffer

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

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

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

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

Keys

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

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

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.

Overflow (Hex FF)

An overflow 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.

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	EO 38	EO B8
64	EO 1D	EO 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	EO 1C	EO 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.

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.		

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,

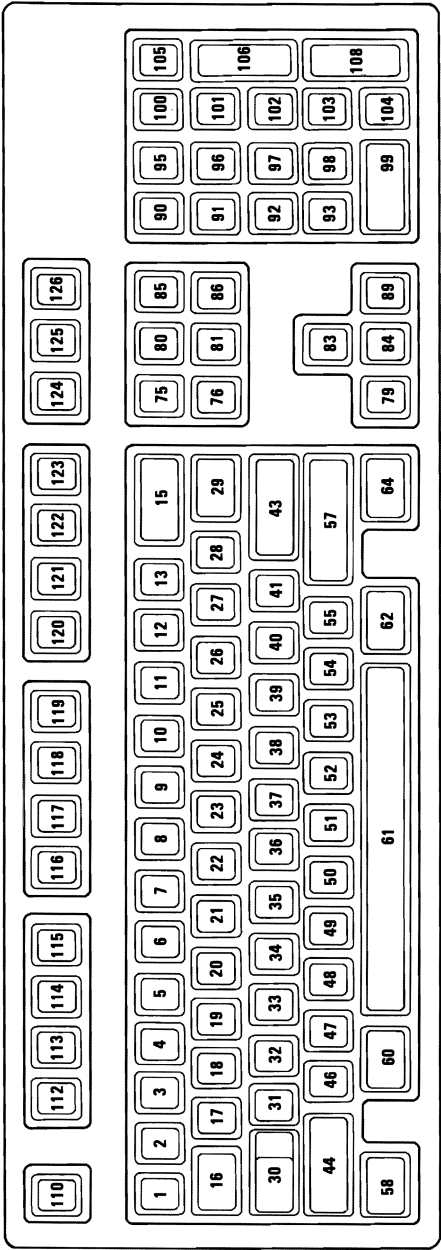
which resides in the read/write memory. This section discusses only the ROM routine.

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.



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

Notes:

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

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)

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.

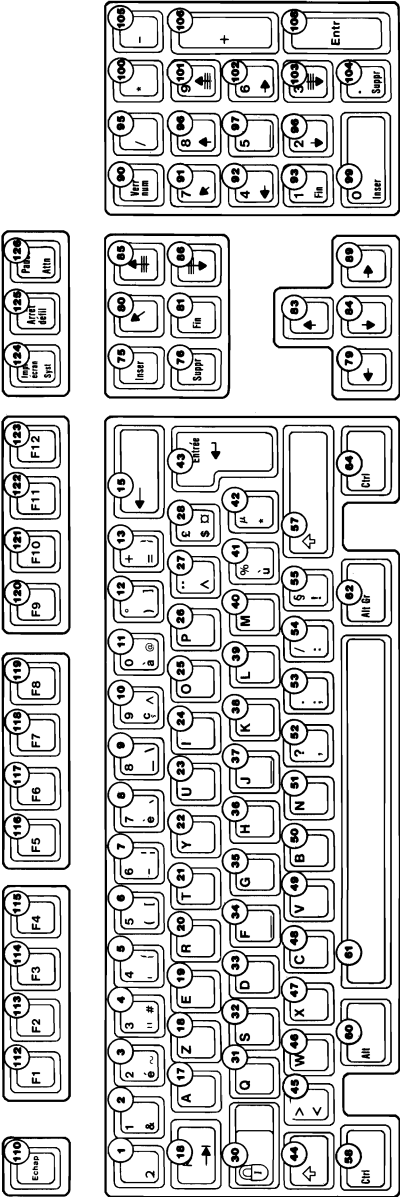
Keyboard Layouts

The keyboard is available in six layouts:

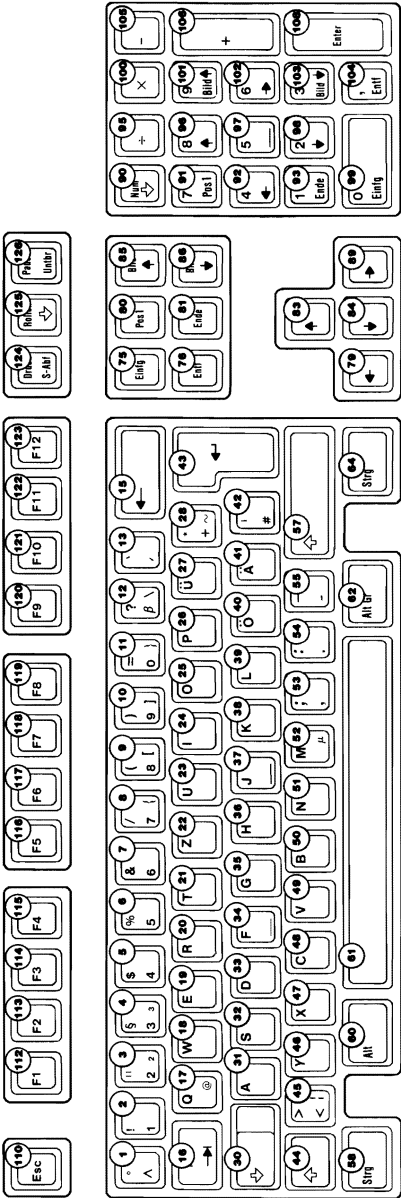
- French
- German
- Italian
- Spanish
- UK English
- US English

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.

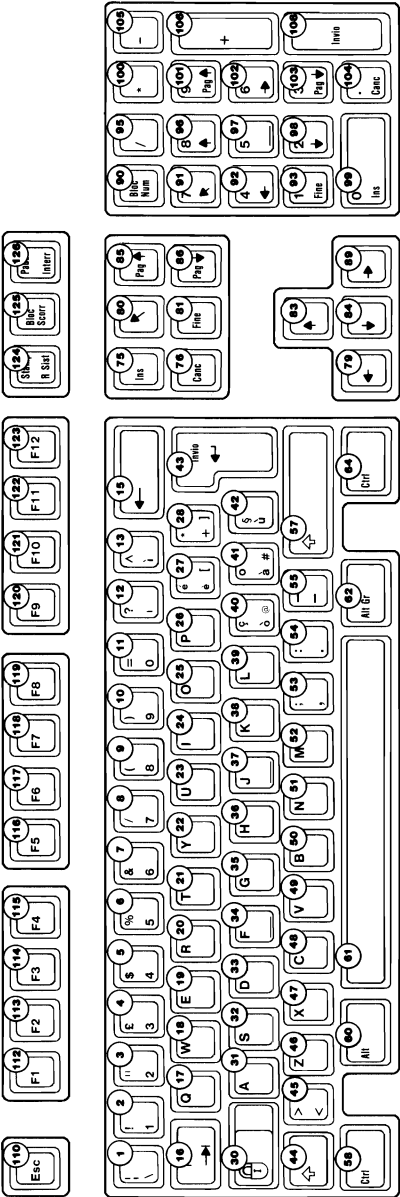
French Keyboard



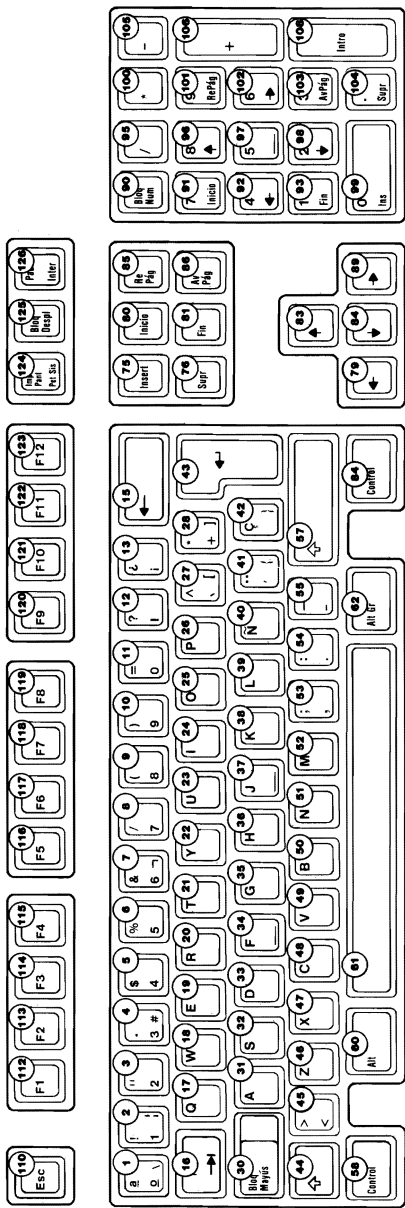
German Keyboard



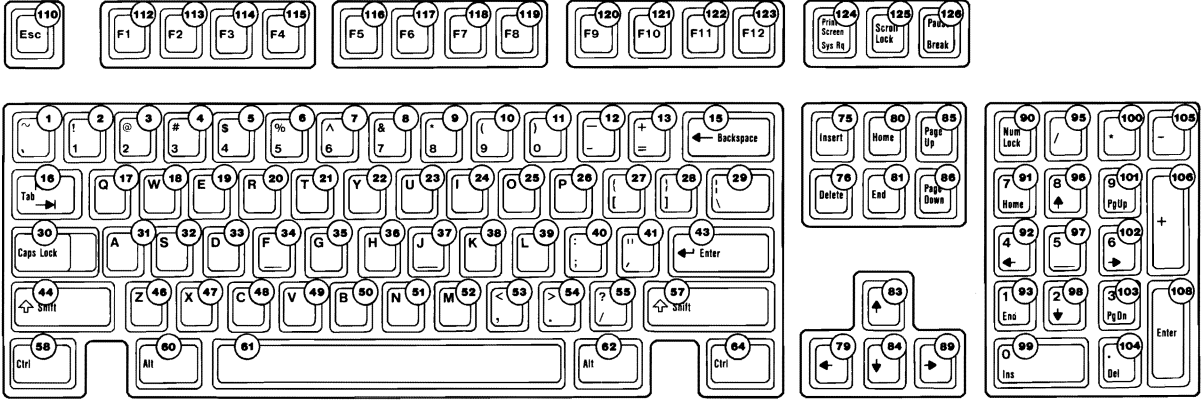
Italian Keyboard



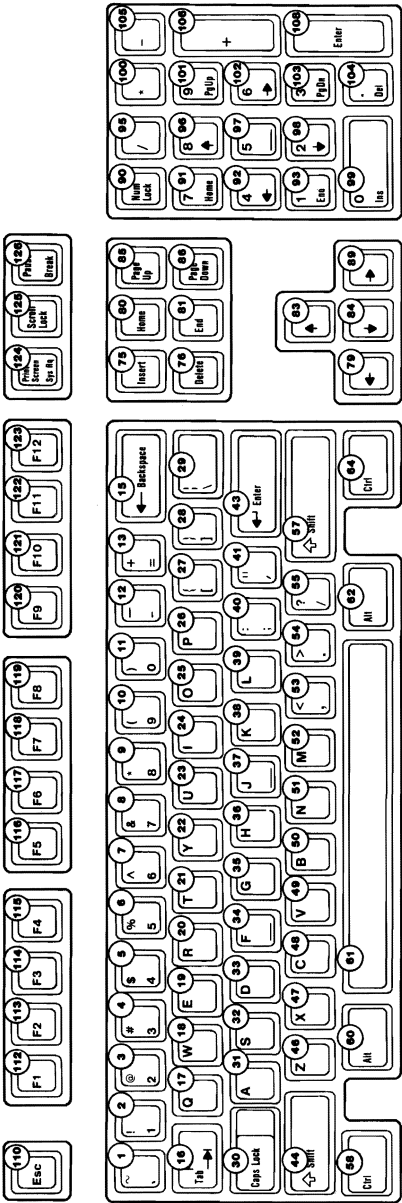
Spanish Keyboard



UK English Keyboard



US English Keyboard



Specifications

The specifications for the keyboard follow.

Power Requirements

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

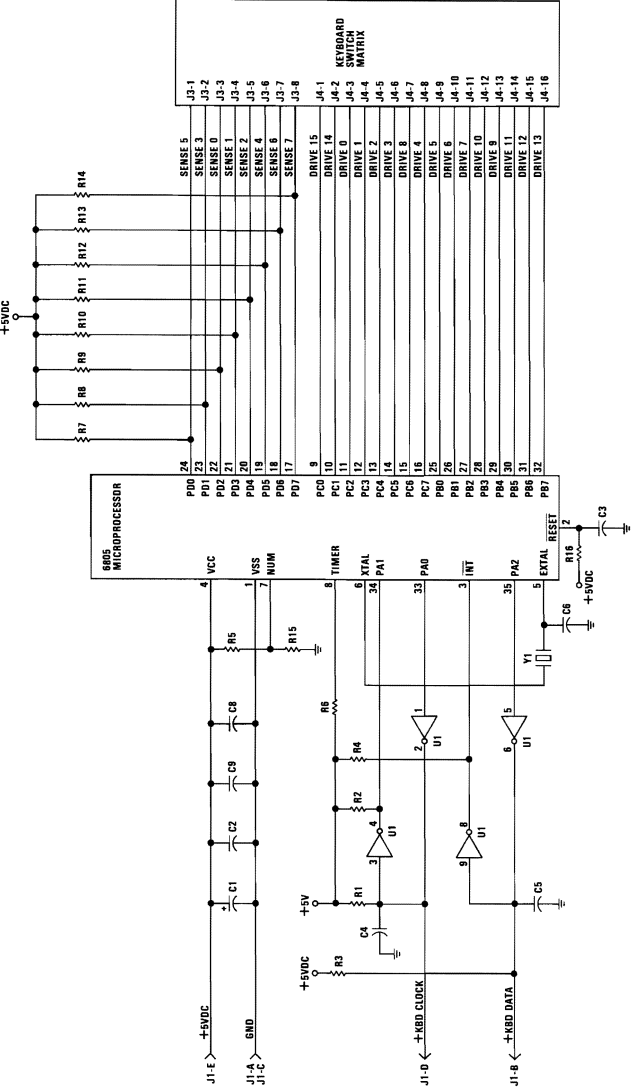
Size

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

Weight

2.25 kilograms (5.0 pounds)

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

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_TO
18	60-63	Resident BASIC	F600:0000
19	64-67	Bootstrap	BOOTSTRAP
1A	68-6B	Time of Day	TIME_OF_DAY
1B	6C-6F	Keyboard Break	DUMMY_RETURN
1C	70-73	Timer Tick	DUMMY_RETURN
1D	74-77	Video Initialization	VIDEO_PARMS
1E	78-7B	Diskette Parameters	DISK_BASE
1F	7C-7F	Video Graphics Chars	0
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	DOS	Reserved for DOS and BASIC
500		Print screen status flag store 0=Print screen not active or successful print screen operation 1=Print screen in progress 255=Error encountered during print screen operation
504		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 ($\text{length}/512$). 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

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	DOS
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:EFAD	MD_TBL4
F000:EFBA	MD_TBL5
F000:0A40	MD_TBL6
F000:12BE	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
F000:1062	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:EFAD	MD_TBL4
F000:EFBA	MD_TBL5
F000:EFC7	DISK_BASE
F000:F0A4	VIDEO_PARMS
F000:F0E4	M5
F000:F0EC	M6
F000:F0F4	M7
F000:FA12	DOS
F000:FA6E	CRT_CHAR_GEN
F000:FFF0	P_O_R

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_10_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_10_1	- INT 16H BIOS ENTRY -INT 16H
		KB_INT_1	- HARDWARE INTERRUPT -INT 09H
		SNB_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 (DSI) WITH DATA SEGMENT
		BEEP	- SPEAKER BEEP CONTROL ROUTINE
		WAITF	- FIXED TIME WAIT ROUTINE

.LIST

```

66      PAGE
67      C INCLUDE POSTEQU.INC
68      C ;-----
69      C ; EQUATES USED BY POST AND BIOS ;
70      C ;-----
71      C
72      = 0000
73      = 00FB
74      = 0000
75      = 0001
76      C
77      C
78      C ;----- KEYBOARD INTERFACE AND DIAGNOSTIC CONTROL REGISTERS -----
79      C PORT_A EQU 060H ; KEYBOARD SCAN CODE/CONTROL PORT
80      C PORT_B EQU 061H ; PORT B READ/WRITE DIAGNOSTIC REGISTER
81      C PORT_C EQU 062H ; 8255 PORT C ADDR
82      C CMD_PORT EQU 063H
83      C
84      C
85      = 0060
86      = 0061
87      = 0054
88      = 00E0
89      = 00E1
90      C
91      C
92      = 00F3
93      = 000C
94      = 00C0
95      = 0001
96      = 0002
97      = 0010
98      = 0020
99      = 0040
100     = 0080
101     C
102     C
103     C RAM_PAR_ON EQU 11110011B ; AND MASK FOR PARITY CHECKING ENABLE ON
104     C RAM_PAR_OFF EQU 00001100B ; OR MASK FOR PARITY CHECKING ENABLE OFF
105     C PARTTY_ERR EQU 11000000B ; R/W MEMORY - I/O CHANNEL PARITY ERROR
106     C GATE2 EQU 00000001B ; TIMER 2 INPUT GATE CLOCK BIT
107     C SPK2 EQU 00000010B ; SPEAKER OUTPUT DATA ENABLE BIT
108     C REFRESH_BIT EQU 00010000B ; REFRESH TEST BIT
109     C OUT2 EQU 00100000B ; SPEAKER TIMER OUT2 INPUT BIT
110     C IO_CHECK EQU 01000000B ; I/O (MEMORY) CHECK OCCURRED BIT MASK
111     C PARITY_CHECK EQU 10000000B ; MEMORY PARITY CHECK OCCURRED BIT MASK
112     C
113     C
114     C
115     C
116     C
117     C
118     C
119     C
120     C
121     C
122     C
123     C
124     C
125     C
126     C
127     C
128     C
129     C
130     C
131     C
132     C
133     C
134     C
135     C
136     C
137     C
138     C
139     C
140     C
141     C
142     C
143     C
144     C
145     C
146     C
147     C
148     C
149     C
150     C
151     C
152     C
153     C
154     C
155     C
156     C
157     C
158     C
159     C
160     C
161     C
162     C
163     C
164     C
165     C
166     C
167     C
168     C
169     C
170     C
171     C
172     C
173     C
174     C
175     C
176     C
177     C
178     C
179     C
180     C
181     C
182     C
183     C
184     C
185     C
186     C
187     C
188     C
189     C
190     C
191     C
192     C
193     C
194     C
195     C
196     C
197     C
198     C
199     C
200     C
201     C
202     C
203     C
204     C
205     C
206     C
207     C
208     C
209     C
210     C
211     C
212     C
213     C
214     C
215     C
216     C
217     C
218     C
219     C
220     C
221     C
222     C
223     C
224     C
225     C
226     C
227     C
228     C
229     C
230     C
231     C
232     C
233     C
234     C
235     C
236     C
237     C
238     C
239     C
240     C
241     C
242     C
243     C
244     C
245     C
246     C
247     C
248     C
249     C
250     C
251     C
252     C
253     C
254     C
255     C
256     C
257     C
258     C
259     C
260     C
261     C
262     C
263     C
264     C
265     C
266     C
267     C
268     C
269     C
270     C
271     C
272     C
273     C
274     C
275     C
276     C
277     C
278     C
279     C
280     C
281     C
282     C
283     C
284     C
285     C
286     C
287     C
288     C
289     C
290     C
291     C
292     C
293     C
294     C
295     C
296     C
297     C
298     C
299     C
300     C
301     C
302     C
303     C
304     C
305     C
306     C
307     C
308     C
309     C
310     C
311     C
312     C
313     C
314     C
315     C
316     C
317     C
318     C
319     C
320     C
321     C
322     C
323     C
324     C
325     C
326     C
327     C
328     C
329     C
330     C
331     C
332     C
333     C
334     C
335     C
336     C
337     C
338     C
339     C
340     C
341     C
342     C
343     C
344     C
345     C
346     C
347     C
348     C
349     C
350     C
351     C
352     C
353     C
354     C
355     C
356     C
357     C
358     C
359     C
360     C
361     C
362     C
363     C
364     C
365     C
366     C
367     C
368     C
369     C
370     C
371     C
372     C
373     C
374     C
375     C
376     C
377     C
378     C
379     C
380     C
381     C
382     C
383     C
384     C
385     C
386     C
387     C
388     C
389     C
390     C
391     C
392     C
393     C
394     C
395     C
396     C
397     C
398     C
399     C
400     C
401     C
402     C
403     C
404     C
405     C
406     C
407     C
408     C
409     C
410     C
411     C
412     C
413     C
414     C
415     C
416     C
417     C
418     C
419     C
420     C
421     C
422     C
423     C
424     C
425     C
426     C
427     C
428     C
429     C
430     C
431     C
432     C
433     C
434     C
435     C
436     C
437     C
438     C
439     C
440     C
441     C
442     C
443     C
444     C
445     C
446     C
447     C
448     C
449     C
450     C
451     C
452     C
453     C
454     C
455     C
456     C
457     C
458     C
459     C
460     C
461     C
462     C
463     C
464     C
465     C
466     C
467     C
468     C
469     C
470     C
471     C
472     C
473     C
474     C
475     C
476     C
477     C
478     C
479     C
480     C
481     C
482     C
483     C
484     C
485     C
486     C
487     C
488     C
489     C
490     C
491     C
492     C
493     C
494     C
495     C
496     C
497     C
498     C
499     C
500     C
501     C
502     C
503     C
504     C
505     C
506     C
507     C
508     C
509     C
510     C
511     C
512     C
513     C
514     C
515     C
516     C
517     C
518     C
519     C
520     C
521     C
522     C
523     C
524     C
525     C
526     C
527     C
528     C
529     C
530     C
531     C
532     C
533     C
534     C
535     C
536     C
537     C
538     C
539     C
540     C
541     C
542     C
543     C
544     C
545     C
546     C
547     C
548     C
549     C
550     C
551     C
552     C
553     C
554     C
555     C
556     C
557     C
558     C
559     C
560     C
561     C
562     C
563     C
564     C
565     C
566     C
567     C
568     C
569     C
570     C
571     C
572     C
573     C
574     C
575     C
576     C
577     C
578     C
579     C
580     C
581     C
582     C
583     C
584     C
585     C
586     C
587     C
588     C
589     C
590     C
591     C
592     C
593     C
594     C
595     C
596     C
597     C
598     C
599     C
600     C
601     C
602     C
603     C
604     C
605     C
606     C
607     C
608     C
609     C
610     C
611     C
612     C
613     C
614     C
615     C
616     C
617     C
618     C
619     C
620     C
621     C
622     C
623     C
624     C
625     C
626     C
627     C
628     C
629     C
630     C
631     C
632     C
633     C
634     C
635     C
636     C
637     C
638     C
639     C
640     C
641     C
642     C
643     C
644     C
645     C
646     C
647     C
648     C
649     C
650     C
651     C
652     C
653     C
654     C
655     C
656     C
657     C
658     C
659     C
660     C
661     C
662     C
663     C
664     C
665     C
666     C
667     C
668     C
669     C
670     C
671     C
672     C
673     C
674     C
675     C
676     C
677     C
678     C
679     C
680     C
681     C
682     C
683     C
684     C
685     C
686     C
687     C
688     C
689     C
690     C
691     C
692     C
693     C
694     C
695     C
696     C
697     C
698     C
699     C
700     C
701     C
702     C
703     C
704     C
705     C
706     C
707     C
708     C
709     C
710     C
711     C
712     C
713     C
714     C
715     C
716     C
717     C
718     C
719     C
720     C
721     C
722     C
723     C
724     C
725     C
726     C
727     C
728     C
729     C
730     C
731     C
732     C
733     C
734     C
735     C
736     C
737     C
738     C
739     C
740     C
741     C
742     C
743     C
744     C
745     C
746     C
747     C
748     C
749     C
750     C
751     C
752     C
753     C
754     C
755     C
756     C
757     C
758     C
759     C
760     C
761     C
762     C
763     C
764     C
765     C
766     C
767     C
768     C
769     C
770     C
771     C
772     C
773     C
774     C
775     C
776     C
777     C
778     C
779     C
780     C
781     C
782     C
783     C
784     C
785     C
786     C
787     C
788     C
789     C
790     C
791     C
792     C
793     C
794     C
795     C
796     C
797     C
798     C
799     C
800     C
801     C
802     C
803     C
804     C
805     C
806     C
807     C
808     C
809     C
810     C
811     C
812     C
813     C
814     C
815     C
816     C
817     C
818     C
819     C
820     C
821     C
822     C
823     C
824     C
825     C
826     C
827     C
828     C
829     C
830     C
831     C
832     C
833     C
834     C
835     C
836     C
837     C
838     C
839     C
840     C
841     C
842     C
843     C
844     C
845     C
846     C
847     C
848     C
849     C
850     C
851     C
852     C
853     C
854     C
855     C
856     C
857     C
858     C
859     C
860     C
861     C
862     C
863     C
864     C
865     C
866     C
867     C
868     C
869     C
870     C
871     C
872     C
873     C
874     C
875     C
876     C
877     C
878     C
879     C
880     C
881     C
882     C
883     C
884     C
885     C
886     C
887     C
888     C
889     C
890     C
891     C
892     C
893     C
894     C
895     C
896     C
897     C
898     C
899     C
900     C
901     C
902     C
903     C
904     C
905     C
906     C
907     C
908     C
909     C
910     C
911     C
912     C
913     C
914     C
915     C
916     C
917     C
918     C
919     C
920     C
921     C
922     C
923     C
924     C
925     C
926     C
927     C
928     C
929     C
930     C
931     C
932     C
933     C
934     C
935     C
936     C
937     C
938     C
939     C
940     C
941     C
942     C
943     C
944     C
945     C
946     C
947     C
948     C
949     C
950     C
951     C
952     C
953     C
954     C
955     C
956     C
957     C
958     C
959     C
960     C
961     C
962     C
963     C
964     C
965     C
966     C
967     C
968     C
969     C
970     C
971     C
972     C
973     C
974     C
975     C
976     C
977     C
978     C
979     C
980     C
981     C
982     C
983     C
984     C
985     C
986     C
987     C
988     C
989     C
990     C
991     C
992     C
993     C
994     C
995     C
996     C
997     C
998     C
999     C
1000    C

```

```

102 C PAGE
103 C ;----- KEYBOARD RESPONSE -----
104 = 00AA C KB_OK EQU 0AAH ; RESPONSE FROM SELF DIAGNOSTIC
105 = 00FA C KB_ACK EQU 0FAH ; ACKNOWLEDGE FROM TRANSMISSION
106 = 00FE C KB_RESEND EQU 0FEH ; RESEND REQUEST
107 = 00FF C KB_OVER_RUN EQU 0FFH ; OVER RUN SCAN CODE
108 C
109 C ;----- FLAG EQUATES WITHIN *KB_FLAG -----
110 = 0001 C RIGHT_SHIFT EQU 00000001B ; RIGHT SHIFT KEY DEPRESSED
111 = 0002 C LEFT_SHIFT EQU 00000100B ; LEFT SHIFT KEY DEPRESSED
112 = 0004 C CTL_SHIFT EQU 00000100B ; CONTROL SHIFT KEY DEPRESSED
113 = 0008 C ALT_SHIFT EQU 00001000B ; ALTERNATE SHIFT KEY DEPRESSED
114 = 0010 C SCROLL_STATE EQU 00010000B ; SCROLL LOCK STATE HAS BEEN TOGGLED
115 = 0020 C NUM_STATE EQU 00100000B ; NUM LOCK STATE HAS BEEN TOGGLED
116 = 0040 C CAPS_STATE EQU 01000000B ; CAPS LOCK STATE HAS BEEN TOGGLED
117 = 0080 C INS_STATE EQU 10000000B ; INSERT STATE IS ACTIVE
118 C
119 C ;----- FLAG EQUATES WITHIN *KB_FLAG_1 -----
120 = 0004 C SYS_SHIFT EQU 00000100B ; SYSTEM KEY DEPRESSED AND HELD
121 = 0008 C HOLD_STATE EQU 00001000B ; SYSTEM KEY HAS BEEN TOGGLED
122 = 0010 C SCROLL_SHIFT EQU 00010000B ; SCROLL LOCK KEY IS DEPRESSED
123 = 0020 C NUM_SHIFT EQU 00100000B ; NUM LOCK KEY IS DEPRESSED
124 = 0040 C CAPS_SHIFT EQU 01000000B ; CAPS LOCK KEY IS DEPRESSED
125 = 0080 C INS_SHIFT EQU 10000000B ; INSERT KEY IS DEPRESSED
126 C
127 C ;----- FLAGS EQUATES WITHIN *KB_FLAG_2 -----
128 = 0007 C KB_LEDS EQU 00000111B ; KEYBOARD LED STATE BITS
129 C ; ; RESERVED (MUST BE ZERO)
130 = 0010 C KB_FA EQU 00010000B ; ACKNOWLEDGMENT RECEIVED
131 = 0020 C KB_FE EQU 00100000B ; RESEND RECEIVED FLAG
132 = 0040 C KB_PR_LED EQU 01000000B ; MODE INDICATOR UPDATE
133 = 0080 C KB_ERR EQU 10000000B ; KEYBOARD TRANSMIT ERROR FLAG
134 C
135 C ;----- FLAGS EQUATES WITHIN *KB_FLAG_3 -----
136 = 0001 C LC_E1 EQU 00000001B ; LAST CODE WAS THE E1 HIDDEN CODE
137 = 0002 C LC_E0 EQU 00000010B ; LAST CODE WAS THE E0 HIDDEN CODE
138 = 0004 C R_CTL_SHIFT EQU 00000100B ; RIGHT CTL KEY DOWN
139 = 0008 C GRAPH_ON EQU 00001000B ; ALL GRAPHICS KEY DOWN (W.T. ONLY)
140 C ; ; RESERVED (MUST BE ZERO)
141 = 0010 C KBX EQU 00010000B ; KBX INSTALLED
142 = 0020 C SET_NUM_LK EQU 00100000B ; FORCE NUM LOCK IF READ ID AND KBX
143 = 0040 C LC_AB EQU 01000000B ; LAST CHARACTER WAS FIRST ID CHARACTER
144 = 0080 C RD_ID EQU 10000000B ; DOING A READ ID (MUST BE BIT0)
145 C
146 C ;----- KEYBOARD SCAN CODES -----
147 = 00AB C ID_1 EQU 0ABH ; 1ST ID CHARACTER FOR KBX
148 = 0041 C ID_2 EQU 041H ; 2ND ID CHARACTER FOR KBX
149 = 0038 C ALT_KEY EQU 56 ; SCAN CODE FOR ALTERNATE SHIFT KEY
150 = 001D C CTL_KEY EQU 29 ; SCAN CODE FOR CONTROL KEY
151 = 003A C CAPS_KEY EQU 58 ; SCAN CODE FOR SHIFT LOCK KEY
152 = 0053 C DEL_KEY EQU 83 ; SCAN CODE FOR DELETE KEY
153 = 0052 C INS_KEY EQU 82 ; SCAN CODE FOR INSERT KEY
154 = 002A C LEFT_KEY EQU 42 ; SCAN CODE FOR LEFT SHIFT
155 = 0045 C NUM_KEY EQU 69 ; SCAN CODE FOR NUMBER LOCK KEY
156 = 0036 C RIGHT_KEY EQU 54 ; SCAN CODE FOR RIGHT SHIFT
157 = 0046 C SCROLL_KEY EQU 70 ; SCAN CODE FOR SCROLL LOCK KEY
158 = 0054 C SYS_KEY EQU 84 ; SCAN CODE FOR SYSTEM KEY
159 = 0057 C F11_M EQU 87 ; F11 KEY MAKE
160 = 0058 C F12_M EQU 88 ; F12 KEY MAKE

```



```

161 C PAGE
162 C ENDIF
163
164 C ----- DISKETTE EQUATES -----
165 C CARD_ID EQU 01010000B ; CONTROLLER CARD I.D. BIT
166 C DUAL EQU 00000001B ; MASK FOR FDC ADAPTER I.D.
167 C INT_FLAG EQU 10000000B ; INTERRUPT OCCURRENCE FLAG
168 C DSK_CHG EQU 10000000B ; DISKETTE CHANGE FLAG MASK BIT
169 C DETERMINED EQU 00010000B ; SET STATE DETERMINED IN STATE BITS
170 C HOME EQU 00010000B ; TRACK 0 MASK
171 C SENSE_DRV_ST EQU 00000100B ; SENSE DRIVE STATUS COMMAND
172 C TRK_SLAP EQU 030H ; CRASH STOP (48 TP) DRIVES)
173 C QUIET_SEEK EQU 00AH ; SEEK TO TRACK 10
174 C MAX_DRV EQU 2 ; MAX NUMBER OF DRIVES
175 C HD12_SETTLE EQU 15 ; 1.2 M HEAD SETTLE TIME
176 C HD320_SETTLE EQU 20 ; 320 K HEAD SETTLE TIME
177 C MOTOR_WAIT EQU 37 ; 2 SECONDS OF COUNTS FOR MOTOR TURN OFF
178
179 C ----- DISKETTE ERRORS -----
180 C TIME_OUT EQU 080H ; ATTACHMENT FAILED TO RESPOND
181 C BAD_SEEK EQU 040H ; SEEK OPERATION FAILED
182 C BAD_NEC EQU 020H ; DISKETTE CONTROLLER HAS FAILED
183 C BAD_CRC EQU 010H ; BAD CRC ON DISKETTE READ
184 C MED_NOT_FND EQU 00CH ; MEDIA TYPE FOUND
185 C DMA_BOUNDARY EQU 009H ; ATTEMPT TO DMA ACROSS 64K BOUNDARY
186 C BAD_DMA EQU 008H ; DMA OVERRUN ON OPERATION
187 C MEDIA_CHANGE EQU 006H ; MEDIA REMOVED ON DUAL ATTACH CARD
188 C RECORD_NOT_FND EQU 004H ; REQUESTED SECTOR NOT FOUND
189 C WRITE_PROTECT EQU 003H ; WRITE ATTEMPTED ON WRITE PROTECT DISK
190 C BAD_ADDR_MARK EQU 002H ; ADDRESS MARK NOT FOUND
191 C BAD_CMD EQU 001H ; BAD COMMAND PASSED TO DISKETTE I/O
192
193 C ----- DISK CHANGE LINE EQUATES -----
194 C NOCHGLN EQU 001H ; NO DISK CHANGE LINE AVAILABLE
195 C CHGLN EQU 002H ; DISK CHANGE LINE AVAILABLE
196
197 C ----- MEDIA/DRIVE STATE INDICATORS -----
198 C TRK_CAPA EQU 00000001B ; 80 TRACK CAPABILITY
199 C FMT_CAPA EQU 00000010B ; MULTIPLE FORMAT CAPABILITY (1.2M)
200 C DRV_DET EQU 00000100B ; DRIVE DETERMINED
201 C MED_DET EQU 00010000B ; MEDIA DETERMINED BIT
202 C DBL_STEP EQU 00100000B ; DOUBLE STEP BIT
203 C RATE_MSK EQU 11000000B ; MASK FOR CLEARING ALL BUT RATE
204 C RATE_500 EQU 00000000B ; 500 KBS DATA RATE
205 C RATE_300 EQU 01000000B ; 300 KBS DATA RATE
206 C RATE_250 EQU 10000000B ; 250 KBS DATA RATE
207 C STRT_MSK EQU 00001100B ; OPERATION START RATE MASK
208 C SEND_MSK EQU 11000000B ; MASK FOR SEND RATE BITS
209
210 C ----- MEDIA/DRIVE STATE INDICATORS COMPATIBILITY -----
211 C M3D3U EQU 00000000B ; 360 MEDIA/DRIVE NOT ESTABLISHED
212 C M3D1U EQU 00000001B ; 360 MEDIA,1.2DRIVE NOT ESTABLISHED
213 C M1D1U EQU 00000010B ; 1.2 MEDIA/DRIVE NOT ESTABLISHED
214 C MED_UNK EQU 00000111B ; NONE OF THE ABOVE

```

```

215 C PAGE
216 C 1----- INTERRUPT EQUATES -----
217 C EQ EQU 020H ; END OF INTERRUPT COMMAND TO 8259
218 C INTA00 EQU 020H ; 8259 PORT
219 C INTA01 EQU 021H ; 8259 PORT
220 C INTB00 EQU 0A0H ; 2ND 8259
221 C INTB01 EQU 0A1H ;
222 C INT_TYPE EQU 010H ; START OF 8259 INTERRUPT TABLE LOCATION
223 C INT_VIDEO EQU 010H ; VIDEO VECTOR
224 C 1-----
225 C DMA08 EQU 008H ; DMA STATUS REGISTER PORT ADDRESS
226 C DMA EQU 000H ; DMA CH.0 ADDRESS REGISTER PORT ADDRESS
227 C DMA18 EQU 0D0H ; 2ND DMA STATUS PORT ADDRESS
228 C DMA1 EQU 0C0H ; 2ND DMA CH.0 ADDRESS REGISTER ADDRESS
229 C 1-----
230 C TIMER EQU 040H ; 8253 TIMER - BASE ADDRESS
231 C TIM_CTL EQU 043H ; 8253 TIMER CONTROL PORT ADDR
232 C TIMER0 EQU 040H ; 8253 TIMER/CNTNR 0 PORT ADDR
233 C 1-----
234 C ;----- MANUFACTURING PORT -----
235 C MFG_PORT EQU 80H ; MANUFACTURING AND POST CHECKPOINT PORT
236 C ; DMA CHANNEL 0 PAGE REGISTER ADDRESS
237 C 1-----
238 C ;----- MANUFACTURING BIT DEFINITION FOR 0MFG.ERR.FLAG+1 -----
239 C MEM_FAIL EQU 00000001B ; STORAGE TEST FAILED (ERROR 20X)
240 C PRO_FAIL EQU 00000010B ; VIRTUAL MODE TEST FAILED (ERROR 104)
241 C LMCS_FAIL EQU 00000100B ; LOW MEG CHIP SELECT FAILED (ERROR 109)
242 C KYCLK_FAIL EQU 00001000B ; KEYBOARD CLOCK TEST FAILED (ERROR 304)
243 C KY_SYS_FAIL EQU 00010000B ; KEYBOARD OR SYSTEM FAILED (ERROR 303)
244 C KYBD_FAIL EQU 00100000B ; KEYBOARD FAILED (ERROR 301)
245 C DSK_FAIL EQU 01000000B ; DISKETTE TEST FAILED (ERROR 601)
246 C KEY_FAIL EQU 10000000B ; KEYBOARD LOCKED (ERROR 302)
247 C 1-----
248 C 1-----
249 C DMA_PAGE EQU 081H ; START OF DMA PAGE REGISTERS
250 C LAST_DMA_PAGE EQU 08FH ; LAST DMA PAGE REGISTER
251 C 1-----
252 C 1-----
253 C 1X287 EQU 0F0H ; MATH COPROCESSOR CONTROL PORT
254 C 1-----
255 C 1-----
256 C POST_SS EQU 000000H ; POST STACK SEGMENT
257 C POST_SP EQU 080000H ; POST STACK POINTER
258 C STACK_SS EQU 30H ; STACK SEGMENT USED DURING POST
259 C TOS EQU 100H ; STACK -- USED DURING POST ONLY
260 C ; USE WILL OVERLAY INTERRUPTS VECTORS
261 C 1-----
262 C 1-----
263 C 1-----
264 C CR EQU 000DH ; CARRIAGE RETURN CHARACTER
265 C LF EQU 000AH ; LINE FEED CHARACTER
266 C RVRT EQU 00001000B ; VIDEO VERTICAL RETRACE BIT
267 C RHRZ EQU 00000001B ; VIDEO HORIZONTAL RETRACE BIT
268 C H EQU 256 ; HIGH BYTE FACTOR (X 100H)
269 C X EQU H+1 ; HIGH AND LOW BYTE FACTOR (X 101H)
270 C
271 .LIST

```

```

272      PAGE
273      C INCLUDE DSEG.INC
274      C ELSE
275      C
276      C ;-----8088 INTERRUPT LOCATIONS-----
277      C ; REFERENCED BY POST & BIOS
278      C ;-----
279      C ENDIF
280
281      0000      ABS0          SEGMENT AT 0          ; ADDRESS= 0000:0000
282
283      0000 ??    *STG_LOC0    DB      ?            ; START OF INTERRUPT VECTOR TABLE
284
285      0008      *NMI_PTR      ORG      4*002H        ;
286      0008 ??????????      DO      ?            ; NON-MASKABLE INTERRUPT VECTOR
287
288      0014      *INT5_PTR     ORG      4*005H        ;
289      0014 ??????????      DO      ?            ; PRINT SCREEN INTERRUPT VECTOR
290
291      0020      *INT_PTR      ORG      4*008H        ;
292      0020 ??????????      DO      ?            ; HARDWARE INTERRUPT POINTER (8-F)
293
294      0040      *VIDEO_INT    ORG      4*010H        ;
295      0040 ??????????      DO      ?            ; VIDEO I/O INTERRUPT VECTOR
296
297      004C      *ORG_VECTOR   ORG      4*013H        ;
298      004C ??????????      DO      ?            ; DISKETTE/DISK INTERRUPT VECTOR
299
300      0060      *BASIC_PTR    ORG      4*018H        ;
301      0060 ??????????      DO      ?            ; POINTER TO CASSETTE BASIC
302
303      0074      *PARM_PTR     ORG      4*01DH        ;
304      0074 ??????????      DO      ?            ; POINTER TO VIDEO PARAMETERS
305
306      0078      *DISK_POINTER ORG      4*01EH        ;
307      0078 ??????????      DO      ?            ; POINTER TO DISKETTE PARAMETER TABLE
308
309      007C      *EXT_PTR      ORG      4*01FH        ;
310      007C ??????????      DO      ?            ; POINTER TO GRAPHIC CHARACTERS 128-255
311
312      0100      *DISK_VECTOR  ORG      4*040H        ;
313      0100 ??????????      DO      ?            ; POINTER TO DISKETTE INTERRUPT CODE
314
315      0104      *HFI_TBL_VEC  ORG      4*041H        ;
316      0104 ??????????      DO      ?            ; POINTER TO FIRST DISK PARAMETER TABLE
317
318      0118      *HFI_TBL_VEC  ORG      4*046H        ;
319      0118 ??????????      DO      ?            ; POINTER TO SECOND DISK PARAMETER TABLE
320
321      01C0      *SLAVE_INT_PTR ORG      4*070H        ;
322      01C0 ??????????      DO      ?            ; POINTER TO SLAVE INTERRUPT HANDLER
323
324      01D8      *HDISK_INT    ORG      4*076H        ;
325      01D8 ??????????      DO      ?            ; POINTER TO FIXED DISK INTERRUPT CODE
326
327      0400      DATA_AREA    ORG      400H          ;
328      0400      DATA_WORD   LABEL  BYTE           ; ABSOLUTE LOCATION OF DATA SEGMENT
329      0400
330
331      0500      *MFG_TEST_RTN ORG      0500H        ;
332      0500      LABEL  FAR           ; LOAD LOCATION FOR MANUFACTURING TESTS
333
334      7C00      *BOOT_LOCN    ORG      7C00H        ;
335      7C00      LABEL  FAR           ; BOOT STRAP CODE LOAD LOCATION
336
337      7C00      ABS0          ENDS

```

```

338 C PAGE -----
339 C |
340 C | ROM BIOS DATA AREAS |
341 C |-----|
342 C |
343 C DATA SEGMENT AT 40H | ADDRESS= 0040:0000
344 C DATA40 LABEL BYTE
345 C *RS232_BASE DW ? | BASE ADDRESSES OF RS232 ADAPTERS
346 C 0000 ? | SECOND LOGICAL RS232 ADAPTER
347 C 0004 ? | RESERVED
348 C 0006 ? | RESERVED
349 C *PRINTER_BASE DW ? | BASE ADDRESSES OF PRINTER ADAPTERS
350 C 0008 ? | SECOND LOGICAL PRINTER ADAPTER
351 C 000C ? | THIRD LOGICAL PRINTER ADAPTER
352 C 000E ? | RESERVED
353 C *EQUIP_FLAG DW ? | INSTALLED HARDWARE FLAGS
354 C *MFG_PST DB ? | INITIALIZATION FLAGS
355 C *MEMORY_SIZE DW ? | BASE MEMORY SIZE IN K BYTES (X 1024)
356 C *MFG_ERR_FLAG DB ? | SCRATCHPAD FOR MANUFACTURING
357 C 0016 ? | ERROR CODES
358 C |-----|
359 C |
360 C | KEYBOARD DATA AREAS |
361 C |-----|
362 C |
363 C *KB_FLAG DB ? | KEYBOARD SHIFT STATE AND STATUS FLAGS
364 C *KB_FLAG_1 DB ? | SECOND BYTE OF KEYBOARD STATUS
365 C *ALT_INPUT DB ? | STORAGE FOR ALTERNATE KEY PAD ENTRY
366 C *BUFFER_HEAD DW ? | POINTER TO HEAD OF KEYBOARD BUFFER
367 C *BUFFER_TAIL DW ? | POINTER TO TAIL OF KEYBOARD BUFFER
368 C |
369 C |----- HEAD = TAIL INDICATES THAT THE BUFFER IS EMPTY
370 C |
371 C *KB_BUFFER DW 16 DUP(?) | ROOM FOR 15 SCAN CODE ENTRIES
372 C |
373 C |-----|
374 C |
375 C | DISKETTE DATA AREAS |
376 C |-----|
377 C |
378 C *SEEK_STATUS DB ? | DRIVE RECALIBRATION STATUS
379 C | BIT 3-0 = DRIVE 3-0 RECALIBRATION
380 C | BEFORE NEXT SEEK IF BIT 15 = 0
381 C *MOTOR_STATUS DB ? | MOTOR STATUS
382 C | BIT 3-0 = DRIVE 3-0 CURRENTLY RUNNING
383 C | BIT 7 = CURRENT OPERATION IS A WRITE
384 C *MOTOR_COUNT DB ? | TIME OUT COUNTER FOR MOTOR(S) TURN OFF
385 C *DSKETTE_STATUS DB ? | RETURN CODE STATUS BYTE
386 C 0041 ? | CMD BLOCK IN STACK FOR DISK OPERATION
387 C *NEC_STATUS DB 7 DUP(?) | STATUS BYTES FROM DISKETTE OPERATION
388 C |
389 C |-----|
390 C |
391 C | VIDEO DISPLAY DATA AREA |
392 C |-----|
393 C |
394 C *CRT_MODE DB ? | CURRENT DISPLAY MODE (TYPE)
395 C *CRT_COLS DW ? | NUMBER OF COLUMNS ON SCREEN
396 C *CRT_LEN DW ? | LENGTH OF REGEN BUFFER IN BYTES
397 C *CRT_START DW ? | STARTING ADDRESS IN REGEN BUFFER
398 C *CURSOR_POSN DW 8 DUP(?) | CURSOR FOR EACH OF UP TO 8 PAGES
399 C |
400 C |-----|
401 C |
402 C *CURSOR_MODE DW ? | CURRENT CURSOR MODE SETTING
403 C *ACTIVE_PAGE DB ? | CURRENT PAGE BEING DISPLAYED
404 C *ADDR_6845 DW ? | BASE ADDRESS FOR ACTIVE DISPLAY CARD
405 C *CRT_MODE_SET DB ? | CURRENT SETTING OF THE 3X6 REGISTER
406 C *CRT_PALETTE DB ? | CURRENT PALETTE SETTING - COLOR CARD
407 C |
408 C |-----|
409 C |
410 C | POST AND BIOS WORK DATA AREA |
411 C |-----|
412 C |
413 C *IO_ROM_INIT DW ? | STACK SAVE, ETC.
414 C *IO_ROM_SEG DW ? | POINTER TO ROM INITIALIZATION ROUTINE
415 C *INTR_FLAG DB ? | POINTER TO I/O ROM SEGMENT
416 C 0067 ? | FLAG INDICATING AN INTERRUPT HAPPENED
417 C |
418 C |-----|
419 C |
420 C | TIMER DATA AREA |
421 C |-----|
422 C |
423 C *TIMER_LOW DW ? | LOW WORD OF TIMER COUNT
424 C *TIMER_HIGH DW ? | HIGH WORD OF TIMER COUNT
425 C *TIMER_OFL DB ? | TIMER HAS ROLLED OVER SINCE LAST READ
426 C |
427 C |-----|
428 C |
429 C | SYSTEM DATA AREA |
430 C |-----|
431 C |
432 C *BIOS_BREAK DB ? | BIT 7=1 IF BREAK KEY HAS BEEN PRESSED
433 C *RESET_FLAG DB ? | WORD=1234H IF KEYBOARD RESET UNDERWAY
434 C |
435 C |-----|
436 C |
437 C | FIXED DISK DATA AREAS |
438 C |-----|
439 C |
440 C *DISK_STATUS1 DB ? | FIXED DISK STATUS
441 C *HP_NOW DB ? | COUNT OF FIXED DISK DRIVES
442 C *CONTROL_BYTE DB ? | HEAD CONTROL BYTE
443 C *PORT_OFF DB ? | RESERVED (PORT OFFSET)

```

```

443 C PAGE
444 C -----
445 C TIME-OUT VARIABLES
446 C -----
447
448 007B ?? C *PRINT_TIM_OUT DB ? ; TIME OUT COUNTERS FOR PRINTER RESPONSE
449 0079 ?? C DB ? ; SECOND LOGICAL PRINTER ADAPTER
450 007A ?? C DB ? ; THIRD LOGICAL PRINTER ADAPTER
451 007B ?? C DB ? ; RESERVED
452 007C ?? C *RS232_TIM_OUT DB ? ; TIME OUT COUNTERS FOR RS232 RESPONSE
453 007D ?? C DB ? ; SECOND LOGICAL RS232 ADAPTER
454 007E ?? C DB ? ; RESERVED
455 007F ?? C DB ? ; RESERVED
456
457 C -----
458 C ADDITIONAL KEYBOARD DATA AREA
459 C -----
460
461
462 0080 ???? C *BUFFER_START DW ? ; BUFFER LOCATION WITHIN SEGMENT 40H
463 0082 ???? C *BUFFER_END DW ? ; OFFSET OF KEYBOARD BUFFER START
464 ; OFFSET OF END OF BUFFER
465
466 C -----
467 C EGA/PGA DISPLAY WORK AREA
468 C -----
469
470 0084 ?? C *ROWS DB ? ; ROWS ON THE ACTIVE SCREEN (LESS 1)
471 0085 ???? C *POINTS DW ? ; BYTES PER CHARACTER
472 0087 ?? C *INFO DB ? ; MODE OPTIONS
473 0088 ?? C *INFO_3 DB ? ; FEATURE BIT SWITCHES
474 0089 ?? C DB ? ; RESERVED FOR DISPLAY ADAPTERS
475 ; RESERVED FOR DISPLAY ADAPTERS
476
477 C -----
478 C ADDITIONAL MEDIA DATA
479 C -----
480
481 008B ?? C *LASTRATE DB ? ; LAST DISKETTE DATA RATE SELECTED
482 008C ?? C *HF_STATUS DB ? ; STATUS REGISTER
483 008D ?? C *HF_ERROR DB ? ; ERROR REGISTER
484 008E ?? C *HF_INT_FLAG DB ? ; FIXED DISK INTERRUPT FLAG
485 008F ?? C *HF_CNTRL DB ? ; BIT 0 -> PC-1/DUAL FDC ADAPTER CARD
486 0090 ?? C *DSK_STATE DB ? ; DRIVE 0 MEDIA STATE
487 0091 ?? C DB ? ; DRIVE 1 MEDIA STATE
488 0092 ?? C DB ? ; DRIVE 0 OPERATION START STATE
489 0093 ?? C DB ? ; DRIVE 1 OPERATION START STATE
490 0094 ?? C *DSK_TRK DB ? ; DRIVE 0 PRESENT CYLINDER
491 ; DRIVE 1 PRESENT CYLINDER
492
493 C -----
494 C ADDITIONAL KEYBOARD FLAGS
495 C -----
496
497 0096 ?? C *KB_FLAG_3 DB ? ; KEYBOARD MODE STATE AND TYPE FLAGS
498 0097 ?? C *KB_FLAG_2 DB ? ; KEYBOARD LED FLAGS
499
500 C IFE SYSTEM
501 C -----
502 C REAL TIME CLOCK DATA AREA
503 C -----
504
505 0098 ???? C *USER_FLAG DW ? ; OFFSET ADDRESS OF USERS WAIT FLAG
506 0099 ???? C *USER_FLAG_SEG DW ? ; SEGMENT ADDRESS OF USER WAIT FLAG
507 009C ???? C *RTC_LOW DW ? ; LOW WORD OF USER WAIT FLAG
508 009E ???? C *RTC_HIGH DW ? ; HIGH WORD OF USER WAIT FLAG
509 00A0 ?? C *RTC_WAIT_FLAG DB ? ; WAIT ACTIVE FLAG (01=BUSY, 80=POSTED)
510 ; (00=POST ACKNOWLEDGED)
511
512 C -----
513 C AREA FOR NETWORK ADAPTER
514 C -----
515
516 00A1 07 [ ?? ] C *NET DB ? DUP(?) ; RESERVED FOR NETWORK ADAPTERS
517
518 C -----
519 C EGA/PGA PALETTE POINTER
520 C -----
521
522 00AB ???????? C *SAVE_PTR DD ? ; POINTER TO EGA PARAMETER CONTROL BLOCK
523
524 C -----
525 C TIMER DATA
526 C -----
527
528
529 00CE C *DAY_COUNT ORG 0CEH ; COUNT OF DAYS FROM 1-1-80
530 00CE ???? C DW ? ;
531 ; RESERVED
532
533 C -----
534 C DATA AREA - PRINT SCREEN
535 C -----
536
537
538 0100 C ORG 100H ; ADDRESS= 0040:0100 (REF 0050:0000)
539 0100 ?? C *STATUS_BYTE DB ? ; PRINT SCREEN STATUS BYTE
540 ; 00=READY/OK, 01=BUSY, FF=ERROR
541
542 0101 C DATA ENDS ; END OF BIOS DATA SEGMENT
543
544 .LIST

```

```
545                                     PAGE
546 0000                                CODE   SEGMENT WORD PUBLIC
547
548                                PUBLIC  HEADER
549
550                                ASSUME  CS:CODE,DS:NOTHING,ES:NOTHING,SS:NOTHING
551
552 0000                                HEADER PROC   NEAR
553
554 = 0000                                BEGIN  EQU    $
555 0000 36 32 58 30 38 35              DB      '62X0854 COPR. IBM CORP. 1981,1986 '  ;COPYRIGHT NOTICE
556                                34 20 43 4F 50 52
557                                2E 20 49 42 4D 20
558                                43 4F 52 50 2E 20
559                                31 39 38 31 2C 31
560                                39 38 36 20
561
562 0022 20 20 20 20 20 20              EVEN   '
563                                DB          '
564                                20 20 20 20 20 20
565                                20 20 20 20 20 20
566 0039 20 20 20 20 20 20              DB      '
567                                20 20 20 20 20 20
568                                20 20 20 20 20 20
569                                20 20 20 20 20 20
570
571 0050                                HEADER ENDP
572 0050                                CODE   ENDS
573                                END
```

```

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

```

```

115 :-----
116 : (AH)=16 DISK CHANGE LINE STATUS
117 : OUTPUT REGISTERS
118 : (AH) - 00 - DISK CHANGE LINE NOT ACTIVE
119 :       06 - DISK CHANGE LINE ACTIVE & CARRY BIT ON
120 : (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
121 :-----
122 :
123 : (AH)=17 SET DASD TYPE FOR FORMAT
124 : INPUT REGISTERS
125 : (AL) - 00 - NOT USED
126 :       01 - DISKETTE 320/360K IN 360K DRIVE
127 :       02 - DISKETTE 360K IN 1.2M DRIVE
128 :       03 - DISKETTE 1.2M IN 1.2M DRIVE
129 :       04 - DISKETTE 120K IN 720K DRIVE
130 : (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED;
131 :       DO NOT USE WHEN DISKETTE ATTACH CARD USED)
132 :-----
133 :
134 : (AH)=18 SET MEDIA TYPE FOR FORMAT
135 : INPUT REGISTERS
136 : (CH) - LOW ORDER 8 OF 10 BITS MAXIMUM NUMBER OF TRACKS
137 : (CL) - BITS 7 & 6 - HIGH ORDER TWO BITS OF MAXIMUM TRACKS
138 : (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
139 : OUTPUT REGISTERS
140 : (ESI) - POINTER TO DRIVE PARAMETERS TABLE FOR THIS MEDIA TYPE,
141 :       UNCHANGED IF (AH) IS NON-ZERO
142 : (AH) - 00H, CY = 0, TRACK AND SECTORS/TRACK COMBINATION IS SUPPORTED
143 :       01H, CY = 1, FUNCTION IS NOT AVAILABLE
144 :       0CH, CY = 1, TRACK AND SECTORS/TRACK COMBINATION IS NOT SUPPORTED
145 :-----
146 : DISK CHANGE STATUS IS ONLY CHECKED WHEN A MEDIA SPECIFIED IS OTHER
147 : THAN 360 KB DRIVE. IF THE DISK CHANGE LINE IS FOUND TO BE
148 : ACTIVE THE FOLLOWING ACTIONS TAKE PLACE:
149 :   ATTEMPT TO RESET DISK CHANGE LINE TO INACTIVE STATE.
150 :   IF ATTEMPT SUCCEEDS SET DASD TYPE FOR FORMAT AND RETURN DISK
151 :   CHANGE ERROR CODE
152 :   IF ATTEMPT FAILS RETURN TIMEOUT ERROR CODE AND SET DASD TYPE
153 :   TO A PREDETERMINED STATE INDICATING MEDIA TYPE UNKNOWN.
154 :   IF THE DISK CHANGE LINE IN INACTIVE PERFORM SET DASD TYPE FOR FORMAT.
155 :-----
156 : DATA VARIABLE -- #DISK POINTER
157 : DOUBLE WORD POINTER TO THE CURRENT SET OF DISKETTE PARAMETERS
158 :-----
159 : OUTPUT FOR ALL FUNCTIONS
160 : AH = STATUS OF OPERATION
161 : STATUS BITS ARE DEFINED IN THE EQUATES FOR #DISKETTE_STATUS
162 : VARIABLE IN THE DATA SEGMENT OF THIS MODULE
163 : CY = 0 SUCCESSFUL OPERATION (AH=0 ON RETURN, EXCEPT FOR READ DASD
164 : TYPE AH=(15))
165 : CY = 1 FAILED OPERATION (AH HAS ERROR REASON)
166 : FOR READ/WRITE/VERIFY
167 : DS,BX,DX,CX PRESERVED
168 : NOTE: IF AN ERROR IS REPORTED BY THE DISKETTE CODE, THE APPROPRIATE
169 : ACTION IS TO RESET THE DISKETTE, THEN RETRY THE OPERATION.
170 : ON READ ACCESSES, NO MOTOR START DELAY IS TAKEN, SO THAT
171 : THREE RETRIES ARE REQUIRED ON READS TO ENSURE THAT THE
172 : PROBLEM IS NOT DUE TO MOTOR START-UP.
173 :-----
174 :
175 :.LIST DISKETTE STATE MACHINE - ABSOLUTE ADDRESS 40190 (DRIVE A) & 91 (DRIVE B)
176 :.LIST
177 :
178 :   7   6   5   4   3   2   1   0
179 :-----
180 :
181 :   |   |   |   |   |   |   |   |
182 :   |   |   |   |   |   |   |   |
183 :   |   |   |   |   |   |   |   |
184 :   |   |   |   |   |   |   |   |
185 :   |   |   |   |   |   |   |   |
186 :   |   |   |   |   |   |   |   |
187 :   |   |   |   |   |   |   |   |
188 :   |   |   |   |   |   |   |   |
189 :   |   |   |   |   |   |   |   |
190 :   |   |   |   |   |   |   |   |
191 :   |   |   |   |   |   |   |   |
192 :   |   |   |   |   |   |   |   |
193 :   |   |   |   |   |   |   |   |
194 :   |   |   |   |   |   |   |   |
195 :   |   |   |   |   |   |   |   |
196 :   |   |   |   |   |   |   |   |
197 :   |   |   |   |   |   |   |   |
198 :   |   |   |   |   |   |   |   |
199 :   |   |   |   |   |   |   |   |
200 :   |   |   |   |   |   |   |   |
201 :   |   |   |   |   |   |   |   |
202 :   |   |   |   |   |   |   |   |
203 :   |   |   |   |   |   |   |   |
204 :   |   |   |   |   |   |   |   |
205 :   |   |   |   |   |   |   |   |
206 :   |   |   |   |   |   |   |   |
207 :   |   |   |   |   |   |   |   |
208 :   |   |   |   |   |   |   |   |
209 :   |   |   |   |   |   |   |   |
210 :   |   |   |   |   |   |   |   |
211 :   |   |   |   |   |   |   |   |
212 :   |   |   |   |   |   |   |   |
213 :   |   |   |   |   |   |   |   |
214 :   |   |   |   |   |   |   |   |

```

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 40192 (DRIVE A) & 93 (DRIVE B)

.PRESENT CYLINDER NUMBER - ABSOLUTE ADDRESS 40194 (DRIVE A) & 95 (DRIVE B)


```

215 PAGE
216
217 MD_STRUC STRUC
218 0000 ?? MD_SPEC1 DB ? ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
219 0001 ?? MD_SPEC2 DB ? ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
220 0002 ?? MD_OFF_TIM DB ? ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
221 0003 ?? MD_BYT_SEC DB ? ; 512 BYTES/SECTOR
222 0004 ?? MD_SEC_TRK DB ? ; EOT (LAST SECTOR ON TRACK)
223 0005 ?? MD_GAP DB ? ; GAP LENGTH
224 0006 ?? MD_DTL DB ? ; DTL
225 0007 ?? MD_GAP3 DB ? ; GAP LENGTH FOR FORMAT
226 0008 ?? MD_FIL_BYT DB ? ; FILL BYTE FOR FORMAT
227 0009 ?? MD_HD_TIM DB ? ; HEAD SETTLE TIME (MILLISECONDS)
228 000A ?? MD_STR_TIM DB ? ; MOTOR START TIME (1/8 SECONDS)
229 000B ?? MD_MAX_TRK DB ? ; MAX. TRACK NUMBER
230 000C ?? MD_RATE DB ? ; DATA TRANSFER RATE
231 000D MD_STRUC ENDS
232
233 = 007F EQU 7FH
234 = 0080 EQU 80H
235
236 PUBLIC DISK_INT_1
237 PUBLIC DISKETTE_SETUP
238 PUBLIC DISKETTE_IO_1
239 PUBLIC NEC_OUTPUT
240 PUBLIC RESULTS
241 PUBLIC SEEK
242
243 EXTRN DDS:NEAR
244 EXTRN DISK_BASE:NEAR
245 EXTRN WAITF:NEAR
246 EXTRN MD_TBL1:NEAR
247 EXTRN MD_TBL2:NEAR
248 EXTRN MD_TBL3:NEAR
249 EXTRN MD_TBL4:NEAR
250 EXTRN MD_TBL5:NEAR
251 EXTRN MD_TBL6:NEAR
252
253 0000 CODE SEGMENT BYTE PUBLIC
254
255 ASSUME CS:CODE,DS:DATA,ES:DATA
256
257 ;-----
258 ; DRIVE TYPE TABLE
259 ;-----
260 0000 OR_TYPE LABEL BYTE
261 0000 01 DB 01 ; DRIVE TYPE, MEDIA TABLE
262 0001 02 DW OFFSET MD_TBL1
263 0002 82 DB 02+BITTON
264 0003 0000 E DW OFFSET MD_TBL2
265 0004 02 DB 02
266 0005 0000 E DW OFFSET MD_TBL3
267 0006 03 DB 03
268 0007 0000 E DW OFFSET MD_TBL4
269 0008 84 DB 04+BITTON
270 0009 0000 E DW OFFSET MD_TBL5
271 000A 04 DB 04
272 000B 0000 E DW OFFSET MD_TBL6
273 = 0012 EQU ; END OF TABLE
274 = 0006 (DR_TYPE_E-DR_TYPE)/3 ; NUMBER OF DRIVE TYPES
275
276 0012 DISKETTE_IO_1 PROC FAR ;>>> ENTRY POINT FOR ORG 0EC59H
277 0013 STI ; INTERRUPTS BACK ON
278 0014 55 PUSH BP ; USER REGISTER
279 0015 57 PUSH DI ; USER REGISTER
280 0016 52 PUSH DX ; HEAD #, DRIVE # OR USER REGISTER
281 0017 53 PUSH BX ; BUFFER OFFSET PARAMETER OR REGISTER
282 0018 51 PUSH CX ; TRACK #-SECTOR # OR USER REGISTER
283 0019 8B EC MOV BP,SP ; BP => PARAMETER LIST DEP. ON AH
284 ; [BP] = SECTOR #
285 ; [BP+1] = TRACK #
286 ; [BP+2] = BUFFER OFFSET
287 ; FOR RETURN OF DRIVE PARAMETERS:
288 ; CL/[BP] = BITS 7-6 HI BITS OF MAX CYL
289 ; CH/[BP+1] = BITS 0-5 MAX SECTORS/TRACK
290 ; BL/[BP+2] = BITS 7-4 = 0
291 ; BH/[BP+3] = 0
292 ; DL/[BP+4] = # DRIVES INSTALLED
293 ; DH/[BP+5] = MAX HEAD #
294 ; DI/[BP+6] = MAX HEAD #
295 ; D1/[BP+6] = BUFFER SEGMENT PARM OR USER REGISTER
296 ; USER REGISTERS
297 001A 1E CALL DS ; SEGMENT OF BIOS DATA AREA TO DS
298 001B 56 PUSH SI ; CHECK FOR > LARGEST FUNCTION
299 001C E8 0000 E CALL AH,(FNC_TAE-FNC_TAB)/2 ; FUNCTION OK
300 001F 80 FC 19 CMP AH,OK_FUNC ; REPLACE WITH KNOWN INVALID FUNCTION
301 0022 72 02 JBE OK_FUNC
302 0024 B4 14 MOV AH,14H
303 0026 OK_FUNC:
304 0026 80 FC 01 CMP AH,1 ; RESET OR STATUS ?
305 0029 76 0C JBE OK_DRV ; IF RESET OR STATUS DRIVE ALWAYS OK
306 002B 80 FC 08 CMP AH,8 ; READ DRIVE PARMS ?
307 002E 74 07 JZ OK_DRV ; IF SO DRIVE CHECKED LATER
308 0030 80 FA 03 CMP DL,3 ; DRIVES 0,1,2 AND 3 OK
309 0033 76 02 JBE OK_DRV ; IF 0 OR 1 THEN JUMP
310 0035 B4 14 MOV AH,14H ; REPLACE WITH KNOWN INVALID FUNCTION
311 0037 OK_DRV:
312 0037 8A CC MOV CL,AH ; CL = FUNCTION
313 0039 32 ED XOR CH,CH ; CX = FUNCTION
314 003B D0 E1 SHL CL,1 ; FUNCTION TIMES 2
315 003D BB 0040 R MOV BX,OFFSET FNC_TAB
316 0040 03 D9 ADD BX,CX ; ADD OFFSET INTO TABLE => ROUTINE
317 0042 8A E6 MOV AH,DH ; AX = HEAD #, # OF SECTORS OR DASD TYPE
318 0044 32 F6 XOR DH,DH ; DX = DRIVE #
319 0046 B8 F0 MOV SI,AX ; SI = HEAD #, # OF SECTORS OR DASD TYPE
320 0048 B8 FA MOV DI,DX ; DI = DRIVE #
321 004A 8A 26 0041 R MOV AH,WDISKETTE_STATUS ; LOAD STATUS TO AH FOR STATUS FUNCTION
322 004E C6 06 0041 R 00 MOV WDISKETTE_STATUS,0 ; INITIALIZE FOR ALL OTHERS
323
324 ;
325 ; THROUGHOUT THE DISKETTE BIOS, THE FOLLOWING INFORMATION IS CONTAINED IN
326 ; THE FOLLOWING MEMORY LOCATIONS AND REGISTERS. NOT ALL DISKETTE BIOS
327 ; FUNCTIONS REQUIRE ALL OF THESE PARAMETERS.
328 ;
329 ; DI ; DRIVE #

```

```

329      :                               SI-HI : HEAD #
330      :                               SI-LOW : # OF SECTORS OR DASD TYPE FOR FORMAT
331      :                               ES      : BUFFER SEGMENT
332      :                               [BP]   : SECTOR #
333      :                               [BP+1] : TRACK #
334      :                               [BP+2] : BUFFER OFFSET
335      :
336      :
337      : ACROSS CALLS TO SUBROUTINES THE CARRY FLAG (CY=1), WHERE INDICATED IN
338      : SUBROUTINE PROLOGUES, REPRESENTS AN EXCEPTION RETURN (NORMALLY AN ERROR
339      : CONDITION). IN MOST CASES, WHEN CY = 1, *DSKETTE_STATUS CONTAINS THE
340      : SPECIFIC ERROR CODE.
341      :
342      :
343      :
344      :
345      :
346      :
347      :
348      :
349      :
350      :
351      :
352      :
353      :
354      :
355      :
356      :
357      :
358      :
359      :
360      :
361      :
362      :
363      :
364      :
365      :
366      :
367      :
368      :
369      :
370      :
371      :
372      :
373      :
374      :
375      :
376      :
377      :
378      :
379      :
380      :
381      :
382      :
383      :
384      :
385      :
386      :
387      :
388      :
389      :
390      :
391      :
392      :
393      :
394      :
395      :
396      :
397      :
398      :
399      :
400      :
401      :
402      :
403      :
404      :
405      :
406      :
407      :
408      :
409      :
410      :
411      :
412      :
413      :
414      :
415      :
416      :
417      :
418      :
419      :
420      :
421      :
422      :
423      :
424      :
425      :
426      :
427      :
428      :
429      :
430      :
431      :
432      :
433      :
434      :
435      :
436      :
437      :
438      :
439      :
440      :
441      :
442      :
443      :
444      :
445      :
446      :
447      :
448      :
449      :
450      :
451      :
452      :
453      :
454      :
455      :
456      :
457      :
458      :
459      :
460      :
461      :
462      :
463      :
464      :
465      :
466      :
467      :
468      :
469      :
470      :
471      :
472      :
473      :
474      :
475      :
476      :
477      :
478      :
479      :
480      :
481      :
482      :
483      :
484      :
485      :
486      :
487      :
488      :
489      :
490      :
491      :
492      :
493      :
494      :
495      :
496      :
497      :
498      :
499      :
500      :
501      :
502      :
503      :
504      :
505      :
506      :
507      :
508      :
509      :
510      :
511      :
512      :
513      :
514      :
515      :
516      :
517      :
518      :
519      :
520      :
521      :
522      :
523      :
524      :
525      :
526      :
527      :
528      :
529      :
530      :
531      :
532      :
533      :
534      :
535      :
536      :
537      :
538      :
539      :
540      :
541      :
542      :
543      :
544      :
545      :
546      :
547      :
548      :
549      :
550      :
551      :
552      :
553      :
554      :
555      :
556      :
557      :
558      :
559      :
560      :
561      :
562      :
563      :
564      :
565      :
566      :
567      :
568      :
569      :
570      :
571      :
572      :
573      :
574      :
575      :
576      :
577      :
578      :
579      :
580      :
581      :
582      :
583      :
584      :
585      :
586      :
587      :
588      :
589      :
590      :
591      :
592      :
593      :
594      :
595      :
596      :
597      :
598      :
599      :
600      :
601      :
602      :
603      :
604      :
605      :
606      :
607      :
608      :
609      :
610      :
611      :
612      :
613      :
614      :
615      :
616      :
617      :
618      :
619      :
620      :
621      :
622      :
623      :
624      :
625      :
626      :
627      :
628      :
629      :
630      :
631      :
632      :
633      :
634      :
635      :
636      :
637      :
638      :
639      :
640      :
641      :
642      :
643      :
644      :
645      :
646      :
647      :
648      :
649      :
650      :
651      :
652      :
653      :
654      :
655      :
656      :
657      :
658      :
659      :
660      :
661      :
662      :
663      :
664      :
665      :
666      :
667      :
668      :
669      :
670      :
671      :
672      :
673      :
674      :
675      :
676      :
677      :
678      :
679      :
680      :
681      :
682      :
683      :
684      :
685      :
686      :
687      :
688      :
689      :
690      :
691      :
692      :
693      :
694      :
695      :
696      :
697      :
698      :
699      :
700      :
701      :
702      :
703      :
704      :
705      :
706      :
707      :
708      :
709      :
710      :
711      :
712      :
713      :
714      :
715      :
716      :
717      :
718      :
719      :
720      :
721      :
722      :
723      :
724      :
725      :
726      :
727      :
728      :
729      :
730      :
731      :
732      :
733      :
734      :
735      :
736      :
737      :
738      :
739      :
740      :
741      :
742      :
743      :
744      :
745      :
746      :
747      :
748      :
749      :
750      :
751      :
752      :
753      :
754      :
755      :
756      :
757      :
758      :
759      :
760      :
761      :
762      :
763      :
764      :
765      :
766      :
767      :
768      :
769      :
770      :
771      :
772      :
773      :
774      :
775      :
776      :
777      :
778      :
779      :
780      :
781      :
782      :
783      :
784      :
785      :
786      :
787      :
788      :
789      :
790      :
791      :
792      :
793      :
794      :
795      :
796      :
797      :
798      :
799      :
800      :
801      :
802      :
803      :
804      :
805      :
806      :
807      :
808      :
809      :
810      :
811      :
812      :
813      :
814      :
815      :
816      :
817      :
818      :
819      :
820      :
821      :
822      :
823      :
824      :
825      :
826      :
827      :
828      :
829      :
830      :
831      :
832      :
833      :
834      :
835      :
836      :
837      :
838      :
839      :
840      :
841      :
842      :
843      :
844      :
845      :
846      :
847      :
848      :
849      :
850      :
851      :
852      :
853      :
854      :
855      :
856      :
857      :
858      :
859      :
860      :
861      :
862      :
863      :
864      :
865      :
866      :
867      :
868      :
869      :
870      :
871      :
872      :
873      :
874      :
875      :
876      :
877      :
878      :
879      :
880      :
881      :
882      :
883      :
884      :
885      :
886      :
887      :
888      :
889      :
890      :
891      :
892      :
893      :
894      :
895      :
896      :
897      :
898      :
899      :
900      :
901      :
902      :
903      :
904      :
905      :
906      :
907      :
908      :
909      :
910      :
911      :
912      :
913      :
914      :
915      :
916      :
917      :
918      :
919      :
920      :
921      :
922      :
923      :
924      :
925      :
926      :
927      :
928      :
929      :
930      :
931      :
932      :
933      :
934      :
935      :
936      :
937      :
938      :
939      :
940      :
941      :
942      :
943      :
944      :
945      :
946      :
947      :
948      :
949      :
950      :
951      :
952      :
953      :
954      :
955      :
956      :
957      :
958      :
959      :
960      :
961      :
962      :
963      :
964      :
965      :
966      :
967      :
968      :
969      :
970      :
971      :
972      :
973      :
974      :
975      :
976      :
977      :
978      :
979      :
980      :
981      :
982      :
983      :
984      :
985      :
986      :
987      :
988      :
989      :
990      :
991      :
992      :
993      :
994      :
995      :
996      :
997      :
998      :
999      :
1000      :

```

```

443 ; ON EXIT:          *DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION ;
444 ;-----
445 DISK_STATUS      PROC NEAR
446 00EA            *DSKETTE_STATUS,AH          ; PUT BACK FOR SETUP_END
447 00EA 88 26 0041 R MOV          *DSKETTE_STATUS,AH          ; PUT BACK FOR SETUP_END
448 00EE E8 0832 R   CALL          SETUP_END              ; VARIOUS CLEANUPS
449 00F1 8B DE       MOV          BX,SI                ; GET SAVED AL TO BL
450 00F3 8A C4       MOV          AL,AH                ; STORE STATUS IN AL
451 00F5 C3         RET
452 00F6            ENDP
453 ;-----
454 DISK_READ         PROC NEAR
455 ; DISKETTE READ.
456 ; ON ENTRY:      DI = DRIVE #
457 ;               SI-HI = HEAD #
458 ;               SI-LOW = # OF SECTORS
459 ;               ES = BUFFER SEGMENT
460 ;               [BP] = SECTOR #
461 ;               [BP+1] = TRACK #
462 ;               [BP+2] = BUFFER OFFSET
463 ; ON EXIT:       *DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION ;
464 ;-----
465 DISK_READ         PROC NEAR
466 00F6            AND          *MOTOR_STATUS,01111111B ; INDICATE A READ OPERATION
467 00F6 B8 E646     MOV          AX,0E646H             ; AX = NEC COMMAND, DMA COMMAND
468 00FE E8 04B3 R   CALL          RD_WR_VF              ; COMMON READ/WRITE/VERIFY
469 0101 C3         RET
470 0102            ENDP
471 ;-----
472 DISK_WRITE        PROC NEAR
473 ; DISKETTE WRITE.
474 ; ON ENTRY:      DI = DRIVE #
475 ;               SI-HI = HEAD #
476 ;               SI-LOW = # OF SECTORS
477 ;               ES = BUFFER SEGMENT
478 ;               [BP] = SECTOR #
479 ;               [BP+1] = TRACK #
480 ;               [BP+2] = BUFFER OFFSET
481 ; ON EXIT:       *DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION ;
482 ;-----
483 DISK_WRITE        PROC NEAR
484 0102            MOV          AX,0C54AH              ; AX = NEC COMMAND, DMA COMMAND
485 0102 B8 C54A     MOV          *MOTOR_STATUS,10000000B ; INDICATE WRITE OPERATION
486 0105 80 DE 003F R 00 CALL          RD_WR_VF              ; COMMON READ/WRITE/VERIFY
487 010A E8 04B3 R   CALL          RD_WR_VF
488 010D C3         RET
489 010E            ENDP
490 ;-----
491 DISK_VERIFY        PROC NEAR
492 ; DISKETTE VERIFY.
493 ; ON ENTRY:      DI = DRIVE #
494 ;               SI-HI = HEAD #
495 ;               SI-LOW = # OF SECTORS
496 ;               ES = BUFFER SEGMENT
497 ;               [BP] = SECTOR #
498 ;               [BP+1] = TRACK #
499 ;               [BP+2] = BUFFER OFFSET
500 ; ON EXIT:       *DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION ;
501 ;-----
502 DISK_VERIFY        PROC NEAR
503 010E            AND          *MOTOR_STATUS,01111111B ; INDICATE A READ OPERATION
504 010E 80 DE 003F R 7F CALL          RD_WR_VF              ; COMMON READ/WRITE/VERIFY
505 0113 B8 E642     MOV          AX,0E642H
506 0116 E8 04B3 R   CALL          RD_WR_VF
507 0119 C3         RET
508 011A            ENDP
509 ;-----
510 DISK_FORMAT        PROC NEAR
511 ; DISKETTE FORMAT.
512 ; ON ENTRY:      DI = DRIVE #
513 ;               SI-HI = HEAD #
514 ;               SI-LOW = # OF SECTORS
515 ;               ES = BUFFER SEGMENT
516 ;               [BP] = SECTOR #
517 ;               [BP+1] = TRACK #
518 ;               [BP+2] = BUFFER OFFSET
519 ; *DISK_POINTER POINTS TO THE PARAMETER TABLE OF THIS DRIVE
520 ; ON EXIT:       *DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION ;
521 ;-----
522 DISK_FORMAT        PROC NEAR
523 011A            CALL          XLAT_NEW              ; TRANSLATE STATE TO PRESENT ARCH.
524 011A E8 0404 R   CALL          FMT_INIT              ; ESTABLISH STATE IF UNESTABLISHED
525 011D E8 05A0 R   OR          *MOTOR_STATUS,10000000B ; INDICATE WRITE OPERATION
526 011E 8B E642     MOV          AX,0E642H
527 0120 80 DE 003F R 80 TEST          *HWF_CNTRL,DUAL ; TEST CONTROLLER I.D.
528 0125 F6 06 00B8 R 01 JZ          NO_CHG_CHECK ; CHECK MEDIA CHANGE AND RESET IF SO
529 012A 74 05      JC          MED_CHANGE ; MEDIA CHANGED, SKIP
530 012C E8 05F5 R   JC          FM_DONE
531 012F 72 41      JC          NO_CHG_CHECK
532 0131            NO_CHG_CHECK:
533 0131            CALL          CHK_LASTRATE ; ZF=1 ATTEMPT RATE IS SAME AS LAST RATE
534 0131 E8 0658 R   JC          FM_WR ; YES, SKIP SPECIFY COMMAND
535 0134 74 06      JC          SEND_SPEC ; SEND SPECIFY COMMAND TO NEC
536 0136 E8 03D1 R   CALL          SEND_RATE ; SEND DATA RATE TO CONTROLLER
537 0139 E8 0637 R   CALL          FM_DONE
538 013C            FM_WR:
539 013C 80 4A      MOV          AL,04AH ; WILL WRITE TO THE DISKETTE
540 013E E8 0665 R   CALL          DMA_SETUP ; SET UP THE DMA
541 0141 72 2F      JC          FM_DONE ; RETURN WITH ERROR
542 0143 B4 4D      MOV          AH,04DH ; ESTABLISH THE FORMAT COMMAND
543 0145 E8 06CB R   CALL          NEC_INIT ; INITIALIZE THE NEC
544 0148 72 28      JC          FM_DONE
545 014A B8 0172 R   MOV          AX,OFFSET FM_DONE
546 014D 50        PUSH        AX ; LOAD ERROR ADDRESS
547 014E B2 03      MOV          DL,3 ; PUSH NEC OUT ERROR RETURN
548 0150 E8 08FE R   CALL          GET_PARM ; BYTES/SECTOR VALUE TO NEC
549 0153 E8 09F0 R   CALL          NEC_OUTPUT ; SECTORS/TRACK VALUE TO NEC
550 0156 B2 04      MOV          DL,4 ;
551 0158 E8 08FE R   CALL          GET_PARM ;
552 015B E8 09F0 R   CALL          NEC_OUTPUT ;
553 015E B2 07      MOV          DL,7 ; GAP LENGTH VALUE TO NEC
554 0160 E8 08FE R   CALL          GET_PARM ;
555 0163 E8 09F0 R   CALL          NEC_OUTPUT ;
556 0166 B2 08      MOV          DL,8 ; FILLER BYTE TO NEC

```

```

557 0168 E8 08FE R      CALL    GET_PARM
558 0168 E8 09F0 R      CALL    NEC_OUTPUT
559 016E 58             POP     AX
560 016F E8 0727 R      CALL    NEC_TERM          ; THROW AWAY ERROR
561 0172             FM_DONE:                ; TERMINATE, RECEIVE STATUS, ETC.
562 0172 E8 0432 R      CALL    XLAT_OLD          ; TRANSLATE STATE TO COMPATIBLE MODE
563 0175 E8 0832 R      CALL    SETUP_END        ; VARIOUS CLEANUPS
564 0178 8B DE          MOV     BX,SI          ; GET SAVED AL TO BL
565 017A 8A C3          MOV     AL,BL          ; PUT BACK FOR RETURN
566 017C 38             RET
567 017D             DISK_FORMAT      ENDP

568 -----
569 ; FNC_ERR
570 ; INVALID FUNCTION REQUESTED OR INVALID DRIVE;
571 ; SET BAD COMMAND IN STATUS.
572 ;
573 ; ON EXIT:      *DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
574 ;
575 017D             FNC_ERR PROC      NEAR
576 017D 8B C6          MOV     AX,S1          ; INVALID FUNCTION REQUEST
577 017F B4 01          MOV     AH,BAD_CMD      ; RESTORE AL
578 0181 8B 26 0041 R   MOV     *DSKETTE_STATUS,AH      ; SET BAD COMMAND ERROR
579 0185 F9             STC                    ; STORE IN DATA AREA
580 0186 C3             RET                    ; SET CARRY INDICATING ERROR
581 0187             FNC_ERR ENDP

582 -----
583 ; DISK_PARAMS
584 ; READ DRIVE PARAMETERS.
585 ; ON ENTRY:
586 ;   DI = DRIVE #
587 ; ON EXIT:
588 ;   CL/[BP] = BITS 7 & 6 HIGH 2 BITS OF MAX CYLINDER
589 ;             BITS 0-5 MAX SECTORS/TRACK
590 ;   CH/[BP+1] = LOW 8 BITS OF MAX CYLINDER
591 ;   BL/[BP+2] = BITS 7-4 = 0
592 ;             BITS 3-0 = VALID CMOS DRIVE TYPE
593 ;   BH/[BP+3] = 0
594 ;   DL/[BP+4] = # DRIVES INSTALLED
595 ;   DH/[BP+5] = MAX HEAD #
596 ;   DI/[BP+6] = OFFSET OF MEDIA/DRIVE PARAMETER TABLE
597 ;   ES = SEGMENT OF MEDIA/DRIVE PARAMETER TABLE
598 ;   AX = 0
599 ;
600 ; NOTE : THE ABOVE INFORMATION IS STORED IN THE USERS STACK AT
601 ; THE LOCATIONS WHERE THE MAIN ROUTINE WILL POP THEM
602 ; INTO THE APPROPRIATE REGISTERS BEFORE RETURNING TO THE
603 ; CALLER
604 ;
605 0187             DISK_PARAMS PROC  NEAR
606 0187 81 FF 0080     CMP     DI,B0H          ; CHECK FOR FIXED MEDIA TYPE REQUEST
607 018B 72 06         JB      DISK_P2         ; CONTINUE IF NOT REQUEST FALL THROUGH
608
609 ;----- FIXED DISK REQUEST FALL THROUGH ERROR
610
611 018D 8B C6          MOV     AX,S1          ; RESTORE AL WITH CALLERS VALUE
612 018F B4 01          MOV     AH,BAD_CMD      ; SET BAD COMMAND ERROR IN (AH)
613 0191 F9             STC                    ; SET ERROR RETURN CODE
614 0192 C3             RET
615
616 0193             DISK_P2:
617 0193 E8 0404 R      CALL    XLAT_NEW          ; TRANSLATE STATE TO PRESENT ARCH.
618 0196 C7 46 02 0000 MOV     WORD_PTR [BP+2],0      ; DRIVE TYPE = 0
619 019B A1 0010 R      MOV     AX,EQUIP_FLAG      ; LOAD EQUIPMENT FLAG FOR # DISKETTES
620 019E 24 C1          AND     AL,11000001B    ; CHECK DISKETTE DRIVE BITS
621 01A0 D0 E8          SHR     AL,1            ; ARE THERE ANY DRIVES INSTALLED?
622 01A2 73 7C          JNC     NON_DRV         ; NC-->NO DRIVES, ZERO PARAMETERS
623 01A4 D0 C0          ROL     AL,T            ; ROTATE TO ORIGINAL POSITION
624 01A6 D0 C0          ROL     AL,I            ; ROTATE BITS 6 AND 7 TO 0 AND 1
625 01A8 D0 C0          ROL     AL,I
626 01AA FE C0          INC     AL
627 01AC 8B 46 04       MOV     [BP+4],AL      ; CONVERT TO RELATIVE 1
628 01AF F6 06 008F R 01 TEST    BF,CNTRL,DUAL      ; STORE NUMBER OF DRIVES
629 01B4 75 03         JNZ     DP1_CONT        ; CHECK CONTROLLER I.D.
630 01B6 E9 0256 R     JMP     DET_PARAMS      ; CONTINUE WITH USUAL PARAMS CHECK
631 01B9             DP1_CONT:                ; RETURN THIS CONTROLLERS PARAMS
632 01B9 83 FF 01      CMP     DI,1
633 01BC 77 66         JA      NON_DRV1        ; CHECK FOR VALID DRIVE
634 01BE C6 46 05 01  MOV     BYTE_PTR [BP+5],1      ; DRIVE INVALID
635 01C2 E8 08CF R     CALL    CMOS_TYPE      ; MAXIMUM HEAD NUMBER = 1
636 01C5 72 16         JC      CHK_EST        ; RETURN DRIVE TYPE IN AL
637 01C7 04 C0         OR      AL,AL          ; ON CMOS BAD CHECK ESTABLISHED
638 01C9 74 12         JZ      CHK_EST        ; TEST FOR NO DRIVE TYPE
639 01CB E8 03B1 R     CALL    DR_TYPE_CHECK   ; JUMP IF 50
640 01CE 72 0D         JC      CHK_EST        ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
641 01D0 8B 46 02       MOV     [BP+2],AL      ; TYPE NOT IN TABLE (POSSIBLE BAD CMOS)
642 01D3 2E: 8A 4F 04  MOV     CL,CS:[BX].MD_SEC_TRK    ; STORE VALID CMOS DRIVE TYPE
643 01D7 2E: 8A 6F 0B  MOV     CH,CS:[BX].MD_MAX_TRK    ; GET SECTOR/TRACK
644 01DB EB 32         JMP     SHORT STO_CX    ; GET MAX. TRACK NUMBER
645                                     ; CMOS GOOD, USE CMOS
646
647 01DD             CHK_EST:
648 01DD 8A A5 0090 R   MOV     AH,POSK STATE[D1]    ; LOAD STATE FOR THIS DRIVE
649 01E1 F6 C4 10      TEST    AH,MD_DET      ; CHECK FOR ESTABLISHED STATE
650 01E4 74 3E         JZ      NON_DRV1        ; CMOS BAD/INVALID AND UNESTABLISHED
651
652 01E6             USE_EST:
653 01E6 80 E4 C0      AND     AH,RATE_MSK      ; ISOLATE STATE
654 01E9 80 FC C0      CMP     AH,RATE_250     ; RATE 250 ?
655 01EC 75 54         JNE     USE_EST2        ; NO, GO CHECK OTHER RATE
656
657 ;--- DATA RATE IS 250 KBS, TRY 360 KB TABLE FIRST
658
659 01EE B0 01         MOV     AL,01          ; DRIVE TYPE 1 (360KB)
660 01F0 E8 03B1 R     CALL    DR_TYPE_CHECK   ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
661 01F3 2E: 8A 4F 04  MOV     CL,CS:[BX].MD_SEC_TRK    ; GET SECTOR/TRACK
662 01F7 2E: 8A 6F 0B  MOV     CH,CS:[BX].MD_MAX_TRK    ; GET MAX. TRACK NUMBER
663 01FB F6 85 0090 R 01 TEST    *STATE[D1],TRK_CAPA    ; *DRIVE STATE?
664 0200 74 0D         JZ      STO_CX          ; MUST BE 360KB DRIVE
665
666 ;--- IT IS HIGH DATA RATE/80 TRACK DRIVE
667
668 0202             PARM_HDR 80T:
669 0202 B0 04         MOV     AL,04          ; DRIVE TYPE 4
670 0204 E8 03B1 R     CALL    DR_TYPE_CHECK   ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
671 0207 2E: 8A 4F 04  MOV     CL,CS:[BX].MD_SEC_TRK    ; GET SECTOR/TRACK

```

```

671 020B 2E: 8A 6F 0B          MOV     CH,CS:[BX].MD_MAX_TRK    ; GET MAX. TRACK NUMBER
672
673 020F                          STO_CX:
674 020F 89 4E 00              MOV     [BP],CX                    ; SAVE IN STACK FOR RETURN
675 0212                          ES_DI:
676 0212 89 5E 06              MOV     [BP+6],BX                ; ADDRESS OF MEDIA/DRIVE PARAM TABLE
677 0215 8C C8                MOV     AX,CS                    ; SEGMENT MEDIA/DRIVE PARAMETER TABLE
678 0217 8E C0                MOV     ES,AX                    ; ES IS SEGMENT OF TABLE
679
680 0219                          DP_OUT:
681 0219 E8 0432 R             CALL    XLAT_OLD                ; TRANSLATE STATE TO COMPATIBLE MODE
682 021C 33 C0                XOR     AX,AX                    ; CLEAR
683 021E F8                    CLC
684 021F C3                    RET
685
686
687
688 0220                          ;----- NO DRIVE PRESENT HANDLER
689 0220 C6 46 04 00          MOV     BYTE PTR [BP+4],0        ; CLEAR NUMBER OF DRIVES
690
691 0224                          NON_DRIVE:
692 0224 81 FF 0080          CMP     DI,80H                ; CHECK FOR FIXED MEDIA TYPE REQUEST
693 0228 72 09              JNB     NON_DRV2              ; CONTINUE IF NOT REQUEST FALL THROUGH
694
695                          ;----- FIXED DISK REQUEST FALL THROUGH ERROR
696
697 022A                          FD_REQ_ERR:
698 022A E8 0432 R             CALL    XLAT_OLD                ; ELSE TRANSLATE TO COMPATIBLE MODE
699 022D 8B C6              MOV     AX,SI                    ; RESTORE AL
700 022F B4 01              MOV     AH,BAD_CMD             ; SET BAD COMMAND ERROR
701 0231 F9                STC                    ; SET ERROR RETURN CODE
702 0232 C3                RET
703
704 0233                          NON_DRV2:
705 0233 33 C0              XOR     AX,AX                    ; CLEAR PARMS IF NO DRIVES OR CMOS BAD
706 0235 89 46 00          MOV     [BP],AX                ; TRACKS, SECTORS/TRACK = 0
707 0238 8B 66 05          MOV     [BP+5],AH              ; HEAD = 0
708 023B 89 46 06          MOV     [BP+6],AX              ; OFFSET TO DISK BASE = 0
709 023E 8E C0              MOV     ES,AX                    ; ES IS SEGMENT OF TABLE
710 0240 EB D7              JMP     SHORT DP_OUT
711
712
713
714 0242                          ;---- DATA RATE IS EITHER 300 KBS OR 500 KBS, TRY 1.2 MB TABLE FIRST
715 0242                          USE_EST2:
716 0242 80 02              CALL    AL,02                  ; DRIVE TYPE 2 (1.2MB)
717 0244 E8 03B1 R             CALL    CL,02                  ; RTH CS:BX = MEDIA/DRIVE PARAM TBL
718 0247 2E: 8A 4F 04          MOV     CL,CS:[BX].MD_SEC_TRK ; GET SECTOR/TRACK
719 024B 2E: 8A 6F 0B          MOV     CH,CS:[BX].MD_MAX_TRK ; GET MAX. TRACK NUMBER
720 024F 80 FC 40             CMP     AH,RATE_300            ; RATE 300 ?
721 0252 74 B8              JE      STO_CX                ; MUST BE 1.2MB DRIVE
722 0254 EB AC              JMP     SHORT PARM_HDR_80T     ; ELSE, HIGH DATA RATE/80 TRACK DRIVE
723 0256
724 0256 83 FF 03          DET_PARMS: CMP     DI,3                    ; REQUEST FOR FIXED DISK?
725 0259 81 09          JA      NON_DRV2              ; YES-->DRIVE NUMBER INVALID
726 025B B1 09          MOV     CL,9
727 025D F6 85 0090 R 01     TEST    #DSK_STATE[DI],TRK_CAPA ; IS DRIVE 80 TRACKS? (RELATIVE ZERO)
728 0262 B0 01          MOV     AL,1                    ; SET CMOS TYPE 1
729 0264 95 27          MOV     CH,39                  ; NUMBER OF TRACKS (RELATIVE ZERO)
730 0266 74 04          JZ      SET_TYP1              ; IF ZERO TYPE = 1
731 0268 B0 03          MOV     AL,3                    ; SET CMOS TYPE 3
732 026A 95 4F          MOV     CH,39                  ; NUMBER OF TRACKS (RELATIVE ZERO)
733 026C
734 026C 88 46 02          SET_TYP1: MOV     [BP+2],AL                ; STORE TYPE
735 026E C6 46 03 00          MOV     BYTE PTR [BP+3],0      ; MAXIMUM HEAD NUMBER = 1
736 0273 C6 46 05 01          MOV     BYTE PTR [BP+5],1      ; ADDRESS OF DISK BASE
737 0277 E8 03B1 R             CALL    DR_TYPE_CHECK          ; GO SET TRKS/SEC,CYL,ES:BX AND EXIT
738 027A EB 93              JMP     STO_CX
739 027C
740
741
742
743
744
745
746
747
748 027C                          DISK_PARMS ENDP
749 027C F6 06 008F R 01     ;-----
750 0281 74 22              ; DISK_TYPE THIS ROUTINE RETURNS THE TYPE OF MEDIA INSTALLED.
751 0283 E8 0404 R             ; ON ENTRY: DI = DRIVE #
752 0286 8A 85 0090 R         ; ON EXIT: AH = DRIVE TYPE, CY=0
753 028A 0A C0              ;-----
754 028C 74 13              DISK_TYPE PROC NEAR
755 028E B4 01              TEST    #HF_CNTRL,DUAL        ; CHECK CONTROLLER I.D.
756 0290 A8 01              JZ      NO_SHNG               ; NO SHNG
757 0292 74 02              CALL    XLAT_NEW              ; TRANSLATE STATE TO PRESENT ARCH.
758 0294 B4 02              MOV     AL,#DSK_STATE[DI]     ; GET PRESENT STATE INFORMATION
759 0296 8A 01              OR      AL,AL                 ; CHECK FOR NO DRIVE
760 0298 8A 01              MOV     AL,AL                 ; NO DRV
761 029A 58                MOV     AH,NOCHGLN            ; NO CHANGE LINE FOR 40 TRACK DRIVE
762 029C 8B DE              TEST    AL,TRK_CAPA          ; IS THIS DRIVE AN 80 TRACK DRIVE?
763 029E 8A C3              JZ      DT_BACR               ; IF NO JUMP
764 02A0 C3                MOV     AH,CHGLN             ; CHANGE LINE FOR 80 TRACK DRIVE
765 02A1
766 02A1 32 E4              DT_BACK: PUSH    AX                    ; SAVE RETURN VALUE
767 02A3 EB F1              CALL    XLAT_OLD              ; TRANSLATE STATE TO COMPATIBLE MODE
768 02A5 58                POP     AX                    ; RESTORE RETURN VALUE
769 02A7 58                DISK_TYPE_EX: EXIT DISK TYPE FUNCTION
770 02A9 F8                CLC                          ; NO ERROR
771 02AB 8B DE              MOV     BX,SI                 ; GET SAVED AL TO BL
772 02AD 8A C3              MOV     AL,BL                 ; PUT BACK FOR RETURN
773 02AF C3                RET
774 02B0
775 02B0 32 E4              NO_DRV: XOR     AH,AH                  ; NO DRIVE PRESENT OR UNKNOWN
776 02B2 EB F1              JMP     SHORT DT_BACK
777 02B4
778 02B4 32 E4              NO_CHNG: MOV     AX,#EQUIP_FLAG        ; LOAD EQUIPMENT FLAG FOR # DISKETTES
779 02B6 58                SHR     AL,1                  ; SHIFT DRIVES PRESENT BIT INTO CARRY
780 02B8 58                JNC     NO_DRV                ; NO DRIVE IN SYSTEM
781 02BA 58                MOV     AH,1                  ; DISKETTE NO CHANGE LINE AVAILABLE
782 02BC 58                JMP     DISK_TYPE_EX
783 02BE
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

```

```

785      : ON EXIT:      AH = %DSKETTE_STATUS      :
786      :              00 - DISK CHANGE LINE INACTIVE, CY = 0      :
787      :              06 - DISK CHANGE LINE ACTIVE, CY = 1      :
788      :
789      DISK_CHANGE     PROC NEAR
790      02B0 F6 06 00BF R 01      TEST     %HF_CNTRL,DUAL      ; TEST CONTROLLER I.D.
791      02B5 75 03              JNZ      DC1              ;
792      02B7 E9 017D R          JMP      FNC_ERR          ; ERROR FOR THIS KIND OF CONTROLLER
793
794      DC1:             CALL     XLAT_NEW              ; TRANSLATE STATE TO PRESENT ARCH.
795      02BD 8A 85 0090 R      MOV      AL,%DSK_STATE[DI] ; GET MEDIA STATE INFORMATION
796      02C1 0A C0              OR       AL,AL            ; DRIVE PRESENT ?
797      02C3 74 19              JZ       DC_NON          ; JUMP IF NO DRIVE
798      02C5 A8 01              TEST     AL,TRK_CAPA      ; 80 TRACK DRIVE ?
799      02C7 74 05              JZ       SETIT_1        ; IF 50 , CHECK CHANGE LINE
800
801      02C9 E8 0B21 R          DC0:    CALL     READ_DSKCHNG ; GO CHECK STATE OF DISK CHANGE LINE
802      02CC 74 05              JZ       FINIS         ; CHANGE LINE NOT ACTIVE
803
804      02CE C6 06 0041 R 06    SETIT:  MOV      %DSKETTE_STATUS,MEDIA_CHANGE ; INDICATE MEDIA REMOVED
805
806      02D3 E8 0432 R          FINIS:  CALL     XLAT_OLD              ; TRANSLATE STATE TO COMPATIBLE MODE
807      02D6 E8 0832 R          CALL     SETUP_END          ; VARIOUS CLEANUPS
808      02D9 8B 03              MOV      BX,SI          ; GET SAVED AL TO BL
809      02DB 8A C3              MOV      AL,BL          ; PUT BACK FOR RETURN
810      02DD C3                RET
811
812      02DE                DC_NON:  OR       %DSKETTE_STATUS,TIME_OUT ; SET TIMEOUT, NO DRIVE
813      02DE 80 0E 0041 R 80    JMP      SHORT FINIS
814      02E3 EB EE              DISK_CHANGE
815      02E5                ENDP
816
817      :-----:
818      : FORMAT SET
819      : THIS ROUTINE IS USED TO ESTABLISH THE TYPE OF
820      : MEDIA TO BE USED FOR THE FOLLOWING FORMAT OPERATION.
821      :
822      : ON ENTRY:      SI LOW = DASD TYPE FOR FORMAT
823      :                DI = DRIVE #
824      :
825      : ON EXIT:       %DSKETTE_STATUS REFLECTS STATUS
826      :                AH = %DSKETTE_STATUS
827      :                CY = 1 IF ERROR
828      :
829      FORMAT_SET     PROC NEAR
830      02E5 E8 0404 R          CALL     XLAT_NEW              ; TRANSLATE STATE TO PRESENT ARCH.
831      02E8 56                PUSH     SI                ; SAVE DASD TYPE
832      02E9 8B C6              MOV      AX,SI          ; AH = ? , AL = DASD TYPE
833      02EB 32 E4              XOR      AH,AH          ; AH = 0 , AL = DASD TYPE
834      02ED 8B F0              MOV      SI,AX          ; SI = DASD TYPE
835      02EF 80 A5 0090 R 0F    AND      %DSK_STATE[DI],NOT MED_DET+DBL_STEP+RATE_MSK ; CLEAR STATE
836      02F4 4E                DEC      SI                ; CHECK FOR 320/360K MEDIA & DRIVE
837      02F5 75 07              JNZ      NOT_320        ; BYPASS IF NOT
838      02F7 80 80 0090 R 90    OR       %DSK_STATE[DI],MED_DET+RATE_250 ; SET TO 320/360
839      02FC EB 3E              JMP      SHORT S0
840
841      02FE                NOT_320: TEST     %HF_CNTRL,DUAL      ; TEST CONTROLLER I.D.
842      02FE F6 06 00BF R 01    JZ       S3              ;
843      0303 74 0A              CALL     MED_CHANGE          ; CHECK FOR TIME_OUT
844      0305 E8 05F5 R          CMP      %DSKETTE_STATUS,TIME_OUT ;
845      0308 80 3E 0041 R 80    JZ       S0              ; IF TIME OUT TELL CALLER
846      030D 74 2D              JZ
847
848      030F 4E                S3:     DEC      SI                ; CHECK FOR 320/360K IN 1.2M DRIVE
849      0310 75 07              JNZ      NOT_320_12        ; BYPASS IF NOT
850      0312 80 80 0090 R 70    OR       %DSK_STATE[DI],MED_DET+DBL_STEP+RATE_300 ; SET STATE
851      0317 EB 23              JMP      SHORT S0
852
853      0319                NOT_320_12: DEC      SI                ; CHECK FOR 1.2M MEDIA IN 1.2M DRIVE
854      0319 4E                JZ
855      031A 75 07              JNZ      NOT_12           ; BYPASS IF NOT
856      031C 80 80 0090 R 10    OR       %DSK_STATE[DI],MED_DET+RATE_500 ; SET STATE VARIABLE
857      0321 EB 19              JMP      SHORT S0         ; RETURN TO CALLER
858
859      0323                NOT_12:  DEC      SI                ; CHECK FOR SET DASD TYPE 04
860      0323 4E                JZ       FS_ERR           ; BAD COMMAND EXIT IF NOT VALID TYPE
861      0324 75 20              JNZ
862
863      0326 F6 85 0090 R 04    TEST     %DSK_STATE[DI],DRV_DET ; DRIVE DETERMINED ?
864      032B 74 09              JZ       ASSUME           ; IF STILL NOT DETERMINED ASSUME
865      032D B0 50              MOV      AL,MED_DET+RATE_300 ;
866      032F F6 85 0090 R 02    TEST     %DSK_STATE[DI],FMT_CAPA ; MULTIPLE FORMAT CAPABILITY ?
867      0334 75 02              JNZ      OR_IT_IN         ; IF 1.2 M THEN DATA RATE 300
868
869      0336                ASSUME:  MOV      AL,MED_DET+RATE_250 ; SET UP
870      0336 B0 90              MOV
871
872      0338                OR_IT_IN: OR       %DSK_STATE[DI],AL ; OR IN THE CORRECT STATE
873      0338 08 85 0090 R      S0:     CALL     XLAT_OLD              ; TRANSLATE STATE TO COMPATIBLE MODE
874
875      033C                CALL     SETUP_END          ; VARIOUS CLEANUPS
876      033C E8 0432 R          CALL     SETUP_END          ;
877      033F E8 0832 R          POP      BX              ; GET SAVED AL TO BL
878      0342 5B                MOV      AL,BL          ; PUT BACK FOR RETURN
879      0343 8A C3              MOV
880      0345 C3              RET
881
882      0346                FS_ERR:  MOV      %DSKETTE_STATUS,BAD_CMD ; UNKNOWN STATE,BAD COMMAND
883      0346 C6 06 0041 R 01    JMP      SHORT S0
884      034B EB EF
885
886      034D                FORMAT_SET
887      034D                ENDP
888
889      :-----:
890      : SET_MEDIA
891      : THIS ROUTINE SETS THE TYPE OF MEDIA AND DATA RATE
892      : TO BE USED FOR THE FOLLOWING FORMAT OPERATION.
893      :
894      : ON ENTRY:      [BP] = SECTOR PER TRACK
895      :                [BP+1] = TRACK #
896      :                DI = DRIVE #
897      :
898      : ON EXIT:       %DSKETTE_STATUS REFLECTS STATUS
899      :                IF NO ERROR:

```

```
899      ; AH = 0 ;
900      ; CY = 0 ;
901      ; ES = SEGMENT OF MEDIA/DRIVE PARAMETER TABLE ;
902      ; DI/[BP+6] = OFFSET OF MEDIA/DRIVE PARAMETER TABLE ;
903      ; IF ERROR: ;
904      ; AH = #DSKETTE_STATUS ;
905      ; CY = 1 ;
906      ;-----
907 0340      SET_MEDIA      PROC    NEAR
908 0340 EB 0404 R      CALL    XLAT_NEW      ; TRANSLATE STATE TO PRESENT ARCH.
909 0350 33 DB          XOR     BX,BX      ; ZERO INDEX POINTER
910 0352 80 TE 01 27    CMP     BYTE PTR [BP+1],39 ; MAX. TRACK = 40 ?
911 0356 75 0B          JNE     TBL_CHK1
912 0358 F6 85 0090 R 0I TEST    #DSK_STATE[DI],TRK_CAPA ; 80 TRACK DRIVE ?
913 035D 74 16          JZ      MD_FND     ; POINT TO TABLE ENTRY 1
914 035F B3 03          MOV     BL,3      ; POINT TO TABLE ENTRY 2
915 0361 EB 12          JMP     SHORT MD_FND
916 0363
917 0363 B3 06          MOV     BL,6      ; POINT TO TABLE ENTRY 3
918 0365 80 TE 00 0F    CMP     BYTE PTR [BP],15 ; SECTORS/TRACK = 15 ?
919 0369 74 0A          JE      MD_FND
920 036B B3 12          MOV     BL,18     ; POINT TO TABLE ENTRY 6
921 036D 80 TE 00 12    CMP     BYTE PTR [BP],18 ; SECTORS/TRACK = 18 ?
922 0371 74 02          JE      MD_FND
923 0373 B3 0C          MOV     BL,12     ; POINT TO TABLE ENTRY 4
924 0375
925 0375 2E: 8B 9F 0001 R LE      MD_FND: MOV    BX,CS:WORD PTR DR_TYPE[BX+1] ; DI = MEDIA/DRIVE PARAMETER TAB
926 037A 2E: 8A 47 04    MOV     AL,CS:[BX].MD_SEC_TRK ; GET SECTOR/TRACK
927 037E F6 8A 67 0B    MOV     AH,CS:[BX].MD_MAX_TRK ; GET MAX. TRACK #
928 0382 39 46 00        CMP     [BP],AX ; MATCH ?
929 0385 75 23          JNE     ER_RTN     ; NOT SUPPORTED
930 0387 2E: 8A 47 0C    MOV     AL,CS:[BX].MD_RATE ; GET RATE
931 038B 3C 40          CMP     AL,RATE_300 ; DOUBLE STEP REQUIRED FOR RATE 300
932 038D 75 02          JNE     MD_SET
933 038F 0C 20          OR      AL,DBL_STEP
934 0391
935 0391 69 5E 06        MOV     [BP+6],BX ; SAVE TABLE POINTER IN STACK
936 0394 0C 10          OR      MD_DET ; SET MEDIA ESTABLISHED
937 0396 80 A5 0090 R 0F AND     #DSK_STATE[DI],NOT MD_DET+DBL_STEP+RATE_MSK ; CLEAR STATE
938 039B 08 85 0090 R 0R OR      #DSK_STATE[DI],AL ; SET STATE
939 039F 8C C5          MOV     AX,CS ; SEGMENT MEDIA/DRIVE PARAMETER TABLE
940 03A1 8E C0          MOV     ES,AX ; ES IS SEGMENT OF TABLE
941 03A3
942 03A3 E8 0432 R      SM_RTN: CALL    XLAT_OLD ; TRANSLATE STATE TO COMPATIBLE MODE
943 03A6 EB 0832 R      CALL    SETUP_END ; VARIOUS CLEANUPS
944 03A9 C3
945 03AA
946 03AA C4 06 0041 R 0C ER_RTN: MOV    #DSKETTE_STATUS,MD_NOT_FND ; ERROR, MEDIA TYPE NOT FOUND
947 03AF EB F2        JMP     SM_RTN
948 03B1
949      SET_MEDIA      ENDP
950
951      ;-----
952      ; DR_TYPE_CHECK
953      ; CHECK IF THE GIVEN DRIVE TYPE IN REGISTER (AL)
954      ; IS SUPPORTED IN BIOS DRIVE TYPE TABLE
955      ; ON ENTRY:
956      ; AL = DRIVE TYPE
957      ; ON EXIT:
958      ; CS = SEGMENT OF MEDIA/DRIVE PARAMETER TABLE (CODE)
959      ; CY = 0 DRIVE TYPE SUPPORTED
960      ; BX = OFFSET TO MEDIA/DRIVE PARAMETER TABLE
961      ; CY = 1 DRIVE TYPE NOT SUPPORTED
962      ; REGISTERS ALTERED: BX
963 03B1      DR_TYPE_CHECK PROC    NEAR
964 03B1 50            PUSH    AX
965 03B2 51            PUSH    CX
966 03B3 33 DB          XOR     BX,BX      ; BX = INDEX TO DR_TYPE TABLE
967 03B5 B9 0006        MOV     CX,DR_CNT ; CX = LOOP COUNT -
968 03B8
969 03B8 2E: 8A AT 0000 R MOV     AH,CS:DR_TYPE[BX] ; GET DRIVE TYPE
970 03BD 3A C4          CMP     AL,AH ; DRIVE TYPE MATCH ?
971 03BF 74 08          JE      DR_TYPE_VALID ; YES, RETURN WITH CARRY RESET
972 03C1 63 C3 03      ADD     BX,3 ; CHECK NEXT DRIVE TYPE
973 03C4 E2 F2          LOOP    TYPE_CHK
974 03C6 F9            STC ; DRIVE TYPE NOT FOUND IN TABLE
975 03C7 EB 05          JMP     SHORT TYPE_RTN
976 03C9
977 03C9 2E: 8B 9F 0001 R LE      DR_TYPE_VALID: MOV    BX,CS:WORD PTR DR_TYPE[BX+1] ; BX = MEDIA TABLE
978 03CE
979 03CE 59            POP     CX
980 03CF 58            POP     AX
981 03D0 C3
982 03D1      DR_TYPE_CHECK ENDP
983
984      ;-----
985      ; SEND_SPEC
986      ; SEND THE SPECIFY COMMAND TO CONTROLLER USING DATA FROM
987      ; THE DRIVE PARAMETER TABLE POINTED BY #DISK_POINTER
988      ; ON ENTRY: #DISK_POINTER = DRIVE PARAMETER TABLE
989      ; ON EXIT: NONE
990      ; REGISTERS ALTERED: AX
991      ;-----
992 03D1      SEND_SPEC   PROC    NEAR
993 03D1 B8 03EB R      MOV     AX,OFFSET SPECBAC ; LOAD ERROR ADDRESS
994 03D4 50            PUSH    AX
995 03D5 B4 03          MOV     AH,03H ; SPECIFY COMMAND
996 03D7 EB 09F0 R      CALL    NEC_OUTPUT ; OUTPUT THE COMMAND
997 03DA 2A D2          SUB     DL,DL ; FIRST SPECIFY BYTE
998 03DC E8 08FE R      CALL    GET_PARM ; GET PARAMETER TO AH
999 03DF EB 09F0 R      CALL    NEC_OUTPUT ; OUTPUT THE COMMAND
1000 03E2 B2 01          MOV     DL,1 ; SECOND SPECIFY BYTE
1001 03E4 E8 08FE R      CALL    GET_PARM ; GET PARAMETER TO AH
1002 03E7 EB 09F0 R      CALL    NEC_OUTPUT ; OUTPUT THE COMMAND
1003 03EA 58            POP     AX ; POP ERROR RETURN
1004 03EB
1005 03EB C3            RET
1006 03EC
1007      SEND_SPEC      ENDP
1008
1009      ;-----
1010      ; SEND_SPEC_MD
1011      ; SEND THE SPECIFY COMMAND TO CONTROLLER USING DATA FROM
1012      ; THE MEDIA/DRIVE PARAMETER TABLE POINTED BY (CS:BX) ;
```

```
1012 ; ON ENTRY: CS:BX = MEDIA/DRIVE PARAMETER TABLE ;
1013 ; ON EXIT : NONE ;
1014 ; REGISTERS ALTERED: AX,CX,DX ;
1015 ;-----;
1016 03EC PROC NEAR
1017 03EC B8 0403 R MOV AX,OFFSET SPEC_ESBAC ; LOAD ERROR ADDRESS
1018 03EF 50 PUSH AX ; PUSH NEC_OUT_ERROR RETURN
1019 03F0 B4 03 MOV AH,03H ; SPECIFY COMMAND
1020 03F2 E8 09F0 R CALL NEC_OUTPUT ; OUTPUT THE COMMAND
1021 03F5 2E: 8A 27 MOV AH,C5:[BX],.MD_SPEC1 ; GET 1ST SPECIFY BYTE
1022 03F8 E8 09F0 R CALL NEC_OUTPUT ; OUTPUT THE COMMAND
1023 03FB 2E: 8A 67 01 MOV AH,C5:[BX],.MD_SPEC2 ; GET SECOND SPECIFY BYTE
1024 03FF E8 09F0 R CALL NEC_OUTPUT ; OUTPUT THE COMMAND
1025 0402 58 POP AX ; POP ERROR RETURN
1026 0403 SPEC_ESBAC:
1027 0403 C3 RET
1028 0404 SEND_SPEC_MD ENDP
1029 ;-----;
1030 ; XLAT_NEW
1031 ; TRANSLATES DISKETTE STATE LOCATIONS FROM COMPATIBLE
1032 ; MODE TO NEW ARCHITECTURE.
1033 ;
1034 ; ON ENTRY: DI : DRIVE
1035 ;-----;
1036 0404 XLAT_NEW PROC NEAR
1037 0404 F6 06 00BF R 01 TEST #0Fh,CNTRL,DUAL ; TEST CONTROLLER I.D.
1038 0409 74 22 JZ XN_OUT ;
1039 040B 83 FF 01 CMP DI,C1 ; VALID DRIVE ?
1040 040E 77 1D JA XN_OUT ; IF INVALID BACK
1041 0410 80 BD 0090 R 00 CMP #0Dsk_STATE[DI],0 ; NO DRIVE ?
1042 0415 74 17 DO_DET ; IF NO DRIVE ATTEMPT DETERMINE
1043 0417 83 0F CMPI CX,DI ; CX = DRIVE NUMBER
1044 0419 D0 E1 SHL CL,1 ; CL = SHIFT COUNT, A=0, B=4
1045 041B D0 E1 SHL CL,1 ;
1046 041D A0 00BF R MOV AL,#0Fh,CNTRL ; DRIVE INFORMATION
1047 0420 D2 C8 ROR AL,1 ; TO LOW NIBBLE
1048 0422 24 07 AND AL,DRV_DET+FMt_CAPA+TRK_CAPA ; KEEP DRIVE BITS
1049 0424 80 A5 0090 R F8 AND #0Dsk_STATE[DI],NOT DRV_DET+FMt_CAPA+TRK_CAPA ;
1050 0429 08 85 0090 R OR #0Dsk_STATE[DI],AL ; UPDATE DRIVE STATE
1051 042D XN_OUT:
1052 042D C3 RET
1053 ;
1054 042E DO_DET:
1055 042E E8 0B2B R CALL DRIVE_DET ; TRY TO DETERMINE
1056 0431 C3 RET
1057 ;
1058 0432 XLAT_NEW ENDP
1059 ;-----;
1060 ; XLAT_OLD
1061 ; TRANSLATES DISKETTE STATE LOCATIONS FROM NEW
1062 ; ARCHITECTURE TO COMPATIBLE MODE.
1063 ;
1064 ; ON ENTRY: DI : DRIVE
1065 ;-----;
1066 0432 XLAT_OLD PROC NEAR
1067 0432 F6 06 00BF R 01 TEST #0Fh,CNTRL,DUAL ; TEST CONTROLLER I.D.
1068 0437 74 79 JZ XO_OUT ;
1069 0439 83 FF 01 CMP DI,C1 ; VALID DRIVE ?
1070 043C 77 74 JA XO_OUT ; IF INVALID BACK
1071 043E 80 BD 0090 R 00 CMP #0Dsk_STATE[DI],0 ; NO DRIVE ?
1072 0443 74 6D JZ XO_OUT ; IF NO DRIVE TRANSLATE DONE
1073 ;-----;
1074 ;----- TEST FOR SAVED DRIVE INFORMATION ALREADY SET
1075 ;
1076 0445 8B CF MOV CX,DI ; CX = DRIVE NUMBER
1077 0447 D0 E1 SHL CL,1 ; CL = SHIFT COUNT, A=0, B=4
1078 0449 D0 E1 SHL CL,1 ;
1079 044B B4 02 MOV AH,FMt_CAPA ; LOAD MULTI DATA RATE BIT MASK
1080 044D D2 CC ROR AH,CL ; ROTATE BY MASK
1081 044F 64 26 00BF R TEST #0Fh,CNTRL,AH ; MULTI-DATA RATE DETERMINED ?
1082 0453 75 16 JNZ SAVE_SET ; IF SO, NO NEED TO RE-SAVE
1083 ;
1084 ;----- ERASE DRIVE BITS IN #0Fh_CNTRL FOR THIS DRIVE
1085 ;
1086 0455 B4 07 MOV AH,DRV_DET+FMt_CAPA+TRK_CAPA ; MASK TO KEEP
1087 0457 D2 CC ROR AH,CL ; FIX MASK TO KEEP
1088 0459 F6 04 NOT AH ; TRANSLATE MASK
1089 045B 20 26 00BF R AND #0Fh_CNTRL,AH ; KEEP BITS FROM OTHER DRIVE INTACT
1090 ;----- ACCESS CURRENT DRIVE BITS AND STORE IN #0Fh_CNTRL
1091 ;
1092 045F 8A 55 0090 R MOV AL,#0Dsk_STATE[DI] ; ACCESS STATE
1093 0463 24 07 AND AL,DRV_DET+FMt_CAPA+TRK_CAPA ; KEEP DRIVE BITS
1094 0465 D2 CC ROR AL,CL ; FIX FOR THIS DRIVE
1095 0467 08 06 00BF R OR #0Fh_CNTRL,AL ; UPDATE SAVED DRIVE STATE
1096 ;----- TRANSLATE TO COMPATIBILITY MODE
1097 ;
1098 046B SAVE_SET:
1099 046B 8A A5 0090 R MOV BH,AH,#0Dsk_STATE[DI] ; ACCESS STATE
1100 046F 8A FC MOV BH,AH ; TO BH FOR LATER
1101 0471 80 E4 C0 AND AH,RATE_MSK ; KEEP ONLY RATE
1102 0474 80 FC 00 CMP AH,RATE_500 ; RATE 500 ?
1103 0477 74 10 JZ CHK_HDR_80T ; YES 1,2/1,2 OR HIGH DATA RATE 80 TRK
1104 0479 B0 01 MOV AH,RATE_300 ; RATE 300 ?
1105 047B 80 FC 40 CMP AH,RATE_300 ; AL = 360 IN 1,2 UNESTABLISHED
1106 047E 75 16 JNZ CHK_250 ; NO, 360/360 ,720/720
1107 0480 F6 C7 20 TEST BH,DBL_STEP ; YES, DOUBLE STEP ?
1108 0483 75 1D JNZ TST_DET ; YES, MUST BE 360 IN 1,2
1109 ;
1110 0485 UNKN0:
1111 0485 80 B7 MOV AL,MED_UNK ; 'NONE OF THE ABOVE'
1112 0487 EB 20 JMP SHORT AL_SET ; PROCESS COMPLETE
1113 ;
1114 0489 CHK_HDR_80T:
1115 0489 8A 08CF R CALL CMDS_TYPE ; RETURN DRIVE TYPE IN (AL)
1116 048C 72 F7 JC ERROR_SET ; ERROR SET 'NONE OF THE ABOVE'
1117 048E 3C 02 CMP AL,02 ; 1,2MB DRIVE ?
1118 0490 F3 F3 JNE UNKN0 ; NO, GO SET 'NONE OF THE ABOVE'
1119 0492 B0 02 MOV AL,MIDIU ; AL = 1,2 IN 1,2 UNESTABLISHED
1120 0494 EB 0C JMP SHORT TST_DET
1121 ;
1122 0496 CHK_250:
1123 0496 80 B0 00 MOV AL,M3D3U ; AL = 360 IN 360 UNESTABLISHED
```



```

1126 0498 80 FC 80          CMP      AH,RATE_250          ; RATE 250 ?
1127 0498 75 E8             JNZ      UNKNO              ; IF SO FALL THRU
1128 049D F6 C7 01         TEST     BH,TRK_CAPA          ; 80 TRACK CAPABILITY ?
1129 04A0 75 E3             JNZ      UNKNO              ; IF SO JUMP, FALL THRU TEST DET
1130
1131 04A2                     TST_DET:
1132 04A2 F6 C7 10         TEST     BH,MED_DET          ; DETERMINED ?
1133 04A5 74 02             JZ       AL_SET              ; IF NOT THEN SET
1134 04A7 04 03             ADD      AL,3              ; MAKE DETERMINED/ESTABLISHED
1135
1136 04A9                     AL_SET:
1137 04A9 80 A5 0090 R F8   AND      0DSK_STATE[DI],NOT DRV_DET+FMT_CAPA+TRK_CAPA ; CLEAR DRIVE
1138 04AE 08 85 0090 R     OR       0DSK_STATE[DI],AL          ; REPLACE WITH COMPATIBLE MODE
1139 04B2                     XO_OUT:
1140 04B2 C3              RET
1141 04B3                     XLAT_OLD:
1142                               ENDP
1143
1144 -----
1145 ; RD_WR_VF          PROC NEAR
1146 ; COMMON READ, WRITE AND VERIFY;
1147 ; MAIN LOOP FOR STATE RETRIES.
1148 ; ON ENTRY:        AH : READ/WRITE/VERIFY NEC PARAMETER
1149 ;                  AL : READ/WRITE/VERIFY DMA PARAMETER
1150 ; ON EXIT:         0DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
1151 ;
1152 RD_WR_VF          PROC NEAR
1153                     PUSH     AX
1154                     CALL     XLAT_NEW          ; SAVE DMA, NEC PARAMETERS
1155                     CALL     SETUP_STATE       ; TRANSLATE STATE TO PRESENT ARCH.
1156                     POP      AX              ; INITIALIZE START AND END RATE
1157                     POP      AX              ; RESTORE READ/WRITE/VERIFY
1158
1159 04BB                     DO_AGAIN:
1160 04BB F6 06 00BF R 01   TEST     0HF_CNTRL,DUAL          ; TEST CONTROLLER I.D.
1161 04C0 74 0A             JZ       RWV              ;
1162 04C2 50               PUSH     AX
1163 04C3 E8 05F5 R         CALL     MED_CHANGE       ; SAVE READ/WRITE/VERIFY PARAMETER
1164 04C6 58               POP      AX              ; MEDIA CHANGE AND RESET IF CHANGED
1165 04C7 C3 03           JNC      RWV              ; RESTORE READ/WRITE/VERIFY
1166 04C9 E9 0552 R        JMP      RWV_END
1167 04CC                     RWV:
1168 04CC 50               PUSH     AX
1169                               ; SAVE READ/WRITE/VERIFY PARAMETER
1170 04CD 8A B5 0090 R     MOV      DH,0DSK_STATE[DI]
1171 04D1 80 E6 C0         AND      DH,RATE_MSK          ; GET RATE STATE OF THIS DRIVE
1172                               ; KEEP ONLY RATE
1173 04D4 E8 08CF R        CALL     CMOS_TYPE          ; RETURN DRIVE TYPE IN (AL)
1174 04D7 0A C0           OR       AL,AL          ; TEST FOR NO DRIVE
1175 04D9 74 2F           JZ       RWV_ASSUME       ; ASSUME TYPE, USE MAX TRACK
1176 04DB E8 03B1 R      CALL     DR_TYPE_CHECK      ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
1177 04DE 72 2A           JC       RWV_ASSUME       ; TYPE NOT IN TABLE ASSUME DEFAULT
1178
1179 ;--- SEARCH FOR MEDIA/DRIVE PARAMETER TABLE
1180
1181 04E0 57               PUSH     DI
1182 04E1 33 DB           XOR      BX,BX          ; SAVE DRIVE #
1183 04E3 B9 0006         MOV      CX,DR_CNT          ; BX = INDEX TO DR_TYPE TABLE
1184 04E6                     RWV_DR_SEARCH:
1185 04E6 2E: 8A A7 0000 R MOV      AH,CS:DR_TYPE[BX] ; GET DRIVE TYPE
1186 04EB 80 E4 7F       AND      AH,BITOFF          ; MASK OUT MSB
1187 04EE 3A C4           CMP      AL,AH          ; DRIVE TYPE MATCH ?
1188 04F0 75 0B           JNE      RWV_NXT_MD       ; NO, CHECK NEXT DRIVE TYPE
1189 04F2                     RWV_DR_FND:
1190 04F2 2E: 8B BF 0001 R MOV      DI,WORD PTR CS:DR_TYPE[BX+I] ; DI = MEDIA/DRIVE PARAMETER TABLE
1191 04F7                     RWV_MD_SEARCH:
1192 04F7 2E: 3A 75 0C       CMP      DH,CS:[DI].MD_RATE ; MATCH ?
1193 04FB 74 1D           JE       RWV_MD_FND          ; YES, GO GET 1ST SPECIFY BYTE
1194 04FD                     RWV_NXT_MD:
1195 04FD 83 C3 03         ADD      BX,3              ; CHECK NEXT DRIVE TYPE
1196 0500 E2 E4           LOOP     RWV_DR_SEARCH
1197 0502 C6 06 0041 R FF MOV      0DSKETTE_STATUS,0FFH ; FORCE IT TO RETRY
1198 0507 5F             POP      DI              ; RESTORE DRIVE #
1199 0508 EB 3F           JMP      SHORT_CHK_RET      ; GO RETRY
1200
1201 ;--- ASSUME PRIMARY DRIVE IS INSTALLED AS SHIPPED
1202
1203 050A                     RWV_ASSUME:
1204 050A BB 0000 E        MOV      BX,OFFSET MD_TBL1 ; POINT TO 40 TK 250 KBS
1205 050D F6 85 0090 R 01 TEST     0DSK_STATE[DI],TRK_CAPA ; TEST FOR 80 TRACK
1206 0512 74 09           JZ       RWV_MD_FND1       ; YES, SKIP SEND RATE COMMAND
1207 0514 BB 0000 E        MOV      BX,OFFSET MD_TBL3 ; POINT TO 80 TK 500 KBS
1208 0517 EB 04 90         JMP      RWV_MD_FND1       ; GO SET SPECIFY PARAMETERS
1209
1210 ;--- CS:BX POINTS TO MEDIA/DRIVE PARAMETER TABLE
1211
1212 051A                     RWV_MD_FND:
1213 051A 8B DF           MOV      BX,DI              ; BX = MEDIA/DRIVE PARAMETER TABLE
1214 051C 5F             POP      DI              ; RESTORE DRIVE #
1215 051D
1216
1217 ;--- SEND THE SPECIFY COMMAND TO THE CONTROLLER
1218
1219 051D E8 03EC R        CALL     SEND_SPEC_MD ; ZF=1 ATTEMPT RATE IS SAME AS LAST RATE
1220 0520 E8 0658 R        CALL     CHK_CAstrate ; YES, SKIP SEND RATE COMMAND
1221 0523 74 03           JZ       RWV_MD_FND1       ; SEND DATA RATE TO NEC
1222 0525 E8 0637 R      CALL     SEND_RATE
1223 0528                     RWV_DBL:
1224 0528 53             PUSH     BX
1225 0529 E8 084C R      CALL     SETUP_DBL          ; SAVE MEDIA/DRIVE PARAM ADDRESS
1226 052C 58             POP      BX              ; CHECK FOR DOUBLE STEP
1227 052D 72 1A           JC      CHK_RET          ; RESTORE ADDRESS
1228 052F 58             POP      AX              ; ERROR FROM READ ID, POSSIBLE RETRY
1229 0530 50             PUSH     AX              ; RESTORE NEC,DMA COMMAND
1230 0531 53             PUSH     BX              ; SAVE NEC COMMAND
1231 0532 E8 0668 R      CALL     DMA_SETUP          ; SAVE MEDIA/DRIVE PARAM ADDRESS
1232 0535 5B             POP      BX              ; SET UP THE DMA
1233 0536 58             POP      AX              ; RESTORE ADDRESS
1234 0537 72 1F           JC      RWV_BAC          ; RESTORE NEC COMMAND
1235 0539 50             PUSH     AX              ; CHECK FOR DMA BOUNDARY ERROR
1236 053A 53             PUSH     BX              ; SAVE NEC COMMAND
1237 053B E8 06CB R      CALL     NEC_INIT          ; SAVE MEDIA/DRIVE PARAM ADDRESS
1238 053E 58             POP      BX              ; INITIALIZE NEC
1239                               ; RESTORE ADDRESS

```

```

1239 053F 72 08          JC      CHK_RET      ; IF ERROR DO NOT SEND MORE COMMANDS
1240 0541 E8 04F1 R      CALL   RWV_CMD      ; IF CODE COMMON TO READ/WRITE/VERIFY
1241 0544 72 03          JC      CHK_RET      ; IF ERROR DO NOT SEND MORE COMMANDS
1242 0546 E8 0727 R      CALL   NEC_TERM     ; TERMINATE, GET STATUS, ETC.
1243
1244 0549                CHK_RET:
1245 0549 E8 07BE R      CALL   RETRY      ; CHECK FOR, SETUP RETRY
1246 054C 58            POP     AX          ; RESTORE READ/WRITE/VERIFY PARAMETER
1247 054D 73 03          JNC    RWV_END     ; CY = 0 NO RETRY
1248 054F E9 04BB R      JMP     DO_AGAIN   ; CY = 1 MEANS RETRY
1249
1250 0552                RWV_END:
1251 0552 E8 071B R      CALL   DSTATE      ; ESTABLISH STATE IF SUCCESSFUL
1252 0555 E8 0805 R      CALL   NUM_TRANS   ; AL = NUMBER TRANSFERRED
1253
1254 0558                RWV_BAC:
1255 0558 50            PUSH    AX          ; BAD DMA ERROR ENTRY
1256 0559 E8 0432 R      CALL   XLAT_OLD   ; SAVE NUMBER TRANSFERRED
1257 055C 58            POP     AX          ; TRANSLATE STATE TO COMPATIBLE MODE
1258 055D E8 0832 R      CALL   SETUP_END  ; RESTORE NUMBER TRANSFERRED
1259 0560 C3            RET              ; VARIOUS CLEANUPS
1260 0561
1261                RD_WV_VF      ENDP
1262
1263                ;-----
1264                ; SETUP_STATE: INITIALIZES START AND END RATES.
1265                ;-----
1266 0561                SETUP_STATE      PROC    NEAR
1267 0561 F6 06 00BF R 01   TEST    %HF_CNTRL,DUAL      ; TEST CONTROLLER I.D.
1268 0566 74 37          JZ      JIC          ; MEDIA DETERMINED ?
1269 0568 F4 85 0090 R 10  TEST    %DSK_STATE[D1],MED_DET ; NO STATES IF DETERMINED
1270 056B 06 75 30          JNC    JIC          ; AH = START RATE, AL = END RATE
1271 056F B8 4080          MOV     AX,RATE_300*H+RATE_250 ; DRIVE ?
1272 0572 F6 85 0090 R 04  TEST    %DSK_STATE[D1],DRV_DET ; DO I KNOW DRIVE
1273 0577 74 0C          JZ      JIC          ; SET UP FOR 1.2 M END RATE
1274 0579 B0 00          MOV     AL,RATE_500 ; 1.2 M ?
1275 057B F6 85 0090 R 02  TEST    %DSK_STATE[D1],FMT_CAPA ; JUMP WITH FIXED END RATE
1276 057D 07 75 33          JNZ     JIC          ; START & END RATE = 250 FOR 360 DRIVE
1277 0582 B8 6080          MOV     AX,SETUP_RATE_250*X
1278
1279 0585                AX_SET:
1280 0585 08 A5 0090 R 1F   AND     %DSK_STATE[D1],NOT RATE_MSK+DBL_STEP ; TURN OFF THE RATE
1281 058A 08 A5 0090 R 1F   AND     %DSK_STATE[D1],AH ; RATE FIRST TO TRY
1282 058E 06 26 00BB R F3  OR      %LASTRATE,NOT STRT_MSK ; ERASE LAST TO TRY RATE BITS
1283 0593 D0 C8          ROR     AL,1 ; TO OPERATION LAST RATE LOCATION
1284 0595 D0 C8          ROR     AL,1
1285 0597 D0 C8          ROR     AL,1
1286 0599 D0 C8          ROR     AL,1
1287 059F C3            JIC:
1288 05A0                RET
1289                SETUP_STATE      ENDP
1290
1291                ;-----
1292                ; FMT_INIT: ESTABLISH STATE IF UNESTABLISHED AT FORMAT TIME.
1293                ;-----
1294 05A0                FMT_INIT      PROC    NEAR
1295 05A0 F6 06 00BF R 01   TEST    %HF_CNTRL,DUAL      ; TEST CONTROLLER I.D.
1296 05A5 74 49          JZ      F1_OUT     ; RATE FIRST TO TRY
1297 05A7 F6 85 0090 R 10  TEST    %DSK_STATE[D1],MED_DET ; IS MEDIA ESTABLISHED
1298 05AC 75 42          JNZ     F1_OUT     ; IF 50 RETURN
1299 05AE E8 08CF R      CALL   CMOS_TYPE ; RETURN DRIVE TYPE IN AL
1300 05B1 72 3E          JC      CL_DRV    ; ERROR IN CMOS ASSUME NO DRIVE
1301 05B3 FE C8          DEC     AL        ; MAKE ZERO ORIGIN
1302 05B5 78 3A          JS      CL_DRV    ; NO DRIVE IF AL 0
1303 05B7 8A A5 0090 R 30  MOV     AH,%DSK_STATE[D1] ; AH = CURRENT STATE
1304 05BB 04 E4 0F          AND     AL,NOT MED_DET+DBL_STEP+RATE_MSK ; CLEAR
1305 05BE 0A C0          OR      AL,AL    ; CHECK FOR 360
1306 05C0 75 05          JNZ     N_360    ; IF 360 WILL BE 0
1307 05C2 06 CC 90          OR      AH,MED_DET+RATE_250 ; ESTABLISH MEDIA
1308 05C5 E8 25          JMP     SHORT SKP_STATE ; SKIP OTHER STATE PROCESSING
1309
1310 05C7                N_360:
1311 05C7 DEC            AL              ; 1.2 M DRIVE
1312 05C9 75 05          JNZ     N_12     ; JUMP IF NOT
1313 05CB E8 1C          JMP     FI_RATE  ; SET FORMAT RATE
1314
1315 05D0                FI_RATE:
1316 05D0 DEC            AL              ; SKIP OTHER STATE PROCESSING
1317 05D2 75 0F          JNZ     N_120   ; CHECK FOR TYPE 3
1318 05D4 F6 04 04          TEST    AH,DRV_DET ; IS DRIVE DETERMINED
1319 05D7 74 10          JZ      ISNT_12  ; TREAT AS NON 1.2 DRIVE
1320 05D9 F6 04 02          TEST    AH,FMT_CAPA ; IS 1.2M
1321 05DB 74 0B          JZ      ISNT_12  ; JUMP IF NOT
1322 05DD 06 CC 50          OR      AH,MED_DET+RATE_300 ; RATE 300
1323 05E1 EB 09          JMP     SHORT SKP_STATE ; CONTINUE
1324
1325 05E3                N_120:
1326 05E3 DEC            AL              ; CHECK FOR TYPE 4
1327 05E5 75 0A          JNZ     CL_DRV    ; NO DRIVE, CMOS BAD
1328 05E7 EB E2          JMP     SHORT FI_RATE
1329
1330 05E9                ISNT_12:
1331 05E9 OR              AH,MED_DET+RATE_250 ; MUST BE RATE 250
1332 05EB SKP_STATE:      MOV     %DSK_STATE[D1],AH ; STORE AWAY
1333
1334 05F1                FI_OUT:
1335 05F1 RET              ;
1336 05F3                CL_DRV:
1337 05F3 XOR            AH,AH          ; CLEAR STATE
1338 05F5 JMP            SHORT SKP_STATE ; SAVE IT
1339
1340                FMT_INIT      ENDP
1341
1342                ;-----
1343                ; MED_CHANGE
1344                ; CHECKS FOR MEDIA CHANGE, RESETS MEDIA CHANGE,
1345                ; CHECKS MEDIA CHANGE AGAIN.
1346                ;
1347                ; ON EXIT: CY = 1 MEANS MEDIA CHANGE OR TIMEOUT
1348                ; %DSKETTE_STATUS = ERROR CODE
1349                ;-----
1350 05F5                MED_CHANGE      PROC    NEAR
1351 05F5 F6 06 00BF R 01   TEST    %HF_CNTRL,DUAL      ; TEST CONTROLLER I.D.
1352 05FA 74 37          JZ      OK2        ; BYPASS DISK CHANGE LINE STATE
1353 05FC E8 0821 R      CALL   READ_DSCHNG  ; READ DISK CHANGE LINE STATE
1354 05FF 74 34          JZ      MC_DET     ; BYPASS HANDLING DISK CHANGE LINE
1355 0601 80 A5 0090 R EF AND     %DSK_STATE[D1],NOT MED_DET ; CLEAR STATE FOR THIS DRIVE
1356
1357                ; THIS SEQUENCE ENSURES WHENEVER A DISKETTE IS CHANGED THAT

```

```

1353      ; ON THE NEXT OPERATION THE REQUIRED MOTOR START UP TIME WILL
1354      ; BE WAITED. (DRIVE MOTOR MAY GO OFF UPON DOOR OPENING).
1355
1356 0606 8B CF      MOV     CX,DI          ; CL = DRIVE #
1357 0608 B0 01      MOV     AL,1          ; MOTOR ON BIT MASK
1358 060A D2 E0      SHL     AL,CL        ; TO APPROPRIATE POSITION
1359 060C F6 D0      NOT     AL           ; KEEP ALL BUT MOTOR ON
1360 060E FA         CLI                ; NO INTERRUPTS
1361 060F 20 06 003F R AND     #MOTOR_STATUS,AL ; TURN MOTOR OFF INDICATOR
1362 0613 FB         STI                ; INTERRUPTS ENABLED
1363 0614 E8 0913 R CALL     MOTOR_ON       ; TURN MOTOR ON
1364
1365
1366      ;----- THIS SEQUENCE OF SEEEKS IS USED TO RESET DISKETTE CHANGE SIGNAL
1367 0617 E8 0092 R CALL     DISK_RESET       ; RESET NEC
1368 061A B5 01      MOV     CH,0TH       ; MOVE TO CYLINDER 1
1369 061C E8 0A14 R CALL     SEEK          ; ISSUE SEEK
1370 061F 32 ED      XOR     CH,CH        ; MOVE TO CYLINDER 0
1371 0621 E8 0A14 R CALL     SEEK          ; ISSUE SEEK
1372 0624 C6 06 0041 R 06 MOV     #DSKETTE_STATUS,MEDIA_CHANGE ; STORE IN STATUS
1373
1374 0629 E8 0B21 R OK1: CALL     READ_DSKCHNG ; CHECK MEDIA CHANGED AGAIN
1375 062C 74 05      JZ      OK2          ; IF ACTIVE, NO DISKETTE, TIMEOUT
1376
1377 062E C6 06 0041 R 80 OK4: MOV     #DSKETTE_STATUS,TIME_OUT; TIMEOUT IF DRIVE EMPTY
1378
1379 0633 F9      OK2: STC                ; MEDIA CHANGED, SET CY
1380 0634 C3      RET
1381 0635      MC_OUT: CLC                ; NO MEDIA CHANGED, CLEAR CY
1382 0635 F8      RET
1383 0636 C3      RET
1384 0637
1385      MED_CHANGE      ENDP
1386
1387      ;-----
1388      ; SEND_RATE
1389      ; SENDS DATA RATE COMMAND TO NEC
1390      ; ON ENTRY: D1 = DRIVE #
1391      ; ON EXIT: NONE
1392      ; REGISTERS ALTERED: NONE
1393      ;-----
1394 0637 F6 06 008F R 01 SEND_RATE PROC NEAR
1395 063C 74 19      JZ      #HF_CNTRL,DUAL ; TEST CONTROLLER I.D.
1396 063E 50      PUSH     AX             ; SAVE REG.
1397 063F 80 26 008B R 3F AND     #LAstrate,NOT SEND_MSK ; ELSE CLEAR LAST RATE ATTEMPTED
1398 0644 8A 85 0090 R MOV     AL,#DSK_STATE[D1] ; GET RATE STATE OF THIS DRIVE
1399 0648 24 C0      AND     AL,SEND_MSK ; KEEP ONLY RATE BITS
1400 064E D0 C0      OR      #LAstrate,AL ; SAVE NEW RATE FOR NEXT CHECK
1401 0650 D0 C0      ROL     AL,1        ; MOVE TO BIT OUTPUT POSITIONS
1402 0652 8A 03F7 MOV     DX,03F7H       ; OUTPUT NEW DATA RATE
1403 0655 EE      OUT     DX,AL         ;
1404 0656 58      POP     AX           ; RESTORE REG.
1405 0657      C_S_OUT: RET
1406 0657 C3      SEND_RATE      ENDP
1407 0658
1408
1409      ;-----
1410      ; CHK_LASTRATE
1411      ; CHECK PREVIOUS DATA RATE SENT TO THE CONTROLLER.
1412      ; ON ENTRY:
1413      ; D1 = DRIVE #
1414      ; ON EXIT:
1415      ; ZF = 1 DATA RATE IS THE SAME AS LAST RATE SENT TO NEC
1416      ; ZF = 0 DATA RATE IS DIFFERENT FROM LAST RATE
1417      ; REGISTERS ALTERED: NONE
1418      ;-----
1419 0658      CHK_LASTRATE PROC NEAR
1420 0658 50      AX             ; SAVE REG
1421 0659 8A 26 008B R MOV     AH,#LAstrate ; GET LAST DATA RATE SELECTED
1422 065D 8A 85 0090 R MOV     AL,#DSK_STATE[D1] ; GET RATE STATE OF THIS DRIVE
1423 0661 25 C0C0 AND     AX,SEND_MSK*X ; KEEP ONLY RATE BITS OF BOTH
1424 0664 3A C4      CMP     AL,AH       ; COMPARE TO PREVIOUSLY TRIED
1425      ; ZF = 1 RATE IS THE SAME
1426 0666 58      POP     AX           ; RESTORE REG.
1427 0667 C3      RET
1428 0668      CHK_LASTRATE      ENDP
1429

```

```

1430 PAGE
1431 -----
1432 DMA_SETUP
1433 THIS ROUTINE SETS UP THE DMA FOR READ/WRITE/VERIFY
1434 OPERATIONS.
1435
1436 ON ENTRY: AL = DMA COMMAND
1437
1438 ON EXIT: 00SKETTE_STATUS, CY REFLECT STATUS OF OPERATION
1439
1440 DMA_SETUP PROC NEAR
1441 1441 0668 FA CL I ; DISABLE INTERRUPTS DURING DMA SET-UP
1442 1442 0669 E6 0C OUT DMA+12,AL ; SET THE FIRST/LAST F/F
1443 1443 066B EB 00 JMP $+2 ; WAIT FOR I/O
1444 1444 066D E6 0B OUT DMA+11,AL ; OUTPUT THE MODE BYTE
1445 1445 066F 3C 42 CMP AL,42H ; DMA VERIFY COMMAND
1446 1446 0671 75 04 JNE NOT_VERF ; NO
1447 1447 0673 33 C0 XOR AX,AX ; START ADDRESS
1448 1448 0675 EB 15 JMP SHORT J33
1449 1449 0677 NOT_VERF: MOV AX,ES ; GET THE ES VALUE
1450 1450 0677 8C C0 ROL AX,1 ; ROTATE LEFT
1451 1451 0679 D1 C0 ROL AX,1
1452 1452 067B D1 C0 ROL AX,1
1453 1453 067D D1 C0 ROL AX,1
1454 1454 067F D1 C0 ROL AX,1
1455 1455 0681 8A E8 MOV CH,AL ; GET HIGHEST NIBBLE OF ES TO CH
1456 1456 0683 24 F0 AND AL,11110000B ; ZERO THE LOW NIBBLE FROM SEGMENT
1457 1457 0685 03 46 02 ADD AX,[BP+2] ; TEST FOR CARRY FROM ADDITION
1458 1458 0688 73 02 JNC J33
1459 1459 068A FE C5 INC CH ; CARRY MEANS HIGH 4 BITS MUST BE INC
1460 1460 068C J33:
1461 1461 068C 50 PUSH AX ; SAVE START ADDRESS
1462 1462 068D E6 04 OUT DMA+4,AL ; OUTPUT LOW ADDRESS
1463 1463 068F EB 00 JMP $+2 ; WAIT FOR I/O
1464 1464 0691 8A C4 MOV AL,AH ; OUTPUT HIGH ADDRESS
1465 1465 0693 E6 04 OUT DMA+4,AL ; GET HIGH 4 BITS
1466 1466 0695 8A C5 MOV AL,CH ; I/O WAIT STATE
1467 1467 0697 EB 00 JMP $+2
1468 1468 0699 24 F0 AND AL,00001111B ; OUTPUT HIGH 4 BITS TO PAGE REGISTER
1469 1469 069B E6 81 OUT 081H,AL
1470
1471 ----- DETERMINE COUNT
1472
1473 1473 069D 8B C6 MOV AX,S1 ; AL = # OF SECTORS
1474 1474 069F 86 C4 XCHG AL,AH ; AH = # OF SECTORS
1475 1475 06A1 2A C0 SUB AL,AL ; AL = 0, AX = # OF SECTORS * 256
1476 1476 06A3 D1 E8 SHR AX,1 ; AX = # SECTORS * 128
1477 1477 06A5 50 PUSH AX ; SAVE # OF SECTORS * 128
1478 1478 06A6 B2 03 MOV DL,3 ; GET BYTES/SECTOR PARAMETER
1479 1479 06A8 E8 08FE R CALL GET_PARM ;
1480 1480 06AB 8A CC MOV CL,AH ; SHIFT COUNT (0=128, 1=256 ETC)
1481 1481 06AD 58 POP AX ; AX = # OF SECTORS * 128
1482 1482 06AE D3 E0 SHL AX,CL ; SHIFT BY PARAMETER VALUE
1483 1483 06B0 48 DEC AX ;-1 FOR DMA VALUE
1484 1484 06B1 50 PUSH AX ; SAVE COUNT VALUE
1485 1485 06B2 E6 05 OUT DMA+5,AL ; LOW BYTE OF COUNT
1486 1486 06B4 EB 00 JMP $+2 ; WAIT FOR I/O
1487 1487 06B6 8A C4 MOV AL,AH ; HIGH BYTE OF COUNT
1488 1488 06B8 E6 05 OUT DMA+5,AL ; RE-ENABLE INTERRUPTS
1489 1489 06BA FB STI ; RECOVER COUNT VALUE
1490 1490 06BB 59 POP CX ; RECOVER ADDRESS VALUE
1491 1491 06BC 58 POP AX ; ADD, TEST FOR 64K OVERFLOW
1492 1492 06BD 03 C1 ADD AX,CX ; MODE FOR 8237
1493 1493 06BF 80 02 MOV AL,2 ; INITIALIZE THE DISKETTE CHANNEL
1494 1494 06C1 E6 0A OUT DMA+10,AL
1495
1496 1496 06C3 73 05 JNC NO_BAD ; CHECK FOR ERROR
1497 1497 06C5 C6 06 0041 R 09 MOV 00SKETTE_STATUS,DMA_BOUNDARY ; SET ERROR
1498
1499 1499 06CA NO_BAD: RET ; CY SET BY ABOVE IF ERROR
1500 1500 06CA C3
1501 1501 06CB DMA_SETUP END
1502
1503 -----
1504 NEC_INIT
1505 THIS ROUTINE SEEKS TO THE REQUESTED TRACK AND
1506 INITIALIZES THE NEC FOR THE READ/WRITE/VERIFY/FORMAT
1507 OPERATION.
1508
1509 ON ENTRY: AH : NEC COMMAND TO BE PERFORMED
1510
1511 ON EXIT: 00SKETTE_STATUS, CY REFLECT STATUS OF OPERATION
1512
1513 NEC_INIT PROC NEAR
1514 1513 06CB 50 PUSH AX ; SAVE NEC COMMAND
1515 1514 06CC E8 0913 R CALL MOTOR_ON ; TURN MOTOR ON FOR SPECIFIC DRIVE
1516
1517 ----- DO THE SEEK OPERATION
1518
1519 1518 06CF 8A E6 01 MOV CH,[BP+1] ; CH = TRACK #
1520 1519 06D2 E8 0A14 R CALL SEEK ; MOVE TO CORRECT TRACK
1521 1520 06D5 58 POP AX ; RECOVER COMMAND
1522 1521 06D6 72 18 JC ER_1 ; ERROR ON SEEK
1523 1522 06D8 BB 06F0 R MOV MOV,OFFSET ER_1 ; LOAD ERROR ADDRESS
1524 1523 06DB 53 PUSH BX ; PUSH NEC_OUT ERROR RETURN
1525
1526 ----- SEND OUT THE PARAMETERS TO THE CONTROLLER
1527
1528 1527 06DC E8 09F0 R CALL NEC_OUTPUT ; OUTPUT THE OPERATION COMMAND
1529 1528 06DF 8B C6 MOV AX,S1 ; AH = HEAD #
1530 1529 06E1 8B DF MOV BX,D1 ; BL = DRIVE #
1531 1530 06E3 D0 E4 SAL AH,1 ; MOVE IT TO BIT 2
1532 1531 06E5 D0 E4 SAL AH,1
1533 1532 06E7 80 E4 04 AND AH,0000100B ; ISOLATE THAT BIT
1534 1533 06EA 0A E3 OR AH,BL ; OR IN THE DRIVE NUMBER
1535 1534 06EC E8 09F0 R CALL NEC_OUTPUT ; FALL THRU CY SET IF ERROR
1536 1535 06EF 5B POP BX ; THROW AWAY ERROR RETURN
1537 1536 06F0 ER_1: RET
1538 1537 06F1 NEC_INIT END
1539
1540 -----
1541 RWV_COM
1542 THIS ROUTINE SENDS PARAMETERS TO THE NEC SPECIFIC
1543 TO THE READ/WRITE/VERIFY OPERATIONS.
1544
1545

```

```

1544      ; ON ENTRY:      CS:BX = ADDRESS OF MEDIA/DRIVE PARAMETER TABLE ;
1545      ; ON EXIT :      #DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION ;
1546      -----
1547      RWV_COM PROC      NEAR
1548      MOV      AX,OFFSET ER_2      ; LOAD ERROR ADDRESS
1549      PUSH     AX      ; PUSH NEC OUT ERROR RETURN
1550      MOV      AH,[BP+1]      ; OUTPUT TRACK #
1551      CALL     NEC_OUTPUT
1552      MOV      AX,S1      ; OUTPUT HEAD #
1553      CALL     NEC_OUTPUT
1554      MOV      AH,[BP]      ; OUTPUT SECTOR #
1555      CALL     NEC_OUTPUT
1556      MOV      DL,S      ; BYTES/SECTOR PARAMETER FROM BLOCK
1557      CALL     GET_PARM      ; . TO THE NEC
1558      CALL     NEC_OUTPUT      ; OUTPUT TO CONTROLLER
1559      MOV      DL,4      ; EOT PARAMETER FROM BLOCK
1560      CALL     GET_PARM      ; . TO THE NEC
1561      CALL     NEC_OUTPUT      ; OUTPUT TO CONTROLLER
1562      MOV      AH,CS:[BX].MD_GAP      ; GET GAP LENGTH
1563      R15:
1564      CALL     NEC_OUTPUT
1565      MOV      DL,6      ; DTL PARAMETER FROM BLOCK
1566      CALL     GET_PARM      ; . TO THE NEC
1567      CALL     NEC_OUTPUT      ; OUTPUT TO CONTROLLER
1568      POP      AX      ; THROW AWAY ERROR EXIT
1569      ER_2:
1570      RET
1571      RWV_COM ENDP
1572      -----
1573      ; NEC_TERM
1574      ; THIS ROUTINE WAITS FOR THE OPERATION THEN ACCEPTS
1575      ; THE STATUS FROM THE NEC FOR THE READ/WRITE/VERIFY/
1576      ; FORMAT OPERATION.
1577      ;
1578      ; ON EXIT:      #DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION ;
1579      -----
1580      NEC_TERM PROC      NEAR
1581      ;----- LET THE OPERATION HAPPEN
1582      1583:
1584      PUSH     SI      ; SAVE HEAD #, # OF SECTORS
1585      CALL     WAIT_INT      ; WAIT FOR THE INTERRUPT
1586      PUSHF
1587      CALL     RESULTS      ; GET THE NEC STATUS
1588      JC       SET_END_POP
1589      POPF
1590      JC       SET_END      ; LOOK FOR ERROR
1591      1592:
1593      ;----- CHECK THE RESULTS RETURNED BY THE CONTROLLER
1594      CLO
1595      MOV      SI,OFFSET #NEC_STATUS      ; SET THE CORRECT DIRECTION
1596      LODS     #NEC_STATUS      ; POINT TO STATUS FIELD
1597      AND      AL,11000000B      ; GET ST0
1598      JZ       SET_END      ; TEST FOR NORMAL TERMINATION
1599      CMP      AL,01000000B      ; TEST FOR ABNORMAL TERMINATION
1600      JNB      J18      ; NOT ABNORMAL, BAD NEC
1601      1602:
1603      ;----- ABNORMAL TERMINATION, FIND OUT WHY
1604      LODS     #NEC_STATUS      ; GET ST1
1605      SAL      AL,1      ; TEST FOR EOT FOUND
1606      MOV      AH,RECORD_NOT_FND
1607      JC       J19
1608      SAL      AL,1
1609      SAL      AL,1
1610      MOV      AH,BAD_CRC
1611      JC       J19
1612      SAL      AL,1
1613      MOV      AH,BAD_DMA      ; TEST FOR DMA OVERRUN
1614      JC       J19
1615      SAL      AL,1
1616      SAL      AL,1      ; TEST FOR RECORD NOT FOUND
1617      MOV      AH,RECORD_NOT_FND
1618      JC       J19
1619      SAL      AL,1
1620      MOV      AH,WRITE_PROTECT      ; TEST FOR WRITE_PROTECT
1621      JC       J19
1622      SAL      AL,1      ; TEST MISSING ADDRESS MARK
1623      MOV      AH,BAD_ADDR_MARK
1624      JC       J19
1625      1626:
1627      ;----- NEC MUST HAVE FAILED
1628      J18:
1629      MOV      AH,BAD_NEC
1630      SET_END:OR      #DSKETTE_STATUS,AH
1631      1632:
1633      CMP      #DSKETTE_STATUS,1      ; SET ERROR CONDITION
1634      CMC
1635      POP      SI      ; RESTORE HEAD #, # OF SECTORS
1636      1637:
1638      SET_END_POP:
1639      POPF
1640      JMP      SHORT SET_END
1641      NEC_TERM ENDP
1642      -----
1643      ; DSTATE: ESTABLISH STATE UPON SUCCESSFUL OPERATION.
1644      DSTATE PROC      NEAR
1645      MOV      BH,CNTRL_DUAL      ; TEST CONTROLLER I.D.
1646      JZ       SETBAC
1647      CMP      #DSKETTE_STATUS,0      ; CHECK FOR ERROR
1648      JNZ      SETBAC      ; IF ERROR JUMP
1649      MOV      #DSK_STATE[D1].MED_DET      ; NO ERROR, MARK MEDIA AS DETERMINED
1650      TEST     #DSK_STATE[D1].DRV_DET      ; DRIVE DETERMINED ?
1651      JNZ      SETBAC      ; IF DETERMINED NO TRY TO DETERMINE
1652      MOV      AL,#DSK_STATE[D1]      ; LOAD STATE
1653      AND      AL,RATE_NSK      ; KEEP ONLY RATE
1654      CMP      AL,RATE_250      ; RATE 250 ?
1655      JNE      M_12      ; NO MUST BE 1.2M OR HI DATA RATE 80 TRK
1656      1657:
1658      ;--- CHECK FOR HIGH DATA RATE 80 TRACK
    
```

```

1658      1659 079F E8 08CF R      CALL    CMOS_TYPE      ; RETURN DRIVE TYPE IN (AL)
1660 07A2 72 14      JC          M_12      ; CMOS BAD ASSUME DEFAULT
1661 07A4 3C 02      CJP         AL,2      ; TYPE 2 DRIVE ?
1662 07A6 74 10      JNE         M_12      ; YES-->ASSUME MULTI FORMAT CAPABILITY
1663 07AB 3C 04      CMP         AL,4      ; TYPE 4 DRIVE ?
1664 07AA 74 0C      JE          M_12      ; YES-->ASSUME MULTI FORMAT CAPABILITY
1665 07AC      M_20:      *DSK_STATE[D1],NOT FMT_CAPA ; TURN OFF FORMAT CAPABILITY
1666 07AC      AND         *DSK_STATE[D1],DRV_DET ; MARK DRIVE DETERMINED
1667 07B1 80 8D 0090 R 04      JMP         SHORT SETBAC ; BACK
1668 07B6 E8 05      M_12:      OR          *DSK_STATE[D1],DRV_DET+FMT_CAPA ; TURN ON DETERMINED & FMT CAPA
1669
1670 07B8      SETBAC:      RET
1671 07B8 80 8D 0090 R 06      DSTATE  ENDP
1672
1673 07BD      ; RETRY
1674 07BD C3          ; DETERMINES WHETHER A RETRY IS NECESSARY. IF RETRY IS
1675 07BE          ; REQUIRED THEN STATE INFORMATION IS UPDATED FOR RETRY.
1676          ;
1677          ; ON EXIT:      CY = 1 FOR RETRY, CY = 0 FOR NO RETRY
1678          ;
1679          ;
1680          ;
1681          ;
1682          ;
1683 07BE      RETRY      PROC    NEAR
1684 07BE 80 3E 0041 R 00      *DSKETTE_STATUS,0 ; GET STATUS OF OPERATION
1685 07C3 74 3E          JZ          NO_RETRY ; SUCCESSFUL OPERATION
1686 07C5 80 3E 0041 R 80      *DSKETTE_STATUS,TIME_OUT ; IF TIME OUT NO RETRY
1687 07CA 74 37          JZ          NO_RETRY
1688 07CC 8A A5 0090 R      MOV         AH,*DSK_STATE[D1] ; GET MEDIA STATE OF DRIVE
1689 07D0 F6 C4 10      TEST        AH,WED_DET ; ESTABLISHED/DETERMINED ?
1690 07D3 75 2E          JNZ         NO_RETRY ; IF ESTABLISHED STATE THEN TRUE ERROR
1691 07D5 80 E4 C0      AND         AND         AH,RATE_MSK ; ISOLATE RATE
1692 07D8 8A 2E 008B R      MOV         CH,*LASTRATE ; GET START OPERATION STATE
1693 07DC D0 C5          ROL          CH,1 ; TO CORRESPONDING BITS
1694 07DE D0 C5          ROL          CH,1
1695 07E0 D0 C5          ROL          CH,1
1696 07E2 D0 C5          ROL          CH,1
1697 07E4 80 E5 C0      AND         CH,RATE_MSK ; ISOLATE RATE BITS
1698 07E7 3A EC          CMP         CH,AH ; ALL RATES TRIED
1699 07E9 74 18          JE          NO_RETRY ; IF YES, THEN TRUE ERROR
1700
1701          ;
1702          ; SETUP STATE INDICATOR FOR RETRY ATTEMPT TO NEXT RATE
1703          ;
1704          ; 00000000B (500) -> 10000000B (250)
1705          ; 10000000B (250) -> 01000000B (300)
1706          ; 01000000B (300) -> 00000000B (500)
1707
1708 07EB 80 FC 01      CMP         AH,RATE_500+1 ; SET CY FOR RATE 500
1709 07EE D0 DC          RCR          AH,1 ; TO NEXT STATE
1710 07F0 80 E4 C0      AND         AH,RATE_MSK ; KEEP ONLY RATE BITS
1711 07F3 80 A5 0090 R IF      AND         *DSK_STATE[D1],NOT RATE_MSK+DBL_STEP ; RATE, DBL STEP OFF
1712 07F8 08 A5 0090 R      MOV         *DSK_STATE[D1],AH ; TURN ON NEW RATE
1713 07FC C6 06 0041 R 00      OR          *DSKETTE_STATUS,0 ; RESET STATUS FOR RETRY
1714 0801 F9          STC          ; SET CARRY FOR RETRY
1715 0802 C3          RET          ; RETRY RETURN
1716
1717 0803      NO_RETRY:      CLC          ; CLEAR CARRY NO RETRY
1718 0804 F8          RET          ; NO RETRY RETURN
1719 0805
1720      RETRY      ENDP
1721
1722          ;
1723          ; NUM_TRANS
1724          ; THIS ROUTINE CALCULATES THE NUMBER OF SECTORS THAT
1725          ; WERE ACTUALLY TRANSFERRED TO/FROM THE DISKETTE.
1726          ;
1727          ; ON ENTRY:      [BP+1] = TRACK
1728          ; SI=HI = HEAD
1729          ; [BP] = START SECTOR
1730          ;
1731          ; ON EXIT:      AL = NUMBER ACTUALLY TRANSFERRED
1732          ;
1733          ;
1734          ;
1735          ;
1736          ;
1737          ;
1738          ;
1739          ;
1740          ;
1741          ;
1742          ;
1743          ;
1744          ;
1745          ;
1746          ;
1747          ;
1748          ;
1749          ;
1750          ;
1751          ;
1752          ;
1753          ;
1754          ;
1755          ;
1756          ;
1757          ;
1758          ;
1759          ;
1760          ;
1761          ;
1762          ;
1763          ;
1764          ;
1765          ;
1766          ;
1767          ;
1768          ;
1769          ;
1770          ;
1771          ;
1772          ;
1773          ;
1774          ;
1775          ;
1776          ;
1777          ;
1778          ;
1779          ;
1780          ;
1781          ;
1782          ;
1783          ;
1784          ;
1785          ;
1786          ;
1787          ;
1788          ;
1789          ;
1790          ;
1791          ;
1792          ;
1793          ;
1794          ;
1795          ;
1796          ;
1797          ;
1798          ;
1799          ;
1800          ;
1801          ;
1802          ;
1803          ;
1804          ;
1805          ;
1806          ;
1807          ;
1808          ;
1809          ;
1810          ;
1811          ;
1812          ;
1813          ;
1814          ;
1815          ;
1816          ;
1817          ;
1818          ;
1819          ;
1820          ;
1821          ;
1822          ;
1823          ;
1824          ;
1825          ;
1826          ;
1827          ;
1828          ;
1829          ;
1830          ;
1831          ;
1832          ;
1833          ;
1834          ;
1835          ;
1836          ;
1837          ;
1838          ;
1839          ;
1840          ;
1841          ;
1842          ;
1843          ;
1844          ;
1845          ;
1846          ;
1847          ;
1848          ;
1849          ;
1850          ;
1851          ;
1852          ;
1853          ;
1854          ;
1855          ;
1856          ;
1857          ;
1858          ;
1859          ;
1860          ;
1861          ;
1862          ;
1863          ;
1864          ;
1865          ;
1866          ;
1867          ;
1868          ;
1869          ;
1870          ;
1871          ;
1872          ;
1873          ;
1874          ;
1875          ;
1876          ;
1877          ;
1878          ;
1879          ;
1880          ;
1881          ;
1882          ;
1883          ;
1884          ;
1885          ;
1886          ;
1887          ;
1888          ;
1889          ;
1890          ;
1891          ;
1892          ;
1893          ;
1894          ;
1895          ;
1896          ;
1897          ;
1898          ;
1899          ;
1900          ;
1901          ;
1902          ;
1903          ;
1904          ;
1905          ;
1906          ;
1907          ;
1908          ;
1909          ;
1910          ;
1911          ;
1912          ;
1913          ;
1914          ;
1915          ;
1916          ;
1917          ;
1918          ;
1919          ;
1920          ;
1921          ;
1922          ;
1923          ;
1924          ;
1925          ;
1926          ;
1927          ;
1928          ;
1929          ;
1930          ;
1931          ;
1932          ;
1933          ;
1934          ;
1935          ;
1936          ;
1937          ;
1938          ;
1939          ;
1940          ;
1941          ;
1942          ;
1943          ;
1944          ;
1945          ;
1946          ;
1947          ;
1948          ;
1949          ;
1950          ;
1951          ;
1952          ;
1953          ;
1954          ;
1955          ;
1956          ;
1957          ;
1958          ;
1959          ;
1960          ;
1961          ;
1962          ;
1963          ;
1964          ;
1965          ;
1966          ;
1967          ;
1968          ;
1969          ;
1970          ;
1971          ;
1972          ;
1973          ;
1974          ;
1975          ;
1976          ;
1977          ;
1978          ;
1979          ;
1980          ;
1981          ;
1982          ;
1983          ;
1984          ;
1985          ;
1986          ;
1987          ;
1988          ;
1989          ;
1990          ;
1991          ;
1992          ;
1993          ;
1994          ;
1995          ;
1996          ;
1997          ;
1998          ;
1999          ;
2000          ;
2001          ;
2002          ;
2003          ;
2004          ;
2005          ;
2006          ;
2007          ;
2008          ;
2009          ;
2010          ;
2011          ;
2012          ;
2013          ;
2014          ;
2015          ;
2016          ;
2017          ;
2018          ;
2019          ;
2020          ;
2021          ;
2022          ;
2023          ;
2024          ;
2025          ;
2026          ;
2027          ;
2028          ;
2029          ;
2030          ;
2031          ;
2032          ;
2033          ;
2034          ;
2035          ;
2036          ;
2037          ;
2038          ;
2039          ;
2040          ;
2041          ;
2042          ;
2043          ;
2044          ;
2045          ;
2046          ;
2047          ;
2048          ;
2049          ;
2050          ;
2051          ;
2052          ;
2053          ;
2054          ;
2055          ;
2056          ;
2057          ;
2058          ;
2059          ;
2060          ;
2061          ;
2062          ;
2063          ;
2064          ;
2065          ;
2066          ;
2067          ;
2068          ;
2069          ;
2070          ;
2071          ;
2072          ;
2073          ;
2074          ;
2075          ;
2076          ;
2077          ;
2078          ;
2079          ;
2080          ;
2081          ;
2082          ;
2083          ;
2084          ;
2085          ;
2086          ;
2087          ;
2088          ;
2089          ;
2090          ;
2091          ;
2092          ;
2093          ;
2094          ;
2095          ;
2096          ;
2097          ;
2098          ;
2099          ;
2100          ;
2101          ;
2102          ;
2103          ;
2104          ;
2105          ;
2106          ;
2107          ;
2108          ;
2109          ;
2110          ;
2111          ;
2112          ;
2113          ;
2114          ;
2115          ;
2116          ;
2117          ;
2118          ;
2119          ;
2120          ;
2121          ;
2122          ;
2123          ;
2124          ;
2125          ;
2126          ;
2127          ;
2128          ;
2129          ;
2130          ;
2131          ;
2132          ;
2133          ;
2134          ;
2135          ;
2136          ;
2137          ;
2138          ;
2139          ;
2140          ;
2141          ;
2142          ;
2143          ;
2144          ;
2145          ;
2146          ;
2147          ;
2148          ;
2149          ;
2150          ;
2151          ;
2152          ;
2153          ;
2154          ;
2155          ;
2156          ;
2157          ;
2158          ;
2159          ;
2160          ;
2161          ;
2162          ;
2163          ;
2164          ;
2165          ;
2166          ;
2167          ;
2168          ;
2169          ;
2170          ;
2171          ;
2172          ;
2173          ;
2174          ;
2175          ;
2176          ;
2177          ;
2178          ;
2179          ;
2180          ;
2181          ;
2182          ;
2183          ;
2184          ;
2185          ;
2186          ;
2187          ;
2188          ;
2189          ;
2190          ;
2191          ;
2192          ;
2193          ;
2194          ;
2195          ;
2196          ;
2197          ;
2198          ;
2199          ;
2200          ;
2201          ;
2202          ;
2203          ;
2204          ;
2205          ;
2206          ;
2207          ;
2208          ;
2209          ;
2210          ;
2211          ;
2212          ;
2213          ;
2214          ;
2215          ;
2216          ;
2217          ;
2218          ;
2219          ;
2220          ;
2221          ;
2222          ;
2223          ;
2224          ;
2225          ;
2226          ;
2227          ;
2228          ;
2229          ;
2230          ;
2231          ;
2232          ;
2233          ;
2234          ;
2235          ;
2236          ;
2237          ;
2238          ;
2239          ;
2240          ;
2241          ;
2242          ;
2243          ;
2244          ;
2245          ;
2246          ;
2247          ;
2248          ;
2249          ;
2250          ;
2251          ;
2252          ;
2253          ;
2254          ;
2255          ;
2256          ;
2257          ;
2258          ;
2259          ;
2260          ;
2261          ;
2262          ;
2263          ;
2264          ;
2265          ;
2266          ;
2267          ;
2268          ;
2269          ;
2270          ;
2271          ;
2272          ;
2273          ;
2274          ;
2275          ;
2276          ;
2277          ;
2278          ;
2279          ;
2280          ;
2281          ;
2282          ;
2283          ;
2284          ;
2285          ;
2286          ;
2287          ;
2288          ;
2289          ;
2290          ;
2291          ;
2292          ;
2293          ;
2294          ;
2295          ;
2296          ;
2297          ;
2298          ;
2299          ;
2300          ;
2301          ;
2302          ;
2303          ;
2304          ;
2305          ;
2306          ;
2307          ;
2308          ;
2309          ;
2310          ;
2311          ;
2312          ;
2313          ;
2314          ;
2315          ;
2316          ;
2317          ;
2318          ;
2319          ;
2320          ;
2321          ;
2322          ;
2323          ;
2324          ;
2325          ;
2326          ;
2327          ;
2328          ;
2329          ;
2330          ;
2331          ;
2332          ;
2333          ;
2334          ;
2335          ;
2336          ;
2337          ;
2338          ;
2339          ;
2340          ;
2341          ;
2342          ;
2343          ;
2344          ;
2345          ;
2346          ;
2347          ;
2348          ;
2349          ;
2350          ;
2351          ;
2352          ;
2353          ;
2354          ;
2355          ;
2356          ;
2357          ;
2358          ;
2359          ;
2360          ;
2361          ;
2362          ;
2363          ;
2364          ;
2365          ;
2366          ;
2367          ;
2368          ;
2369          ;
2370          ;
2371          ;
2372          ;
2373          ;
2374          ;
2375          ;
2376          ;
2377          ;
2378          ;
2379          ;
2380          ;
2381          ;
2382          ;
2383          ;
2384          ;
2385          ;
2386          ;
2387          ;
2388          ;
2389          ;
2390          ;
2391          ;
2392          ;
2393          ;
2394          ;
2395          ;
2396          ;
2397          ;
2398          ;
2399          ;
2400          ;
2401          ;
2402          ;
2403          ;
2404          ;
2405          ;
2406          ;
2407          ;
2408          ;
2409          ;
2410          ;
2411          ;
2412          ;
2413          ;
2414          ;
2415          ;
2416          ;
2417          ;
2418          ;
2419          ;
2420          ;
2421          ;
2422          ;
2423          ;
2424          ;
2425          ;
2426          ;
2427          ;
2428          ;
2429          ;
2430          ;
2431          ;
2432          ;
2433          ;
2434          ;
2435          ;
2436          ;
2437          ;
2438          ;
2439          ;
2440          ;
2441          ;
2442          ;
2443          ;
2444          ;
2445          ;
2446          ;
2447          ;
2448          ;
2449          ;
2450          ;
2451          ;
2452          ;
2453          ;
2454          ;
2455          ;
2456          ;
2457          ;
2458          ;
2459          ;
2460          ;
2461          ;
2462          ;
2463          ;
2464          ;
2465          ;
2466          ;
2467          ;
2468          ;
2469          ;
2470          ;
2471          ;
2472          ;
2473          ;
2474          ;
2475          ;
2476          ;
2477          ;
2478          ;
2479          ;
2480          ;
2481          ;
2482          ;
2483          ;
2484          ;
2485          ;
2486          ;
2487          ;
2488          ;
2489          ;
2490          ;
2491          ;
2492          ;
2493          ;
2494          ;
2495          ;
2496          ;
2497          ;
2498          ;
2499          ;
2500          ;
2501          ;
2502          ;
2503          ;
2504          ;
2505          ;
2506          ;
2507          ;
2508          ;
2509          ;
2510          ;
2511          ;
2512          ;
2513          ;
2514          ;
2515          ;
2516          ;
2517          ;
2518          ;
2519          ;
2520          ;
2521          ;
2522          ;
2523          ;
2524          ;
2525          ;
2526          ;
2527          ;
2528          ;
2529          ;
2530          ;
2531          ;
2532          ;
2533          ;
2534          ;
2535          ;
2536          ;
2537          ;
2538          ;
2539          ;
2540          ;
2541          ;
2542          ;
2543          ;
2544          ;
2545          ;
2546          ;
2547          ;
2548          ;
2549          ;
2550          ;
2551          ;
2552          ;
2553          ;
2554          ;
2555          ;
2556          ;
2557          ;
2558          ;
2559          ;
2560          ;
2561          ;
2562          ;
2563          ;
2564          ;
2565          ;
2566          ;
2567          ;
2568          ;
2569          ;
2570          ;
2571          ;
2572          ;
2573          ;
2574          ;
2575          ;
2576          ;
2577          ;
2578          ;
2579          ;
2580          ;
2581          ;
2582          ;
2583          ;
2584          ;
2585          ;
2586          ;
2587          ;
2588          ;
2589          ;
2590          ;
2591          ;
2592          ;
2593          ;
2594          ;
2595          ;
2596          ;
2597          ;
2598          ;
2599          ;
2600          ;
2601          ;
2602          ;
2603          ;
2604          ;
2605          ;
2606          ;
2607          ;
2608          ;
2609          ;
2610          ;
2611          ;
2612          ;
2613          ;
2614          ;
2615          ;
2616          ;
2617          ;
2618          ;
2619          ;
2620          ;
2621          ;
2622          ;
2623          ;
2624          ;
2625          ;
2626          ;
2627          ;
2628          ;
2629          ;
2630          ;
2631          ;
2632          ;
2633          ;
2634          ;
2635          ;
2636          ;
2637          ;
2638          ;
2639          ;
2640          ;
2641          ;
2642          ;
2643          ;
2644          ;
2645          ;
2646          ;
2647          ;
2648          ;
2649          ;
2650          ;
2651          ;
2652          ;
2653          ;
2654          ;
2655          ;
2656          ;
2657          ;
2658          ;
2659          ;
2660          ;
2661          ;
2662          ;
2663          ;
2664          ;
2665          ;
2666          ;
2667          ;
2668          ;
2669          ;
2670          ;
2671          ;
2672          ;
2673          ;
2674          ;
2675          ;
2676          ;
2677          ;
2678          ;
2679          ;
2680          ;
2681          ;
2682          ;
2683          ;
2684          ;
2685          ;
2686          ;
2687          ;
2688          ;
2689          ;
2690          ;
2691          ;
2692          ;
2693          ;
2694          ;
2695          ;
2696          ;
2697          ;
2698          ;
2699          ;
2700          ;
2701          ;
2702          ;
2703          ;
2704          ;
2705          ;
2706          ;
2707          ;
2708          ;
2709          ;
2710          ;
2711          ;
2712          ;
2713          ;
2714          ;
2715          ;
2716          ;
2717          ;
2718          ;
2719          ;
2720          ;
2721          ;
2722          ;
2723          ;
2724          ;
2725          ;
2726          ;
2727          ;
2728          ;
2729          ;
2730          ;
2731          ;
2732          ;
2733          ;
2734          ;
2735          ;
2736          ;
2737          ;
2738          ;
2739          ;
2740          ;
2741          ;
2742          ;
2743          ;
2744          ;
2745          ;
2746          ;
2747          ;
2748          ;
2749          ;
2750          ;
2751          ;
2752          ;
2753          ;
2754          ;
2755          ;
2756          ;
2757          ;
2758          ;
2759          ;
2760          ;
2761          ;
2762          ;
2763          ;
2764          ;
2765          ;
2766          ;
2767          ;
2768          ;
2769          ;
2770          ;
2771          ;
2772          ;
2773          ;
2774          ;
2775          ;
2776          ;
2777          ;
2778          ;
2779          ;
2780          ;
2781          ;
2782          ;
2783          ;
2784          ;
2785          ;
2786          ;
2787          ;
2788          ;
2789          ;
2790          ;
2791          ;
2792          ;
2793          ;
2794          ;
2795          ;
2796          ;
2797          ;
2798          ;
2799          ;
2800          ;
2801          ;
2802          ;
2803          ;
2804          ;
2805          ;
2806          ;
2807          ;
2808          ;
2809          ;
2810          ;
2811          ;
2812          ;
2813          ;
2814          ;
2815          ;
2816          ;
2817          ;
2818          ;
2819          ;
2820          ;
2821          ;
2822          ;
2823          ;
2824          ;
2825          ;
2826          ;
2827          ;
2828          ;
2829          ;
2830          ;
2831          ;
2832          ;
2833          ;
2834          ;
2835          ;
2836          ;
2837          ;
2838          ;
2839          ;
2840          ;
2841          ;
2842          ;
2843          ;
2844          ;
2845          ;
2846          ;
2847          ;
2848          ;
2849          ;
2850          ;
2851          ;
2852          ;
2853          ;
2854          ;
2855          ;
2856          ;
2857          ;
2858          ;
2859          ;
2860          ;
2861          ;
2862          ;
2863          ;
2864          ;
2865          ;
2866          ;
2867          ;
2868          ;
2869          ;
2870          ;
2871          ;
2872          ;
2873          ;
2874          ;
2875          ;
2876          ;
2877          ;
2878          ;
2879          ;
2880          ;
2881          ;
2882          ;
2883          ;
2884          ;
2885          ;
2886          ;
2887          ;
2888          ;
2889          ;
2890          ;
2891          ;
2892          ;
2893          ;
2894          ;
2895          ;
2896          ;
2897          ;
2898          ;
2899          ;
2900          ;
2901          ;
2902          ;
2903          ;
2904          ;
2905          ;
2906          ;
2907          ;
2908          ;
2909          ;
2910          ;
2911          ;
2912          ;
2913          ;
2914          ;
2915          ;
2916          ;
2917          ;
2918          ;
2919          ;
2920          ;
2921          ;
2922          ;
2923          ;
2924          ;
2925          ;
2926          ;
2927          ;
2928          ;
2929          ;
2930          ;
2931          ;
2932          ;
2933          ;
2934          ;
2935          ;
2936          ;
2937          ;
2938          ;
2939          ;
2940          ;
2941          ;
2942          ;
2943          ;
2944          ;
2945          ;
2946          ;
2947          ;
2948          ;
2949          ;
2950          ;
2951          ;
2952          ;
2953          ;
2954          ;
2955          ;
2956          ;
2957          ;
2958          ;
2959          ;
2960          ;
2961          ;
2962          ;
2963          ;
2964          ;
2965          ;
2966          ;
2967          ;
2968          ;
2969          ;
2970          ;
2971          ;
2972          ;
2973          ;
2974          ;
2975          ;
2976          ;
2977          ;
2978          ;
2979          ;
2980          ;
2981          ;
2982          ;
2983          ;
2984          ;
2985          ;
2986          ;
2987          ;
2988          ;
2989          ;
2990          ;
2991          ;
2992          ;
2993          ;
2994          ;
2995          ;
2996          ;
2997          ;
2998          ;
2999          ;
3000          ;
3001          ;
3002          ;
3003          ;
3004          ;
3005          ;
3006          ;
3007          ;
3008          ;
3009          ;
3010          ;
3011          ;
3012          ;
3013          ;
3014          ;
3015          ;
3016          ;
3017          ;
3018          ;
3019          ;
3020          ;
3021          ;
3022          ;
3023          ;
3024          ;
3025          ;
3026          ;
3027          ;
3028          ;
3029          ;
3030          ;
3031          ;
3032          ;
3033          ;
3034          ;
3035          ;
3036          ;
3037          ;
3038          ;
3039          ;
3040          ;
3041          ;
3042          ;
3043          ;
3044          ;
3045          ;
3046          ;
3047          ;
3048          ;
3049          ;
3050          ;
3051          ;
3052          ;
3053          ;
3054          ;
3055          ;
3056          ;
3057          ;
3058          ;
3059          ;
3060          ;
3061          ;
3062          ;
3063          ;
3064          ;
3065          ;
3066          ;
3067          ;
3068          ;
3069          ;
3070          ;
3071          ;
3072          ;
3073          ;
3074          ;
3075          ;
3076          ;
3077          ;
3078          ;
3079          ;
3080          ;
3081          ;
3082          ;
3083          ;
3084          ;
3085          ;
3086          ;
3087          ;
3088          ;
3089          ;
3090          ;
3091          ;
3092          ;
3093          ;
3094          ;
3095          ;
3096          ;
3097          ;
3098          ;
3099          ;
3100          ;
3101          ;
3102          ;
3103          ;
3104          ;
3105          ;
3106          ;
3107          ;
3108          ;
3109          ;
3110          ;
3111          ;
3112          ;
3113          ;
3114          ;
3115          ;
3116          ;
3117          ;
3118          ;
3119          ;
3120          ;
3121          ;
3122          ;
3123          ;
3124          ;
3125          ;
3126          ;
3127          ;
3128          ;
3129          ;
3130          ;
3131          ;
3132          ;
3133          ;
3134          ;
3135          ;
3136          ;
3137          ;
3138          ;
3139          ;
3140          ;
3141          ;
3142          ;
3143          ;
3144          ;
3145          ;
3146          ;
3147          ;
3148          ;
3149          ;
3150          ;
3151          ;
3152          ;
3153          ;
3154          ;
3155          ;
3156          ;
3157          ;
3158          ;
3159          ;
3160          ;
3161          ;
3162          ;
3163          ;
3164          ;
3165          ;
3166          ;
3167          ;
3168          ;
3169          ;
3170          ;
3171          ;
3172          ;
3173          ;
3174          ;
3175          ;
3176          ;
3177          ;
3178          ;
3179          ;
3180          ;
3181          ;
3182          ;
3183          ;
3184          ;
3185          ;
3186          ;
3187          ;
3188          ;
3189          ;
3190          ;
3191          ;
3192          ;
3193          ;
3194          ;
3195          ;
3196          ;
3197          ;
3198          ;
3199          ;
3200          ;
3201          ;
3202          ;
3203          ;
3204          ;
3205          ;
3206          ;
3207          ;
3208          ;
3209          ;
3210          ;
3211          ;
3212          ;
3213          ;
3214          ;
3215          ;
3216          ;
3217          ;
3218          ;
3219          ;
3220          ;
3221          ;
3222          ;
3223          ;
3224          ;
3225          ;
3226          ;
3227          ;
3228          ;
3229          ;
3230          ;
3231          ;
3232          ;
3233          ;
3234          ;
3235          ;
3236          ;
3237          ;
3238          ;
3239          ;
3240          ;
3241          ;
3242          ;
32
```

```

1772
1773 0847      NUN_ERR:
1774 0847 80 FC 01      CMP      AH,1          ; SET THE CARRY FLAG TO INDICATE
1775 084A F5           CMC              ; NO DOUBLE STEPPING REQUIRED
1776 084B C3          RET              ; SUCCESS OR FAILURE
1777 084C      SETUP_END      ENDP
1778
1779      ;-----
1780      ; SETUP_DBL
1781      ; ON ENTRY:    DI = DRIVE
1782      ; ON EXIT:    CY = 1 MEANS ERROR
1783      ;-----
1784
1785 084C      SETUP_DBL      PROC      NEAR
1786 084C F6 06 00BF R 01  TEST     #0H,CNTRL,DUAL      ; TEST CONTROLLER I.D.
1787 0851 74 65          JZ         NO_DBL      ; NO DOUBLE STEPPING REQUIRED
1788 0853 8A A5 0090 R   MOV      AH,#DSK_STATE[DI]    ; ACCESS STATE
1789 0857 F6 C4 10      TEST     AH,MED_DET    ; ESTABLISHED STATE ?
1790 085A 75 5C          JNZ        NO_DBL      ; IF ESTABLISHED THEN DOUBLE DONE
1791
1792      ;----- CHECK FOR TRACK 0 TO SPEED UP ACKNOWLEDGE OF UNFORMATTED DISKETTE
1793
1794 085C C6 06 003E R 00  MOV      #SEEK_STATUS,0      ; SET RECALIBRATE REQUIRED ON ALL DRIVES
1795 0861 E8 0913 R      CALL     MOTOR_ON      ; ENSURE MOTOR STAY ON
1796 0864 B5 00          MOV      CH,0      ; LOAD TRACK 0
1797 0866 E8 0A14 R      CALL     SEEK          ; SEEK TO TRACK 0
1798 0869 E8 08BA R      CALL     READ_ID      ; READ ID FUNCTION
1799 086C 72 35          JC         SD_ERR      ; IF ERROR NO TRACK 0
1800
1801      ;----- INITIALIZE START AND MAX TRACKS (TIMES 2 FOR BOTH HEADS)
1802
1803 086E B9 0450      MOV      CX,0450H      ; START, MAX TRACKS
1804 0871 F6 85 0090 R 01  TEST     #DSK_STATE[DI],TRK_CAPA ; TEST FOR 80 TRACK CAPABILITY
1805 0876 74 02          JZ         CNT_OK      ; IF NOT COUNT IS SETUP
1806 0878 B1 A0          MOV      CL,0A0H      ; MAXIMUM TRACK 1.2 MB
1807
1808      ; ATTEMPT READ ID OF ALL TRACKS, ALL HEADS UNTIL SUCCESS; UPON SUCCESS,
1809      ; MUST SEE IF ASKED FOR TRACK IN SINGLE STEP MODE = TRACK ID READ; IF NOT,
1810      ; THEN SET DOUBLE STEP ON.
1811
1812 087A      CNT_OK:
1813 087A 51      PUSH     CX          ; SAVE TRACK, COUNT
1814 087B C6 06 0041 R 00  MOV      #DSKETTE_STATUS,0      ; CLEAR STATUS, EXPECT ERRORS
1815 0880 33 C0          XOR      AX,AX      ; CLEAR AX
1816 0882 D0 ED          SHR      CH,1      ; HALVE TRACK, CY = HEAD
1817 0884 D0 D0          RCL      AL,1      ; AX = HEAD IN CORRECT BIT
1818 0886 D0 D0          RCL      AL,1
1819 0888 D0 D0          RCL      AL,1
1820 088A 50      PUSH     AX          ; SAVE HEAD
1821 088B E8 0A14 R      CALL     SEEK          ; SEEK TO TRACK
1822 088E 58          POP      AX          ; RESTORE HEAD
1823 088F 0B F8          OR      DI,AX      ; DI = HEAD OR'ED DRIVE
1824 0891 E8 08BA R      CALL     READ_ID      ; READ ID HEAD 0
1825 0894 9C          PUSHF      ; SAVE RETURN FROM READ_ID
1826 0895 81 E7 00FB      AND      DI,11111011B      ; TURN OFF HEAD 1 BIT
1827 0899 9D          POPF      ; RESTORE ERROR RETURN
1828 089A 59          POP      CX          ; RESTORE COUNT
1829 089B 73 08          JNC      DO_CHK      ; IF OK, ASKED = RETURNED TRACK ?
1830 089D FE C5          INC      CH      ; INC FOR NEXT TRACK
1831 089F 3A E9          CMP      CH,CL      ; REACHED MAXIMUM YET
1832 08A1 75 D7          JNZ      CNT_OK      ; CONTINUE TILL ALL TRIED
1833
1834      ;----- FALL THRU, READ ID FAILED FOR ALL TRACKS
1835
1836 08A3      SD_ERR:
1837 08A3 F9          STC              ; SET CARRY FOR ERROR
1838 08A4 C3          RET              ; SETUP_DBL ERROR EXIT
1839
1840 08A5      DO_CHK:
1841 08A5 8A 0E 0045 R   MOV      CL,#NEC_STATUS+3      ; LOAD RETURNED TRACK
1842 08A9 86 8D 0094 R   MOV      #DSK_TRK[DI],CL      ; STORE TRACK NUMBER
1843 08AB D0 ED          SHR      CH,1      ; HALVE TRACK
1844 08AF 3A E9          CMP      CH,CL      ; IS IT THE SAME AS ASKED FOR TRACK
1845 08B1 74 05          JZ         NO_DBL      ; IF SAME THEN NO DOUBLE STEP
1846 08B3 80 8D 0090 R 20  OR      #DSK_STATE[DI],DBL_STEP    ; TURN ON DOUBLE STEP REQUIRED
1847
1848 08B8      NO_DBL:
1849 08B8 F8          CLC              ; CLEAR ERROR FLAG
1850 08B9 C3          RET
1851 08BA      SETUP_DBL      ENDP
1852
1853      ;-----
1854      ; READ_ID
1855      ; ON ENTRY:    DI = BIT 2 = HEAD; BITS 1,0 = DRIVE
1856      ; ON EXIT:    DI = BIT 2 IS RESET, BITS 1,0 = DRIVE
1857      ;              #DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
1858      ;-----
1859
1860 08BA      READ_ID      PROC      NEAR
1861 08BA B8 08CE R      MOV      AX,OFFSET ER_3      ; MOVE NEC OUTPUT ERROR ADDRESS
1862 08BD 50      PUSH     AX
1863 08BE B4 4A          MOV      AH,4AH      ; READ ID COMMAND
1864 08C0 E8 09F0 R      CALL     NEC_OUTPUT      ; TO CONTROLLER
1865 08C3 8B C7          MOV      AX,DI      ; DRIVE # TO AH, HEAD 0
1866 08C5 8A E0          MOV      AH,AL
1867 08C7 E8 09F0 R      CALL     NEC_OUTPUT      ; TO CONTROLLER
1868 08CA E8 0727 R      CALL     NEC_TERM      ; WAIT FOR OPERATION, GET STATUS
1869 08CD 58          POP      AX          ; THROW AWAY ERROR ADDRESS
1870 08CE      ER_3:
1871 08CE C3          RET
1872 08CF      READ_ID      ENDP
1873
1874      ;-----
1875      ; CMOS_TYPE
1876      ; RETURNS DISKETTE TYPE FROM CMOS
1877      ; ON ENTRY:    DI = DRIVE #
1878      ; ON EXIT:    AL = TYPE; CY REFLECTS STATUS
1879      ;-----
1880
1881 08CF      CMOS_TYPE      PROC      NEAR
1882 08CF A0 0010 R      MOV      AL,BYTE PTR #EQUIP_FLAG ; LOAD EQUIPMENT FLAG FOR # DISKETTES
1883 08D2 24 C1          AND      AL,11000001B ; KEEP DISKETTE DRIVE BITS
1884 08D4 D0 E8          SHR      AL,1      ; ARE THERE ANY DRIVES INSTALLED?
1885 08D6 73 20          JNC      TYP_ZERO      ; NO-->NO DRIVES TYPE ZERO

```

```

1886 08D8 D0 C0          ROL    AL,1          ; ROTATE TO ORIGINAL POSITION
1887 08DA D0 C0          ROL    AL,1          ; ROTATE BITS 6 AND 7 TO 0 AND 1
1888 08DC D0 C0          ROL    AL,1
1889 08DE 32 E4          XOR     AH,AH          ; AX=NUMBER OF DRIVES
1890 08E0 3B C7          CMP     AX,D1          ; IS DRIVE REQUESTED PRESENT
1891 08E2 72 14          JC      TYP_ZERO      ; C->REQUESTED DRIVE NOT PRESENT
1892 08E4 F6 06 008F R 01 TEST    %HF_CNTRL,DUAL      ; TEST CONTROLLER I.O.
1893 08E9 75 10          JNZ     CR2
1894 08EB F6 85 0090 R 01 TEST    %DSK_STATE[D1],TRK_CAPA ; TEST FOR 80 TRACKS
1895 08FD B0 01          MOV     AL,1          ; DRIVE TYPE HAS 40 TRACKS
1896 08FF 74 06          JZ      CR1
1897 08F4 B0 03          MOV     AL,3          ; DRIVE TYPE HAS 80 TRACKS
1898 08F6 EB 02          JMP     SHORT CR1
1899 08F8
1900 08F8 32 C0          XOR     AL,AL          ; DRIVE TYPE 0
1901 08FA          CR1:      RET
1902 08FA C3          RET
1903 08FB          CR2:      STC
1904 08FB F9          JMP     CR1          ; EXIT WITH CARRY IF DUAL CARD
1905 08FC EB FC
1906 08FE
1907
1908          ; GET_PARM
1909          ; THIS ROUTINE FETCHES THE INDEXED POINTER FROM THE
1910          ; DISK BASE BLOCK POINTED TO BY THE DATA VARIABLE
1911          ; %DISK_POINTER, A BYTE FROM THAT TABLE IS THEN MOVED
1912          ; INTO AH, THE INDEX OF THAT BYTE BEING THE PARAMETER
1913          ; IN DL.
1914
1915          ; ON ENTRY: DL = INDEX OF BYTE TO BE FETCHED
1916
1917          ; ON EXIT: AH = THAT BYTE FROM BLOCK
1918          ; AL,DH DESTROYED
1919
1920 08FE          GET_PARM PROC NEAR
1921 08FE IE          PUSH    DS
1922 08FF 56          SUB     AX,AX          ; DS = 0 , BIOS DATA AREA
1923 0900 2B C0          MOV     DS,AX
1924 0902 8E D8          XCHG    DX,BX
1925 0904 87 D3          SUB     BH,BH
1926 0906 2A FF          ASSUME DS:ABSO
1927          LDS     SI,%DISK_POINTER ; POINT TO BLOCK
1928 0908 C5 36 0078 R  MOV     AH,[SI+BX] ; GET THE WORD
1929 090C 8A 20          XCHG    DX,BX ; RESTORE BX
1930 090E 87 D3          POP     SI
1931 0910 5E          POP     DS
1932 0911 IF
1933 0912 C3          RET
1934          ASSUME DS:DATA
1935 0913          GET_PARM ENDP
1936
1937          ; MOTOR_ON
1938          ; TURN MOTOR ON AND WAIT FOR MOTOR START UP TIME. THE %MOTOR_COUNT
1939          ; IS REPLACED WITH A SUFFICIENTLY HIGH NUMBER (0FFFH) TO ENSURE
1940          ; THAT THE MOTOR DOES NOT GO OFF DURING THE OPERATION. IF THE
1941          ; MOTOR NEEDED TO BE TURNED ON, THE MULTITASKING HOOK FUNCTION
1942          ; (AX=90FDH, INT 15H) IS CALLED TELLING THE OPERATING SYSTEM
1943          ; THAT THE BIOS IS ABOUT TO WAIT FOR MOTOR START UP. IF THIS
1944          ; FUNCTION RETURNS WITH CY = 1, IT MEANS THAT THE MINIMUM WAIT
1945          ; HAS BEEN COMPLETED. AT THIS POINT A CHECK IS MADE TO ENSURE
1946          ; THAT THE MOTOR WASN'T TURNED OFF BY THE TIMER. IF THE HOOK DID
1947          ; NOT WAIT, THE WAIT FUNCTION (AH=086H) IS CALLED TO WAIT THE
1948          ; PRESCRIBED AMOUNT OF TIME. IF THE CARRY FLAG IS SET ON RETURN,
1949          ; IT MEANS THAT THE FUNCTION IS IN USE AND DID NOT PERFORM THE
1950          ; WAIT. A TIMER ! WAIT LOOP WILL THEN DO THE WAIT.
1951
1952          ; ON ENTRY: DI = DRIVE #
1953
1954          ; ON EXIT: AX,CX,DX DESTROYED
1955
1956 0913          MOTOR_ON PROC NEAR
1957 0913 53          PUSH    BX
1958 0914 E8 095E R      CALL    TURN_ON ; SAVE REG.
1959 0917 72 43          JC      IF_CY_1_NO_WAIT ; TURN ON MOTOR
1960 0919 E8 0432 R      CALL    XLAT_OLD ; IF CY=1 NO WAIT
1961 091C B8 90FD      MOV     AX,090FDH ; TRANSLATE STATE TO COMPATIBLE MODE
1962 091F CD 15          INT     15H ; LOAD WAIT CODE & TYPE
1963 0921 9C          PUSHF ; TELL OPERATING SYSTEM ABOUT TO WAIT
1964 0922 E8 0404 R      CALL    XLAT_NEW ; SAVE CY FOR TEST
1965 0925 9D          POPF ; TRANSLATE STATE TO PRESENT ARCH.
1966 0926 73 05          JNC     M_WAIT ; RESTORE CY FOR TEST
1967 0928 E8 095E R      CALL    TURN_ON ; BYPASS LOOP IF OP SYSTEM HANDLED WAIT
1968 092B 72 2F          JC      MOT_TS_ON ; CHECK AGAIN IF MOTOR ON
1969          ; IF NO WAIT MEANS IT IS ON
1970 092D
1971 092D B2 0A          MOV     DL,10 ; GET THE MOTOR WAIT PARAMETER
1972 092F E8 08FE R      CALL    GET_PARM
1973 0932 8A CA          MOV     AL,AH ; AL = MOTOR WAIT PARAMETER
1974 0934 32 E4          XOR     AH,AH ; MOTOR WAIT PARAMETER
1975 0936 3C 08          CMP     AL,8 ; SEE IF AT LEAST A SECOND IS SPECIFIED
1976 0938 73 02          JAE     GP2 ; IF YES, CONTINUE
1977 093A B0 08          MOV     AL,8 ; ONE SECOND WAIT FOR MOTOR START UP
1978
1979          ;----- AX CONTAINS NUMBER OF 1/8 SECONDS (125000 MICROSECONDS) TO WAIT
1980
1981 093C 50          GP2:   PUSH    AX ; SAVE WAIT PARAMETER
1982 093D B4 F424      MOV     DX,62500 ; LOAD LARGEST POSSIBLE MULTIPLIER
1983 0940 F7 E2          MUL     DX ; MULTIPLY BY HALF OF WHAT'S NECESSARY
1984 0942 8B CA          MOV     CX,DX ; CX = HIGH WORD
1985 0944 8B D0          MOV     DX,AX ; DX,DX = 1/2 * (# OF MICROSECONDS)
1986 0946 F8          CLC ; CLEAR CARRY FOR ROTATE
1987 0947 D1 D2          RCL     DX,1 ; DOUBLE LOW WORD, CY CONTAINS OVERFLOW
1988 0949 D1 D1          RCL     CX,1 ; DOUBLE HI, INCLUDING LOW WORD OVERFLOW
1989 094B B4 86          MOV     AH,86H ; LOAD WAIT CODE
1990 094D CD 15          INT     15H ; PERFORM WAIT
1991 094F 58          POP     AX ; RESTORE WAIT PARAMETER
1992 0950 73 0A          JNC     MOT_IS_ON ; CY MEANS WAIT COULD NOT BE DONE
1993
1994          ;----- FOLLOWING LOOPS REQUIRED WHEN RTC WAIT FUNCTION IS ALREADY IN USE
1995
1996 0952          J13:      MOV     CX,8286 ; WAIT FOR 1/8 SECOND PER (AL)
1997 0952 B9 205E      CALL    WAITF ; COUNT FOR 1/8 SECOND AT 15.085737 US
1998 0955 E8 0000 E    DEC     AL ; GO TO FIXED WAIT ROUTINE
1999 0958 FE C8          DEC     AL ; DECREMENT TIME VALUE

```



```

2000 095A T5 F6                                JNZ     J13                                ; ARE WE DONE YET
2001
2002 095C                                MOT_IS_ON:
2003 095C 5B                                POP     BX                                ; RESTORE REG.
2004 095D C3                                RET
2005 095E                                MOTOR_ON ENDP
2006
2007                                ;-----
2008                                ; TURN_ON
2009                                ; TURN MOTOR ON AND RETURN WAIT STATE.
2010                                ;
2011                                ; ON ENTRY: DI = DRIVE #
2012                                ;
2013                                ; ON EXIT: CY = 0 MEANS WAIT REQUIRED
2014                                ;           CY = 1 MEANS NO WAIT REQUIRED
2015                                ;           AX,BX,CX,DX DESTROYED
2016                                ;-----
2017 095E 0B 0F                                TURN_ON PROC
2018 0960 8A CB                                MOV     BX,DI                                ; BX = DRIVE #
2019 0962 D0 C3                                MOV     CL,BL                                ; CL = DRIVE #
2020 0964 D0 C3                                ROL     BL,1                                ; BL = DRIVE SELECT
2021 0966 D0 C3                                ROL     BL,1
2022 0968 D0 C3                                ROL     BL,1
2023 096A FA                                CLI
2024 096B C6 06 0040 R FF                    MOV     BH,00000000H                        ; NO INTERRUPTS WHILE DETERMINING STATUS
2025 0970 A0 003F R                          MOV     AL,MOTOR_STATUS                    ; ENSURE MOTOR STAYS ON FOR OPERATION
2026 0973 24 30                                AND     AL,00110000B                      ; GET DIGITAL OUTPUT REGISTER REFLECTION
2027 0975 84 01                                MOV     AH,1                                ; KEEP ONLY DRIVE SELECT BITS
2028 0977 D2 E4                                SHL     AH,CL                                ; MASK FOR DETERMINING MOTOR BIT
2029                                ; AH = MOTOR ON, A=00000001, B=00000010
2030                                ;
2031                                ; AL = DRIVE SELECT FROM MOTOR_STATUS
2032                                ; BL = DRIVE SELECT
2033                                ; AH = MOTOR ON MASK DESIRED
2034 0979 3A C3                                CMP     AL,BL                                ; REQUESTED DRIVE ALREADY SELECTED ?
2035 097B 75 06                                JNZ     J13                                ; IF NOT SELECTED JUMP
2036 097D 84 26 003F R                        TEST     AH,MOTOR_STATUS                    ; TEST MOTOR ON BIT
2037 0981 75 31                                JNZ     NO_MOT_WAIT                        ; JUMP IF MOTOR ON AND SELECTED
2038
2039 0983                                TURN_IT_ON:
2040 0983 0A E3                                OR      AH,BL                                ; AH = DRIVE SELECT AND MOTOR ON
2041 0985 8A 3E 003F R                        MOV     BH,MOTOR_STATUS                    ; SAVE COPY OF MOTOR STATUS BEFORE
2042 0989 80 E7 0F                                AND     BH,00001111B                      ; KEEP ONLY MOTOR BITS
2043 098B 80 26 003F R C0                    AND     MOTOR_STATUS,11000000B            ; CLEAR OUT DRIVE SELECT AND MOTORS
2044 0991 08 26 003F R                        OR      MOTOR_STATUS,AH                    ; OR IN DRIVE SELECTED
2045 0995 A0 003F R                          MOV     AL,MOTOR_STATUS                    ; GET DIGITAL OUTPUT REGISTER REFLECTION
2046 0998 8A D8                                BL,AH                                        ; BL=MOTOR STATUS AFTER, BH=BEFORE
2047 099A 80 E3 0F                                AND     BL,00001111B                      ; KEEP ONLY MOTOR BITS
2048 099D FB                                STI
2049 099E 24 3F                                AND     AL,00111111B                      ; ENABLE INTERRUPTS AGAIN
2050 09A0 D0 C0                                ROL     AL,1                                ; STRIP AWAY UNWANTED BITS
2051 09A2 D0 C0                                ROL     AL,1                                ; PUT BITS IN DESIRED POSITIONS
2052 09A4 D0 C0                                ROL     AL,1
2053 09A6 D0 C0                                ROL     AL,1
2054 09A8 0C 0C                                OR      AL,00001100B                      ; NO RESET, ENABLE DMA/INTERRUPT
2055 09AA BA 03F2                             MOV     DX,03F2H                          ; SELECT DRIVE AND TURN ON MOTOR
2056 09AD EE                                OUT     DX,AL
2057 09AE 3A DF                                CMP     BL,BH
2058 09B0 74 02                                JZ      NO_MOT_WAIT                        ; NEW MOTOR TURNED ON ?
2059 09B2 FB                                CLC
2060 09B3 C3                                RET
2061
2062 09B4                                NO_MOT_WAIT:
2063 09B4 F9                                STC
2064 09B5 FB                                RET
2065 09B6 C3                                RET
2066 09B7                                TURN_ON ENDP
2067                                ;-----
2068                                ; HD_WAIT
2069                                ; WAIT FOR HEAD SETTLE TIME.
2070                                ;
2071                                ; ON ENTRY: DI = DRIVE #
2072                                ;
2073                                ; ON EXIT: AX,BX,CX,DX DESTROYED
2074                                ;-----
2075 09B7                                HD_WAIT PROC
2076 09B7 B2 09                                MOV     DL,9                                ; POINT TO HEAD SETTLE PARAMETER
2077 09B9 E8 08FE R                            CALL     GET_PARM                          ; GET PARAMETER
2078 09BC F6 06 003F R 80                    TEST     MOTOR_STATUS,10000000B            ; SEE IF A WRITE OPERATION
2079 09C1 74 09                                JZ      ISNT_WRITE                        ; IF NOT, DO NOT ENFORCE ANY VALUES
2080 09C3 80 FC 0F                                CMP     AH,15                              ; IS WAIT 15 MILLISECONDS OR GREATER?
2081 09C6 73 08                                JAE     DO_WAIT                           ; IF THERE DO NOT ENFORCE
2082 09C8 B4 0F                                MOV     AH,15                              ; HEAD SETTLE MINIMUM
2083 09CA EB 04                                JMP     SHORT DO_WAIT                     ; PERFORM WAIT
2084 09CC                                ISNT_WRITE:
2085 09CC 0A E4                                OR      AH,AH
2086 09CE 74 1F                                JZ      HW_DONE                           ; CHECK FOR WAIT TO BE ZERO
2087                                ; IF NOT WRITE AND 0 THEN EXIT
2088
2089                                ;----- AH CONTAINS NUMBER OF MILLISECONDS TO WAIT
2090
2090 09D0                                DO_WAIT:
2091 09D0 8A C4                                MOV     AL,AH                                ; AL = # MILLISECONDS
2092 09D2 32 E4                                XOR     AH,AH                                ; AX = # MILLISECONDS
2093 09D4 50                                PUSH    AX                                ; SAVE HEAD SETTLE PARAMETER
2094 09D6 BA 03EB                             MOV     DX,1000                            ; SET UP FOR MULTIPLY TO MICROSECONDS
2095 09D8 F7 E2                                MUL     DI
2096 09DA BB CA                                MOV     CX,DX                                ; CX,AX = # MICROSECONDS
2097 09DC BB D0                                MOV     DX,AX                                ; CX,DX = # MICROSECONDS
2098 09DE BA 66H                             MOV     AH,66H                            ; LOAD WAIT CODE
2099 09E0 CD 15                                INT     15H                                ; RESTORE HEAD SETTLE PARAMETER
2100 09E2 58                                POP     AX                                ; CHECK FOR EVENT WAIT ACTIVE
2101 09E3 73 0A                                JNC     HW_DONE
2102
2103 09E5                                J29:
2104 09E5 B9 0042                             MOV     CX,66                            ; 1 MILLISECOND LOOP
2105 09E8 EB 0000 E                           CALL     WAITF                            ; COUNT AT 15.085737 US PER COUNT
2106 09EB FE C9                             DEC     CX                                ; DELAY FOR 1 MILLISECOND
2107 09ED 75 F6                                JNZ     J29                                ; DECREMENT TO COUNT
2108 09EF                                ; DO AL MILLISECOND # OF TIMES
2109 09EF                                HW_DONE:
2110 09F0                                RET
2111                                HD_WAIT ENDP
2112                                ;-----
2113                                ; NEC_OUTPUT
2114                                ; THIS ROUTINE SENDS A BYTE TO THE NEC CONTROLLER AFTER
    
```

```
2114      ; TESTING FOR CORRECT DIRECTION AND CONTROLLER READY THIS ;
2115      ; ROUTINE WILL TIME OUT IF THE BYTE IS NOT ACCEPTED WITHIN ;
2116      ; A REASONABLE AMOUNT OF TIME, SETTING THE DISKETTE STATUS ;
2117      ; ON COMPLETION. ;
2118      ; ;
2119      ; ON ENTRY: ;
2120      ; AH = BYTE TO BE OUTPUT ;
2121      ; ON EXIT: ;
2122      ; CY = 0 SUCCESS ;
2123      ; CY = 1 FAILURE -- DISKETTE STATUS UPDATED ;
2124      ; IF A FAILURE HAS OCCURRED, THE RETURN IS MADE ;
2125      ; ONE LEVEL HIGHER THAN THE CALLER OF NEC_OUTPUT. ;
2126      ; THIS REMOVES THE REQUIREMENT OF TESTING AFTER ;
2127      ; EVERY CALL OF NEC_OUTPUT. ;
2128      ; AX,CX,DX DESTROYED ;
2129      ;-----;
2130 09F0      NEC_OUTPUT      PROC      NEAR
2131 09F0 53      PUSH      BX ; SAVE REG.
2132 09F1 BA 03F4 MOV      DX,03F4H ; STATUS PORT
2133 09F4 B3 02 MOV      BL,2 ; HIGH ORDER COUNTER
2134 09F6 33 C9 XOR      CX,CX ; COUNT FOR TIME OUT
2135
2136 09F8 EC      J23:      IN      AL,DX ; GET STATUS
2137 09F9 24 C0 AND      AL,11000000B ; KEEP STATUS AND DIRECTION
2138 09FB 3C 80 CMP      AL,10000000B ; STATUS AND DIRECTION 0 ?
2139 09FD 74 0F JZ      J23 ; STATUS AND DIRECTION OK
2140 09FF E2 F7 LOOP     J23 ; CONTINUE TILL CX EXHAUSTED
2141
2142 0A01 FE CB DEC      BL ; DECREMENT COUNTER
2143 0A03 75 F3 JNZ      J23 ; REPEAT TILL DELAY FINISHED, CX = 0
2144
2145      ;----- FALL THRU TO ERROR RETURN
2146
2147 0A05 80 0E 0041 R 80 OR      0DSKETTE_STATUS,TIME_OUT
2148
2149 0A0A 5B      POP      BX ; RESTORE REG.
2150
2151 0A0B 58      POP      AX ; DISCARD THE RETURN ADDRESS
2152 0A0C F9      STC ; INDICATE ERROR TO CALLER
2153 0A0D C3      RET
2154
2155      ;----- DIRECTION AND STATUS OK; OUTPUT BYTE
2156
2157 0A0E 8A C4 J2T:      MOV      AL,AH ; GET BYTE TO OUTPUT
2158 0A10 42      INC      DX ; DATA PORT = STATUS PORT + 1
2159 0A11 EE      OUT      DX,AL ; OUTPUT THE BYTE
2160
2161 0A12 5B      POP      BX ; RESTORE REG.
2162 0A13 C3      RET ; CY = 0 FROM TEST INSTRUCTION
2163 0A14
2164      NEC_OUTPUT      ENDP
2165      ;-----;
2166      ; SEEK ;
2167      ; THIS ROUTINE WILL MOVE THE HEAD ON THE NAMED DRIVE ;
2168      ; TO THE NAMED TRACK. IF THE DRIVE HAS NOT BEEN ACCESSSED ;
2169      ; SINCE THE DRIVE RESET COMMAND WAS ISSUED, THE DRIVE ;
2170      ; WILL BE RECALIBRATED. ;
2171      ; ;
2172      ; ON ENTRY: DI = DRIVE # ;
2173      ; CH = TRACK # ;
2174      ; ON EXIT: 0DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION. ;
2175      ; AX,BX,CX,DX DESTROYED ;
2176      ;-----;
2177 0A14      SEEK      PROC      NEAR
2178 0A14 8B DF MOV      BX,DI ; BX = DRIVE #
2179 0A16 BA 0A7B R MOV      DX,OFFSET NEC_ERR ; LOAD RETURN ADDRESS
2180 0A19 52      PUSH     DX ; ON STACK FOR NEC_OUTPUT ERROR
2181 0A1A 80 01 MOV      AL,1 ; ESTABLISH MASK FOR RECALIBRATE TEST
2182 0A1C 86 CB XCHG     CL,BL ; GET DRIVE VALUE INTO CL
2183 0A1E D2 C0 ROL      AL,CL ; SHIFT MASK BY THE DRIVE VALUE
2184 0A20 86 CB XCHG     CL,BL ; RECOVER TRACK VALUE
2185 0A22 84 06 TEST     AL,003E R ; TEST FOR RECALIBRATE REQUIRED
2186 0A26 75 21 JNZ      J28A ; JUMP IF RECALIBRATE NOT REQUIRED
2187
2188 0A28 08 06 OR      0DSKETTE_STATUS,AL ; TURN ON THE NO RECALIBRATE BIT IN FLAG
2189 0A2C E8 0A7C R CALL     RECAL ; RECALIBRATE DRIVE
2190 0A2F 73 0A JNC      AFT_RECAL ; RECALIBRATE DONE
2191
2192      ;----- ISSUE RECALIBRATE FOR 80 TRACK DISKETTES
2193
2194 0A31 C6 06 0041 R 00 MOV      0DSKETTE_STATUS,0 ; CLEAR OUT INVALID STATUS
2195 0A36 E8 0A7C R CALL     RECAL ; RECALIBRATE DRIVE
2196 0A39 72 3F JC      RB ; IF RECALIBRATE FAILS TWICE THEN ERROR
2197
2198 0A3B      AFT_RECAL:
2199 0A3B 83 FF 01 CMP      DI,1 ; IF REQUEST FOR DRIVE > 2
2200 0A3E 77 13 JA      R8 ; DO SEEK EVERY TIME
2201 0A40 C6 85 0094 R 00 MOV      0DSK_TRK[DI],0 ; SAVE NEW CYLINDER AS PRESENT POSITION
2202 0A45 0A ED CH      CH,CH ; CHECK FOR SEEK TO TRACK 0
2203 0A47 74 2C JZ      DO_WAIT ; HEAD SETTLE, CY = 0 IF JUMP
2204
2205      ;----- DRIVE IS IN SYNCHRONIZATION WITH CONTROLLER, SEEK TO TRACK
2206
2207 0A49      J28A:
2208 0A49 83 FF 01 CMP      DI,1 ; IF REQUEST FOR DRIVE > 2
2209 0A4C 77 13 JA      R8 ; DO SEEK EVERY TIME
2210 0A4E F6 85 0090 R 20 TEST     0DSK_STATE[DI],DBL_STEP ; CHECK FOR DOUBLE STEP REQUIRED
2211 0A53 74 02 JZ      R7 ; SINGLE STEP REQUIRED BYPASS DOUBLE
2212 0A55 D0 E5 SHL      CH,1 ; DOUBLE NUMBER OF STEP TO TAKE
2213
2214 0A57 3A AD 0094 R R7:      CMP      CH,0DSK_TRK[DI] ; SEE IF ALREADY AT THE DESIRED TRACK
2215 0A5B 74 1D JE      RB ; IF YES, DO NOT NEED TO SEEK
2216
2217 0A5D 88 AD 0094 R R8:      MOV      0DSK_TRK[DI],CH ; SAVE NEW CYLINDER AS PRESENT POSITION
2218 0A61
2219 0A61 51      PUSH     CX ; SAVE CYLINDER #
2220 0A62 84 0F MOV      AH,0FH ; SEEK COMMAND TO NEC
2221 0A64 E8 09F0 R CALL     NEC_OUTPUT
2222 0A67 8B DF MOV      BX,DI ; BX = DRIVE #
2223 0A69 8A E3 MOV      AH,BL ; OUTPUT DRIVE NUMBER
2224 0A6B E8 09F0 R CALL     NEC_OUTPUT
2225 0A6E 58      POP      AX ; RESTORE CYLINDER # FOR NEC_OUTPUT
2226 0A6F E8 09F0 R CALL     NEC_OUTPUT
2227 0A72 E8 0A93 R CALL     CHK_STAT_2 ; ENDING INTERRUPT AND SENSE STATUS
```

```

2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3039
3040
3041
3042
3043
3044
3045
3046
304
```

```

2342      ;----- WAIT FOR REQUEST FOR MASTER
2343
2344      R10:  MOV  BH,2           ; HIGH ORDER COUNTER
2345      XOR    CX,CX             ; COUNTER
2346      J39:   IN     AL,DX        ; WAIT FOR MASTER
2347      AND    AL,11000000B      ; GET STATUS
2348      CMP    AL,11000000B      ; KEEP ONLY STATUS AND DIRECTION
2349      JZ     J42                ; STATUS 1 AND DIRECTION 0 ?
2350      JZ     J42                ; STATUS AND DIRECTION OK
2351      LOOP   J39                ; LOOP TILL TIMEOUT
2352
2353      DEC    BH                 ; DECREMENT HIGH ORDER COUNTER
2354      JNZ    J39                ; REPEAT TILL DELAY DONE
2355
2356      OR     #DSKETTE_STATUS,TIME_OUT
2357      STC
2358      JMP    SHORT POPRES       ; SET ERROR RETURN
2359                                     ; POP REGISTERS AND RETURN
2360
2361      ;----- READ IN THE STATUS
2362
2363      J42:   INC    DX           ; POINT AT DATA PORT
2364      IN     AL,DX              ; GET THE DATA
2365      MOV    [DI],AL            ; STORE THE BYTE
2366      INC    DI                 ; INCREMENT THE POINTER
2367
2368      MOV    CX,2                ; MINIMUM 12 MICROSECONDS FOR NEC
2369      CALL   WAITF               ; WAIT 15 TO 30 MICROSECONDS
2370      DEC    DX                  ; POINT AT STATUS PORT
2371      IN     AL,DX              ; GET STATUS
2372      TEST   AL,00010000B        ; TEST FOR NEC STILL BUSY
2373      JZ     POPRES              ; RESULTS DONE ?
2374
2375      DEC    BL                 ; DECREMENT THE STATUS COUNTER
2376      JNZ    R10                ; GO BACK FOR MORE
2377      OR     #DSKETTE_STATUS,BAD_NEC
2378      STC
2379
2380      ;----- RESULT OPERATION IS DONE
2381
2382      POPRES:
2383      POP    DI                 ; RETURN WITH CARRY SET
2384      RET
2385
2386      RESULTS ENOP
2387
2388      ;----- READ_DSCHNG
2389      ; READS THE STATE OF THE DISK CHANGE LINE.
2390      ; ON ENTRY:  DI = DRIVE #
2391      ; ON EXIT:   DI = DRIVE #
2392      ;             ZF = 0 : DISK CHANGE LINE INACTIVE
2393      ;             ZF = 1 : DISK CHANGE LINE ACTIVE
2394      ;             AX,CX,DX DESTROYED
2395
2396      READ_DSCHNG PROC NEAR
2397      CALL   MOTOR_ON           ; TURN ON THE MOTOR IF OFF
2398      MOV    DX,03F7H           ; ADDRESS DIGITAL INPUT REGISTER
2399      IN     AL,DX              ; INPUT DIGITAL INPUT REGISTER
2400      TEST   AL,DSK_CHG         ; CHECK FOR DISK CHANGE LINE ACTIVE
2401      RET
2402      READ_DSCHNG ENDP
2403
2404      ;----- DRIVE_DET
2405      ; DETERMINES WHETHER DRIVE IS 80 OR 40 TRACKS AND
2406      ; UPDATES STATE INFORMATION ACCORDINGLY.
2407      ; ON ENTRY:  DI = DRIVE #
2408
2409      DRIVE_DET PROC NEAR
2410      CALL   MOTOR_ON           ; TURN ON MOTOR IF NOT ALREADY ON
2411      CALL   RECAL              ; RECALIBRATE DRIVE
2412      JC     DD_BAC             ; ASSUME NO DRIVE PRESENT
2413      MOV    CH,TRK_SLAP        ; SEEK TO TRACK 48
2414      CALL   SEEK               ; SEEK TO TRACK 10
2415      JC     DD_BAC             ; ERROR NO DRIVE
2416      MOV    CH,QUIET_SEEK+1    ; SEEK TO TRACK 10
2417
2418      SK_GIN: DEC    CH           ; DECREMENT TO NEXT TRACK
2419      JS     IS_40              ; END LOOP IF CYLINDER COUNT NEGATIVE
2420      PUSH   CX                 ; SAVE TRACK
2421      CALL   SEEK               ; SEEK
2422      JC     POP_BAC            ; POP AND RETURN
2423      MOV    AX,OFFSET DD_BAC   ; LOAD NEC OUTPUT ERROR ADDRESS
2424      PUSH   AX                 ;
2425      MOV    AH,SENSE_DRV_ST    ; SENSE DRIVE STATUS COMMAND BYTE
2426      CALL   NEC_OUTPUT         ; OUTPUT TO NEC
2427      MOV    AX,DI              ; AL = DRIVE
2428      MOV    AH,AL              ; AH = DRIVE
2429      CALL   NEC_OUTPUT         ; OUTPUT TO NEC
2430      CALL   RESULTS            ; GO GET STATUS
2431      POP    AX                 ; THROW AWAY ERROR ADDRESS
2432      POP    CX                 ; RESTORE TRACK
2433      TEST   #NEC_STATUS,HOME   ; TRACK 0 ?
2434      JZ     SK_GIN            ; GO TILL TRACK 0
2435      OR     SK_CH              ; IS HOME AT TRACK 0 ?
2436      JZ     IS_80              ; MUST BE 80 TRACK DRIVE
2437
2438      ; DRIVE IS A 360i SET DRIVE TO DETERMINED;
2439      ; SET MEDIA TO DETERMINED AT RATE 250.
2440      IS_40: OR     #DSK_STATE[DI],DRV_DET+MED_DET+RATE_250
2441      RET
2442
2443      IS_80: OR     #DSK_STATE[DI],TRK_CAPA ; SETUP 80 TRACK CAPABILITY
2444      DD_BAC: RET
2445
2446      POP_BAC:
2447      POP    CX                 ; THROW AWAY
2448      RET
2449
2450      DRIVE_DET ENDP

```

```

2456 ;-----
2457 ; DISK_INT
2458 ; THIS ROUTINE HANDLES THE DISKETTE INTERRUPT.
2459 ;
2460 ; ON EXIT: THE INTERRUPT FLAG IS SET IN *SEEK STATUS.
2461 ;-----
2462 0B74 DISK_INT PROC FAR ; ENTRY POINT FOR ORG 0EF57H
2463 0B74 FB ; RE-ENABLE INTERRUPTS
2464 0B75 50 ; SAVE WORK REGISTER
2465 0B76 1E ; SAVE REGISTERS
2466 0B77 E8 0000 E ; SETUP DATA ADDRESSING
2467 0B7A 80 0E 003E R 80 ; TURN ON INTERRUPT OCCURRED
2468 0B7F 1F ; RESTORE USER (DS)
2469 0B80 80 20 ; END OF INTERRUPT MARKER
2470 0B82 E6 20 ; INTERRUPT CONTROL PORT
2471 0B84 88 9101 ; INTERRUPT POST CODE AND TYPE
2472 0B87 CD 15 ; GO PERFORM OTHER TASK
2473 0B89 58 ; RECOVER REGISTER
2474 0B8A CF ; RETURN FROM INTERRUPT
2475 0B8B ;
2476 DISK_INT ENDP
2477 ;-----
2478 ; DSKETTE SETUP
2479 ; THIS ROUTINE DOES A PRELIMINARY CHECK TO SEE WHAT TYPE
2480 ; OF DISKETTE DRIVES ARE ATTACHED TO THE SYSTEM.
2481 ;-----
2481 0B8B DSKETTE_SETUP PROC NEAR
2482 0B8B 50 ; SAVE REGISTERS
2483 0B8C 53 ;
2484 0B8D 51 ;
2485 0B8E 52 ;
2486 0B8F 57 ;
2487 0B90 56 ;
2488 0B91 1E ;
2489 0B92 E8 0000 E ; CALL DDS
2490 0B95 80 0E 00A0 R 01 ; *RTC_WAIT_FLAG,01 ; POINT DATA SEGMENT TO BIOS DATA AREA
2491 0B9A C7 06 0090 R 0000 ; WORD_PTR *DSK_STATE,0 ; NO RTC WAIT, FORCE USE OF LOOP
2492 0BA0 80 26 008B R 33 ; AND *LAstrate,NOT *STRT_MSK ; SEND MSK ; CLEAR START & SEND
2493 0BA5 80 0E 008B R C0 ; OR *LAstrate,SEND_MSK ; INITIALIZE SENT TO IMPOSSIBLE
2494 0BA6 C6 06 003E R 00 ; MOV *SEEK_STATUS,0 ; INDICATE RECALIBRATE NEEDED
2495 0BAF C6 06 0040 R 00 ; MOV *MOTOR_COUNT,0 ; INITIALIZE MOTOR COUNT
2496 0BB4 C6 06 003F R 00 ; MOV *MOTOR_STATUS,0 ; INITIALIZE DRIVES TO OFF STATE
2497 0BB9 C6 06 0041 R 00 ; MOV *DSKETTE_STATUS,0 ; NO ERRORS
2498 0BBE A0 0010 R ; MOV AL,BYTE PTR *EQUIP_FLAG ; GET EQUIPMENT STATUS
2499 0BC1 D0 C0 ; ROL AL,1 ; SHIFT BITS 7,6 TO 1,0
2500 0BC3 D0 C0 ; ROL AL,1
2501 0BC5 24 03 ; AND AL,3 ; MASK DRIVE BITS
2502 0BC7 32 E4 ; XOR AH,AH ; AX=NUMBER OF DRIVES(RELATIVE ZERO)
2503 0BC9 33 FF ; XOR DI,DI ; DI=INITIAL DRIVE TO BE ESTABLISHED
2504 0BCB BE 0010 ; MOV SI,HOMES ; SI=HOMES MASK FOR ALL DRIVES
2505 0BCE ;
2506 0BCE F6 06 008F R 01 ; SUP0: TEST *HF_CNTRL,DUAL ; TEST CONTROLLER TYPE
2507 0BD3 75 05 ; JNZ SUP1
2508 0BD5 C6 85 0090 R 94 ; MOV *DSK_STATE[DI],DRV_DET+MED_DET+RATE_250
2509 0BDA ;
2510 0BDA 50 ; SUP1: PUSH AX ; SAVE DRIVE COUNT
2511 0BDB E8 0B2B R ; CALL DRIVE_DET ; DETERMINE DRIVE
2512 0BDE E8 0432 R ; CALL XLAT_OLD ; TRANSLATE STATE TO COMPATIBLE MODE
2513 0BE1 23 36 0042 R ; AND SI,WORD PTR *NEC_STATUS ; AND *NEC_STATUS WITH HOMES MASK
2514 0BE5 58 ; POP AX ; RESTORE DRIVE COUNT
2515 0BE6 47 ; INC DI ; POINT TO NEXT DRIVE
2516 0BE7 3B F8 ; CMP DI,AX ;
2517 0BE9 76 E3 ; JNA SUP2 ; REPEAT FOR EACH DRIVE
2518 0BEB ;
2519 0BEB C6 06 003E R 00 ; MOV *SEEK_STATUS,0 ; FORCE RECALIBRATE
2520 0BF0 80 26 00A0 R FE ; AND *RTC_WAIT_FLAG,0FEH ; ALLOW FOR RTC WAIT
2521 0BF5 E8 0832 R ; CALL SETUP_END ; VARIOUS CLEANUPS
2522 0BF8 72 05 ; JC HOME_OK ; EXIT WITH CY FLAG FROM SETUP_END
2523 0BFA 0B F6 ; OR SI,SI ; TEST HOME INDICATORS FOR ALL DRIVES
2524 0BFC 75 01 ; JNZ HOME_OK
2525 0BFE F9 ; STC ; ERROR-->HOME INDICATOR BAD
2526 0BFF ;
2527 0BFF 1F ; SUP2: POP DS ; RESTORE CALLERS REGISTERS
2528 0C00 5E ; POP SI
2529 0C01 5F ; POP DI
2530 0C02 5A ; POP DX
2531 0C03 59 ; POP CX
2532 0C04 5B ; POP BX
2533 0C05 58 ; POP AX
2534 0C06 C3 ; RET
2535 0C07 ;
2536 0C07 DSKETTE_SETUP ENDP
2537 CODE ENDS
END

```

```

1 PAGE 118,121
2 TITLE KEYBRD --- 01/10/86 KEYBOARD ADAPTER BIOS
3 ----- INT 16 -----
4 ; KEYBOARD I/O
5 ; THESE ROUTINES PROVIDE KEYBOARD SUPPORT
6 ;
7 ; INPUT
8 (AH)=0 READ THE NEXT ASCII CHARACTER STRUCK FROM THE KEYBOARD
9 RETURN THE RESULT IN (AL), SCAN CODE IN (AH)
10 ; (AH)=1 SET THE Z FLAG TO INDICATE IF AN ASCII CHARACTER IS
11 ; AVAILABLE TO BE READ.
12 ; (ZF)=1 -- NO CODE AVAILABLE
13 ; (ZF)=0 -- CODE IS AVAILABLE
14 ; IF ZF = 0, THE NEXT CHARACTER IN THE BUFFER TO BE READ
15 ; IS IN AX, AND THE ENTRY REMAINS IN THE BUFFER
16 ; (AH)=2 RETURN THE CURRENT SHIFT STATUS IN AL REGISTER
17 ; THE BIT SETTINGS FOR THIS CODE ARE INDICATED IN THE
18 ; THE EQUATES FOR *KB_FLAG
19 ; (AH)=5 PLACE ASCII CHARACTER/SCAN CODE COMBINATION IN KEYBOARD
20 ; BUFFER AS IF STRUCK FROM KEYBOARD
21 ;
22 ENTRY: (CL) = ASCII CHARACTER
23 (CH) = SCAN CODE
24 ;
25 EXIT: (AL) = 00H = SUCCESSFUL OPERATION
26 (AL) = 01H = UNSUCCESSFUL - BUFFER FULL
27 ;
28 ;
29 (AH)=10H EXTENDED READ INTERFACE FOR THE ENHANCED KEYBOARD
30 (AH)=11H EXTENDED ASCII STATUS FOR THE ENHANCED KEYBOARD,
31 OTHERWISE SAME AS FUNCTION AH=1
32 (AH)=12H RETURN THE EXTENDED SHIFT STATUS IN AX REGISTER
33 AL = BITS FROM *KB_FLAG, AH = BITS FOR LEFT AND RIGHT
34 CTL AND ALT KEYS FROM *KB_FLAG_1 AND *KB_FLAG_3
35 ;
36 ;
37 ;
38 ;
39 ;
40 ;
41 ;
42 ;
43 ;
44 ;
45 ;
46 ;
47 ;
48 ;
49 ;
50 ;
51 ;
52 ;
53 ;
54 ;
55 ;
56 ;
57 ;
58 ;
59 ;
60 ;
61 ;
62 ;
63 ;
64 ;
65 ;
66 ;
67 ;
68 ;
69 ;
70 ;
71 ;
72 ;
73 ;
74 ;
75 ;
76 ;
77 ;
78 ;
79 ;
80 ;
81 ;
82 ;
83 ;
84 ;
85 ;
86 ;
87 ;
88 ;
89 ;
90 ;
91 ;
92 ;
93 ;
94 ;
95 ;
96 ;
97 ;
98 ;
99 ;
100 ;
101 ;
102 ;
103 ;
104 ;
105 ;
106 ;
107 ;
108 ;
109 ;
110 ;
111 ;
112 ;
113 ;
114 ;

```

[7] [6] [5] [4] [3] [2] [1] [0] AH REGISTER
 [7] [6] [5] [4] [3] [2] [1] [0] AL 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

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

```

67 EXTRN DDS:NEAR
68 EXTRN RESET:NEAR
69 EXTRN BEEP:NEAR
70
71 PUBLIC KEYBOARD_IO_1
72 PUBLIC KB_INT_1
73
74 .LIST
75
76 0000 CODE SEGMENT BYTE PUBLIC
77 ASSUME CS:CODE,DS:DATA
78 KEYBOARD_IO_1 PROC FAR ;
79 0000 FB STI ; INTERRUPTS BACK ON
80 0001 IE PUSH DS ; SAVE CURRENT DS
81 0002 53 PUSH BX ; SAVE BX TEMPORARILY
82 0003 51 PUSH CX
83 0004 E8 0000 E CALL DDS ; ESTABLISH POINTER TO DATA REGION
84 0007 0A E4 OR AH,AH ; AH=0
85 0009 74 26 JZ K1 ; ASCII_READ
86 000B FE CC DEC AH ; AH=1
87 000D 74 37 JZ K2 ; ASCII_STATUS
88 000F FE CC DEC AH ; AH=2
89 0011 74 64 JZ K3 ; SHIFT_STATUS
90 0013 80 EC 03 SUB AH,3 ; AH=5
91 0016 74 64 JZ K500 ; KEYBOARD_WRITE
92 0018 80 EC 0B SUB AH,0BH ; AH=10
93 001B 74 0C JZ K1E ; EXTENDED_ASCII_READ
94 001D FE CC DEC AH ; AH=11
95 001F 74 1A JZ K2E ; EXTENDED_ASCII_STATUS
96 0021 FE CC DEC AH ; AH=12
97 0023 74 39 JZ K3E ; EXTENDED_SHIFT_STATUS
98 0025 K1O_EXIT:
99 0025 59 POP CX
100 0026 5B POP BX ; RECOVER REGISTER
101 0027 1F POP DS ; RECOVER SEGMENT
102 0028 CF IRET ; INVALID COMMAND
103
104 ;----- ASCII CHARACTER
105
106 0029 E8 009E R K1E: CALL K1S ; GET A CHARACTER FROM THE BUFFER (EXTENDED)
107 002C EB 00D1 R CALL K1O_E_XLAT ; ROUTINE TO XLATE FOR EXTENDED CALLS
108 002F EB F4 JMP K1O_EXIT ; GIVE IT TO THE CALLER
109
110 0031 E8 009E R K1: CALL K1S ; GET A CHARACTER FROM THE BUFFER
111 0034 E8 00DC R CALL K1O_S_XLAT ; ROUTINE TO XLATE FOR STANDARD CALLS
112 0037 72 F8 JC K1 ; CARRY SET MEANS THROW CODE AWAY
113 0039 EB EA JMP K1O_EXIT
114

```

```

115          ;----- ASCII STATUS
116
117          K2E: CALL K25          ; TEST FOR CHARACTER IN BUFFER (EXTENDED)
118              JZ K2B            ; RETURN IF BUFFER EMPTY
119              PUSHF             ; SAVE ZF FROM TEST
120              CALL K10_E_XLAT   ; ROUTINE TO XLATE FOR EXTENDED CALLS
121              JMP SHORT K2A     ; GIVE IT TO THE CALLER
122
123          K2: CALL K25          ; TEST FOR CHARACTER IN BUFFER
124              JZ K2B            ; RETURN IF BUFFER EMPTY
125              PUSHF             ; SAVE ZF FROM TEST
126              CALL K10_E_XLAT   ; ROUTINE TO XLATE FOR STANDARD CALLS
127              JNC K2A          ; CARRY CLEAR MEANS PASS VALID CODE
128              POPF             ; INVALID CODE FOR THIS TYPE OF CALL
129              CALL K15         ; THROW THE CHARACTER AWAY
130              JMP K2           ; GO LOOK FOR NEXT CHAR, IF ANY
131
132          K2A: POPF             ; RESTORE ZF FROM TEST
133          K2B: POP CX           ;
134              POP BX           ; RECOVER REGISTER
135              POP DS           ; RECOVER SEGMENT
136              RET 2            ; THROW AWAY FLAGS
137
138          ;----- SHIFT STATUS
139
140          K3E: MOV AH,0KB_FLAG_1 ; GET THE EXTENDED SHIFT STATUS FLAGS
141              AND AH,SYS_SHIFT_1 ; GET SYSTEM SHIFT KEY STATUS
142              MOV CL,5          ; MASK ALL BUT SYS KEY BIT
143              SHL AH,CL         ; SHIFT THE SYSTEM KEY BIT OVER TO
144                              ; BIT 7 POSITION
145              MOV AL,0KB_FLAG_1 ; GET SHIFT STATES BACK
146              AND AL,011T0011B ; ELIMINATE SYS_SHIFT, HOLD_STATE, AND INS_SHIFT
147              OR AH,AL         ; MERGE THE REMAINING BITS INTO AH
148              MOV AL,0KB_FLAG_3 ; GET RIGHT CTL AND ALT
149              AND AL,00001100B ; ELIMINATE LC ED AND LC E1
150              OR AH,AL         ; OR THE SHIFT FLAGS TOGETHER
151              MOV AL,0KB_FLAG   ; GET THE SHIFT STATUS FLAGS
152              JMP K10_EXIT      ; RETURN TO CALLER
153
154          ;----- WRITE TO KEYBOARD BUFFER
155
156          K500: PUSH SI
157              CLT
158              MOV SI,BX         ; GET THE "IN TO" POINTER TO THE BUFFER
159              MOV SI,BX         ; SAVE A COPY IN CASE BUFFER NOT FULL
160              CALL K4           ; BUMP THE POINTER TO SEE IF BUFFER IS FULL
161              CMP BX,[0BUFFER_HEAD] ; WILL THE BUFFER OVERRUN IF WE STORE THIS?
162              JE K502          ; YES - INFORM CALLER OF ERROR
163              MOV [SI],CX       ; NO - PUT THE ASCII/SCAN CODE INTO BUFFER
164              MOV [0BUFFER_TAIL],BX ; ADJUST "IN TO" POINTER TO REFLECT CHANGE
165              SUB AL,AL         ; TELL CALLER THAT OPERATION WAS SUCCESSFUL
166              JMP K504         ; SUB INSTRUCTION ALSO RESETS CARRY FLAG
167
168          K502: MOV AL,01H      ; BUFFER FULL INDICATION
169
170          K504: STI
171              POP SI
172              JMP K10_EXIT      ; RETURN TO CALLER WITH STATUS IN AL
173
174          KEYBOARD_IO_1 ENDP
175
176          009E

```

```

176      PAGE
177      ;----- READ THE KEY TO FIGURE OUT WHAT TO DO -----
178
179      K1S  PROC    NEAR
180      MOV    BX,0BUFFER_HEAD ; GET POINTER TO HEAD OF BUFFER
181      CMP    BX,0BUFFER_TAIL ; TEST END OF BUFFER
182      JNE    KIT         ; IF ANYTHING IN BUFFER DONT DO INTERRUPT
183
184      MOV    AX,09002H      ; MOVE IN WAIT CODE & TYPE
185      INT    15H           ; PERFORM OTHER FUNCTION
186
187      KIT:  STI           ; ASCII READ
188      NOP           ; INTERRUPTS BACK ON DURING LOOP
189      CLI           ; ALLOW AN INTERRUPT TO OCCUR
190      MOV    BX,0BUFFER_HEAD ; INTERRUPTS BACK OFF
191      CMP    BX,0BUFFER_TAIL ; GET POINTER TO HEAD OF BUFFER
192      JNE    KIT         ; TEST END OF BUFFER
193      MOV    AX,[BX]      ; LOOP UNTIL SOMETHING IN BUFFER
194      CALL   K4           ; GET SCAN CODE AND ASCII CODE
195      MOV    0BUFFER_HEAD,BX ; MOVE POINTER TO NEXT POSITION
196      RET              ; STORE VALUE IN VARIABLE
197
198      K1S  ENDP
199
200      ;----- READ THE KEY TO SEE IF ONE IS PRESENT -----
201
202      K2S  PROC    NEAR
203      CLI           ; INTERRUPTS OFF
204      MOV    BX,0BUFFER_HEAD ; GET HEAD POINTER
205      CMP    BX,0BUFFER_TAIL ; IF EQUAL (Z=1) THEN NOTHING THERE
206      MOV    AX,[BX]      ; INTERRUPTS BACK ON
207      STI           ; RETURN
208      RET
209
210      K2S  ENDP
211
212      ;----- ROUTINE TO TRANSLATE SCAN CODE PAIRS FOR EXTENDED CALLS
213
214      K1O_E_XLAT:
215      CMP    AL,0F0h      ; IS IT ONE OF THE FILL-INs?
216      JNE    K1O_E_RET   ; NO, PASS IT ON
217      OR     AH,AH        ; AH = 0 IS SPECIAL CASE
218      JZ     K1O_E_RET   ; PASS THIS ON UNCHANGED
219      XOR    AL,AL       ; OTHERWISE SET AL = 0
220
221      K1O_E_RET:
222      RET              ; GO BACK
223
224      ;----- ROUTINE TO TRANSLATE SCAN CODE PAIRS FOR STANDARD CALLS
225
226      K1O_S_XLAT:
227      CMP    AH,0E0h      ; IS IT KEYPAD ENTER OR / ?
228      JNE    K1O_S2      ; NO, CONTINUE
229      CMP    AL,00h      ; KEYPAD ENTER CODE?
230      JE     K1O_S1      ; YES, MESSAGE A BIT
231      CMP    AL,0Ah      ; CTRL KEYPAD ENTER CODE?
232      JE     K1O_S1      ; YES, MESSAGE THE SAME
233      MOV    AH,55h      ; NO, MUST BE KEYPAD /
234      JMP    K1O_USE     ; GIVE TO CALLER
235
236      K1O_S1: MOV    AH,Tch ; CONVERT TO COMPATIBLE OUTPUT
237      JMP    K1O_USE     ; GIVE TO CALLER
238
239      K1O_S2: CMP    AH,84h ; IS IT ONE OF THE EXTENDED ONES?
240      JA     K1O_DIS     ; YES, THROW AWAY AND GET ANOTHER CHAR
241
242      CMP    AL,0F0h      ; IS IT ONE OF THE FILL-INs?
243      JNE    K1O_S3      ; NO, TRY LAST TEST
244      OR     AH,AH        ; AH = 0 IS SPECIAL CASE
245      JZ     K1O_USE     ; PASS THIS ON UNCHANGED
246      JMP    K1O_DIS     ; THROW AWAY THE REST
247
248      K1O_S3: CMP    AL,0E0h ; IS IT AN EXTENSION OF A PREVIOUS ONE?
249      JNE    K1O_USE     ; NO, MUST BE A STANDARD CODE
250      OR     AH,AH        ; AH = 0 IS SPECIAL CASE
251      JZ     K1O_USE     ; JUMP IF AH = 0
252      XOR    AL,AL       ; CONVERT TO COMPATIBLE OUTPUT
253      JMP    K1O_USE     ; PASS IT ON TO CALLER
254
255      K1O_USE:
256      CLC              ; CLEAR CARRY TO INDICATE GOOD CODE
257      RET              ; RETURN
258
259      K1O_DIS:
260      STC              ; SET CARRY TO INDICATE DISCARD CODE
261      RET              ; RETURN

```



```

260
261
262
263
264
265 0114
266 0114 43
267 0115 43
268
269 0116 3B 1E 0082 R
270 011A 72 04
271 011C 8B 1E 0080 R
272 0120 C3
273 0121
274
275
276
277
278 0121
279 0121 50
280 0122 53
281 0123 51
282 0124 52
283 0125 56
284 0126 57
285 0127 1E
286 0128 06
287 0129 FC
288 012A E8 0000 E
289 012D E4 60
290 012F 93
291
292
293
294
295 0130 E4 61
296 0132 8A E0
297 0134 0C 80
298 0136 E6 61
299 0138 86 E0
300 013A E6 61
301 013C FB
302 013D 93
303
304
305
306 013E B4 4F
307 0140 F9
308 0141 CD 15
309
310 0143 72 03
311 0145 E9 02CA R
312
313 0148
314 0148 8A E0
315
316
317
318 014A 3C FF
319 014C 75 03
320 014E E9 0540 R
321
322 0151 0E
323 0152 07
324 0153 8A 3E 0096 R
325
326 0157
327 0157 3C E0
328 0159 75 07
329 015B 80 0E 0096 R 12
330 0160 EB 09
331
332 0162
333 0162 3C E1
334 0164 75 08
335 0166 80 0E 0096 R 11
336 016B E9 02CF R
337
338 016E
339 016E 24 7F
340 0170 F6 C7 02
341 0173 74 0C
342
343 0175 B9 0002
344 017B BF 0555 R
345 017B F2/ AE
346 017D 75 54
347 017F EB 3D
348
349 0181
350 0181 F6 C7 01
351 0184 74 16
352
353 0186 B9 0004
354 0189 BF 0553 R
355 018C F2/ AE
356 018E 74 DB
357
358 0190 3C 45
359 0192 75 2A
360 0194 F6 C4 80
361 0197 75 25
362 0199 E9 03FF R

```

PAGE

INCREMENT BUFFER POINTER ROUTINE

```

K4 PROC NEAR
INC BX
INC BX ; MOVE TO NEXT WORD IN LIST
K5: CMP BX,0BUFFER_END ; AT END OF BUFFER?
JB K5 ; NO, CONTINUE
MOV BX,0BUFFER_START ; YES, RESET TO BUFFER BEGINNING
K4: RET
ENDP

```

----- KEYBOARD INTERRUPT ROUTINE

```

KB_INT_1 PROC FAR
PUSH AX ; SAVE THE STI UNTIL AFTER KEYBOARD RESET
PUSH BX
PUSH CX
PUSH DX
PUSH SI
PUSH DI
PUSH DS
PUSH ES
CALL CLD ; FORWARD DIRECTION
IN AL,KB_DATA ; SET UP ADDRESSING TO DATA SEGMENT
XCHG BX,AX ; READ IN THE CHARACTER
; SAVE IT
;----- RESET THE SHIFT REGISTER ON THE PLANAR IF ENABLED, OR DO NOTHING IF IT IS DISABLED
IN AL,KB_CTL ; GET THE CONTROL PORT
MOV AH,AL ; SAVE VALUE
OR AL,80H ; RESET BIT FOR KEYBOARD
OUT KB_CTL,AL
XCHG AH,AL ; GET BACK ORIGINAL CONTROL
OUT KB_CTL,AL ; KB HAS BEEN RESET
STI
XCHG AX,BX ; RESTORE DATA IN
;----- SYSTEM HOOK INT 15H - FUNCTION 4FH (ON HARDWARE INTERRUPT LEVEL 9H)
MOV AH,04FH ; SYSTEM INTERCEPT - KEY CODE FUNCTION
STC ; SET CY=1 (IN CASE OF IRET)
INT 15H ; CASSETTE CALL (AL)= KEY SCAN CODE
; RETURNS CY=1 FOR INVALID FUNCTION
; CONTINUE IF CARRY FLAG SET ((AL)=CODE)
; EXIT IF SYSTEM HANDLED SCAN CODE
; EXIT HANDLES HARDWARE EOI AND ENABLE
KB_INT_PC: MOV AH,AL ; SAVE SCAN CODE IN AH ALSO
;----- TEST FOR OVERRUN SCAN CODE FROM KEYBOARD
CMP AL,OFFH ; IS THIS AN OVERRUN CHAR
JNZ K16 ; NO, TEST FOR SHIFT KEY
JMP K62 ; BUFFER_FULL_BEEP
K16: PUSH CS
POP ES ; ESTABLISH ADDRESS OF TABLES
MOV BH,0KB_FLAG_3 ; LOAD FLAGS FOR TESTING
TEST_E0: CMP AL,MC_E0 ; IS THIS THE GENERAL MARKER CODE?
JNE TEST_E1 ; WAS LAST CODE THE E0 MARKER CODE?
OR 0KB_FLAG_3,LC_E0+KBX ; SET FLAG BIT, SET KBX, AND
SHORT EXIT_K ; THROW AWAY THIS CODE
TEST_E1: CMP AL,MC_E1 ; IS THIS THE PAUSE KEY?
JNE NOT_HC ; NO, THROW AWAY & RESET FLAG
OR 0KB_FLAG_3,LC_E1+KBX ; SET FLAG, PAUSE KEY MARKER CODE
EXIT_K: JMP K26A ; THROW AWAY THIS CODE
NOT_HC: AND AL,07FH ; TURN OFF THE BREAK BIT
TEST BH,LC_E0 ; WAS LAST CODE THE E0 MARKER CODE?
JZ NOT_LC_E0 ; JUMP IF NOT
MOV CX,2 ; LENGTH OF SEARCH
DI,OFFSET K6+6 ; IS THIS A SHIFT KEY?
SCASB ; CHECK IT
JNE K16A ; NO, CONTINUE KEY PROCESSING
JMP SHORT K16B ; YES, THROW AWAY & RESET FLAG
NOT_LC_E0: TEST BH,LC_E1 ; WAS LAST CODE THE E1 MARKER CODE?
JZ T_SYS_KEY ; JUMP IF NOT
MOV CX,4 ; LENGTH OF SEARCH
DI,OFFSET K6+4 ; IS THIS AN ALT, CTL, OR SHIFT?
SCASB ; CHECK IT
JNE EXIT_K ; THROW AWAY IF SO
CMP AL,NUM_KEY ; IS IT THE PAUSE KEY?
JNE K16B ; NO, THROW AWAY & RESET FLAG
TEST AH,80H ; YES, IS IT THE BREAK OF THE KEY?
JNZ K16B ; YES, THROW THIS AWAY, TOO
JMP K39P ; NO, THIS IS THE REAL PAUSE STATE

```

```

363 PAGE
364 1----- TEST FOR SYSTEM KEY
365
366 T_SYS_KEY:
367 CMP AL,SYS_KEY      ; IS IT THE SYSTEM KEY?
368 JNE K16A             ; CONTINUE IF NOT
369
370 TEST AH,080H         ; CHECK IF THIS A BREAK CODE
371 JNZ K16C             ; DONT TOUCH SYSTEM INDICATOR IF TRUE
372
373 TEST *KB_FLAG_1,SYS_SHIFT ; SEE IF IN SYSTEM KEY HELD DOWN
374 JNZ K16B             ; IF YES, DONT PROCESS SYSTEM INDICATOR
375
376 MOV *KB_FLAG_1,SYS_SHIFT ; INDICATE SYSTEM KEY DEPRESSED
377 OR AL,E0I            ; END OF INTERRUPT COMMAND
378 OUT 020H,AL          ; SEND COMMAND TO INTERRUPT CONTROL PORT
379
380 MOV AX,08500H         ; INTERRUPT-RETURN-NO-E0I
381 STI                  ; FUNCTION VALUE FOR MAKE OF SYSTEM KEY
382 INT 15H              ; MAKE SURE INTERRUPTS ENABLED
383 JMP K27               ; USER INTERRUPT
384                     ; END PROCESSING
385 K16B: JMP K26         ; IGNORE SYSTEM KEY
386
387 K16C: AND *KB_FLAG_1,NOT SYS_SHIFT; TURN OFF SHIFT KEY HELD DOWN
388 OUT 020H,AL          ; END OF INTERRUPT COMMAND
389
390 MOV AX,08501H         ; SEND COMMAND TO INTERRUPT CONTROL PORT
391 STI                  ; INTERRUPT-RETURN-NO-E0I
392 INT 15H              ; FUNCTION VALUE FOR BREAK OF SYSTEM KEY
393 JMP K27               ; MAKE SURE INTERRUPTS ENABLED
394                     ; USER INTERRUPT
395                     ; IGNORE SYSTEM KEY
396
397 1----- TEST FOR SHIFT KEYS
398
399 K16A: MOV BL,*KB_FLAG ; PUT STATE FLAGS IN BL
400 MOV DI,OFFSET K6     ; SHIFT KEY TABLE
401 MOV CX,K6            ; LENGTH
402 REPNE SCASB          ; LOOK THROUGH THE TABLE FOR A MATCH
403 MOV AL,AH            ; RECOVER SCAN CODE
404 JE K17               ; JUMP IF MATCH FOUND
405 JMP K25               ; IF NO MATCH, THEN SHIFT NOT FOUND
406
407 1----- SHIFT KEY FOUND
408
409 K17: SUB DI,OFFSET K6+1 ; ADJUST PTR TO SCAN CODE MTCH
410 MOV AH,CS:[DI]        ; GET MASK INTO AH
411 MOV CL,2              ; SET UP COUNT FOR FLAG SHIFTS
412 TEST AL,AH            ; TEST FOR KEY BREAK
413 JZ K17C              ; JUMP IF BREAK
414 JMP K23               ; JUMP IF BREAK
415
416 1----- SHIFT MAKE FOUND, DETERMINE SET OR TOGGLE
417
418 K17C: CMP AH,SCROLL_SHIFT ; IF SCROLL SHIFT OR ABOVE, TOGGLE KEY
419 JAE K18
420
421 1----- PLAIN SHIFT KEY, SET SHIFT ON
422
423 OR *KB_FLAG,AH        ; TURN ON SHIFT BIT
424 TEST AH,CTL_SHIFT+ALT_SHIFT ; IS IT ALT OR CTRL?
425 JNZ K17D              ; YES, MORE FLAGS TO SET
426 JMP K26               ; NO, INTERRUPT RETURN
427 K17D: TEST BH,LC_E0   ; IS THIS ONE OF THE NEW KEYS?
428 JZ K17E               ; NO, JUMP
429 OR *KB_FLAG_3,AH      ; SET BITS FOR RIGHT CTRL, ALT
430 JZ K26                ; INTERRUPT RETURN
431 SHR AH,CL             ; MOVE FLAG BITS TWO POSITIONS
432 OR *KB_FLAG_1,AH      ; SET BITS FOR LEFT CTRL, ALT
433 JMP K26                ; INTERRUPT RETURN
434
435 1----- TOGGLED SHIFT KEY, TEST FOR 1ST MAKE OR NOT
436
437 K18: TEST BL,CTL_SHIFT ; SHIFT-TOGGLE
438 JZ K18A               ; CHECK CTL SHIFT STATE
439 JMP K25                ; JUMP IF NOT CTL STATE
440 K18A: CMP AL,INS_KEY  ; JUMP IF CTL STATE
441 JNE K22               ; CHECK FOR INSERT KEY
442 TEST BL,ALT_SHIFT    ; JUMP IF NOT INSERT KEY
443 JZ K18B               ; CHECK FOR ALTERNATE SHIFT
444 OR *KB_FLAG,AH        ; JUMP IF NOT ALTERNATE SHIFT
445 TEST BH,LC_E0        ; JUMP IF ALTERNATE SHIFT
446 JNZ K22               ; IS THIS THE NEW INSERT KEY?
447 TEST BL,NUM_STATE    ; YES, THIS ONE'S NEVER A "0"
448 JNZ K21               ; CHECK FOR BASE STATE
449 TEST BL,LEFT_SHIFT+RIGHT_SHIFT ; JUMP IF NUM LOCK IS ON
450 JZ K22                ; TEST FOR SHIFT STATE
451                     ; JUMP IF BASE STATE
452
453 K20: MOV AH,AL         ; PUT SCAN CODE BACK INTO AH
454 JMP K25                ; NUMERAL "0", STDNRD. PROCESSING
455
456 K21: TEST BL,LEFT_SHIFT+RIGHT_SHIFT ; MIGHT BE NUMERIC
457 JZ K20                 ; IS NUMERIC, STD. PROC.
458
459 K22: 1 SHIFT TOGGLE KEY HIT! PROCESS IT
460 TEST AH,*KB_FLAG_1    ; IS KEY ALREADY DEPRESSED
461 JZ K22A               ; JUMP IF KEY ALREADY DEPRESSED
462 OR *KB_FLAG_1,AH      ; INDICATE THAT THE KEY IS DEPRESSED
463 XOR *KB_FLAG,AH       ; TOGGLE THE SHIFT STATE
464 CMP AL,INS_KEY        ; TEST FOR 1ST MAKE OF INSERT KEY
465 JNE K26               ; JUMP IF NOT INSERT KEY
466 MOV AH,AL             ; SCAN CODE IN BOTH HALVES OF AX
467 JMP K28                ; FLAGS UPDATED, PROC. FOR BUFFER
468
469 1----- BREAK SHIFT FOUND
470
471 K23: 1 BREAK-SHIFT-FOUND
472 NOT AH,SCROLL_SHIFT  ; IS THIS A TOGGLE KEY?
473 JZ K24                ; INVERT MASK
474 AND *KB_FLAG,AH      ; YES, HANDLE BREAK TOGGLE
475 CMP AH,NOT CTL_SHIFT ; TURN OFF SHIFT BIT
476                     ; IS THIS ALT OR CTL?

```

```

477 0274 77 26          JA      K23D          ; NO, ALL DONE
478
479 0276 F6 C7 02       TEST     BH,LC_E0      ; 2ND ALT OR CTL?
480 0279 74 06          JZ       K23A          ; NO, HANDLE NORMALLY
481 027B 20 26 0096 R   AND      *KB_FLAG_3,AH      ; RESET BIT FOR RIGHT ALT OR CTL
482 027F E9 06          JMP      SHORT K23B     ; CONTINUE
483 0281 D2 FC          SAR      AH,CL          ; MOVE THE MASK BIT TWO POSITIONS
484 0283 20 26 0018 R   AND      *KB_FLAG_1,AH      ; RESET BIT FOR LEFT ALT OR CTL
485 0287 8A E0          MOV      AH,AL          ; SAVE SCAN CODE
486 0289 A0 0096 R      MOV      AL,*KB_FLAG_3    ; GET RIGHT ALT & CTRL FLAGS
487 028C D2 E8          SHR      AL,CL          ; MOVE TO BITS 1 & 0
488 028E 0A 06 0018 R   OR       AL,*KB_FLAG_1    ; PUT IN LEFT ALT & CTRL FLAGS
489 0292 D2 E0          SHL      AL,CL          ; MOVE BACK TO BITS 3 & 2
490 0294 24 0C          AND      AL,ALT_SHIFT+CTL_SHIFT ; FILTER OUT OTHER GARBAGE
491 0296 08 06 0017 R   OR       *KB_FLAG,AL      ; PUT RESULT IN THE REAL FLAGS
492 029A 8A C4          MOV      AL,AH          ; RECOVER SAVED SCAN CODE
493
494 029C 3C B8          K23D:  CMP     AL,ALT_KEY+80H    ; IS THIS ALTERNATE SHIFT RELEASE
495 029E 75 2A          JNE      K26             ; INTERRUPT_RETURN
496
497 ;----- ALTERNATE SHIFT KEY RELEASED, GET THE VALUE INTO BUFFER
498
499 02A0 A0 0019 R      MOV      AL,*ALT_INPUT
500 02A3 B4 00          MOV      AH,0             ; SCAN CODE OF 0
501 02A5 88 26 0019 R   MOV      *ALT_INPUT,AH      ; ZERO OUT THE FIELD
502 02A9 3C 00          CMP      AL,0             ; WAS THE INPUT = 0?
503 02AB 74 1D          JE       K26             ; INTERRUPT_RETURN
504 02AD E9 0519 R      JMP      K61             ; IT WASN'T, SO PUT IN BUFFER
505
506 02B0                K24:
507 02B0 20 26 0018 R   AND      *KB_FLAG_1,AH      ; BREAK-TOGGLE
508 02B4 EB 14          JMP      SHORT K26             ; INDICATE NO LONGER DEPRESSED
509 ;----- TEST FOR HOLD STATE
510
511 02B6                K25:
512 02B6 3C 80          CMP      AL,80H          ; AL, AH = SCAN CODE
513 02B8 73 10          JAE      K26             ; NO-SHIFT-FOUND
514 02BB 73 10          JAE      K26             ; TEST FOR BREAK KEY
515 02BA F6 06 0018 R 0B TEST     *KB_FLAG_1,HOLD_STATE ; NOTHING FOR BREAK CHARS FROM HERE ON
516 02BF 74 1C          JE       K28             ; ARE WE IN HOLD STATE
517 02C1 3C 45          CMP      AL,NUM_KEY      ; BRANCH AROUND TEST IF NOT
518 02C3 74 05          JE       K26             ; CAN'T END HOLD ON NUM LOCK
519 02C5 80 26 0018 R F7 AND      *KB_FLAG_1,NOT_HOLD_STATE ; END OFF THE HOLD STATE BIT
520
521 02CA                K26:
522 02CA 80 26 0096 R FC AND      *KB_FLAG_3,NOT LC_E0+LC_E1 ; RESET LAST CHAR H.C. FLAG
523
524 02CF                K26A:
525 02CF FA            CLI             ; INTERRUPT_RETURN
526 02D0 B0 20          MOV      AL,E01          ; TURN OFF INTERRUPTS
527 02D2 E6 20          OUT     020H,AL          ; END OF INTERRUPT COMMAND
528 ; SEND COMMAND TO INTERRUPT CONTROL PORT
529 02D4                K27:
530 02D4 07            POP      E5             ; INTERRUPT_RETURN-NO-E01
531 02D5 1F            POP      D5             ; RESTORE REGISTERS
532 02D6 5F            POP      D1
533 02D7 5E            POP      SI
534 02D8 5A            POP      DI
535 02D9 59            POP      CX
536 02DA 5B            POP      BX
537 02DB 58            POP      AX
538 02DC CF            IRET          ; RETURN, INTERRUPTS BACK ON
539 ; WITH FLAG CHANGE

```

```

540      PAGE
541      :----- NOT IN HOLD STATE
542
543      02DD      K28:
544      02DD 3C 58      CMP     AL,88
545      02DF 77 E9      JA      K26
546
547      02E1 F6 C3 08      TEST    BL,ALT_SHIFT
548      02E4 74 0C      JZ      K28A
549
550      02E6 F6 C7 10      TEST    BH,KBX
551      02E9 74 0A      JZ      K28
552
553      02EB F6 06 0018 R 04      TEST    *KB_FLAG_1,SYS_SHIFT
554      02F0 74 03      JZ      K28
555      02F2 E9 03CC R      JMP     K38
556
557      :----- TEST FOR RESET KEY SEQUENCE (CTL ALT DEL)
558
559      02F5      K29:
560      02F5 F6 C3 04      TEST    BL,CTL_SHIFT
561      02F6 74 37      JZ      K31
562      02FA 3C 53      CMP     AL,DEL_KEY
563      02FC 75 33      JNE     K31
564
565      :----- CTL-ALT-DEL HAS BEEN FOUND, DO I/O CLEANUP
566
567      02FE C7 06 0072 R 1234      MOV     *RESET_FLAG,1234H
568      0304 81 26 0096 R 0010      AND     WORD PTR *KB_FLAG_3,KBX
569      030A E9 0000 E      JMP     RESET
570
571      :----- ALT-INPUT-TABLE
572
573      030D      K30
574      030D 52 4F 50 51 4B      LABEL  BYTE
575      0312 4C 4D 47 48 49      DB      82,79,80,81,75
576
577      0317 0310 10 11 12 13 14 15      DB      76,77,71,72,73
578      031D 16 17 18 19 1E 1F      : 10 NUMBERS ON KEYPAD
579      0323 20 21 22 23 24 25      :----- SUPER-SHIFT-TABLE
580      0329 26 2C 2D 2E 2F 30      DB      16,17,18,19,20,21
581      032F 31 32      DB      22,23,24,25,30,31
582      DB      22,33,34,35,36,37      : A-Z TYPEWRITER CHARS
583      DB      38,44,45,46,47,48      DB
584      DB      49,50      DB
585
586      :----- IN ALTERNATE SHIFT, RESET NOT FOUND
587
588      0331      K31:
589      0331 3C 39      CMP     AL,57
590      0333 75 15      JNE     K31I
591      0335 B0 20      MOV     AL,' '
592      0337 E9 050D R      JMP     K57
593
594      033A      K31I:
595      033A 3C 0F      CMP     AL,15
596      033C 75 06      JNE     K312
597      033E B8 A500      MOV     AX,0A500h
598      0341 E9 050D R      JMP     K57
599
600      0344      K312:
601      0344 3C 4A      CMP     AL,74
602      0346 74 79      JE      K37B
603      0348 3C 4E      CMP     AL,78
604      034A 74 75      JE      K37B
605
606      :----- LOOK FOR KEY PAD ENTRY
607
608      034C      K32:
609      034C BF 030D R      MOV     DI,OFFSET K30
610      034F B9 000A      MOV     CX,10
611      0352 F2 / AE      REPNE  SCASB
612      0354 75 18      JNE     K33
613      0356 F6 C7 02      TEST    BH,LC_E0
614      0359 75 6B      JNZ     K37C
615      035B 81 EF 030E R      SUB     DI,OFFSET K30+1
616      035F A0 0019 R      MOV     AL,*ALT_INPUT
617      0362 B4 0A      MOV     AH,10
618      0364 F6 E4      MUL     AH
619      0366 03 C7      ADD     AX,DI
620      0368 A2 0019 R      MOV     *ALT_INPUT,AL
621      036B E9 02CA R      JMP     K26
622
623      :----- LOOK FOR SUPERSHIFT ENTRY
624
625      036E      K33:
626      036E C6 06 0019 R 00      MOV     *ALT_INPUT,0
627      0373 B9 001A      MOV     CX,26
628      0376 F2 / AE      REPNE  SCASB
629      0378 74 42      JE      K37A
630
631      :----- LOOK FOR TOP ROW OF ALTERNATE SHIFT
632
633      037A      K34:
634      037A 3C 02      CMP     AL,2
635      037C 72 43      JB      K37B
636      037E 3C 0D      CMP     AL,13
637      0380 77 05      JA      K35
638      0382 80 C4 76      ADD     AH,18
639      0385 EB 35      JMP     SHORT K37A
640
641      :----- TRANSLATE ALTERNATE SHIFT PSEUDO SCAN CODES
642
643      0387      K35:
644      0387 3C 57      CMP     AL,F11_M
645      0389 72 09      JB      K35A
646      038B 3C 58      CMP     AL,F12_M
647      038D 77 05      JA      K35A
648      038F 80 C4 34      ADD     AH,52
649      0392 EB 28      JMP     SHORT K37A
650
651      0394      K35A:
652      0394 F6 C7 02      TEST    BH,LC_E0
653      0397 74 18      JZ      K35B
654      0399 3C 1C      CMP     AL,28
655      039B 75 06      JNE     K35B
656      039D B8 A600      MOV     AX,0A600h
657      03A0 E9 050D R      JMP     K57
658
659      03A3      K35B:
660      03A3 3C 53      CMP     AL,83
661      03A5 74 1F      JE      K37C

```

```

654 03A7 3C 35      CMP     AL,53      ; TEST FOR KEYPAD /
655 03A9 75 C0      JNE      K32A      ; NOT THERE, NO OTHER E0 SPECIALS
656 03AB B8 A400    MOV      AX,0A400h    ; SPECIAL CODE
657 03AE E9 050D R  JMP      K57      ; BUFFER_FILL
658
659 03B1 3C 3B      K37:  CMP     AL,59      ; TEST FOR IN TABLE
660 03B3 72 0C      JB      K37B      ; ALT-CONTINUE
661 03B5 3C 44      CMP     AL,60      ; IN KEYPAD REGION?
662                                ; OR NUMLOCK, SCROLLLOCK?
663 03B7 77 B2      JA      K32A      ; IF SO, IGNORE
664 03B9 80 C4 2D   ADD      AH,45      ; CONVERT TO PSEUDO SCAN CODE
665
666 03BC B0 00      K37A:  MOV     AL,0      ; ASCII CODE OF ZERO
667 03BE E9 050D R  JMP      K57      ; PUT IT IN THE BUFFER
668
669 03C1 B0 F0      K37B:  MOV     AL,0F0h    ; USE SPECIAL ASCII CODE
670 03C3 E9 050D R  JMP      K57      ; PUT IT IN THE BUFFER
671
672 03C6 04 50      K37C:  ADD     AL,80      ; CONVERT SCAN CODE (EDIT KEYS)
673 03C8 8A E0      MOV     AH,AL      ; (SCAN CODE NOT IN AH FOR INSERT)
674 03CA EB F0      JMP      K37A      ; PUT IT IN THE BUFFER
675
676             I----- NOT IN ALTERNATE SHIFT
677
678 03CC      K38:      ; NOT-ALT-SHIFT
679                                ; BL STILL HAS SHIFT FLAGS
680 03CC F6 C3 04    TEST     BL,CTL_SHIFT ; ARE WE IN CONTROL SHIFT?
681 03CF 75 03      JNZ     K38A      ; YES, START PROCESSING
682 03D1 E9 0454 R  JMP      K44      ; NOT-CTL-SHIFT
683
684             I----- CONTROL SHIFT, TEST SPECIAL CHARACTERS
685             I----- TEST FOR BREAK
686
687 03DA 3C 46      K38A:  CMP     AL,SCROLL_KEY ; TEST FOR BREAK
688 03DB 75 1E      JNE      K39      ; JUMP, NO-BREAK
689 03DD F4 07 10    TEST     BH,KBX      ; IS THIS THE ENHANCED KEYBOARD?
690 03DF 74 05      JZ      K38B      ; NO, BREAK IS VALID
691 03E0 F6 C7 02    TEST     BH,LC_E0     ; YES, WAS LAST CODE AN E0?
692 03E2 74 14      JZ      K39      ; NO-BREAK, TEST FOR PAUSE
693
694 03E2 8B 1E 00 1A R K38B:  MOV     BX,#BUFFER_HEAD ; RESET BUFFER TO EMPTY
695 03E6 89 1E 00 1C R MOV     #BUFFER_TAIL,BX ; TURN ON BIOS BREAK BIT
696 03EA C6 06 00 1F R 0B MOV     #BIOS_BREAK,80H ; BREAK INTERRUPT VECTOR
697 03EF CD 1B      INT     1BH          ;
698 03F1 2B C0      SUB     AX,AX      ; PUT OUT DUMMY CHARACTER
699 03F3 E9 050D R  JMP      K57      ; BUFFER_FILL
700
701             I----- TEST FOR PAUSE
702
703 03F6      K39:      ; NO-BREAK
704 03F6 F6 C7 10    TEST     BH,KBX      ; IS THIS THE ENHANCED KEYBOARD?
705 03F9 75 25      JNZ     K41      ; YES, THEN THIS CAN'T BE PAUSE
706 03FB 3C 45      CMP     AL,NUM_KEY   ; LOOK FOR PAUSE KEY
707 03FD 75 21      JNE      K41      ; NO-PAUSE
708 03FF 80 0E 00 1B R 0B OR      #KB_FLAG_1,HOLD_STATE ; TURN ON THE HOLD FLAG
709 0400 80 20      MOV     AL,E01      ; END OF INTERRUPT TO CONTROL PORT
710 0404 E6 20      OUT     020H,AL      ; ALLOW FURTHER KEYSTROKE INTS
711
712             I----- DURING PAUSE INTERVAL, TURN CRT BACK ON
713
714 0408 80 3E 00 49 R 07 CMP     #CRT_MODE,7 ; IS THIS BLACK AND WHITE CARD
715 040D 74 07 10    JE      K40      ; YES, NOTHING TO DO
716 040F BA 0308     MOV     DX,03D8H    ; PORT FOR COLOR CARD
717 0412 A0 0665 R   MOV     AL,#CRT_MODE_SET ; GET THE VALUE OF THE CURRENT MODE
718 0415 EE          OUT     DX,AL      ; SET THE CRT MODE, SO THAT CRT IS ON
719 0416 F6 06 00 1B R 0B TEST     #KB_FLAG_1,HOLD_STATE ; PAUSE-LOOP
720 0418 75 F9      JNZ     K40      ; LOOP UNTIL FLAG TURNED OFF
721 041D E9 02D4 R   JMP      K27      ; INTERRUPT_RETURN_NO_E0I
722
723             I----- TEST SPECIAL CASE KEY 55
724
725 0420      K41:      ; NO-PAUSE
726 0420 3C 37      CMP     AL,55      ; TEST FOR */PRTSK KEY
727 0422 75 10      JNE      K42      ; NOT-KEY-55
728 0424 F6 C7 10    TEST     BH,KBX      ; IS THIS THE ENHANCED KEYBOARD?
729 0427 74 05      JZ      K41A      ; NO, CTL-PRTSK IS VALID
730 0429 F6 C7 02    TEST     BH,LC_E0     ; YES, WAS LAST CODE AN E0?
731 042C 74 20      JZ      K42B      ; NO, TRANSLATE TO A FUNCTION
732 042E B8 7200     MOV     AX,114*256   ; START/STOP PRINTING SWITCH
733 0431 E9 050D R   JMP      K57      ; BUFFER_FILL
734
735             I----- SET UP TO TRANSLATE CONTROL SHIFT
736
737 0434      K42:      ; NOT-KEY-55
738 0434 3C 0F      CMP     AL,15      ; IS IT THE TAB KEY?
739 0436 74 16      JE      K42B      ; YES, XLATE TO FUNCTION CODE
740 0438 3C 35      CMP     AL,53      ; IS IT THE / KEY?
741 043A 75 0B      JNE      K42A      ; NO, NO MORE SPECIAL CASES
742 043C F6 C7 02    TEST     BH,LC_E0     ; YES, IS IT FROM THE KEYPAD?
743 043F 74 06      JZ      K42A      ; NO, JUST TRANSLATE
744 0441 B8 9500     MOV     AX,9500h     ; YES, SPECIAL CODE FOR THIS ONE
745 0444 E9 050D R   JMP      K57      ; BUFFER_FILL
746
747 0447 B8 055F R   K42A:  MOV     BX,OFFSET K8 ; SET UP TO TRANSLATE CTL
748 044A 3C 3B      CMP     AL,59      ; IS IT IN CHARACTER TABLE?
749 044C 72 57      JB      K45F      ; YES, GO TRANSLATE CHAR
750 044E B8 055F R   K42B:  MOV     BX,OFFSET K8 ; SET UP TO TRANSLATE CTL
751 0451 E9 04FC R   JMP      K64      ; NO, GO TRANSLATE_SCAN
752
753             I----- NOT IN CONTROL SHIFT
754
755 0454 3C 37      K44:  CMP     AL,55      ; PRINT SCREEN KEY?
756 0456 75 1F      JNE      K45      ; NOT-PRINT-SCREEN
757 0458 F6 C7 10    TEST     BH,KBX      ; IS THIS ENHANCED KEYBOARD?
758 045B 74 07      JZ      K44A      ; NO, TEST FOR SHIFT STATE
759 045D F6 C7 02    TEST     BH,LC_E0     ; YES, LAST CODE A MARKER?
760 0460 75 07      JNZ     K44B      ; YES, IS PRINT SCREEN
761 0462 EB 34      JMP     SHORT K45C    ; NO, XLATE TO ** CHARACTER
762 0464 F6 C3 03    K44A:  TEST     BL,LEFT_SHIFT+RIGHT_SHIFT ; NOT 101 KBD, SHIFT KEY DOWN?
763 0467 74 2F      JZ      K45C      ; NO, XLATE TO ** CHARACTER
764
765             I----- ISSUE INTERRUPT TO PERFORM PRINT SCREEN FUNCTION
766
767
```

```

768 0469 B0 20      K44B:  MOV     AL,E01          ; END OF CURRENT INTERRUPT
769 046B E6 20      OUT      020H,AL          ; 50 FURTHER THINGS CAN HAPPEN
770 046D CD 05      INT      $H              ; ISSUE PRINT SCREEN INTERRUPT
771 046F B0 26 0096 R FC  AND     $KB FLAG_3,NOT LC_E0+LC_E0
772 0474 E9 02D4 R   JMP      K27              ; GO BACK WITHOUT E01 OCCURRING
773
774
775
776 0477
777 0477 3C 3A      K45:    HANDLE THE IN-CORE KEYS
778 0479 77 2C      CMP      AL,58          ; NOT-PRINT-SCREEN
779                      JA      K46          ; TEST FOR IN-CORE AREA
780                      ; JUMP IF NOT
781 047B 3C 35      CMP      AL,53          ; IS THIS THE "/" KEY?
782 047D 75 05      JNE      K45A          ; NO, JUMP
783 0482 75 14      TEST     BH,LC_E0        ; WAS LAST CODE THE MARKER?
784                      JNZ      K45C        ; YES, TRANSLATE TO CHARACTER
785
786 048A B9 001A      K45A:  MOV      CX,26          ; LENGTH OF SEARCH
787 048B BF 0317 R   MOV      D1,OFFSET K30+10        ; POINT TO TABLE OF A-Z CHARS
788 048C F2/ AE      REPNE    SCASB          ; IS THIS A LETTER KEY?
789                      JNE      K45B        ; NO, SYMBOL KEY
790
791 048E F6 C3 40      TEST     BL,CAPS_STATE      ; ARE WE IN CAPS_LOCK?
792 0491 75 0A      JNZ      K45D          ; TEST FOR SURE
793 0493 F6 C3 03      K45B:  TEST     BL,LEFT_SHIFT+RIGHT_SHIFT ; ARE WE IN SHIFT STATE?
794 0496 75 0A      JNZ      K45E          ; YES, UPPERCASE
795                      ; NO, LOWERCASE
796 049B BB 05B7 R   K45C:  MOV      BX,OFFSET K10         ; TRANSLATE TO LOWERCASE LETTERS
797 049D EB 50      JMP      SHORT K56          ;
798
799 049D F6 C3 03      K45D:  TEST     BL,LEFT_SHIFT+RIGHT_SHIFT ; ALMOST-CAPS-STATE
800 04A2 BB 060F R   JNZ      K45C          ; CL ON: IS SHIFT ON, TOO?
801 04A5 EB 46      K45E:  MOV      BX,OFFSET K11         ; SHIFTED TEMP OUT OF CAPS STATE
802                      JMP      SHORT K56          ; TRANSLATE TO UPPERCASE LETTERS
803
804
805 04A7
806 04A7 3C 44      K46:    TEST FOR KEYS F1 - F10
807 04A9 77 02      CMP      AL,68          ; NOT IN-CORE AREA
808 04AB EB 36      JA      K47          ; TEST FOR F1 - F10
809                      JMP      SHORT K53      ; JUMP IF NOT
810                      ; YES, GO DO FN KEY PROCESS
811
812
813 04AD
814 04AD 3C 53      K47:    HANDLE THE NUMERIC PAD KEYS
815 04AF 77 2C      CMP      AL,83          ; NOT F1 - F10
816                      JA      K52          ; TEST FOR NUMPAD KEYS
817                      ; JUMP IF NOT
818
819 04B1 3C 4A      K48:    KEYPAD KEYS, MUST TEST NUM LOCK FOR DETERMINATION
820 04B3 74 ED      CMP      AL,74          ; SPECIAL CASE FOR MINUS
821 04B7 74 E9      JE      K45E          ; GO TRANSLATE
822 04B9 F6 C7 02      CMP      AL,78          ; SPECIAL CASE FOR PLUS
823 04BC 75 0A      JE      K45E          ; GO TRANSLATE
824                      TEST     BH,LC_E0        ; IS THIS ONE OF THE NEW KEYS?
825                      JNZ      K49          ; YES, TRANSLATE TO BASE STATE
826
827 04BE F6 C3 20      TEST     BL,NUM_STATE      ; ARE WE IN NUM LOCK?
828 04C1 75 13      JNZ      K50          ; TEST FOR SURE
829 04C3 F6 C3 03      JNZ      K51          ; ARE WE IN SHIFT STATE?
830 04C6 75 13      TEST     BL,LEFT_SHIFT+RIGHT_SHIFT ; IF SHIFTED, REALLY NUM STATE
831
832 04CA 3C 4C      K49:    BASE CASE FOR KEYPAD
833 04CC 75 05      CMP      AL,76          ; SPECIAL CASE FOR BASE STATE 5
834 04CE BB 05B7 R   JNE      K49A          ; CONTINUE IF NOT KEYPAD 5
835 04D0 EB 26      MOV      AL,0F0h          ; SPECIAL ASCII CODE
836 04D4 EB 26      MOV      K51,0          ; BUFFER FILL
837                      ; BASE CASE TABLE
838                      JMP      SHORT K64      ; CONVERT TO PSEUDO SCAN
839
840 04D6 F6 C3 03      K50:    MIGHT BE NUM LOCK, TEST SHIFT STATUS
841 04D9 75 ED      TEST     BL,LEFT_SHIFT+RIGHT_SHIFT ; ALMOST-NUM-STATE
842 04DB EB C5      JNZ      K49          ; SHIFTED TEMP OUT OF NUM STATE
843                      K51:    JMP      SHORT K45E      ; REALLY_NUM_STATE
844
845
846 04DD
847 04DD 3C 56      K52:    TEST FOR THE NEW KEY ON WT KEYBOARDS
848 04DF 75 02      CMP      AL,86          ; NOT A NUMPAD KEY
849 04E1 EB B0      JNE      K53          ; IS IT THE NEW WT KEY?
850                      JMP      SHORT K45B      ; JUMP IF NOT
851                      ; HANDLE WITH REST OF LETTER KEYS
852
853
854 04E3 F6 C3 03      K53:    MUST BE F11 OR F12
855 04E6 74 E0      TEST     BL,LEFT_SHIFT+RIGHT_SHIFT ; F1 - F10 COME HERE, TOO
856                      JZ      K49          ; TEST SHIFT STATE
857                      ; JUMP, LOWERCASE PSEUDO SC'S
858 04EB BB 060F R   MOV      BX,OFFSET K11         ; UPPERCASE PSEUDO SCAN CODES
859 04ED EB 0F      JMP      SHORT K64          ; TRANSLATE_SCAN
860
861
862 04ED
863 04ED FE C8      K56:    TRANSLATE THE CHARACTER
864 04EF 2E/ D7      DEC      AL          ; TRANSLATE-CHAR
865 04F1 F6 06 0096 R 02 XLAT     CS:K11          ; CONVERT ORIGIN
866 04F6 74 15      AND     $KB FLAG_3,LC_E0        ; CONVERT THE SCAN CODE TO ASCII
867 04FB B4 E0      TEST     JZ          ; IS THIS A NEW KEY?
868 04FA EB 11      MOV      AH,MC_E0          ; NO, GO FILL BUFFER
869                      JMP      SHORT K57          ; YES, PUT SPECIAL MARKER IN AH
870                      ; PUT IT INTO THE BUFFER
871
872 04FC
873 04FC FE C8      K64:    TRANSLATE SCAN FOR PSEUDO SCAN CODES
874 04FE 2E/ D7      DEC      AL          ; TRANSLATE-SCAN-ORGD
875 0500 8A E0      XLAT     CS:K8          ; CONVERT ORIGIN
876 0502 B0 00      MOV      AH,AL          ; CTL TABLE SCAN
877 0504 F6 06 0096 R 02 AL      0          ; PUT VALUE INTO AH
878 0509 74 02      TEST     $KB FLAG_3,LC_E0        ; ZERO ASCII CODE
879 050B B0 E0      JZ      K57          ; IS THIS A NEW KEY?
880                      MOV      AL,MC_E0          ; NO, GO FILL BUFFER
881                      ; YES, PUT SPECIAL MARKER IN AL
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900

```

```

882
883 050D                K57:
884 050D 3C FF          CMP     AL,-1
885 050F 74 05          JE      K59
886 0511 80 FC FF      CMP     AH,-1
887 0514 75 03          JNE     K61
888
889 0516                K59:
890 0516 E9 02CA R      JMP     K26
891
892 0519                K61:
893 0519 FA             CLI
894 051A 8B 1E 001C R   MOV     BX,0BUFFER_TAIL
895 051E 8B F3          MOV     SI,BX
896 0520 EB 0114 R      CALL    K4
897 0523 3B 1E 001A R   CMP     BX,0BUFFER_HEAD
898 0527 74 17          JE      K62
899 0529 89 04          MOV     [SI],AX
900 052B 89 1E 001C R   MOV     0BUFFER_TAIL,BX
901 052F 80 20          MOV     AL,E01
902 0531 E6 20          OUT     020H,AL
903 0533 8B 9102        MOV     AX,09102H
904 0536 CD 15          INT     15H
905 0538 80 26 0096 R FC AND     0KB_FLAG_3,NOT LC_E0+LC_E1
906 053D E9 02D4 R      JMP     K27
907
908 ;----- BUFFER IS FULL SOUND THE BEEPER
909
910 0540                K62:
911 0540 80 20          MOV     AL,E01
912 0542 E6 20          OUT     INTA00,AL
913 0544 B9 02A6        MOV     CX,678
914 0547 B3 04          MOV     BL,4
915 0549 E8 0000 E      CALL    BEEP
916 054C E9 02D4 R      JMP     K27
917
918 054F                KB_INT_1 ENDP

```

```

919 PAGE
920 -----
921 1 KEY IDENTIFICATION SCAN TABLES
922 -----
923
924 1----- TABLE OF SHIFT KEYS AND MASK VALUES -----
925 1-----
926 KEY TABLE
927 LABEL BYTE
928 054F DB INS KEY ; INSERT KEY
929 0550 3A 45 46 38 1D DB CAPS_KEY,NUM_KEY,SCROLL_KEY,ALT_KEY,CTL_KEY
930 055A 2A 36 DB LEFT_KEY,RIGHT_KEY
931 055D 02 01 EQU -K6
932
933 1----- MASK TABLE
934 LABEL BYTE
935 0557 DB INS_SHIFT ; INSERT MODE SHIFT
936 0558 40 20 10 08 04 DB CAPS_SHIFT,NUM_SHIFT,SCROLL_SHIFT,ALT_SHIFT,CTL_SHIFT
937 055D 02 01 DB LEFT_SHIFT,RIGHT_SHIFT
938
939 1----- TABLES FOR CTRL CASE -----
940 1-----
941 LABEL BYTE
942 055F 1B FF 00 FF FF FF DB 27,-1,00,-1,-1,-1 ; Esc, 1, 2, 3, 4, 5
943 0560 1E FF FF FF FF 1F DB 30,-1,-1,-1,-1,31 ; 6, 7, 8, 9, 0,
944 0561 7F 94 11 17 05 DB -1,127,148,17,23,5 ; =, Bksp, Tab, Q, W, E
945 0562 12 14 19 15 09 0F DB 18,20,25,21,09,15 ; R, T, Y, U, I, O
946 0563 10 18 1D 0A FF 01 DB 16,27,29,10,-1,01 ; P, [ ], Enter, Ctrl, A
947 0564 13 04 06 07 08 0A DB 19,04,06,07,08,10 ; S, D, F, G, H, J
948 0565 0C 0F FF FF FF DB 11,12,-1,-1,-1,-1 ; K, L, ;, ', LShift
949 0566 1A 18 03 16 02 DB 28,26,24,03,22,02 ; I, Z, X, C, V, B
950 0567 0E 0D FF FF FF DB 14,13,-1,-1,-1,-1 ; N, M, ., /, RShift
951 0568 96 FF 20 FF DB 150,-1,' ',-1 ; *, Alt, Space, CL
952
953 1----- FUNCTIONS -----
954 LABEL BYTE
955 0569 5E 5F 60 61 62 63 DB 94,95,96,97,98,99 ; F1 - F6
956 056A 64 65 66 67 FF FF DB 100,101,102,103,-1,-1 ; F7 - F10, NL, SL
957 056B 77 80 84 8E 73 8F DB 119,141,132,142,115,143 ; Home, Up, PgUp, -, Left, Pad5
958 056C 14 90 75 91 76 92 DB 116,144,117,145,118,146 ; Right, *, End, Down, PgDn, Ins
959 056D 93 FF FF FF 89 8A DB 147,-1,-1,-1,137,138 ; Del, SysReq, Undef, WT, F11, F12
960
961 1----- TABLES FOR LOWER CASE -----
962 1-----
963 LABEL BYTE
964 05B7 1B 31 32 33 34 35 DB 27,'12345'
965 05B8 36 37 38 39 30 2D DB '67890-'
966 05B9 3D 08 09 71 77 65 DB '=,08,09,'qw'
967 05BA 3C 74 79 75 69 6F DB 'rtyui'
968 05BB 10 5B 5D 50 51 4F DB 'p[ ],ODH,-1,'a' ; LETTERS, Return, Ctrl
969 05BC 73 64 66 67 68 6A DB 'adfgj'
970 05BD 6B 6C 3B 27 60 FF DB 'kl; ',-1 ; LETTERS, L Shift
971 05BE 5C 7A 78 63 76 62 DB '\xcv'
972 05BF 6E 6D 2C 2E 2F DB 'nm,/'
973 05C0 2A FF 20 FF DB -1,'*',-1,' ',-1 ; R Shift,*, Alt, Space, CL
974
975 1----- LC TABLE SCAN -----
976 LABEL BYTE
977 05F1 3B 3C 3D 3E 3F DB 59,60,61,62,63 ; BASE STATE OF F1 - F10
978 05F2 40 41 42 43 44 DB 64,65,66,67,68 ; NL, SL
979 05F3 FF FF
980
981 1----- KEYPAD TABLE -----
982 LABEL BYTE
983 05FD 71 72 73,-1,75,-1 ; BASE STATE OF KEYPAD KEYS
984 05FE 77,-1,79,80,81,82
985 0600 83
986 0601 -1,-1,'\',133,134 ; SysRq, Undef, WT, F11, F12
987
988 1----- TABLES FOR UPPER CASE -----
989 1-----
990 LABEL BYTE
991 060F 1B 21 40 23 24 25 DB 27,'0123456789'
992 0610 5E 26 2A 28 29 5F DB '\A'(-1)
993 0611 2B 08 00 51 57 45 DB '+,08,00,'QWE'
994 0612 52 54 59 55 49 4F DB 'RTYUIO'
995 0613 50 7B 7D 0D FF 41 DB 'P[ ],ODH,-1,'A' ; LETTERS, Return, Ctrl
996 0614 62 54 46 47 48 4A DB 'SDFGHJ'
997 0615 4B 4C 3A 22 7E FF DB 'KL; ',-1 ; LETTERS, L Shift
998 0616 39 7C 5A 68 43 56 42 DB '\ZXCVB'
999 0617 4E 4D 3C 3E 3F DB '\MBC?'
1000 0618 FF 2A FF 20 FF DB -1,'*',-1,' ',-1 ; R Shift,*, Alt, Space, CL
1001
1002 1----- UC TABLE SCAN -----
1003 LABEL BYTE
1004 0649 54 55 56 57 58 DB 84,85,86,87,88 ; SHIFTED STATE OF F1 - F10
1005 064A 59 5A 5B 5C 5D DB 89,90,91,92,93
1006 064B FF FF DB -1,-1
1007
1008 1----- NUM STATE TABLE -----
1009 LABEL BYTE
1010 0653 37 38 39 2D 34 35 DB '789-456+1230.' ; NUMLOCK STATE OF KEYPAD KEYS
1011 36 2B 31 32 33 30
1012 2E
1013 FF FF 7C 87 88 ; SysRq, Undef, WT, F11, F12
1014
1015 CODE ENDS
1016 END

```



```

1      PAGE 118,121
2      TITLE PRT ----- 01/10/86 PRINTER ADAPTER BIOS
3      .LIST
4
5      0000      CODE      SEGMENT BYTE PUBLIC
6
7      PUBLIC  PRINTER_IO_1
8
9      EXTRN  DDS:NEAR
10
11
12      INT 17 H
13      PRINTER_IO
14      THIS ROUTINE PROVIDES COMMUNICATION WITH THE PRINTER
15      INPUT
16      (AH)= 00H PRINT THE CHARACTER IN (AL)
17      ON RETURN, (AH)= 1 IF CHARACTER NOT PRINTED (TIME OUT)
18      OTHER BITS SET AS ON NORMAL STATUS CALL
19      (AH)= 01H INITIALIZE THE PRINTER PORT
20      RETURNS WITH (AH) SET WITH PRINTER STATUS
21      (AH)= 02H READ THE PRINTER STATUS INTO (AH)
22
23      7 6 5 4 3 2-1 0
24      | | | | | | |
25      | | | | | | |
26      | | | | | | |
27      | | | | | | |
28      | | | | | | |
29      | | | | | | |
30      | | | | | | |
31      | | | | | | |
32      | | | | | | |
33      | | | | | | |
34      | | | | | | |
35      | | | | | | |
36      | | | | | | |
37      | | | | | | |
38      | | | | | | |
39      | | | | | | |
40      | | | | | | |
41      | | | | | | |
42      | | | | | | |
43      | | | | | | |
44      | | | | | | |
45      | | | | | | |
46      | | | | | | |
47      | | | | | | |
48      | | | | | | |
49      | | | | | | |
50      | | | | | | |
51      | | | | | | |
52      | | | | | | |
53      | | | | | | |
54      | | | | | | |
55      | | | | | | |
56      | | | | | | |
57      | | | | | | |
58      | | | | | | |
59      | | | | | | |
60      | | | | | | |
61      | | | | | | |
62      | | | | | | |
63      | | | | | | |
64      | | | | | | |
65      | | | | | | |
66      | | | | | | |
67      | | | | | | |
68      | | | | | | |
69      | | | | | | |
70      | | | | | | |
71      | | | | | | |
72      | | | | | | |
73      | | | | | | |
74      | | | | | | |
75      | | | | | | |
76      | | | | | | |
77      | | | | | | |
78      | | | | | | |
79      | | | | | | |
80      | | | | | | |
81      | | | | | | |
82      | | | | | | |
83      | | | | | | |
84      | | | | | | |
85      | | | | | | |
86      | | | | | | |
87      | | | | | | |
88      | | | | | | |
89      | | | | | | |
90      | | | | | | |
91      | | | | | | |
92      | | | | | | |
93      | | | | | | |
94      | | | | | | |
95      | | | | | | |

```

(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

```

50      0000      ASSUME  CS:CODE,DS:DATA
51
52      PRINTER_IO_1  PROC  FAR
53
54      STI          ; INTERRUPTS BACK ON
55      PUSH DX      ; SAVE WORK REGISTERS
56      CMP DX,03H   ; CHECK FOR PRINTER NUMBER VALID 0-3
57      JA B10       ; ERROR EXIT IF OUT OF RANGE
58
59      MOV BH,AL     ; SAVE CHARACTER TO BE PRINTED
60      PUSH DS      ; SAVE SEGMENT
61      CALL DDS      ; ADDRESS DATA SEGMENT
62
63      PUSH SI       ; SAVE WORK POINTER REGISTER
64      MOV SI,DX     ; GET PRINTER PARAMETER
65      MOV BL,*PRINT_TIM_OUT[SI] ; LOAD TIMEOUT VALUE
66      SHL SI,1      ; WORD OFFSET INTO TABLE INTO (SI)
67      MOV DX,*PRINTER_BASE[SI] ; GET BASE ADDRESS FOR PRINTER CARD
68      POP SI        ; RECOVER CALLERS (SI) REGISTER
69      POP DS        ; AND (DS) SEGMENT REGISTER
70      ASSUME DS:NOTHING
71      OR DX,DX      ; TEST DX = ZERO, INDICATING NO PRINTER
72      JZ B10        ; EXIT, NO PRINTER ADAPTER AT OFFSET
73
74      OR AH,AH      ; TEST FOR (AH)= 00H
75      JZ B30        ; PRINT CHARACTER IN (AL)
76
77      DEC AH        ; TEST FOR (AH)= 01H
78      JZ B80        ; INITIALIZE PRINTER
79
80      DEC AH        ; TEST FOR (AH)= 02H
81      JZ B60        ; GET PRINTER STATUS
82
83      B10:         MOV AH,029H ; RETURN ERROR BITS FOR INVALID CALLS
84
85      B20:         POP BX      ; RETURN
86      POP DX       ; RECOVER REGISTERS
87      IRET        ; RETURN TO CALLING PROGRAM (AH)= STATUS
88
89
90
91      ;----- PRINT THE CHARACTER IN (AL)
92
93      B30:         OUT DX,AL   ; OUTPUT CHARACTER TO DATA PORT
94      INC DX       ; POINT TO STATUS PORT
95

```

```

96      PAGE
97      i----- CHECK FOR PRINTER BUSY
98
99      0034 EC      IN      AL,DX      ; PRE-CHARGE +BUSY LINE IF FLOATING
100     0035 EC      IN      AL,DX      ; GET STATUS PORT VALUE
101     0036 A8 E0    TEST     AL,B0H    ; IS THE PRINTER CURRENTLY BUSY
102     0038 75 05    JNZ     B40       ; SKIP SYSTEM DEVICE BUSY CALL IF NOT
103
104     i----- INT 15 H -- DEVICE BUSY
105
106     003A B8 90FE  MOV     AX,90FEH    ; FUNCTION 90 PRINTER ID
107     003D CD 15    INT      15H      ; SYSTEM CALL
108
109     i----- WAIT BUSY
110
111     003F          B40:    PUSH    CX      ; SAVE CALLERS (CX) REGISTER
112     003F 51      SUB     CX,CX        ; INNER LOOP (64K)
113     0040 2B C9    B45:    IN      AL,DX      ; GET STATUS
114     0042          MOV     AH,AL      ; STATUS TO (AH) ALSO
115     0042 EC      TEST     AL,B0H    ; IS THE PRINTER CURRENTLY BUSY
116     0043 8A E0    JNZ     B50       ; GO TO OUTPUT STROBE
117     0045 A8 B0    ;
118     0047 75 0F    ; LOOP IF NOT
119
120     0049 E2 F7    ; DECREMENT OUTER LOOP COUNT
121     004B FE CB    ; MAKE ANOTHER PASS IF NOT ZERO
122     004D 75 F3    ;
123
124     004F 59      POP     CX      ; RESTORE (CX) WITH CALLERS VALUE
125     0050 80 CC 01 OR      AH,1      ; SET ERROR FLAG
126     0053 80 E4 F9 AND     AH,0F9H    ; TURN OFF THE UNUSED BITS
127     0056 EB 15    JMP     SHORT B70   ; RETURN WITH ERROR FLAG SET
128
129     B50:          ; SEND STROBE PULSE
130     0058 59      POP     CX      ; RESTORE (CX) WITH CALLERS VALUE
131     0058 59      MOV     DX,0DH      ; SET THE STROBE LOW (BIT ON)
132     0059 B0 0D    INC     DX         ; OUTPUT STROBE TO CONTROL PORT
133     005B 42      CLT      ; PREVENT INTERRUPT PULSE STRETCHING
134     005C FA      OUT     DX,AL      ; OUTPUT STROBE BIT > 1us < 5us
135     005D EE      JMP     $+2        ; I/O DELAY TO ALLOW FOR LINE LOADING
136     005E EB 00    ; AND FOR CORRECT PULSE WIDTH
137     0060 B0 0C    MOV     AL,0CH      ; SET THE -STROBE HIGH
138     0062 EE      OUT     DX,AL      ;
139     0063 FB      STI      ; INTERRUPTS BACK ON
140     0064 4A      DEC     DX         ; ADJUST BACK TO BASE ADDRESS
141     0066 4A      DEC     DX         ; FOR STATUS ROUTINE EXIT
142
143     i----- PRINTER STATUS
144
145     B60:          ; POINT TO CONTROL PORT
146     0066 42      INC     DX         ; PRE-CHARGE +BUSY LINE IF FLOATING
147     0067 EC      IN      AL,DX      ; GET PRINTER STATUS HARDWARE BITS
148     0068 EC      IN      AL,DX      ; TURN OFF UNUSED BITS
149     0069 24 F8    AND     AL,0F8H    ; SAVE
150     006B 8A E0    MOV     AH,AL
151
152     B70:          ; RECOVER CHARACTER INTO (AL) REGISTER
153     006D          ; FLIP A COUPLE OF BITS IN STATUS
154     006D 8A C7    MOV     AL,BH      ; RETURN FROM ROUTINE WITH STATUS IN AH
155     006F 80 F4 48 XOR     AH,48H
156     0072 EB BB    JMP     B20
157
158     i----- INITIALIZE THE PRINTER PORT
159
160     B80:          ; POINT TO OUTPUT PORT
161     0074          INC     DX         ; SET INIT LINE LOW
162     0074 42      INC     DX         ;
163     0075 42      MOV     AL,8        ; ADJUST FOR INITIALIZATION DELAY LOOP
164     0076 B0 08    OUT     DX,AL      ;
165     0078 EE      MOV     AX,1000    ;
166     0079 B8 03EB B90:    DEC     AX      ; DECREMENT DELAY COUNTER
167     007C          JNZ     B90       ; LOOP FOR RESET TO TAKE
168     007C 48      DEC     AX         ;
169     007D 75 FD    ; NO INTERRUPTS, NON AUTO LF, INIT HIGH
170
171     007F B0 0C    MOV     AL,0CH      ; SET DEFAULT INITIAL OUTPUTS
172     0081 EE      OUT     DX,AL      ; ADJUST BACK TO BASE ADDRESS
173     0082 4A      DEC     DX         ; FOR STATUS ROUTINE EXIT
174     0083 4A      DEC     DX         ; EXIT THROUGH STATUS ROUTINE
175     0084 EB E0    JMP     B60
176
177     0086          PRINTER_IO_1 ENDP
178
179     0086          CODE ENDS
180     0086          ENDS

```

```

1 PAGE 118,121
2 TITLE RS232 ---- 01/10/86 COMMUNICATIONS BIOS (RS232)
3 .LIST
4 CODE SEGMENT BYTE PUBLIC
5
6 PUBLIC RS232_IO_1
7 EXTRN A1:NEAR
8 EXTRN D0S:NEAR
9
10
11 ;--- INT 14 H
12 ;RS232_IO
13 ; THIS ROUTINE PROVIDES BYTE STREAM I/O TO THE COMMUNICATIONS
14 ; PORT ACCORDING TO THE PARAMETERS:
15 ;
16 ; (AH)= 00H INITIALIZE THE COMMUNICATIONS PORT
17 ; (AL) HAS PARAMETERS FOR INITIALIZATION
18 ;
19 ; 7 6 5 4 3 2 1 0
20 ; ---- BAUD RATE -- -PARITY-- STOPBIT -WORD LENGTH--
21 ;
22 ; 000 - 110 X0 - NONE 0 - 1 10 - 7 BITS
23 ; 001 - 150 01 - ODD 1 - 2 11 - 8 BITS
24 ; 010 - 300 11 - EVEN
25 ; 011 - 600
26 ; 100 - 1200
27 ; 101 - 2400
28 ; 110 - 4800
29 ; 111 - 9600
30 ; ON RETURN, CONDITIONS SET AS IN CALL TO COMMO STATUS (AH=03H)
31 ;
32 ; (AH)= 01H SEND THE CHARACTER IN (AL) OVER THE COMMO LINE
33 ; (AL) REGISTER IS PRESERVED
34 ; ON EXIT, BIT 7 OF AH IS SET IF THE ROUTINE WAS UNABLE TO
35 ; TO TRANSMIT THE BYTE OF DATA OVER THE LINE.
36 ; IF BIT 7 OF AH IS NOT SET, THE
37 ; REMAINDER OF (AH) IS SET AS IN A STATUS REQUEST,
38 ; REFLECTING THE CURRENT STATUS OF THE LINE.
39 ; (AH)= 02H RECEIVE A CHARACTER IN (AL) FROM COMMO LINE BEFORE
40 ; RETURNING TO CALLER
41 ; ON EXIT, (AH) HAS THE CURRENT LINE STATUS, AS SET BY THE
42 ; THE STATUS ROUTINE, EXCEPT THAT ONLY BITS
43 ; LEFT ON ARE THE ERROR BITS (7,4,3,2,1)
44 ; IF (AH) HAS BIT 7 ON (TIME OUT) THE REMAINING
45 ; BITS ARE NOT PREDICTABLE
46 ; THUS, (AH) IS NON ZERO ONLY WHEN AN ERROR OCCURRED.
47 ; (AH)= 03H RETURN THE COMMO PORT STATUS IN (AX)
48 ; (AH) CONTAINS THE LINE CONTROL STATUS
49 ; BIT 7 = TIME OUT
50 ; BIT 6 = TRANSMIT SHIFT REGISTER EMPTY
51 ; BIT 5 = TRANSMIT HOLDING REGISTER EMPTY
52 ; BIT 4 = BREAK DETECT
53 ; BIT 3 = FRAMING ERROR
54 ; BIT 2 = PARITY ERROR
55 ; BIT 1 = OVERRUN ERROR
56 ; BIT 0 = DATA READY
57 ; (AL) CONTAINS THE MODEM STATUS
58 ; BIT 7 = RECEIVE LINE SIGNAL DETECT
59 ; BIT 6 = RING INDICATOR
60 ; BIT 5 = DATA SET READY
61 ; BIT 4 = CLEAR TO SEND
62 ; BIT 3 = DELTA RECEIVE LINE SIGNAL DETECT
63 ; BIT 2 = TRAILING EDGE RING DETECTOR
64 ; BIT 1 = DELTA DATA SET READY
65 ; BIT 0 = DELTA CLEAR TO SEND
66 ;
67 ; (DX) = PARAMETER INDICATING WHICH RS232 CARD (0,1 ALLOWED)
68 ;
69 ; DATA AREA #RS232 BASE CONTAINS THE BASE ADDRESS OF THE 8250 ON THE CARD
70 ; LOCATION 400H CONTAINS UP TO 4 RS232 ADDRESSES POSSIBLE
71 ; DATA AREA LABEL #RS232 TIM OUT (BYTE) CONTAINS OUTER LOOP COUNT
72 ; VALUE FOR TIMEOUT (DEFAULT=1)
73 ;
74 ; OUTPUT
75 ; AX MODIFIED ACCORDING TO PARAMETERS OF CALL
76 ;
77 ;-----
78 ASSUME CS:CODE,DS:DATA
79
80 RS232_IO_1 PROC FAR
81 ; INTERRUPTS BACK ON
82 ; SAVE SEGMENT
83 STI
84 PUSH DS
85 PUSH SI
86 PUSH DI
87 PUSH CX
88 PUSH BX
89 CMP DX,03H
90 JNC A3E
91 JNC A3E
92 MOV SI,DX
93 MOV DI,DX
94 SHL SI,1
95 CALL D0S
96 MOV DX,#RS232_BASE[SI]
97 OR DX,DX
98 JZ A3E
99 OR AH,AH
100 JZ A4
101 DEC AH
102 JZ A5
103 DEC AH
104 JZ A12
105
106 A2: DEC AH
107 JNZ A3E
108 JMP A1B
109
110 A3E: MOV AH,080H
111
112 A3: POP BX
113 POP CX
114 POP DI
115 POP SI
116 POP DX
117 IRET

```

```

115 PAGE
116 I----- INITIALIZE THE COMMUNICATIONS PORT
117
118 0039 A4:
119 0039 8A E0      MOV     AH,AL          ; SAVE INITIALIZATION PARAMETERS IN (AH)
120 003B 83 C2 03    ADD     DX,3          ; POINT TO 8250 CONTROL REGISTER
121 003E B0 80      MOV     AL,80H
122 0040 EE          OUT     DX,AL          ; SET DLAB=1
123
124 I----- DETERMINE BAUD RATE DIVISOR
125
126 0041 8A D4      MOV     DL,AH          ; GET PARAMETERS TO (DL)
127 0043 B1 04      MOV     CL,4
128 0045 D2 C2      ROL     CL,CL
129 0047 81 E2 00DE AND     DX,0EH
130 004B BF 0000 E   MOV     DI,OFFSET A1      ; ISOLATE THEM
131 004E 03 FA      ADD     DI,DX          ; BASE OF TABLE
132 0050 8B 94 0000 R MOV     DX,0RS232_BASE[SI] ; PUT INTO INDEX REGISTER
133 0054 42          INC     DX          ; POINT TO HIGH ORDER OF DIVISOR
134 0055 2E: 8A 45 01 MOV     AL,CS:[DI]+1      ; GET HIGH ORDER OF DIVISOR
135 0059 EE          OUT     DX,AL          ; SET ms OF DIVISOR TO 0
136 005A 4A          DEC     DX
137 005B 90          NOP
138 005C 2E: 8A 05   MOV     AL,CS:[DI]
139 005F EE          OUT     DX,AL          ; GET LOW ORDER OF DIVISOR
140 0060 83 C2 03    ADD     DX,3          ; SET LOW OF DIVISOR
141 0063 8A C4      MOV     AL,AH          ; GET PARAMETERS BACK
142 0065 24 1F      AND     AL,01FH        ; STRIP OFF THE BAUD BITS
143 0067 EE          OUT     DX,AL          ; LINE CONTROL TO 8 BITS
144 0068 4A          DEC     DX
145 0069 4A          DEC     DX
146 006A 90          NOP
147 006B 80 B0      MOV     AL,0          ; I/O DELAY
148 006D EE          OUT     DX,AL          ; INTERRUPT ENABLES ALL OFF
149 006E EB 4B      JMP     SHORT A18      ; COM_STATUS
150
151 I----- SEND CHARACTER IN (AL) OVER COMMO LINE
152
153 0070 A5:
154 0070 50          PUSH    AX          ; SAVE CHAR TO SEND
155 0071 83 C2 04    ADD     DX,4          ; MODEM CONTROL REGISTER
156 0074 B0 03      MOV     AL,3          ; DTR AND RTS
157 0076 EE          OUT     DX,AL          ; DATA TERMINAL READY, REQUEST TO SEND
158 0077 42          INC     DX          ; MODEM STATUS REGISTER
159 0078 42          INC     DX
160 0079 B7 30      MOV     BH,30H      ; DATA SET READY & CLEAR TO SEND
161 007B E8 00CA R   CALL    WAIT_FOR_STATUS    ; ARE BOTH TRUE
162 007E 74 08      JE      A9          ; YES, READY TO TRANSMIT CHAR
163 0080 A7:
164 0080 59          POP     CX          ; RELOAD DATA BYTE
165 0081 8A C1      MOV     AL,CL
166 0083 A8:
167 0083 80 CC 80    OR      AH,80H        ; INDICATE TIME OUT
168 0086 EB AA      JMP     A3          ; RETURN
169
170 0088 A9:
171 0088 4A          DEC     DX          ; CLEAR TO SEND
172 0089 A10:
173 0089 B7 20      MOV     BH,20H      ; LINE STATUS REGISTER
174 008B E8 00CA R   CALL    WAIT_FOR_STATUS    ; WAIT SEND
175 008E 75 F0      JNZ     A7          ; IS TRANSMITTER READY
176 0090 A11:
177 0090 50          SUB     DX,5          ; TEST FOR TRANSMITTER READY
178 0093 59          POP     CX          ; RETURN WITH TIME OUT SET
179 0094 8A C1      MOV     AL,CL          ; OUT CHAR
180 0096 EE          OUT     DX,AL          ; DATA PORT
181 0097 EB 99      JMP     A3          ; RECOVER IN CX TEMPORARILY
182
183 I----- RECEIVE CHARACTER FROM COMMO LINE
184
185 0099 A12:
186 0099 83 C2 04    ADD     DX,4          ; MODEM CONTROL REGISTER
187 009C B0 01      MOV     AL,1          ; DATA TERMINAL READY
188 009E EE          OUT     DX,AL
189 009F 42          INC     DX          ; MODEM STATUS REGISTER
190 00A0 42          INC     DX
191 00A1 A13:
192 00A1 B7 20      MOV     BH,20H      ; WAIT_DSR
193 00A3 E8 00CA R   CALL    WAIT_FOR_STATUS    ; DATA SET READY
194 00A6 75 DB      JNZ     A8          ; TEST FOR DSR
195 00AB A15:
196 00AB 4A          DEC     DX          ; RETURN WITH ERROR
197 00A9 A16:
198 00A9 B7 01      MOV     BH,1          ; WAIT_DSR_END
199 00AB E8 00CA R   CALL    WAIT_FOR_STATUS    ; LINE STATUS REGISTER
200 00AE 75 D3      JNZ     A8          ; WAIT RECV
201 00B0 A17:
202 00B0 80 E4 1E   AND     AH,0001110B      ; RECEIVE BUFFER FULL
203 00B3 8B 94 0000 R MOV     DX,0RS232_BASE[SI] ; TEST FOR RECEIVE BUFFER FULL
204 00B7 EC          IN      AL,DX          ; SET TIME OUT ERROR
205 00B8 E9 0032 R   JMP     A3          ; GET CHAR
206
207 I----- COMMO PORT STATUS ROUTINE
208
209 00BB A18:
210 00BB 8B 94 0000 R MOV     DX,0RS232_BASE[SI] ; DATA PORT
211 00BB 83 C2 05    ADD     DX,5          ; GET CHARACTER FROM LINE
212 00C2 EC          IN      AL,DX          ; RETURN
213 00C3 8A E0      MOV     AH,E0          ; GET MODEM CONTROL STATUS
214 00C5 42          INC     DX          ; RETURN
215 00C6 EC          IN      AL,DX
216 00C7 E9 0032 R   JMP     A3

```

```

218                                     PAGE
219                                     ;-----
220                                     ; WAIT FOR STATUS ROUTINE
221                                     ; ENTRY: (BH)= STATUS BIT(S) TO LOOK FOR
222                                     ; (DX)= ADDRESS OF STATUS REG
223                                     ; EXIT: ZERO FLAG ON = STATUS FOUND
224                                     ; ZERO FLAG OFF = TIMEOUT.
225                                     ; (AH)= LAST STATUS READ
226                                     ;-----
227
228 00CA                                WAIT_FOR_STATUS PROC    NEAR
229
230 00CA 8A 9D 007C R                    WFS0:  MOV     BL,0RS232_TIM_OUT[DI]    ; LOAD OUTER LOOP COUNT
231 00CE                                WFS1:  SUB     CX,CX
232 00CE 2B C9                          ; GET STATUS
233 00D0                                MOV     AH,AL
234 00D0 EC                                AND     AL,BH
235 00D1 8A E0                            MOV     AL,BH
236 00D3 22 C7                            CMP     AL,BH
237 00D5 3A C7                            JE      WFS_END
238 00D7 74 08                            ; RETURN WITH ZERO FLAG ON
239
240 00D9 E2 F5                            LOOP    WFS1
241
242 00DB FE CB                            DEC     BL
243 00DD 75 EF                            JNZ     WFS0
244
245 00DF 0A FF                            OR      BH,BH
246 00E1                                WFS_END: RET
247 00E1 C3
248
249 00E2                                WAIT_FOR_STATUS ENDP
250
251 00E2                                RS232_10_1 ENDP
252
253 00E2                                CODE ENDS
254                                END

```

```

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

```

```

115      (CX) = COUNT OF CHARACTERS TO WRITE
116      (AL) = CHAR TO WRITE
117      (BL) = ATTRIBUTE OF CHARACTER (ALPHA)/COLOR OF CHAR (GRAPHICS)
118      SEE NOTE ON WRITE DOT FOR BIT 7 OF BL = 1.
119      (AH) = 0AH WRITE CHARACTER ONLY AT CURRENT CURSOR POSITION
120      (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY)
121      (CX) = COUNT OF CHARACTERS TO WRITE
122      (AL) = CHAR TO WRITE
123      NOTE: USE FUNCTION (AH) = 09H IN GRAPHICS MODES
124      FOR READ/WRITE CHARACTER INTERFACE WHILE IN GRAPHICS MODE. THE
125      CHARACTERS ARE FORMED FROM A CHARACTER GENERATOR IMAGE
126      MAINTAINED IN THE SYSTEM ROM. ONLY THE 1ST 128 CHARS
127      ARE CONTAINED THERE. TO READ/WRITE THE SECOND 128 CHARS,
128      THE USER MUST INITIALIZE THE POINTER AT INTERRUPT 1FH
129      (LOCATION 0007CH) TO POINT TO THE 1K BYTE TABLE CONTAINING
130      THE CODE POINTS FOR THE SECOND 128 CHARS (128-255).
131      FOR WRITE CHARACTER INTERFACE IN GRAPHICS MODE, THE REPLICATION FACTOR
132      CONTAINED IN (CX) ON ENTRY WILL PRODUCE VALID RESULTS ONLY
133      FOR CHARACTERS CONTAINED ON THE SAME ROW. CONTINUATION TO
134      SUCCEEDING LINES WILL NOT PRODUCE CORRECTLY.
135
136      GRAPHICS INTERFACE
137
138      (AH) = 0BH SET COLOR PALETTE
139      (BH) = PALETTE COLOR ID BEING SET (0-127)
140      (BL) = COLOR VALUE TO BE USED WITH THAT COLOR ID
141      NOTE: FOR THE CURRENT COLOR CARD, THIS ENTRY POINT HAS
142      MEANING ONLY FOR 320X200 GRAPHICS.
143      COLOR ID = 0 SELECTS THE BACKGROUND COLOR (0-15)
144      COLOR ID = 1 SELECTS THE PALETTE TO BE USED:
145      0 = GREEN(1)/RED(2)/YELLOW(3)
146      1 = CYAN(1)/MAGENTA(2)/WHITE(3)
147      IN 40X25 OR 80X25 ALPHA MODES, THE VALUE SET FOR
148      PALETTE COLOR 0 INDICATES THE BORDER COLOR
149      TO BE USED (VALUES 0-31, WHERE 16-31 SELECT
150      THE HIGH INTENSITY BACKGROUND SET.
151      (AH) = 0CH WRITE DOT
152      (DX) = ROW NUMBER
153      (CX) = COLUMN NUMBER
154      (AL) = COLOR VALUE
155      IF BIT 7 OF AL = 1, THEN THE COLOR VALUE IS EXCLUSIVE
156      OR'ed WITH THE CURRENT CONTENTS OF THE DOT
157      (AH) = 0DH READ DOT
158      (DX) = ROW NUMBER
159      (CX) = COLUMN NUMBER
160      (AL) RETURNS THE DOT READ
161
162      ASCII TELETYPE ROUTINE FOR OUTPUT
163
164      (AH) = 0EH WRITE TELETYPE TO ACTIVE PAGE
165      (AL) = CHAR TO WRITE
166      (BL) = FOREGROUND COLOR IN GRAPHICS MODE
167      NOTE -- SCREEN WIDTH IS CONTROLLED BY PREVIOUS MODE SET
168      (AH) = 0FH CURRENT VIDEO STATE
169      RETURNS THE CURRENT VIDEO STATE
170      (AL) = MODE CURRENTLY SET (SEE (AH) = 00H FOR EXPLANATION)
171      (AH) = NUMBER OF CHARACTER COLUMNS ON SCREEN
172      (BH) = CURRENT ACTIVE DISPLAY PAGE
173      (AH) = 10H RESERVED
174      (AH) = 11H RESERVED
175      (AH) = 12H RESERVED
176      (AH) = 13H WRITE STRING
177      ES:BP - POINTER TO STRING TO BE WRITTEN
178      CX - LENGTH OF CHARACTER STRING TO WRITTEN
179      DX - CURSOR POSITION FOR STRING TO BE WRITTEN
180      BH - PAGE NUMBER
181      (AL) = 00H WRITE CHARACTER STRING
182      BL - ATTRIBUTE
183      STRING IS <CHAR,CHAR, ... ,CHAR>
184      CURSOR NOT MOVED
185      (AL) = 01H WRITE CHARACTER STRING AND MOVE CURSOR
186      BL - ATTRIBUTE
187      STRING IS <CHAR,CHAR, ... ,CHAR>
188      CURSOR IS MOVED
189      (AL) = 02H WRITE CHARACTER AND ATTRIBUTE STRING
190      (VALID FOR ALPHA MODES ONLY)
191      STRING IS <CHAR,ATTR,CHAR,ATTR .. ,CHAR,ATTR>
192      CURSOR IS NOT MOVED
193      (AL) = 03H WRITE CHARACTER AND ATTRIBUTE STRING AND MOVE CURSOR
194      (VALID FOR ALPHA MODES ONLY)
195      STRING IS <CHAR,ATTR,CHAR,ATTR .. ,CHAR,ATTR>
196      CURSOR IS MOVED
197      NOTE: CARRIAGE RETURN, LINE FEED, BACKSPACE, AND BELL ARE
198      TREATED AS COMMANDS RATHER THAN PRINTABLE CHARACTERS.
199
200      BX,CX,DX,S1,D1,BP,SP,DS,ES,SS PRESERVED DURING CALLS EXCEPT FOR
201      BX,CX,DX RETURN VALUES ON FUNCTIONS 03H,04H,0DH AND 0DH. ON ALL CALLS
202      AX IS MODIFIED.
203
204      -----
205      ASSUME CS:CODE,DS:DATA,ES:NOTHING
206
207      MI DW OFFSET SET_MODE ; TABLE OF ROUTINES WITHIN VIDEO I/O
208      DW OFFSET SET_CTYPE
209      DW OFFSET SET_CPOS
210      DW OFFSET READ_CURSOR
211      DW OFFSET READ_PAGE
212      DW OFFSET ACT_DISP_PAGE
213      DW OFFSET SCROLL_UP
214      DW OFFSET SCROLL_DOWN
215      DW OFFSET READ_AC_CURRENT
216      DW OFFSET WRITE_AC_CURRENT
217      DW OFFSET WRITE_C_CURRENT
218      DW OFFSET SET_COLOR
219      DW OFFSET WRITE_DOT
220      DW OFFSET READ_DOT
221      DW OFFSET WRITE_TTY
222      DW OFFSET VIDEO_STATE
223      DW OFFSET VIDEO_RETURN ; RESERVED
224      DW OFFSET VIDEO_RETURN ; RESERVED
225      DW OFFSET VIDEO_RETURN ; RESERVED
226      DW OFFSET VIDEO_STRING ; CASE 13H, WRITE STRING
227      EQU $-MI
228

```

```
229 0028 VIDEO_10_1 PROC NEAR ; ENTRY POINT FOR ORG 0F065H
230 0028 FB STI ; INTERRUPTS BACK ON
231 0029 FC CLD ; SET DIRECTION FORWARD
232 002A 80 FC 14 CMP AH,MIL/2 ; TEST FOR WITHIN TABLE RANGE
233 002D 73 2F JNB M4 ; BRANCH TO EXIT IF NOT A VALID COMMAND
234
235 002F 06 PUSH ES
236 0030 1E PUSH DS ; SAVE WORK AND PARAMETER REGISTERS
237 0031 52 PUSH DX
238 0032 51 PUSH CX
239 0033 53 PUSH BX
240 0034 56 PUSH SI
241 0035 57 PUSH DI
242 0036 55 PUSH BP
243 0037 BE ---- R MOV SI,DATA ; POINT DS: TO DATA SEGMENT
244 003A 8E DE MOV DS,SI
245 003C 8B F0 MOV SI,AX ; SAVE COMMAND/DATA INTO (SI) REGISTER
246 003E A0 0010 R MOV AL,BYTE PTR @EQUIP_FLAG ; GET THE EQUIPMENT FLAG VIDEO BITS
247 0041 24 30 AND AL,30H ; ISOLATE CRT SWITCHES
248 0043 3C 40 CMP AL,30H ; IS SETTING FOR MONOCHROME CARD?
249 0045 BF B800 MOV DI,0B800H ; GET SEGMENT FOR COLOR CARD
250 0048 75 03 JNE M2 ; SKIP IF NOT MONOCHROME CARD
251 004A BF B000 MOV DI,0B000H ; ELSE GET SEGMENT FOR MONOCHROME CARD
252 004D C7AD M2: MOV ES,DI ; SET UP TO POINT AT VIDEO MEMORY AREAS
253 004D 8E C7 MOV AL,AH ; PLACE COMMAND IN LOW BYTE OF (AX)
254 004F 8A C4 AND AL,30H ; AND FORM BYTE OFFSET WITH COMMAND
255 0051 98 CBW ; TIMES 2 FOR WORD TABLE LOOKUP
256 0052 D1 E0 SAL AX,1 ; MOVE OFFSET INTO LOOK UP REGISTER (SI)
257 0054 96 XCHG SI,AX ; AND RESTORE COMMAND/DATA INTO (AX)
258 ; MOVE CURRENT MODE INTO (AH) REGISTER
259 0055 8A 26 0049 R MOV AH,@CRT_MODE
260 ; JUMP WORD PTR CS:[SI-OFFSET MI] ; GO TO SELECTED FUNCTION
261 0059 2E: FF A4 0000 R JMP WORD PTR CS:[SI-OFFSET MI]
262
263 005E M4: ; COMMAND NOT VALID
264 005E CF IRET ; DO NOTHING IF NOT IN VALID RANGE
265 005F VIDEO_10_1 ENDP
266
267 ; SET_MODE
268 ; THIS ROUTINE INITIALIZES THE ATTACHMENT TO
269 ; THE SELECTED MODE. THE SCREEN IS BLANKED.
270 ; INPUT
271 ; @EQUIP_FLAG BITS 5-4 = MODE/WIDTH
272 ; 1 = MONOCHROME (FORCES MODE 7)
273 ; 01 = COLOR ADAPTER 40x25 (MODE 0 DEFAULT)
274 ; 10 = COLOR ADAPTER 80x25 (MODE 2 DEFAULT)
275 ; (AL) = COLOR MODE REQUESTED ( RANGE 0 - 6 )
276 ; OUTPUT
277 ; NONE
278
279 005F SET_MODE PROC NEAR
280 005F BA 03D4 MOV DX,03D4H ; ADDRESS OF COLOR CARD
281 0062 8B 3E 0010 R MOV DI,@EQUIP_FLAG ; GET EQUIPMENT FLAGS SETTING
282 0066 81 E7 0030 AND DI,30H ; ISOLATE CRT SWITCHES
283 006A 83 FF 30 CMP DI,30H ; IS BW CARD INSTALLED AS PRIMARY
284 006D 78 06 M8: JNB JNE ; SKIP AND CHECK IF COLOR
285 006F 80 07 MOV AL,7 ; ELSE INDICATE INTERNAL BW CARD MODE
286 0071 B2 B4 MOV DL,0B4H ; SET ADDRESS OF BW (MONOCHROME) CARD
287 0073 EB D0 JMP SHORT M8 ; CONTINUE WITH FORCED MODE 7
288
289 0075 3C 07 CMP AL,7 ; CHECK FOR VALID COLOR MODES 0-6
290 0077 72 09 JB M8 ; CONTINUE IF BELOW MODE 7
291 0079 B0 00 MOV AL,0 ; FORCE DEFAULT 40x25 BW MODE
292 007B 83 FF 20 CMP DI,20H ; CHECK FOR @EQUIP_FLAG AT 80x25 BW
293 007E 74 02 JE M8 ; CONTINUE WITH MODE 0 IF NOT
294 0080 B0 A2 MOV AL,2 ; ELSE FORCE MODE 2
295
296 0082 A2 0049 R M8: MOV @CRT_MODE,AL ; SAVE MODE IN GLOBAL VARIABLE
297 0085 89 16 0063 R MOV @ADDR_6845,DX ; SAVE ADDRESS OF BASE
298 0089 C6 06 0084 R 18 MOV @ROWS,25-1 ; INITIALIZE DEFAULT ROW COUNT OF 25
299 008E 1E M8: DS PUSH ; SAVE POINTER TO DATA SEGMENT
300 008F 50 PUSH AX ; SAVE MODE NUMBER (AL)
301 0090 98 CBW ; CLEAR HIGH BYTE OF MODE
302 0091 8B F0 MOV SI,AX ; SET TABLE POINTER, INDEXED BY MODE
303 0093 2E: 8A 84 0000 E MOV AL,CS:[SI + OFFSET M7] ; GET THE MODE SET VALUE FROM TABLE
304 0095 A2 0065 R MOV @CRT_MODE_SET,AL ; SAVE THE MODE SET VALUE
305 009B 24 37 AND AL,037H ; VIDEO OFF, SAVE HIGH RESOLUTION BIT
306 009D 52 DS PUSH ; SAVE OUTPUT PORT VALUE
307 009E 83 C2 04 ADD DX,4 ; POINT TO CONTROL REGISTER
308 00A1 EE OUT DX,AL ; RESET VIDEO TO OFF TO SUPPRESS ROLLING
309 00A2 5A POP DX ; BACK TO BASE REGISTER
310
311 00A3 2B DB SUB BX,BX ; SET UP FOR AB50 SEGMENT
312 00A5 8E DB MOV DS,BX ; ESTABLISH VECTOR TABLE ADDRESSING
313 00A7 C5 1E 0074 R LDS BX,@PARM_PTR ; GET POINTER TO VIDEO PARMS
314
315 00AB 58 ASSUME DS:CODE
316 00AC B9 0010 POP AX ; RECOVER MODE NUMBER IN (AL)
317 00AF 3C 02 CMP CX,16 ; LENGTH OF EACH ROW OF TABLE
318 00B1 72 0E JC M9 ; DETERMINE WHICH ONE TO USE
319 00B3 03 D9 ADD BX,CX ; NEXT ROW OF INITIALIZATION TABLE
320 00B5 3C 04 CMP AL,4 ;
321 00B7 72 08 JC M9 ; MODE IS 2 OR 3
322 00B9 03 D9 ADD BX,CX ; MOVE TO GRAPHICS ROW OF INIT_TABLE
323 00BB 3C 07 CMP AL,7 ;
324 00BD 72 02 JC M9 ; MODE IS 4,5, OR 6
325 00BF 03 D9 ADD BX,CX ; MOVE TO BW CARD ROW OF INIT_TABLE
326
327 ;----- BX POINTS TO CORRECT ROW OF INITIALIZATION TABLE
328
329 00C1 M9: ; OUT_INIT
330 00C1 50 PUSH AX ; SAVE MODE IN (AL)
331 00C2 8B 47 0A MOV AX,[BX+10] ; GET THE CURSOR MODE FROM THE TABLE
332 00C5 86 E0 XCHG AX,AL ; PUT CURSOR MODE IN CORRECT POSITION
333 00C7 1E DS PUSH ; SAVE TABLE SEGMENT POINTER
334
335 00C8 E8 0000 E CALL D05 ; POINT DS TO DATA SEGMENT
336 00CB A3 0060 R MOV @CURSOR_MODE,AX ; PLACE INTO BIOS DATA SAVE AREA
337
338 00CE 1F POP DS ; RESTORE THE TABLE SEGMENT POINTER
339 00CF C2 E4 XOR AH,AH ; AH IS REGISTER NUMBER DURING LOOP
340
341 ;----- LOOP THROUGH TABLE, OUTPUTTING REGISTER ADDRESS, THEN VALUE FROM TABLE
342
```



```

343 00D1
344 00D1 8A C4      M10:      MOV     AL,AH          ; INITIALIZATION LOOP
345 00D3 EE          OUT     DX,AL          ; GET 6845 REGISTER NUMBER
346 00D4 42          INC     DX              ; POINT TO DATA PORT
347 00D5 FE C4      INC     AH              ; NEXT REGISTER VALUE
348 00D7 8A 07      MOV     AL,[BX]         ; GET TABLE VALUE
349 00D9 EE          OUT     DX,AL          ; OUT TO CHIP
350 00DA 43          INC     BX              ; NEXT IN TABLE
351 00DB 4A          DEC     DX              ; BACK TO POINTER REGISTER
352 00DC E2 F3      LOOP    M10           ; DO THE WHOLE TABLE
353 00DE 58          POP     AX              ; GET MODE BACK INTO (AL)
354 00DF 1F          POP     DS              ; RECOVER SEGMENT VALUE
355
356
357 ;----- FILL REGEN AREA WITH BLANK
358
359 00E0 33 FF      XOR     DI,DI          ; SET UP POINTER FOR REGEN
360 00E2 89 3E 004E R MOV     DI,CRT_START,DI ; START ADDRESS SAVED IN GLOBAL
361 00E6 C6 06 0062 R 00 MOV     DI,ACTIVE_PAGE,0 ; SET PAGE VALUE
362 00EB B9 2000      MOV     CX,8192         ; NUMBER OF WORDS IN COLOR CARD
363 00EE 3C 04      CMP     AL,4           ; TEST FOR GRAPHICS
364 00F0 72 0A      JC      M12           ; NO GRAPHICS INIT
365 00F2 3C 07      CMP     AL,7           ; TEST FOR BW CARD
366 00F4 74 04      JBE     M11          ; BW CARD INIT
367 00F6 33 C0      XOR     AX,AX          ; FILL FOR GRAPHICS MODE
368 00F8 EB 05      JMP     SHORT M13       ; CLEAR BUFFER
369 00FA          MOV     CH,08H        ; BW CARD INIT
370 00FB B5 08      MOV     AX,AX          ; BUFFER SIZE ON BW CARD (2048)
371 00FC          M12:      MOV     AX,' '*7*MH ; NO GRAPHICS INIT
372 00FC B8 0720      MOV     AX,' '*7*MH ; FILL CHAR FOR ALPHA + ATTRIBUTE
373 00FF          M13:      REP     STOSW      ; CLEAR BUFFER
374 00FF F3/ AB      ; FILL THE REGEN BUFFER WITH BLANKS
375
376 ;----- ENABLE VIDEO AND CORRECT PORT SETTING
377
378 0101 8B 16 0063 R MOV     DX,#ADDR_6845   ; PREPARE TO OUTPUT TO VIDEO ENABLE PORT
379 0105 83 C2 04      ADD     DX,4            ; POINT TO THE MODE CONTROL REGISTER
380 0108 A0 0065 R      MOV     AL,CRT_MODE_SET ; GET THE MODE SET VALUE
381 010B EE          OUT     DX,AL          ; SET VIDEO ENABLE PORT
382
383 ;----- DETERMINE NUMBER OF COLUMNS, BOTH FOR ENTIRE DISPLAY
384 ;----- AND THE NUMBER TO BE USED FOR TTY INTERFACE
385
386 010C 2E: 8A 84 0000 E MOV     AL,CS:[SI + OFFSET M6] ; GET NUMBER OF COLUMNS ON THIS SCREEN
387 0111 98          CMBW          ; CLEAR HIGH BYTE
388 0112 A3 004A R      MOV     CRT_COLS,AX     ; INITIALIZE NUMBER OF COLUMNS COUNT
389
390 ;----- SET CURSOR POSITIONS
391
392 0115 81 E6 000E      AND     SI,000EH        ; WORD OFFSET INTO CLEAR LENGTH TABLE
393 0119 2E: 8B 84 0000 E AX,CS:[SI + OFFSET M5] ; LENGTH TO CLEAR
394 011E A3 004C R      MOV     CRT_LEN,AX     ; SAVE LENGTH OF CRT -- NOT USED FOR BW
395 0121 B9 0008      MOV     CX,8            ; CLEAR ALL CURSOR POSITIONS
396 0124 B4 0050 R      MOV     M1,OFFSET_CURSOR_POSN ; ESTABLISH SEGMENT
397 0127 1E          DS      PUSH     DS ; ADDRESSING
398 0128 07          POP     ES
399 0129 33 C0      XOR     AX,AX          ; FILL WITH ZEROES
400 012B F3/ AB      REP     STOSW
401
402 ;----- SET UP OVERSCAN REGISTER
403
404 012D 42          INC     DX              ; SET OVERSCAN PORT TO A DEFAULT
405 012E B0 30      MOV     AL,30H         ; 30H VALUE FOR ALL MODES EXCEPT 640X200
406 0130 80 3E 0049 R 06 CMP     CRT_MODE,6       ; SEE IF THE MODE IS 640X200 BW
407 0135 75 02      JNZ     M14           ; IF NOT 640X200, THEN GO TO REGULAR
408 0137 B0 3F      MOV     AL,3FH         ; IF IT IS 640X200, THEN PUT IN 3FH
409 0139          M14:      OUT     DX,AL          ; OUTPUT THE CORRECT VALUE TO 309 PORT
410 0139 EE          MOV     CRT_PALETTE,AL ; SAVE THE VALUE FOR FUTURE USE
411 013A A2 0066 R
412
413 ;----- NORMAL RETURN FROM ALL VIDEO RETURNS
414
415 013D          VIDEO_RETURN:
416 013D 5D          POP     BP
417 013E 5F          POP     DI
418 013F 5E          POP     SI
419 0140 5B          POP     BX
420 0141          M15:      ; VIDEO_RETURN_C
421 0141 59          POP     CX
422 0142 5A          POP     DX
423 0143 1F          POP     DS
424 0144 07          POP     ES
425 0145 CF          IRET          ; RECOVER SEGMENTS
426 0146          ; ALL DONE
427
428 ;----- SET_CTYPE
429 ; THIS ROUTINE SETS THE CURSOR VALUE
430 ; INPUT (CX) HAS CURSOR VALUE CH-START LINE, CL-STOP LINE
431 ; OUTPUT
432 ; NONE
433
434 ;----- SET_CTYPE PROC NEAR
435 0146          MOV     AH,10          ; 6845 REGISTER FOR CURSOR SET
436 0146 B4 0A      MOV     CURSOR_MODE,CX    ; SAVE IN DATA AREA
437 0148 B9 0E 0060 R CALL     M16             ; OUTPUT CX REGISTER
438 014C E8 0151 R      JMP     VIDEO_RETURN
439 014F EB EC
440
441 ;----- THIS ROUTINE OUTPUTS THE CX REGISTER TO THE 6845 REGISTERS NAMED IN (AH)
442
443 0151          M16:      MOV     DX,#ADDR_6845   ; ADDRESS REGISTER
444 0151 8B 16 0063 R MOV     AL,AH          ; GET VALUE
445 0155 8A C4      OUT     DX,AL          ; REGISTER SET
446 0157 EE          INC     DX              ; DATA REGISTER
447 0158 42          INC     DX              ; DATA
448 0158 42          MOV     AL,CH          ; DATA
449 0159 8A C5      OUT     DX,AL          ; DATA
450 015B EE          DEC     DX
451 015C 4A          INC     AL,AH          ; POINT TO OTHER DATA REGISTER
452 015D 8A C4      OUT     DX,AL          ; SET FOR SECOND REGISTER
453 015F FE C0      INC     DX
454 0161 EE          INC     DX
455 0162 42          MOV     AL,CL          ; SECOND DATA VALUE
456 0163 8A C1

```

```

457 0165 EE          OUT      DX,AL
458 0166 C3          RET
459 0167             SET_CTYPE      ENDP
460
461 -----
462 ; SET_CPOS
463 ; THIS ROUTINE SETS THE CURRENT CURSOR POSITION TO THE
464 ; NEW X-Y VALUES PASSED
465 ; INPUT
466 ;   DX - ROW,COLUMN OF NEW CURSOR
467 ;   BH - DISPLAY PAGE OF CURSOR
468 ; OUTPUT
469 ;   CURSOR IS SET AT 6845 IF DISPLAY PAGE IS CURRENT DISPLAY
470 -----
471 0167             SET_CPOS      PROC    NEAR
472 0167 8A C7        MOV     AL,BH          ; MOVE PAGE NUMBER TO WORK REGISTER
473 0169 98          CBW          ; CONVERT PAGE TO WORD VALUE
474 016A D1 E0        SAL      AX,1        ; WORD OFFSET
475 016C 96          XCHG     AX,SI        ; USE INDEX REGISTER
476 016D 89 94 0050 R MOV     [SI+OFFSET *CURSOR_POSN],DX ; SAVE THE POINTER
477 0171 38 3E 0062 R CMP     *ACTIVE_PAGE,BH
478 0175 75 05        JNZ     M17         ; SET_CPOS_RETURN
479 0177 8B C2        MOV     AX,DX        ; GET ROW/COLUMN TO AX
480 0179 E8 017E R    CALL    M18         ; CURSOR_SET
481 017C             M17:         JUMP     VIDEO_RETURN ; SET_CPOS_RETURN
482 017C EB BF        SET_CPOS      ENDP
483 017E
484
485 -----
486 ; ---- SET CURSOR POSITION, AX HAS ROW/COLUMN FOR CURSOR
487 017E             M18         PROC    NEAR
488 017E E8 0200 R    CALL    POSITION          ; DETERMINE LOCATION IN REGEN BUFFER
489 0181 8B C8        MOV     CX,AX
490 0183 03 0E 004E R ADD     CX,*CRT_START      ; ADD IN THE START ADDRESS FOR THIS PAGE
491 0187 D1 F9        SAR     CX,1        ; DIVIDE BY 2 FOR CHAR ONLY COUNT
492 0189 B4 0E        MOV     AH,14      ; REGISTER NUMBER FOR CURSOR
493 018B E8 0151 R    CALL    M16         ; OUTPUT THE VALUE TO THE 6845
494 018E C3          RET
495 018F             M18         ENDP
496
497 -----
498 ; READ_CURSOR
499 ; THIS ROUTINE READS THE CURRENT CURSOR VALUE FROM THE
500 ; 6845, FORMATS IT, AND SENDS IT BACK TO THE CALLER
501 ; INPUT
502 ;   BH - PAGE OF CURSOR
503 ; OUTPUT
504 ;   DX - ROW, COLUMN OF THE CURRENT CURSOR POSITION
505 ;   CX - CURRENT CURSOR MODE
506 -----
507 018F             READ_CURSOR  PROC    NEAR
508 018F 8A DF        MOV     BL,BH
509 0193 D1 E3        SAL      BX,1        ; WORD OFFSET
510 0195 8B 97 0050 R MOV     DX,[BX+OFFSET *CURSOR_POSN]
511 0199 8B 0E 0060 R MOV     CX,*CURSOR_MODE
512 019D 5D          POP      BP
513 019E 5F          POP      DI
514 019F 5E          POP      SI
515 01A0 5B          POP      BX
516 01A1 58          POP      AX          ; DISCARD SAVED CX AND DX
517 01A2 58          POP      AX
518 01A3 1F          POP      DS
519 01A4 07          POP      ES
520 01A5 CF          IRET
521 01A6             READ_CURSOR  ENDP
522
523 -----
524 ; ACT_DISP_PAGE
525 ; THIS ROUTINE SETS THE ACTIVE DISPLAY PAGE, ALLOWING
526 ; THE FULL USE OF THE MEMORY SET ASIDE FOR THE VIDEO ATTACHMENT
527 ; INPUT
528 ;   AL HAS THE NEW ACTIVE DISPLAY PAGE
529 ; OUTPUT
530 ;   THE 6845 IS RESET TO DISPLAY THAT PAGE
531 -----
532 01A6             ACT_DISP_PAGE PROC    NEAR
533 01A6 A2 0062 R    MOV     *ACTIVE_PAGE,AL      ; SAVE ACTIVE PAGE VALUE
534 01A9 98          CBW          ; CONVERT (AL) TO WORD
535 01AB F7 26 004C R PUSH     BH,BH          ; SAVE PAGE VALUE
536 01AF A3 004E R    MOV     *CRT_START,AX      ; DISPLAY PAGE TIMES REGEN LENGTH
537 01B2 8B C8        MOV     CX,AX          ; SAVE START ADDRESS FOR LATER
538 01B4 D1 F9        SAR     CX,1        ; START ADDRESS TO CX
539 01B6 B4 0C        MOV     AH,12      ; DIVIDE BY 2 FOR 6845 HANDLING
540 01B8 E8 0151 R    CALL    M16         ; 6845 REGISTER FOR START ADDRESS
541 01BB 5B          POP      BX          ; RECOVER PAGE VALUE
542 01BC D1 E3        SAL      BX,1        ; *2 FOR WORD OFFSET
543 01BE 8B 87 0050 R MOV     AX,[BX + OFFSET *CURSOR_POSN] ; GET CURSOR FOR THIS PAGE
544 01C2 E8 017E R    CALL    M18         ; SET THE CURSOR POSITION
545 01C5 E9 013D R    JMP     VIDEO_RETURN
546 01C8             ACT_DISP_PAGE ENDP
547
548 -----
549 ; SET_COLOR
550 ; THIS ROUTINE WILL ESTABLISH THE BACKGROUND COLOR, THE OVERSCAN COLOR,
551 ; AND THE FOREGROUND COLOR SET FOR MEDIUM RESOLUTION GRAPHICS
552 ; INPUT
553 ;   (BH) HAS COLOR ID
554 ;       IF BH=0, THE BACKGROUND COLOR VALUE IS SET
555 ;       FROM THE LOW BITS OF BL (0-31)
556 ;       IF BH=1, THE PALETTE SELECTION IS MADE
557 ;       BASED ON THE LOW BIT OF BL
558 ;       0 = GREEN, RED, YELLOW FOR COLORS 1,2,3
559 ;       1 = BLUE, CYAN, MAGENTA FOR COLORS 1,2,3
560 ; OUTPUT
561 ;   THE COLOR SELECTION IS UPDATED
562 -----
563 01C8             SET_COLOR    PROC    NEAR
564 01C8 8B 16 0063 R MOV     DX,*ADDR_6845      ; I/O PORT FOR PALETTE
565 01CC 83 C2 05      ADD     DX,5          ; OVERSCAN PORT
566 01CF A0 0066 R    MOV     AL,*CRT_PALETTE      ; GET THE CURRENT PALETTE VALUE
567 01D2 0A FF        OR      BH,BH          ; IS THIS COLOR 0?
568 01D4 75 0E        JNZ     M20         ; OUTPUT COLOR 1
569
570 ; ---- HANDLE COLOR 0 BY SETTING THE BACKGROUND COLOR

```

```
571
572 01D6 24 E0          AND     AL,0E0H          ; TURN OFF LOW 5 BITS OF CURRENT
573 01D8 80 E3 IF      OR      BL,01FH          ; TURN OFF HIGH 3 BITS OF INPUT VALUE
574 01DB 0A C3        M19:  OR      AL,BL          ; PUT VALUE INTO REGISTER
575 01DD              OUT      DX,AL          ; OUTPUT THE PALETTE
576 01DD EE          MOV      MOV     CRT_PALETTE,AL ; OUTPUT COLOR SELECTION TO 3D9 PORT
577 01E2 A2 0066 R    JMP      VIDEO_RETURN ; SAVE THE COLOR VALUE
578 01E1 E9 013D R
579
580 ;----- HANDLE COLOR 1 BY SELECTING THE PALETTE TO BE USED
581
582 01E4 24 DF      M20:  AND     AL,0DFH          ; TURN OFF PALETTE SELECT BIT
583 01E6 D0 EF      SHR      BL,1          ; TEST THE LOW ORDER BIT OF BL
584 01E8 13 F3      JNC      M19          ; ALREADY DONE
585 01E8 13 F3      OR      AL,20H          ; TURN ON PALETTE SELECT BIT
586 01EA 0C 20      JMP      M19          ; GO DO IT
587 01EC EB EF
588 01EE
589
590 ;-----
591 ; VIDEO STATE
592 ; RETURNS THE CURRENT VIDEO STATE IN AX
593 ; AH = NUMBER OF COLUMNS ON THE SCREEN
594 ; AL = CURRENT VIDEO MODE
595 ; BH = CURRENT ACTIVE PAGE
596
597 01EE 8A 26 004A R  VIDEO_STATE PROC NEAR
598 01F2 A0 0049 R    MOV      AH,BYTE PTR CRT_COLS ; GET NUMBER OF COLUMNS
599 01F5 8A 3E 0062 R MOV      AL,CRT_MODE ; CURRENT MODE
600 01F9 5F          POP      BH,ACTIVE_PAGE ; GET CURRENT ACTIVE PAGE
601 01FA 5F          POP      BP ; RECOVER REGISTERS
602 01FB 5E          POP      SI
603 01FC 59          POP      CX
604 01FD E9 0141 R  JMP      M15 ; DISCARD SAVED BX
605 0200          ENDP ; RETURN TO CALLER
606
607 ;-----
608 ; POSITION
609 ; THIS SERVICE ROUTINE CALCULATES THE REGEN BUFFER ADDRESS
610 ; OF A CHARACTER IN THE ALPHA MODE
611 ; INPUT
612 ; AX = ROW, COLUMN POSITION
613 ; AX = OFFSET OF CHAR POSITION IN REGEN BUFFER
614
615 0200          POSITION PROC NEAR
616 0200 53          PUSH     BX ; SAVE REGISTER
617 0201 93          XCHG     BX,AX ; SAVE ROW/COLUMN POSITION IN (BX)
618 0202 A0 004A R  MOV      AL,BYTE PTR CRT_COLS ; GET COLUMNS PER ROW COUNT
619 0205 F6 ET      MUL      BH ; DETERMINE BYTES TO ROW
620 0207 32 FF      XOR      BH,BH
621 0209 03 C3      ADD      AX,BX ; ADD IN COLUMN VALUE
622 020B D1 E0      SAL      AX,1 ; * 2 FOR ATTRIBUTE BYTES
623 020D 5B          POP      BX
624 020E C3          RET
625 020F          POSITION ENDP
626
627 ;-----
628 ; SCROLL UP
629 ; THIS ROUTINE MOVES A BLOCK OF CHARACTERS UP
630 ; ON THE SCREEN
631 ; INPUT
632 ; (AH) = CURRENT CRT MODE
633 ; (AL) = NUMBER OF ROWS TO SCROLL
634 ; (CX) = ROW/COLUMN OF UPPER LEFT CORNER
635 ; (DX) = ROW/COLUMN OF LOWER RIGHT CORNER
636 ; (BH) = ATTRIBUTE TO BE USED ON BLANKED LINE
637 ; (DS) = DATA SEGMENT
638 ; (ES) = REGEN BUFFER SEGMENT
639 ; OUTPUT
640 ; NONE -- THE REGEN BUFFER IS MODIFIED
641
642 020F          ASSUME DS:DATA,ES:DATA
643          SCROLL_UP PROC NEAR
644 020F E8 02EA R    CALL     TEST_LINE_COUNT ; TEST FOR GRAPHICS MODE
645 0212 80 FC 04    CMP      N1,4 ; HANDLE SEPARATELY
646 0215 72 08      JC      N1 ; TEST FOR BW CARD
647 0217 80 FC 07    CMP      N1,7
648 021A 74 03      JE      N1
649 021C E9 04AC R  JMP      GRAPHICS_UP
650 021F
651 021F 53          N1:  PUSH     BX ; UP CONTINUE
652 0220 8B C1      MOV      MOV     AX,CX ; SAVE FILL ATTRIBUTE IN BH
653 0222 E8 025C R  CALL     SCROLL_POSITION ; UPPER LEFT POSITION
654 0225 74 31      JZ      N7 ; DO SETUP FOR SCROLL
655 0227 03 F6      JZ      N7 ; BLANK FIELD
656 0229 8A E0      ADD      SI,AX ; FROM ADDRESS
657 022B 8A E6      MOV      AH,DH ; # ROWS IN BLOCK
658 022D          SUB      AH,BL ; # ROWS TO BE MOVED
659 022D          N2:  CALL     N10 ; ROW LOOP
660 0230 03 F5      ADD      D1,BP ; MOVE ONE ROW
661 0232 03 FD      ADD      D1,BP
662 0234 FE CC      DEC      AH ; POINT TO NEXT LINE IN BLOCK
663 0236 75 F5      JNZ     N2 ; COUNT OF LINES TO MOVE
664 0238          N3:  CLEAR_ENTRY
665 0238 58          POP      AX ; RECOVER ATTRIBUTE IN AH
666 0239 B0 20      MOV      AL,' ' ; FILL WITH BLANKS
667 023B          N4:  CALL     N11 ; CLEAR_LOOP
668 023B E8 02A6 R  CALL     N11 ; CLEAR THE ROW
669 023E 03 FD      ADD      D1,BP ; POINT TO NEXT LINE
670 0240 FE CB      DEC      BL ; COUNTER OF LINES TO SCROLL
671 0242 75 F7      JNZ     N4 ; CLEAR_LOOP
672 0244          N5:  DD      DD5 ; SCROLL_END
673 0244 E8 0000 E    CALL     CRT_MODE,7 ; IS THIS THE BLACK AND WHITE CARD
674 0247 80 3E 0049 R 07 CMP      N6,3E ; IF SO, SKIP THE MODE RESET
675 024C 74 07      JE      N6 ; GET THE VALUE OF THE MODE SET
676 024E A0 0065 R  MOV      AL,CRT_MODE_SET ; ALWAYS SET COLOR CARD PORT
677 0251 BA 03D8     MOV      DX,03D8H
678 0253 EE          OUT      DX,AL
679 0255          N6:  VIDEO_RET_HERE
680 0255 E9 013D R  JMP      VIDEO_RETURN
681 0258          N7:  MOV      BL,DH ; BLANK FIELD
682 025B 8A DE      MOV      N3 ; GET ROW COUNT
683 025A EB DC      JMP      N3 ; GO CLEAR THAT AREA
684 025C          SCROLL_UP ENDP
```

```

685
686
687
688      025C
689      025C E8 0200 R
690      025F 03 06 004E R
691      0263 B8 F8
692      0265 B8 F0
693      0267 2B D1
694      0269 FE C6
695      026B FE C2
696      026D 32 ED
697      026F BB 2E 004A R
698      0273 03 ED
699      0275 A0 004A R
700      0278 F4 E3
701      027A 03 C0
702      027C 50
703      027D A0 0049 R
704      0280 06
705      0281 IF
706      0282 3C 02
707      0284 72 13
708      0286 3C 03
709      0288 77 0F
710
711      028A 52
712      028B BA 03DA
713      028E EC
714      028F A8 08
715      0291 74 FB
716      0293 B0 25
717      0295 B0 D8
718      0297 EE
719      0298 5A
720      0299
721      0299 58
722      029A 0A DB
723      029C 03
724      029D
725
726
727
728      029D
729      029D BA CA
730      029F 56
731      02A0 57
732      02A1 F3/ A5
733      02A3 6F
734      02A4 5E
735      02A5 C3
736      02A6
737
738
739      02A6
740      02A6 BA CA
741      02A8 57
742      02A9 F3/ AB
743      02AB 5F
744      02AC C3
745      02AD
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762      02AD
763      02AD FD
764      02AE E8 02EA R
765      02B1 80 FC 04
766      02B4 72 08
767      02B6 80 FC 07
768      02B9 74 03
769      02BB E9 0503 R
770      02BE
771      02BE 53
772      02BF BB C2
773      02C1 E8 025C R
774      02C4 74 20
775      02C6 2B F0
776      02C8 BA E6
777      02CA 2A E3
778      02CC
779      02CC E8 029D R
780      02CF 2B F5
781      02D1 2B FD
782      02D3 FE CC
783      02D5 F5 F5
784      02D7
785      02D7 58
786      02D8 B0 20
787      02DA
788      02DA E8 02A6 R
789      02DD 2B FD
790      02DF FE CB
791      02E1 75 F7
792      02E3 E9 0244 R
793      02E6
794      02E6 BA DE
795      02E8 EB ED
796      02EA

;----- HANDLE COMMON SCROLL SET UP HERE
SCROLL_POSITION PROC NEAR
    CALL POSITION NEAR
    ADD AX,CRT_START
    MOV DI,AX
    MOV SI,AX
    SUB DX,CX
    INC DH
    INC DL
    XOR CH,CH
    MOV BP,CRT_COLS
    ADD BP,BP
    MOV BL,BYTE PTR *CRT_COLS
    MUL BL
    ADD AX,AX
    PUSH AX
    MOV AL,*CRT_MODE
    PUSH ES
    POP DS
    CMP AL,2
    JB N9
    CMP AL,3
    JA N9
; CONVERT TO REGEN POINTER
; OFFSET OF ACTIVE PAGE
; TO ADDRESS FOR SCROLL
; FROM ADDRESS FOR SCROLL
; DX = #ROWS, #COLS IN BLOCK
; INCREMENT FOR 0 ORIGIN
; SET HIGH BYTE OF COUNT TO ZERO
; GET NUMBER OF COLUMNS IN DISPLAY
; TIMES 2 FOR ATTRIBUTE BYTE
; GET CHARACTERS PER LINE COUNT
; DETERMINE OFFSET TO FROM ADDRESS
; *2 FOR ATTRIBUTE BYTE
; SAVE LINE COUNT
; GET CURRENT MODE
; ESTABLISH ADDRESSING TO REGEN BUFFER
; FOR BOTH POINTERS
; TEST FOR COLOR CARD SPECIAL CASES HERE
; HAVE TO HANDLE 80X25 SEPARATELY

;-----
; 80X25 COLOR CARD SCROLL
N8:    PUSH DX
        MOV DX,3DAH
; GUARANTEED TO BE COLOR CARD HERE
; WAIT DISP_ENABLE
        IN AL,DX
        TEST AL,RVRT
        JZ N8
        MOV AL,25H
        DL,OD8H
        OUT DX,AL
        POP DX
; ADDRESS CONTROL PORT
; TURN OFF VIDEO DURING VERTICAL RETRACE
N9:    POP AX
        OR BL,BL
        RET
; RESTORE LINE COUNT
; 0 SCROLL MEANS BLANK FIELD
; RETURN WITH FLAGS SET
SCROLL_POSITION ENDP

;----- MOVE_ROW
N10:   PROC NEAR
        MOV CL,DL
        PUSH SI
        PUSH DI
        REP MOVSW
        POP DI
        POP SI
; GET # OF COLS TO MOVE
; SAVE START ADDRESS
; MOVE THAT LINE ON SCREEN
; RECOVER ADDRESSES
        RET
N10:   ENDP

;----- CLEAR_ROW
N11:   PROC NEAR
        MOV CL,DL
        PUSH DI
        REP STOSW
        POP DI
        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
;-----
SCROLL_DOWN PROC NEAR
    CALL TEST_LINE_COUNT
    CMP AH,4
    JC N12
    CMP AH,7
    JE N12
    JMP GRAPHICS_DOWN
; DIRECTION FOR SCROLL DOWN
; TEST FOR GRAPHICS
; TEST FOR BW CARD
N12:   PUSH BX
        MOV AX,DX
        CALL SCROLL_POSITION
        JZ N16
        SUB SI,AX
        MOV AH,DH
        SUB AH,BL
; CONTINUE DOWN
; SAVE ATTRIBUTE IN BH
; LOWER RIGHT CORNER
; GET REGEN LOCATION
; SI IS FROM ADDRESS
; GET TOTAL # ROWS
; COUNT TO MOVE IN SCROLL
N13:   CALL N10
        SUB SI,BP
        SUB DI,BP
        DEC AH
        JNZ N13
; MOVE ONE ROW
N14:   POP AX
        MOV AL,
; RECOVER ATTRIBUTE IN AH
N15:   CALL N11
        SUB DI,BP
        DEC BL
        JNZ N15
; CLEAR ONE ROW
; GO TO NEXT ROW
N16:   JMP N5
; SCROLL_END
        MOV BL,DH
        JMP N14
SCROLL_DOWN ENDP

```

```

797 PAGE
798 ----- IF AMOUNT OF LINES TO BE SCROLLED = AMOUNT OF LINES IN WINDOW
799 ----- THEN ADJUST AL; ELSE RETURN;
800
801 TEST_LINE_COUNT PROC NEAR
802
803 02EA MOV BL,AL ; SAVE LINE COUNT IN BL
804 02EC OR AL,AL ; TEST IF AL IS ALREADY ZERO
805 02EE JZ BL_SET ; IF IT IS THEN RETURN...
806 02F0 PUSH AX ; SAVE AX
807 02F1 MOV AL,DH ; SUBTRACT LOWER ROW FROM UPPER ROW
808 02F3 SUB AL,CH
809 02F5 INC AL ; ADJUST DIFFERENCE BY 1
810 02F7 CMP AL,BL ; LINE COUNT = AMOUNT OF ROWS IN WINDOW?
811 02F9 POP AX ; RESTORE AX
812 02FA JNE BL_SET ; IF NOT THEN WE'RE ALL SET
813 02FC SUB BL,BL ; OTHERWISE SET BL TO ZERO
814 02FE BL_SET: RET
815 02FE RET
816 02FF
817
818 TEST_LINE_COUNT ENDP
819
820 -----
821 ; READ_AC_CURRENT
822 ; THIS ROUTINE READS THE ATTRIBUTE AND CHARACTER AT THE CURRENT
823 ; CURSOR POSITION AND RETURNS THEM TO THE CALLER
824 ; INPUT
825 ; (AH) = CURRENT CRT MODE
826 ; (BH) = DISPLAY PAGE ( ALPHA MODES ONLY )
827 ; (DS) = DATA SEGMENT
828 ; (ES) = REGEN SEGMENT
829 ; OUTPUT
830 ; (AL) = CHARACTER READ
831 ; (AH) = ATTRIBUTE READ
832 -----
833 ASSUME DS:DATA,ES:DATA
834
835 READ_AC_CURRENT PROC NEAR
836 CMP AH,4 ; IS THIS GRAPHICS
837 JC P10
838
839 CMP AH,7 ; IS THIS BW CARD
840 JE P10
841
842 JMP GRAPHICS_READ
843
844 P10: CALL FIND_POSITION ; READ AC CONTINUE
845 MOV SI,DI ; GET REGEN LOCATION AND PORT ADDRESS
846 PUSH ES ; ESTABLISH ADDRESSING IN SI
847 POP DS ; GET REGEN SEGMENT FOR QUICK ACCESS
848
849 ;----- WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
850
851 OR BL,BL ; CHECK MODE FLAG FOR COLOR CARD IN 80
852 JNZ P11 ; ELSE SKIP RETRACE WAIT - DO FAST READ
853
854 STI ; ENABLE INTERRUPTS FIRST
855 NOP ; ALLOW FOR SMALL INTERRUPT WINDOW
856 CLD ; BLOCK INTERRUPTS FOR SINGLE LOOP
857 IN AL,DX ; GET STATUS FROM THE ADAPTER
858 TEST AL,RHRZ ; IS HORIZONTAL RETRACE LOW
859 JNZ P11 ; WAIT UNTIL IT IS
860
861 IN AL,DX ; NOW WAIT FOR EITHER RETRACE HIGH
862 TEST AL,RVRT+RHRZ ; GET STATUS
863 JZ P12 ; IS HORIZONTAL OR VERTICAL RETRACE HIGH
864
865 LODSW ; IS HORIZONTAL OR VERTICAL RETRACE HIGH
866 JMP VIDEO_RETURN ; WAIT UNTIL EITHER IS ACTIVE
867
868 READ_AC_CURRENT ENDP
869
870
871 FIND_POSITION PROC NEAR
872 XCHG AH,BL ; SETUP FOR BUFFER READ OR WRITE
873 MOV BP,AX ; SWAP MODE TYPE WITH ATTRIBUTE
874 SUB BL,2 ; SAVE CHARACTER/ATTR IN (BP) REGISTER
875 SHR BL,1 ; CONVERT DISPLAY MODE TYPE TO A
876 MOV AL,BH ; ZERO VALUE FOR COLOR IN 80 COLUMN
877 CBW ; MOVE DISPLAY PAGE TO LOW BYTE
878 MOV DI,AX ; CLEAR HIGH BYTE FOR BYTE OFFSET
879 MOV DI,DI ; MOVE DISPLAY PAGE (COUNT) TO WORK REG
880 JZ P21 ; TIMES 2 FOR WORD OFFSET
881
882 XOR DI,DI ; GET ROW/COLUMN OF THAT PAGE
883 ; SKIP BUFFER ADJUSTMENT IF PAGE ZERO
884
885 XOR DI,DI ; ELSE SET BUFFER START ADDRESS TO ZERO
886
887 P20: ADD DI,*CRT_LEN ; ADD LENGTH OF BUFFER FOR ONE PAGE
888 DEC AX ; DECREMENT PAGE COUNT
889 JNZ P20 ; LOOP TILL PAGE COUNT EXHAUSTED
890
891 P21: MOV AL,BYTE PTR *CRT_COLS ; DETERMINE LOCATION IN REGEN IN PAGE
892 MUL DH ; GET COLUMNS PER ROW COUNT
893 XOR DH,DH ; DETERMINE BYTES TO ROW
894 ADD AX,DX ; ADD IN COLUMN VALUE
895 SAL AX,1 ; * 2 FOR ATTRIBUTE BYTES
896 ADD DI,AX ; ADD LOCATION TO START OF REGEN PAGE
897 MOV DX,*ADDR_6845 ; GET BASE ADDRESS OF ACTIVE DISPLAY
898 ADD DX,6 ; DX= STATUS PORT ADDRESS OF ADAPTER
899 RET ; BP= ATTRIBUTE/CHARACTER (FROM BL/AL)
900 ; DI= POSITION (OFFSET IN REGEN BUFFER)
901 ; BL= MODE FLAG (ZERO FOR 80X25 COLOR)
902
903 FIND_POSITION ENDP

```

```

900 PAGE
901 -----
902 | WRITE_C_CURRENT |
903 | THIS ROUTINE WRITES THE CHARACTER AT |
904 | THE CURRENT CURSOR POSITION, ATTRIBUTE UNCHANGED |
905 | INPUT |
906 | (AH) = CURRENT CRT MODE |
907 | (BH) = DISPLAY PAGE |
908 | (CX) = COUNT OF CHARACTERS TO WRITE |
909 | (AL) = CHAR TO WRITE |
910 | (BL) = ATTRIBUTE OF CHAR TO WRITE |
911 | (DS) = DATA SEGMENT |
912 | (ES) = REGEN SEGMENT |
913 | OUTPUT |
914 | DISPLAY REGEN BUFFER UPDATED |
915 -----
916
917 035C 80 FC 04 WRITE_C_CURRENT PROC NEAR
918 035F 72 08 JC P30
919 0361 80 FC 07 CMP AH,7
920 0364 74 03 JE P30
921 0366 E9 058A R JMP GRAPHICS_WRITE
922 0369
923 0369 E8 0328 R P30: CALL FIND_POSITION
924 036C 0A DB OR BL,BL
925 036E 74 06 JZ P32
926
927 0370 95 XCHG AX,BP
928 0371 F3/ AB REP STOSW
929 0373 EB 16 JMP SHORT P35
930
931 ----- WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
932
933 P31: XCHG BP,AX
934
935 0375 95 STI
936 0376 FB NOP
937 0377 90 CLI
938 0378 FA IN AL,DX
939 0379 EC TEST AL,RVRT
940 037A A8 08 JNZ P34
941 037C 75 09 JZ P32
942 037E A8 01 TEST AL,RHRZ
943 0380 75 F4 JNZ P32
944 0382 EC P33: IN AL,DX
945 0383 A8 09 TEST AL,RVRT+RHRZ
946 0385 74 FB JZ P33
947 0387 95 XCHG AX,BP
948 0388 AB STOSW
949 0389 E2 EA LOOP P31
950 038B P35: JMP VIDEO_RETURN
951 038D E9 013D R
952 038E
953 038E ENDP
954
955 -----
956 | WRITE_C_CURRENT |
957 | THIS ROUTINE WRITES THE CHARACTER AT |
958 | THE CURRENT CURSOR POSITION, ATTRIBUTE UNCHANGED |
959 | INPUT |
960 | (AH) = CURRENT CRT MODE |
961 | (BH) = DISPLAY PAGE |
962 | (CX) = COUNT OF CHARACTERS TO WRITE |
963 | (AL) = CHAR TO WRITE |
964 | (DS) = DATA SEGMENT |
965 | (ES) = REGEN SEGMENT |
966 | OUTPUT |
967 | DISPLAY REGEN BUFFER UPDATED |
968 -----
969
970 038E 80 FC 04 WRITE_C_CURRENT PROC NEAR
971 0391 72 08 JC P40
972 0393 80 FC 07 CMP AH,7
973 0396 74 03 JE P40
974 0398 E9 058A R JMP GRAPHICS_WRITE
975 039B
976 039B E8 0328 R P40: CALL FIND_POSITION
977
978 ----- WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
979
980 P41: STI
981 039E FB OR BL,BL
982 039F 0A DB JNZ P43
983 03A1 75 0F CLI
984 03A3 FA IN AL,DX
985 03A4 EC TEST AL,RVRT
986 03A5 A8 08 JNZ P43
987 03A7 75 09 TEST AL,RHRZ
988 03A9 A8 01 JNZ P41
989 03AB 75 F1 JZ P41
990 03AD EC P42: IN AL,DX
991 03AE A8 09 TEST AL,RVRT+RHRZ
992 03B0 74 FB JZ P42
993 03B2 P43: MOV AX,BP
994 03B2 8B C5 GET THE CHARACTER SAVE IN (BP)
995 03B4 AA PUT THE CHARACTER INTO REGEN BUFFER
996 03B5 47 INC DI
997 03B6 E2 E6 BUMP POINTER PAST ATTRIBUTE
998 03B8 E9 013D R
999 03BB
1000 03BB JMP VIDEO_RETURN
1001
1002 03BB WRITE_C_CURRENT ENDP

```

```

1009 PAGE
1010 -----
1011 : WRITE_STRING
1012 : THIS ROUTINE WRITES A STRING OF CHARACTERS TO THE CRT.
1013 : INPUT
1014 : (AL) = WRITE STRING COMMAND 0 - 3
1015 : (BH) = DISPLAY PAGE (ACTIVE PAGE)
1016 : (CX) = COUNT OF CHARACTERS TO WRITE, IF (CX) = 0 THEN RETURN
1017 : (DX) = CURSOR POSITION FOR START OF STRING WRITE
1018 : (BL) = ATTRIBUTE OF CHARACTER TO WRITE IF (AL) = 0 OR (AL) = 1
1019 : (BP) = SOURCE STRING OFFSET
1020 : [0E] = SOURCE STRING SEGMENT (FOR USE IN (ES) IN STACK +14)
1021 : OUTPUT
1022 : NONE
1023 -----
1024 WRITE_STRING PROC NEAR
1025 PUSH BP
1026 MOV BP,SP
1027 MOV ES,[BP]+14*2
1028 POP BP
1029 CBW
1030 MOV DI,AX
1031 CMP AL,04
1032 JNB P59
1033
1034 JCXZ P59
1035
1036 MOV SI,BX
1037 MOV BL,BH
1038 XOR BH,BH
1039 XCHG SI,BX
1040 MOV SI,1
1041 [SI+OFFSET *CURSOR_POSN]
1042 MOV AX,0200H
1043 INT 10H
1044
1045 P50: MOV AL,ES:[BP]
1046 INC BP
1047
1048 I----- TEST FOR SPECIAL CHARACTER'S
1049
1050 CMP AL,08H
1051 JE P51
1052 CMP AL,CR
1053 JE P51
1054 CMP AL,LF
1055 JE P51
1056 CMP AL,07H
1057 JNE P52
1058
1059 P51: MOV AH,0EH
1060 INT 10H
1061 MOV DX,[SI+OFFSET *CURSOR_POSN]
1062 JMP SHORT P54
1063
1064 P52: PUSH CX
1065 PUSH BX
1066 MOV CX,1
1067 MOV DI,2
1068 CMP DI,2
1069 JB P53
1070 MOV BL,ES:[BP]
1071 INC BP
1072
1073 P53: MOV AH,09H
1074 INT 10H
1075 POP BX
1076 POP CX
1077 INC DL
1078 CMP DL,BYTE PTR *CRT_COLS
1079 JB P54
1080 INC DH
1081 SUB DL,DL
1082 CMP DH,25
1083 JB P54
1084
1085 MOV AX,0E0AH
1086 INT 10H
1087 DEC DH
1088
1089 P54: MOV AX,0200H
1090 INT 10H
1091 LOOP P50
1092
1093 POP DX
1094 XCHG AX,DI
1095 TEST AL,01H
1096 JNZ P59
1097 MOV AX,0200H
1098 INT 10H
1099
1100 P59: JMP VIDEO_RETURN
1101
1102 WRITE_STRING ENDP

```

```

1103 PAGE
1104 -----
1105 ; READ DOT -- WRITE DOT
1106 ; THESE ROUTINES WILL WRITE A DOT, OR READ THE
1107 ; DOT AT THE INDICATED LOCATION
1108 ; ENTRY --
1109 ; DX = ROW (0-199) (THE ACTUAL VALUE DEPENDS ON THE MODE)
1110 ; CX = COLUMN (0-639) (THE VALUES ARE NOT RANGE CHECKED)
1111 ; AL = DOT VALUE TO WRITE (1,2 OR 4 BITS DEPENDING ON MODE,
1112 ; REQUIRED FOR WRITE DOT ONLY, RIGHT JUSTIFIED)
1113 ; BIT 7 OF AL = 1 INDICATES XOR THE VALUE INTO THE LOCATION
1114 ; DS = DATA SEGMENT
1115 ; ES = REGEN SEGMENT
1116
1117 ; EXIT
1118 ; AL = DOT VALUE READ, RIGHT JUSTIFIED, READ ONLY
1119 -----
1120 ASSUME DS:DATA,ES:DATA
1121 READ_DOT PROC NEAR
1122 CALL R3 ; DETERMINE BYTE POSITION OF DOT
1123 MOV AL,ES:[SI] ; GET THE BYTE
1124 AND AL,AH ; MASK OFF THE OTHER BITS IN THE BYTE
1125 SHL AL,CL ; LEFT JUSTIFY THE VALUE
1126 MOV CL,DH ; GET NUMBER OF BITS IN RESULT
1127 ROL AL,CL ; RIGHT JUSTIFY THE RESULT
1128 JMP VIDEO_RETURN ; RETURN FROM VIDEO I/O
1129 ENDP
1130
1131 WRITE_DOT PROC NEAR
1132 PUSH AX ; SAVE DOT VALUE
1133 PUSH AX ; TWICE
1134 CALL R3 ; DETERMINE BYTE POSITION OF THE DOT
1135 SHR AL,CL ; SHIFT TO SET UP THE BITS FOR OUTPUT
1136 AND AL,AH ; STRIP OFF THE OTHER BITS
1137 MOV CL,ES:[SI] ; GET THE CURRENT BYTE
1138 POP BX ; RECOVER XOR FLAG
1139 TEST BL,80H ; IS IT ON
1140 JNZ R3 ; YES, XOR THE DOT
1141 NOT AH ; SET MASK TO REMOVE THE INDICATED BITS
1142 AND CL,AH ;
1143 OR AL,CL ; OR IN THE NEW VALUE OF THOSE BITS
1144 MOV ES:[SI],AL ; FINISH DOT
1145 POP AX ; RESTORE THE BYTE IN MEMORY
1146 JMP VIDEO_RETURN ; RETURN FROM VIDEO I/O
1147
1148 R2: XOR AL,CL ; XOR DOT
1149 JMP R3 ; EXCLUSIVE OR THE DOT
1150 ; FINISH UP THE WRITING
1151 WRITE_DOT ENDP
1152 -----
1153 ; THIS SUBROUTINE DETERMINES THE REGEN BYTE LOCATION OF THE
1154 ; INDICATED ROW COLUMN VALUE IN GRAPHICS MODE.
1155 ; ENTRY --
1156 ; DX = ROW VALUE (0-199)
1157 ; CX = COLUMN VALUE (0-639)
1158 ; EXIT --
1159 ; SI = OFFSET INTO REGEN BUFFER FOR BYTE OF INTEREST
1160 ; AH = MASK TO STRIP OFF THE BITS OF INTEREST
1161 ; CL = BITS TO SHIFT TO RIGHT JUSTIFY THE MASK IN AH
1162 ; DH = # BITS IN RESULT
1163 ; BX = MODIFIED
1164 -----
1165 R3 PROC NEAR
1166
1167 ;----- DETERMINE 1ST BYTE IN INDICATED ROW BY MULTIPLYING ROW VALUE BY 40
1168 ; ( LOW BIT OF ROW DETERMINES EVEN/ODD, 80 BYTES/ROW )
1169
1170 XCHG SI,AX ; WILL SAVE AL AND AH DURING OPERATION
1171 MOV AL,40
1172 MUL DL ; AX= ADDRESS OF START OF INDICATED ROW
1173 TEST AL,008H ; TEST FOR EVEN/ODD ROW CALCULATED
1174 JZ R4 ; JUMP IF EVEN ROW
1175 ADD AX,2000H-40 ; OFFSET TO LOCATION OF ODD ROWS ADJUST
1176 ; EVEN ROW
1177 XCHG SI,AX ; MOVE POINTER TO (SI) AND RECOVER (AX)
1178 MOV DX,CX ; COLUMN VALUE TO DX
1179
1180 ;----- DETERMINE GRAPHICS MODE CURRENTLY IN EFFECT
1181
1182 ; SET UP THE REGISTERS ACCORDING TO THE MODE
1183 ; CH = MASK FOR LOW OF COLUMN ADDRESS ( 7/3 FOR HIGH/MED RES )
1184 ; CL = # OF ADDRESS BITS IN COLUMN VALUE ( 3/2 FOR H/M )
1185 ; BL = MASK TO SELECT BITS FROM POINTED BYTE ( 80H/COH FOR H/M )
1186 ; BH = NUMBER OF VALID BITS IN POINTED BYTE ( 1/2 FOR H/M )
1187
1188 MOV BX,2C0H
1189 MOV CX,302H ; SET PARMS FOR MED RES
1190 CMP @CRT_MODE,6
1191 JC R5 ; HANDLE IF MED RES
1192 MOV BX,180H
1193 MOV CX,703H ; SET PARMS FOR HIGH RES
1194
1195 ;----- DETERMINE BIT OFFSET IN BYTE FROM COLUMN MASK
1196 R5: AND CH,DL ; ADDRESS OF PEL WITHIN BYTE TO CH
1197
1198 ;----- DETERMINE BYTE OFFSET FOR THIS LOCATION IN COLUMN
1199
1200 SHR DX,CL ; SHIFT BY CORRECT AMOUNT
1201 ADD SI,DX ; INCREMENT THE POINTER
1202 MOV DH,BH ; GET THE # OF BITS IN RESULT TO DH
1203
1204 ;----- MULTIPLY BH (VALID BITS IN BYTE) BY CH (BIT OFFSET)
1205
1206 SUB CL,CL ; ZERO INTO STORAGE LOCATION
1207
1208 R6: ROR AL,1 ; LEFT JUSTIFY VALUE IN AL (FOR WRITE)
1209 ADD AL,CH ; ADD IN THE BIT OFFSET VALUE
1210 DEC BH ; LOOP CONTROL
1211 JNZ R6 ; ON EXIT, CL HAS COUNT TO RESTORE BITS
1212 MOV AH,BL ; GET MASK TO AH
1213 SHR AH,CL ; MOVE THE MASK TO CORRECT LOCATION
1214 RET ; RETURN WITH EVERYTHING SET UP
1215
1216 R3 ENDP

```



```

1217 ;-----
1218 ; SCROLL UP
1219 ; THIS ROUTINE SCROLLS UP THE INFORMATION ON THE CRT
1220 ; ENTRY --
1221 ; CH,CL = UPPER LEFT CORNER OF REGION TO SCROLL
1222 ; DH,DL = LOWER RIGHT CORNER OF REGION TO SCROLL
1223 ; BOTH OF THE ABOVE ARE IN CHARACTER POSITIONS
1224 ; BH = FILL VALUE FOR BLANKED LINES
1225 ; AL = # LINES TO SCROLL (AL=0 MEANS BLANK THE ENTIRE FIELD)
1226 ; DS = DATA SEGMENT
1227 ; ES = REGEN SEGMENT
1228 ; EXIT --
1229 ; NOTHING, THE SCREEN IS SCROLLED
1230 ;-----
1231 04AC PROC NEAR
1232 04AC 8A D8 MOV BL,AL ; SAVE LINE COUNT IN BL
1233 04AE 8B C1 MOV AX,CX ; GET UPPER LEFT POSITION INTO AX REG
1234 ;-----
1235 ; USE CHARACTER SUBROUTINE FOR POSITIONING
1236 ;----- ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE
1237
1238 04B0 E8 06EC R CALL GRAPH_POSN
1239 04B3 8B F8 MOV DI,AX ; SAVE RESULT AS DESTINATION ADDRESS
1240
1241 ;----- DETERMINE SIZE OF WINDOW
1242
1243 04B5 2B D1 SUB DX,CX
1244 04B7 81 C2 0101 ADD DX,101H ; ADJUST VALUES
1245 04B8 D0 E6 SAL DH,1 ; MULTIPLY ROWS BY 4 AT 8 VERT DOTS/CHAR
1246 04BD D0 E6 SAL DH,1 ; AND EVEN/ODD ROWS
1247
1248 ;----- DETERMINE CRT MODE
1249
1250 04BF 80 3E 0049 R 06 CMP #CRT_MODE,6 ; TEST FOR MEDIUM RES
1251 04C4 73 04 JNC RT ; FIND_SOURCE
1252
1253 ;----- MEDIUM RES UP
1254 04C6 D0 E2 SAL DL,1 ; # COLUMNS * 2, SINCE 2 BYTES/CHAR
1255 04CB D1 E7 SAL DI,1 ; OFFSET *2 SINCE 2 BYTES/CHAR
1256
1257 ;----- DETERMINE THE SOURCE ADDRESS IN THE BUFFER
1258 04CA RT ; FIND_SOURCE
1259 04CA 06 PUSH ES ; GET SEGMENTS BOTH POINTING TO REGEN
1260 04CB 1F POP DS
1261 04CC 2A ED SUB CH,CH ; ZERO TO HIGH OF COUNT REGISTER
1262 04CE D0 E3 SAL BL,1 ; MULTIPLY NUMBER OF LINES BY 4
1263 04D0 D0 E3 SAL BL,1
1264 04D2 74 2B JZ ; IF ZERO, THEN BLANK ENTIRE FIELD
1265 04D4 B0 50 MOV AL,80 ; 80 BYTES/ROW
1266 04D6 F6 E3 MOV BL,AL ; DETERMINE OFFSET TO SOURCE
1267 04D8 8B F7 MOV SI,DI ; SET UP SOURCE
1268 04DA 03 F0 ADD SI,AX ; ADD IN OFFSET TO IT
1269 04DC 8A E6 MOV AH,DH ; NUMBER OF ROWS IN FIELD
1270 04DE 2A E3 SUB AH,BL ; DETERMINE NUMBER TO MOVE
1271
1272 ;----- LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD FIELDS
1273 04E0 R8: ; ROW LOOP
1274 04E0 E8 0560 R CALL R17 ; MOVE ONE ROW
1275 04E3 81 EE 1FB0 SUB SI,2000H-80 ; MOVE TO NEXT ROW
1276 04E7 81 EF 1FB0 SUB DI,2000H-80
1277 04EB FE CC DEC AH ; NUMBER OF ROWS TO MOVE
1278 04ED 75 F1 JNZ R8 ; CONTINUE TILL ALL MOVED
1279
1280 ;----- FILL IN THE VACATED LINE(S)
1281 04EF R9: ; CLEAR ENTRY
1282 04EF 8A C7 MOV AL,BH ; ATTRIBUTE TO FILL WITH
1283 04F1 R10: ; CLEAR THAT ROW
1284 04F1 E8 0579 R CALL R18 ; POINT TO NEXT LINE
1285 04F4 81 EF 1FB0 SUB DI,2000H-80 ; NUMBER OF LINES TO FILL
1286 04F8 FE CB DEC BL ; CLEAR_LOOP
1287 04FA 75 F5 JNZ R10 ; EVERYTHING DONE
1288 04FC E9 013D R JMP VIDEO_RETURN
1289
1290 04FF R11: ; BLANK FIELD
1291 04FF 8A DE MOV BL,DH ; SET BLANK COUNT TO EVERYTHING IN FIELD
1292 0501 EB EC JMP R9 ; CLEAR THE FIELD
1293 0503
1294 GRAPHICS_UP
1295 ;-----
1296 ; SCROLL DOWN
1297 ; THIS ROUTINE SCROLLS DOWN THE INFORMATION ON THE CRT
1298 ; ENTRY --
1299 ; CH,CL = UPPER LEFT CORNER OF REGION TO SCROLL
1300 ; DH,DL = LOWER RIGHT CORNER OF REGION TO SCROLL
1301 ; BOTH OF THE ABOVE ARE IN CHARACTER POSITIONS
1302 ; BH = FILL VALUE FOR BLANKED LINES
1303 ; AL = # LINES TO SCROLL (AL=0 MEANS BLANK THE ENTIRE FIELD)
1304 ; DS = DATA SEGMENT
1305 ; ES = REGEN SEGMENT
1306 ; EXIT --
1307 ; NOTHING, THE SCREEN IS SCROLLED
1308 ;-----
1309 0503 PROC NEAR
1310 0503 FD STI
1311 0504 8A D8 MOV BL,AL ; SET DIRECTION
1312 0506 8B C2 MOV AX,DX ; SAVE LINE COUNT IN BL
1313 ; GET LOWER RIGHT POSITION INTO AX REG
1314 ;-----
1315 ; USE CHARACTER SUBROUTINE FOR POSITIONING
1316 ;----- ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE
1317
1318 0508 E8 06EC R CALL GRAPH_POSN
1319 050B 8B F8 MOV DI,AX ; SAVE RESULT AS DESTINATION ADDRESS
1320
1321 ;----- DETERMINE SIZE OF WINDOW
1322
1323 050D 2B D1 SUB DX,CX
1324 050F 81 C2 0101 ADD DX,101H ; ADJUST VALUES
1325 0513 D0 E6 SAL DH,1 ; MULTIPLY ROWS BY 4 AT 8 VERT DOTS/CHAR
1326 0515 D0 E6 SAL DH,1 ; AND EVEN/ODD ROWS
1327
1328 ;----- DETERMINE CRT MODE
1329
1329 0517 80 3E 0049 R 06 CMP #CRT_MODE,6 ; TEST FOR MEDIUM RES
1330 051C 73 05 JNC R12 ; FIND_SOURCE_DOWN

```

```

1331
1332
1333 051E D0 E2          ;----- MEDIUM RES DOWN
1334 0520 D1 E7          SAL DL,1          ; # COLUMNS * 2, SINCE 2 BYTES/CHAR
1335 0522 47             SAL DI,1          ; OFFSET *2 SINCE 2 BYTES/CHAR
1336                     INC DI             ; POINT TO LAST BYTE
1337
1338 0523
1339 0523 06             ;----- DETERMINE THE SOURCE ADDRESS IN THE BUFFER
1340 0524 1F             R12:             ; FIND_SOURCE DOWN
1341 0525 2A E0          PUSH ES          ; BOTH SEGMENTS TO REGEN
1342 0527 B1 C7 00F0     POP DS          ;
1343 0528 D0 E3          SUB CH,CH        ; ZERO TO HIGH OF COUNT REGISTER
1344 052D D0 E3          ADD DI,240       ; POINT TO LAST ROW OF PIXELS
1345 052F 74 2B         SAL BL,1         ; MULTIPLY NUMBER OF LINES BY 4
1346 0531 B0 50         JZ R16           ; IF ZERO, THEN BLANK ENTIRE FIELD
1347 0533 F6 E3         MOV AL,80        ; 80 BYTES/ROW
1348 0535 BB F7         MUL BL           ; DETERMINE OFFSET TO SOURCE
1349 0537 2B F0         MOV SI,D1        ; SET UP SOURCE
1350 0539 BA E6         SUB SI,AX        ; SUBTRACT THE OFFSET
1351 053B 2A E3         MOV AH,DH        ; NUMBER OF ROWS IN FIELD
1352                     SUB AH,BL        ; DETERMINE NUMBER TO MOVE
1353
1354 053D
1355 053D E8 0560 R       ;----- LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD FIELDS
1356 0540 81 EE 2050     R13:             ; ROW LOOP DOWN
1357 0544 81 EF 2050     CALL R17         ; MOVE ONE ROW
1358 0548 FE CC         SUB SI,2000H+80   ; MOVE TO NEXT ROW
1359 054A 75 F1         DEC AH           ; NUMBER OF ROWS TO MOVE
1360                     JNZ R13          ; CONTINUE TILL ALL MOVED
1361
1362 054C
1363 054C 8A C7           ;----- FILL IN THE VACATED LINE(S)
1364 054E             R14:             ; CLEAR ENTRY DOWN
1365 054E E8 0579 R       MOV AL,BH        ; ATTRIBUTE TO FILL WITH
1366 0551 81 EF 2050     R15:             ; CLEAR_LOOP DOWN
1367 0555 FE CB         CALL R18         ; CLEAR A ROW
1368 0557 75 F5         SUB DI,2000H+80   ; POINT TO NEXT LINE
1369                     DEC BL           ; NUMBER OF LINES TO FILL
1370                     JNZ R15          ; CLEAR_LOOP_DOWN
1371
1372 0559             JMP VIDEO_RETURN     ; EVERYTHING DONE
1373
1374 055C
1375 055C 8A DE           R16:             ; BLANK_FIELD DOWN
1376 055E EB EC         JMP R14           ; SET BLANK COUNT TO EVERYTHING IN FIELD
1377                     GRAPHICS_DOWN   ; CLEAR THE FIELD
1378                     ENDP
1379
1380 0560
1381 0560 8A CA           ;----- ROUTINE TO MOVE ONE ROW OF INFORMATION
1382 0563 57             R17             ; PROC NEAR
1383 0564 F3/ A4         MOV CL,DL        ; NUMBER OF BYTES IN THE ROW
1384 0566 5F             PUSH SI         ;
1385 0567 5E             PUSH DI        ; SAVE POINTERS
1386 0568 B1 C6 2000     POP SI          ; MOVE THE EVEN FIELD
1387 056C 81 C7 2000     ADD SI,2000H    ;
1388 0570 56             ADD DI,2000H    ; POINT TO THE ODD FIELD
1389 0571 57             PUSH DI        ;
1390 0572 8A CA         MOV CL,DL        ; SAVE THE POINTERS
1391 0574 F3/ A4         REP MOVSB       ; COUNT BACK
1392 0576 5F             POP SI          ; MOVE THE ODD FIELD
1393 0577 5E             POP SI          ; POINTERS BACK
1394 0578 C3             RET            ; RETURN TO CALLER
1395 0579
1396             R17     ENDP
1397
1398
1399 0579
1400 0579 8A CA           ;----- CLEAR A SINGLE ROW
1401 057B 57             R18             ; PROC NEAR
1402 057C F3/ AA         MOV CL,DL        ; NUMBER OF BYTES IN FIELD
1403 057E 5F             PUSH DI        ; SAVE POINTER
1404 057F 81 C7 2000     REP STOSB       ; STORE THE NEW VALUE
1405 0583 57             POP DI          ; POINTER BACK
1406 0584 8A CA         ADD DI,2000H    ; POINT TO ODD FIELD
1407 0586 F3/ AA         MOV CL,DL        ;
1408 0588 5F             REP STOSB       ; FILL THE ODD FIELD
1409 0589 C3             POP DI          ; RETURN TO CALLER
1410 058A             R18     ENDP
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442 058A
1443 058A B4 00         ;-----
1444 058C 50           GRAPHICS_WRITE PROC NEAR
1445                     MOV AH,0          ; ZERO TO HIGH OF CODE POINT
1446                     PUSH AX           ; SAVE CODE POINT VALUE

```

```

1445 ;----- DETERMINE POSITION IN REGEN BUFFER TO PUT CODE POINTS
1446
1447 CALL S26 ; FIND LOCATION IN REGEN BUFFER
1448 0580 E8 06E9 R MOV DI,AX ; REGEN POINTER IN DI
1449 0590 8B F8
1450
1451 ;----- DETERMINE REGION TO GET CODE POINTS FROM
1452
1453 0592 58 POP AX ; RECOVER CODE POINT
1454 0593 3C 80 CMP DI,80H ; IS IT IN SECOND HALF
1455 0595 73 06 JAE S1 ; YES
1456
1457 ;----- IMAGE IS IN FIRST HALF, CONTAINED IN ROM
1458
1459 0597 BE 0000 E MOV SI,OFFSET CRT_CHAR_GEN ; OFFSET OF IMAGES
1460 059A 0E PUSH CS ; SAVE SEGMENT ON STACK
1461 059B EB 18 JMP SHORT S2 ; DETERMINE_MODE
1462
1463 ;----- IMAGE IS IN SECOND HALF, IN USER MEMORY
1464
1465 059D S1: ; EXTEND CHAR
1466 059D 2C 80 SUB AL,80H ; ZERO ORIGIN FOR SECOND HALF
1467 059F 1E PUSH DS ; SAVE DATA POINTER
1468 05A0 2B F6 SUB DS,S1 ;
1469 05A2 8E DE MOV DS,S1 ; ESTABLISH VECTOR ADDRESSING
1470 ASSUME DS:ABS0
1471 05A4 C5 36 007C R LDS SI,0EXT_PTR ; GET THE OFFSET OF THE TABLE
1472 05A8 8C DA MOV DX,SI ; GET THE SEGMENT OF THE TABLE
1473 ASSUME DS:DATA
1474 05AA 1F POP DS ; RECOVER DATA SEGMENT
1475 05AB 52 PUSH DX ; SAVE TABLE SEGMENT ON STACK
1476 05AC 0B D6 OR DX,S1 ; CHECK FOR VALID TABLE DEFINED
1477 05AE 75 05 JNZ S2 ; CONTINUE IF DS:SI NOT 0000:0000
1478
1479 05B0 58 POP AX ; ELSE SET (AX)= 0000 FOR "NULL"
1480 05B1 BE 0000 E MOV SI,OFFSET CRT_CHAR_GEN ; POINT TO DEFAULT TABLE OFFSET
1481 05B4 0E PUSH CS ; IN THE CODE SEGMENT
1482
1483 ;----- DETERMINE GRAPHICS MODE IN OPERATION
1484
1485 05B5 S2: ; DETERMINE_MODE
1486 05B5 D1 E0 SAL AX,1 ; MULTIPLY CODE POINT VALUE BY 8
1487 05B7 D1 E0 SAL AX,1 ;
1488 05B9 D1 E0 SAL AX,1 ;
1489 05BB 03 F0 ADD SI,AX ; SI HAS OFFSET OF DESIRED CODES
1490 05BD 80 36 0049 R 06 CMP #CRT_MODE,6 ;
1491 05C2 1F POP DS ; RECOVER TABLE POINTER SEGMENT
1492 05C3 72 2C JC S7 ; TEST FOR MEDIUM RESOLUTION MODE
1493
1494 ;----- HIGH RESOLUTION MODE
1495
1496 05C5 S3: ; HIGH_CHAR
1497 05C5 57 PUSH DI ; SAVE REGEN POINTER
1498 05C7 B6 04 PUSH SI ; SAVE CODE POINTER
1499 05C9 MOV DH,4 ; NUMBER OF TIMES THROUGH LOOP
1500 05C9 AC S4: ;
1501 05CA F6 C3 80 LODSB ; GET BYTE FROM CODE POINTS
1502 05CB 75 16 JNZ S6 ; SHOULD WE USE THE FUNCTION
1503 05CC AF S6: ; TO PUT CHAR IN
1504 05DD AC STOSB ; STORE IN REGEN BUFFER
1505 05DD LODSB
1506 05DD 26: 88 85 IFFF S5: ;
1507 05DE 83 C7 4F MOV ES:[DI+2000H-1],AL ; STORE IN SECOND HALF
1508 05DF FE CE ADD DI,79 ; MOVE TO NEXT ROW IN REGEN
1509 05E0 75 EC DEC DH ; DONE WITH LOOP
1510 05E0 5E POP SI ;
1511 05E2 5F POP DI ; RECOVER REGEN POINTER
1512 05DF 47 INC DI ; POINT TO NEXT CHAR POSITION
1513 05E0 E2 E3 LOOP S3 ; MORE CHARS TO WRITE
1514 05E2 E9 013D R JMP VIDEO_RETURN
1515
1516 05E5 S6: ;
1517 05E5 26: 32 05 XOR AL,ES:[DI] ; EXCLUSIVE OR WITH CURRENT
1518 05E8 AA STOSB ; STORE THE CODE POINT
1519 05E9 AC LODSB ; AGAIN FOR ODD FIELD
1520 05EA 26: 32 85 IFFF XOR AL,ES:[DI+2000H-1] ;
1521 05EF EB E0 JMP S5 ; BACK TO MAINSTREAM
1522
1523 ;----- MEDIUM RESOLUTION WRITE
1524
1525 05F1 S7: ; MED RES WRITE
1526 05F1 8A D3 MOV DL,BL ; SAVE HIGH COLOR BIT
1527 05F3 D1 E7 SAL DI,1 ; OFFSET*2 SINCE 2 BYTES/CHAR
1528 ; EXPAND BL TO FULL WORD OF COLOR
1529 05F5 80 E3 03 AND BL,3 ; ISOLATE THE COLOR BITS ( LOW 2 BITS )
1530 05F6 F6 E3 MOV BL,055H ; GET BIT CONVERSION MULTIPLIER
1531 05F7 8A D8 MUL BL ; EXPAND 2 COLOR BITS TO 4 REPLICATIONS
1532 05F8 BA F8 MOV BL,AL ; PLACE BACK IN WORK REGISTER
1533 0600 MOV BH,AL ; EXPAND TO 8 REPLICATIONS OF COLOR BITS
1534 0600 S8: ; MED CHAR
1535 0601 57 PUSH DI ; SAVE REGEN POINTER
1536 0601 56 PUSH SI ; SAVE THE CODE POINTER
1537 0602 B6 04 MOV DH,4 ; NUMBER OF LOOPS
1538 0604 S9: ;
1539 0604 AC LODSB ; GET CODE POINT
1540 0605 E8 06C0 R CALL S21 ; DOUBLE UP ALL THE BITS
1541 0608 23 C3 AND AX,BX ; CONVERT TO FOREGROUND COLOR ( 0 BACK )
1542 060A 86 E0 XCHG AH,AL ; SWAP HIGH/LOW BYTES FOR WORD MOVE
1543 060F 74 03 TEST DL,80H ; IS THIS XOR FUNCTION
1544 0611 26: 33 05 JZ S10 ; NO, STORE IT IN AS.IT IS
1545 0614 XOR AX,ES:[DI] ; DO FUNCTION WITH LOW/HIGH
1546 0614 26: 89 05 MOV ES:[DI],AX ; STORE FIRST BYTE HIGH, SECOND LOW
1547 0617 AC LODSB ; GET CODE POINT
1548 0618 E8 06C0 R CALL S21 ;
1549 061B 23 C3 AND AX,BX ; CONVERT TO COLOR
1550 061D 86 E0 XCHG AH,AL ; SWAP HIGH/LOW BYTES FOR WORD MOVE
1551 061F F6 C2 80 TEST DL,80H ; AGAIN, IS THIS XOR FUNCTION
1552 0622 74 05 JZ S11 ; NO, JUST STORE THE VALUES
1553 0624 26: 33 85 2000 XOR AX,ES:[DI+2000H] ; FUNCTION WITH FIRST HALF LOW
1554 0629 S11: ;
1555 0629 26: 89 85 2000 MOV ES:[DI+2000H],AX ; STORE SECOND PORTION HIGH
1556 062E 83 C7 50 ADD DI,80 ; POINT TO NEXT LOCATION
1557 0631 FE DEC DH ;
1558 0633 75 CF JNZ S9 ; KEEP GOING

```

```

1559 0635 5E          POP     SI          ; RECOVER CODE POINTER
1560 0636 5F          POP     DI          ; RECOVER REGEN POINTER
1561 0637 47          INC     DI          ; POINT TO NEXT CHAR POSITION
1562 0638 47          INC     DI
1563 0639 E2 C5       LOOP    S8          ; MORE TO WRITE
1564 063B E9 013D R   JMP     VIDEO_RETURN
1565 063E              GRAPHICS_WRITE ENDP
1566              ;-----
1567              ; GRAPHICS_READ
1568              ;-----
1569 063E          PROC    NEAR
1570 063E E8 06E9 R   CALL    S26          ; CONVERTED TO OFFSET IN REGEN
1571 0641 8B F0       MOV     SI,AX      ; SAVE IN SI
1572 0643 83 EC 08    SUB     SP,8      ; ALLOCATE SPACE FOR THE READ CODE POINT
1573 0646 8B EC      MOV     BP,SP      ; POINTER TO SAVE AREA
1574
1575              ;----- DETERMINE GRAPHICS MODES
1576
1577 0648 80 3E 0049 R 06 CMP     #CRT_MODE,6
1578 064D 06          PUSH    ES
1579 064E 1F          POP     DS          ; POINT TO REGEN SEGMENT
1580 064F 72 19       JC      S13        ; MEDIUM RESOLUTION
1581
1582              ;----- HIGH RESOLUTION READ
1583
1584              ;----- GET VALUES FROM REGEN BUFFER AND CONVERT TO CODE POINT
1585 0651 B6 04       MOV     DH,4          ; NUMBER OF PASSES
1586 0653              S12:
1587 0653 8A 04       MOV     AL,[SI]      ; GET FIRST BYTE
1588 0655 8A 46 00   MOV     [BP],AL      ; SAVE IN STORAGE AREA
1589 0658 45          INC     BP          ; NEXT LOCATION
1590 0659 8A 84 2000 MOV     AL,[SI+2000H] ; GET LOWER REGION BYTE
1591 065D 88 46 00   MOV     [BP],AL      ; ADJUST AND STORE
1592 0660 45          INC     BP
1593 0661 83 C6 50   ADD     SI,80        ; POINTER INTO REGEN
1594 0664 FE CE      DEC     DH          ; LOOP CONTROL
1595 0666 75 EB      JNZ     S12        ; DO IT SOME MORE
1596 0668 EB 16     JMP     SHORT S15      ; GO MATCH THE SAVED CODE POINTS
1597
1598              ;----- MEDIUM RESOLUTION READ
1599 066A          S13:
1600 066A D1 E6      SAL     SI,1          ; MED RES READ
1601 066C B6 04     MOV     DH,4          ; OFFSET*2 SINCE 2 BYTES/CHAR
1602 066E              S14:
1603 066E E8 06CF R   CALL    S23          ; GET BYTES FROM REGEN INTO SINGLE SAVE
1604 0671 81 C6 1FFE ADD     SI,2000H-2    ; GO TO LOWER REGION
1605 0675 E8 06CF R   CALL    S23          ; GET THIS PAIR INTO SAVE
1606 0678 81 EE 1FB2 SUB     SI,2000H-80+2 ; ADJUST POINTER BACK INTO UPPER
1607 067C FE CE      DEC     DH
1608 067E 75 EE      JNZ     S14          ; KEEP GOING UNTIL ALL 8 DONE
1609
1610              ;----- SAVE AREA HAS CHARACTER IN IT, MATCH IT
1611 0680          S15:
1612 0680 BF 0000 E   MOV     DI,OFFSET CRT_CHAR_GEN ; FIND CHAR
1613 0683 0E          PUSH    CS          ; ESTABLISH ADDRESSING
1614 0684 07          POP     ES
1615 0685 83 ED 08    SUB     BP,8          ; CODE POINTS IN CS
1616 0688 8B F5      MOV     SI,BP      ; ADJUST POINTER TO START OF SAVE AREA
1617 068A B0 00      MOV     AL,0          ; CURRENT CODE POINT BEING MATCHED
1618 068C              S16:
1619 068C 16          PUSH    DS          ; ESTABLISH ADDRESSING TO STACK
1620 068D 1F          POP     SS
1621 068E BA 0080    MOV     DX,128        ; FOR THE STRING COMPARE
1622 0691              S17:
1623 0691 56          PUSH    SI          ; SAVE SAVE AREA POINTER
1624 0692 57          PUSH    DI          ; SAVE CODE POINTER
1625 0693 B9 0004    MOV     CX,4          ; NUMBER OF WORDS TO MATCH
1626 0696 F3/ A7     REPE    CMPSW      ; COMPARE THE 8 BYTES AS WORDS
1627 0698 5F          POP     DI          ; RECOVER THE POINTERS
1628 0699 5E          POP     SI
1629 069A 74 1E     JZ      S18          ; IF ZERO FLAG SET, THEN MATCH OCCURRED
1630 069C FE C0      INC     AL          ; NO MATCH, MOVE ON TO NEXT
1631 069E 83 C7 08   ADD     DI,8          ; NEXT CODE POINT
1632 06A1 4A          DEC     DX          ; LOOP CONTROL
1633 06A2 75 ED      JNZ     S17          ; DO ALL OF THEM
1634
1635              ;----- CHAR NOT MATCHED, MIGHT BE IN USER SUPPLIED SECOND HALF
1636
1637 06A4 3C 00       CMP     AL,0          ; AL<= 0 IF ONLY 1ST HALF SCANNED
1638 06A6 74 12      JE      S18          ; IF = 0, THEN ALL HAS BEEN SCANNED
1639 06A8 2B C0      SUB     AX,AX
1640 06AA BE D8      MOV     DS,AX
1641              ; ESTABLISH ADDRESSING TO VECTOR
1642 06AC C4 3E 007C R ASSUME DS:ABS0
1643 06B0 8C C0      LES     DI,0EXT_PTR ; GET POINTER
1644 06B2 0B C7      MOV     AX,ES ; SEE IF THE POINTER REALLY EXISTS
1645 06B4 74 04      OR      SI,B ; IF ALL 0, THEN DOESN'T EXIST
1646 06B6 80 80      JZ      S18          ; NO SENSE LOOKING
1647 06B8 B0 80      MOV     AL,128        ; ORIGIN FOR SECOND HALF
1648 06BA EB D2      JMP     S16          ; GO BACK AND TRY FOR IT
1649
1650              ;----- CHARACTER IS FOUND (AL=0 IF NOT FOUND)
1651 06BA          S18:
1652 06BA 83 C4 08    ADD     SP,8          ; READJUST THE STACK, THROW AWAY SAVE
1653 06BD E9 013D R   JMP     VIDEO_RETURN ; ALL DONE
1654 06C0              GRAPHICS_READ ENDP
1655              ;-----
1656              ; EXPAND BYTE
1657              ; THIS ROUTINE TAKES THE BYTE IN AL AND DOUBLES ALL
1658              ; OF THE BITS, TURNING THE 8 BITS INTO 16 BITS.
1659              ; THE RESULT IS LEFT IN AX
1660              ;-----
1661 06C0          S21:
1662 06C0 51          PROC    NEAR
1663 06C1 B9 0008    PUSH    CX          ; SAVE REGISTER
1664 06C4          S22:
1665 06C4 D0 C8      MOV     CX,8          ; SHIFT COUNT REGISTER FOR ONE BYTE
1666 06C6 D1 D0      ROR     AL,1          ; SHIFT BITS, LOW BIT INTO CARRY FLAG
1667 06C8 D1 FD      RCR     BP,1          ; MOVE CARRY FLAG (LOW BIT) INTO RESULTS
1668 06CA E2 F8      SAR     BP,1          ; SIGN EXTEND HIGH BIT (DOUBLE IT)
1669              ; REPEAT FOR ALL 8 BITS
1670 06CC 95          XCHG    AX,BP          ; MOVE RESULTS TO PARAMETER REGISTER
1671 06CD 59          POP     CX          ; RECOVER REGISTER
1672 06CE C3          RET              ; ALL DONE

```

```

1673 06CF          S21      ENDP
1674          ;-----
1675          ; MED READ BYTE
1676          ; THIS ROUTINE WILL TAKE 2 BYTES FROM THE REGEN BUFFER,
1677          ; COMPARE AGAINST THE CURRENT FOREGROUND COLOR, AND PLACE
1678          ; THE CORRESPONDING ON/OFF BIT PATTERN INTO THE CURRENT
1679          ; POSITION IN THE SAVE AREA
1680          ; ENTRY --
1681          ; SI,DS = POINTER TO REGEN AREA OF INTEREST
1682          ; BX = EXPANDED FOREGROUND COLOR
1683          ; BP = POINTER TO SAVE AREA
1684          ; EXIT --
1685          ; SI AND BP ARE INCREMENTED
1686          ;-----
1687 06CF          S23      PROC    NEAR
1688 06CF AD          LODSW    NEAR          ; GET FIRST BYTE AND SECOND BYTES
1689 06D0 86 C4        XCHG     AL,AH          ; SWAP FOR COMPARE
1690 06D2 B9 C000      MOV      CX,0C000H    ; 2 BIT MASK TO TEST THE ENTRIES
1691 06D5 B2 00        MOV      DL,0          ; RESULT REGISTER
1692 06D7          S24:      JZ       AX,CX          ; IS THIS SECTION BACKGROUND?
1693 06D7 85 C1        JZ       S25          ; IF ZERO, IT IS BACKGROUND (CARRY=0)
1694 06D9 74 01        JC      S24          ; WASN'T, SO SET CARRY
1695 06DB F9          STC
1696 06DC          S25:      RCL      DL,1          ; MOVE THAT BIT INTO THE RESULT
1697 06DC D0 D2        SHR      CX,1          ; MOVE THE MASK TO THE RIGHT BY 2 BITS
1698 06DE D1 E9        SHR      CX,1          ; DO IT AGAIN IF MASK DIDN'T FALL OUT
1699 06E0 D3 F3        JNC      S24          ; STORE RESULT IN SAVE AREA
1700 06E2 73 F3        JNC      S24          ; ADJUST POINTER
1701 06E4 88 56 00     MOV      [BP],DL
1702 06E7 45          INC      RET          ; ALL DONE
1703 06E8 C3          RET
1704 06E9          S23      ENDP
1705          ;-----
1706          ; V4 POSITION
1707          ; THIS ROUTINE TAKES THE CURSOR POSITION CONTAINED IN
1708          ; THE MEMORY LOCATION, AND CONVERTS IT INTO AN OFFSET
1709          ; INTO THE REGEN BUFFER, ASSUMING ONE BYTE/CHAR.
1710          ; FOR MEDIUM RESOLUTION GRAPHICS, THE NUMBER MUST
1711          ; BE DOUBLED.
1712          ; ENTRY -- NO REGISTERS, MEMORY LOCATION *CURSOR_POSN IS USED
1713          ; EXIT --
1714          ; AX CONTAINS OFFSET INTO REGEN BUFFER
1715          ;-----
1716 06E9          S26      PROC    NEAR
1717 06E9 A1 0050 R     MOV      AX,CURSOR_POSN ; GET CURRENT CURSOR
1718 06EC          GRAPH_POSN LABEL    NEAR
1719 06EC 53          PUSH     BX          ; SAVE REGISTER
1720 06ED 8B DB        MOV      BX,AX          ; SAVE A COPY OF CURRENT CURSOR
1721 06EF A0 004A R     MOV      AL,BYTE PTR *CRT_COLS ; GET BYTES PER COLUMN
1722 06F2 F6 E4        MUL      AH          ; MULTIPLY BY ROWS
1723 06F4 D1 E0        SHL      AX,1          ; MULTIPLY * 4 SINCE 4 ROWS/BYTE
1724 06F6 D1 E0        SHL      AX,1          ; ISOLATE COLUMN VALUE
1725 06F8 2A FF        SUB      BH,BH          ; DETERMINE OFFSET
1726 06FA 03 C3        ADD      AX,BX          ; RECOVER POINTER
1727 06FC 5B          POP      BX          ; ALL DONE
1728 06FD C3          RET
1729 06FE          S26      ENDP
1730          ;-----
1731          ; WRITE_TTY
1732          ; THIS INTERFACE PROVIDES A TELETYPE LIKE INTERFACE TO THE
1733          ; VIDEO CARDS. THE INPUT CHARACTER IS WRITTEN TO THE CURRENT
1734          ; CURSOR POSITION, AND THE CURSOR IS MOVED TO THE NEXT POSITION.
1735          ; IF THE CURSOR LEAVES THE LAST COLUMN OF THE FIELD, THE COLUMN
1736          ; IS SET TO ZERO, AND THE ROW VALUE IS INCREMENTED. IF THE ROW
1737          ; ROW VALUE LEAVES THE FIELD, THE CURSOR IS PLACED ON THE LAST ROW,
1738          ; FIRST COLUMN, AND THE ENTIRE SCREEN IS SCROLLED UP ONE LINE.
1739          ; WHEN THE SCREEN IS SCROLLED UP, THE ATTRIBUTE FOR FILLING THE
1740          ; NEWLY BLANKED LINE IS READ FROM THE CURSOR POSITION ON THE PREVIOUS
1741          ; LINE BEFORE THE SCROLL, IN CHARACTER MODE. IN GRAPHICS MODE,
1742          ; THE 0 COLOR IS USED.
1743          ; ENTRY --
1744          ; (AH) = CURRENT CRT MODE
1745          ; (AL) = CHARACTER TO BE WRITTEN
1746          ; NOTE THAT BACK SPACE, CARRIAGE RETURN, BELL AND LINE FEED ARE
1747          ; HANDLED AS COMMANDS RATHER THAN AS DISPLAY GRAPHICS CHARACTERS
1748          ; (BL) = FOREGROUND COLOR FOR CHAR WRITE IF CURRENTLY IN A GRAPHICS MODE
1749          ; EXIT --
1750          ; ALL REGISTERS SAVED THROUGH VIDEO_EXIT (INCLUDING (AX))
1751          ;-----
1752          ASSUME DS:DATA
1753 06FE          WRITE_TTY PROC    NEAR
1754 06FE 97          XCHG     DI,AX          ; SAVE (AX) REGISTER IN (DI) FOR EXIT
1755 06FF B4 03        MOV      AH,03H          ; READ CURSOR POSITION
1756 0701 8A 3E 0062 R MOV      BH,*ACTIVE_PAGE ; GET CURRENT PAGE SETTING
1757 0705 CD 10        INT      10H          ; READ THE CURRENT CURSOR POSITION
1758 0707 8B C7        MOV      AX,DI          ; RECOVER CHARACTER FROM (DI) REGISTER
1759          ;-----
1760          ; DX NOW HAS THE CURRENT CURSOR POSITION
1761          ;-----
1762 0709 3C D0        CMP      AL,CR          ; IS IT CARRIAGE RETURN OR CONTROL
1763 070B 76 46        JBE      U8          ; GO TO CONTROL CHECKS IF IT IS
1764          ;-----
1765          ; WRITE THE CHAR TO THE SCREEN
1766          ;-----
1767 070D B4 0A        MOV      AH,0AH          ; WRITE CHARACTER ONLY COMMAND
1768 070F B9 0001      MOV      CX,1          ; ONLY ONE CHARACTER
1769 0712 CD 10        INT      10H          ; WRITE THE CHARACTER
1770          ;-----
1771          ; POSITION THE CURSOR FOR NEXT CHAR
1772          ;-----
1773 0714 FE C2        INC      DL          ; TEST FOR COLUMN OVERFLOW
1774 0716 3A 16 004A R CMP      DL,BYTE PTR *CRT_COLS ; SET CURSOR
1775 071A 75 33        JNZ      U7          ; COLUMN FOR CURSOR
1776 071C B2 00        MOV      DL,0          ; CHECK FOR LAST ROW
1777 071E 80 FE 18     CMP      DH,25-1
1778 0721 75 2A        JNZ      U6          ; SET_CURSOR_INC
1779          ;-----
1780          ; SCROLL REQUIRED
1781 0723          U1:      MOV      AH,02H          ; SET THE CURSOR
1782 0723 B4 02        MOV      AH,02H
1783 0725 CD 10        INT      10H
1784          ;-----
1785          ; DETERMINE VALUE TO FILL WITH DURING SCROLL
1786          ;-----

```

```

1787 0727 A0 0049 R      MOV     AL,0CRT_MODE      ; GET THE CURRENT MODE
1788 072A 3C 04          CMP     AL,4              ; READ-CURSOR
1789 072C 72 06          JCP     BH,0              ; READ-CURSOR
1790 072E 3C 07          CMP     BH,0              ; READ-CURSOR
1791 0730 B7 00          MOV     U3,BH              ; READ-CURSOR
1792 0732 75 06          JNE     U3                ; READ-CURSOR
1793 0734              ; READ-CURSOR
1794 073A B4 08          MOV     AH,0BH          ; READ CURSOR COMMAND
1795 0736 CD 10          INT     10H              ; READ CHAR/ATTR AT CURRENT CURSOR
1796 0738 BA FC          MOV     BH,AH              ; STORE IN BH
1797 073A              ; READ-CURSOR
1798 073A B8 0601        MOV     AX,0601H        ; SCROLL-UP
1799 073D 2B C9          SUB     CX,CX              ; SCROLL ONE LINE
1800 073F B6 18          MOV     DH,25-1          ; UPPER LEFT CORNER
1801 0741 8A 16 004A R    MOV     DL,BYTE PTR 0CRT_COLS ; LOWER LEFT CORNER
1802 0745 FE CA          DEC     DL              ; LOWER RIGHT CORNER
1803 0747              ; VIDEO-CALL-RETURN
1804 0747 CD 10          INT     10H              ; SCROLL UP THE SCREEN
1805 0749              ; TTY-RETURN
1806 0749 97              JMP     AX,DI              ; RESTORE THE ENTRY CHARACTER FROM (DI)
1807 074A E9 013D R      JMP     VIDEO_RETURN      ; RETURN TO CALLER
1808
1809 074D              ; SET-CURSOR-INC
1810 074D FE C6          INC     DH              ; NEXT ROW
1811 074F              ; SET-CURSOR
1812 074F B4 02          MOV     AH,02H          ; SET-CURSOR
1813 0751 EB F4          JMP     U4                ; ESTABLISH THE NEW CURSOR
1814
1815
1816
1817 0753              ; CHECK FOR CONTROL CHARACTERS
1818 0753 74 13          JE      U9                ; WAS IT A CARRIAGE RETURN
1819 0755 3C 0A          CMP     AL,LF              ; IS IT A LINE FEED
1820 0757 74 13          JE      U10             ; GO TO LINE FEED
1821 0759 3C 07          CMP     AL,07H           ; IS IT A BELL
1822 075B 74 16          JE      U11             ; GO TO BELL
1823 075D 3C 08          CMP     AL,0BH           ; IS IT A BACKSPACE
1824 075F 75 AC          JNE     U0                ; IF NOT A CONTROL, DISPLAY IT
1825
1826
1827 0761 0A D2          OR      DL,DL              ; IS IT ALREADY AT START OF LINE
1828 0763 74 EA          JE      U7                ; SET_CURSOR
1829 0765 4A          DEC     DX              ; NO -- JUST MOVE IT BACK
1830 0766 EB E7          JMP     U7                ; SET_CURSOR
1831
1832
1833
1834 0768              ; CARRIAGE RETURN FOUND
1835 0768 B2 00          MOV     DL,0              ; MOVE TO FIRST COLUMN
1836 076A EB E3          JMP     U7                ; SET_CURSOR
1837
1838
1839
1840 076C              ; LINE FEED FOUND
1841 076C 80 FE 18        CMP     DH,25-1          ; BOTTOM OF SCREEN
1842 076F 75 DC          JNE     U6                ; YES, SCROLL THE SCREEN
1843 0771 EB B0          JMP     U1                ; NO, JUST SET THE CURSOR
1844
1845
1846
1847 0773              ; BELL FOUND
1848 0773 B9 0533        MOV     CX,1331          ; DIVISOR FOR 896 HZ TONE
1849 0776 B3 1F          MOV     BL,31             ; SET COUNT FOR 31/64 SECOND FOR BEEP
1850 0778 EB 0000 E      CALL    BEEP              ; SOUND THE POD BELL
1851 077B EB CC          JMP     U5                ; TTY_RETURN
1852 077D
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868 077D 03 03 05 05 03 03 V1 ASSUME DS:DATA      ; SUBTRACT_TABLE
1869 03 04          DB      3,3,5,5,3,3,3,4
1870
1871
1872 0785              ; WAIT FOR LIGHT PEN TO BE DEPRESSSED
1873 0785 B4 00          MOV     AH,0              ; READ_LPEN
1874 0787 B8 16 0063 R    MOV     DX,0ADDR_6845     ; SET NO LIGHT PEN RETURN CODE
1875 078B 83 C2 06          ADD     DX,6              ; GET BASE ADDRESS OF 6845
1876 078E EC          IN      AL,DX              ; POINT TO STATUS REGISTER
1877 078F AB 04          TEST   AL,004H           ; GET STATUS REGISTER
1878 0791 74 03          JZ      V6_A              ; TEST LIGHT PEN SWITCH
1879 0793 E9 0816 R      JMP     V6                ; GO IF YES
1880
1881
1882
1883 0796 A8 02          ; NOW TEST FOR LIGHT PEN TRIGGER
1884 0798 75 03          JNZ     V7A              ; TEST LIGHT PEN TRIGGER
1885 079A E9 0820 R      JMP     V7                ; RETURN WITHOUT RESETING TRIGGER
1886
1887
1888
1889 079D              ; TRIGGER HAS BEEN SET, READ THE VALUE IN
1890 079D B4 10          V7A:  MOV     AH,16             ; LIGHT PEN REGISTERS ON 6845
1891
1892
1893
1894 079F B8 16 0063 R    MOV     DX,0ADDR_6845     ; INPUT REGISTERS POINTED TO BY AH, AND CONVERT TO ROW COLUMN IN (DX)
1895 07A3 BA C4          MOV     AL,AH              ; ADDRESS REGISTER FOR 6845
1896 07A5 EE          OUT     DX,AL              ; REGISTER TO READ
1897 07A6 90          NOP                       ; SET IT UP
1898 07A7 42          INC     DX                ; I/O DELAY
1899 07AB EC          IN      AL,DX              ; DATA REGISTER
1900 07A9 BA E8          MOV     CH,AL              ; GET THE VALUE
                               ; SAVE IN CX

```

```

1901 07AB 4A          DEC     DX          ; ADDRESS REGISTER
1902 07AC FE C4      INC     AH          ;
1903 07AE 8A C4      MOV     AL,AH       ; SECOND DATA REGISTER
1904 07B0 EE          OUT     DX,AL      ;
1905 07B1 42          INC     DX          ;
1906 07B2 90          NOP              ; I/O DELAY
1907 07B3 EC          IN      AL,DX      ; GET SECOND DATA VALUE
1908 07B4 8A E5      MOV     AH,CH      ; AX HAS INPUT VALUE
1909
1910 ;----- AX HAS THE VALUE READ IN FROM THE 6845
1911
1912 07B6 8A 1E 0049 R MOV     BL,*CRT_MODE
1913 07BA 2A FF      SUB     BH,BH          ; MODE VALUE TO BX
1914 07BC 2E1 8A 9F 07DD R MOV     BL,CS:VI[BX]
1915 07C1 2B C3      SUB     AX,BX          ; DETERMINE AMOUNT TO SUBTRACT
1916 07C3 8B 1E 004E R MOV     BX,*CRT_START
1917 07C7 D1 E8      SHR     BX,1          ; TAKE IT AWAY
1918 07C9 2B C3      SUB     AX,BX          ; CONVERT TO CORRECT PAGE ORIGIN
1919 07CB 19 02      JNS     V2          ; IF POSITIVE, DETERMINE MODE
1920 07CD 2B C0      SUB     AX,AX          ; <0 PLAYS AS 0
1921
1922 ;----- DETERMINE MODE OF OPERATION
1923
1924 07CF          V2:
1925 07CF B1 03      MOV     CL,3          ; DETERMINE MODE
1926 07D1 80 3E 0049 R 04 CMP     *CRT_MODE,4      ; SET *8 SHIFT COUNT
1927 07D6 72 2A      JB      V4          ; DETERMINE IF GRAPHICS OR ALPHA
1928 07D8 80 3E 0049 R 07 CMP     *CRT_MODE,7      ; ALPHA_PEN
1929 07DD 74 23      JE      V4          ; ALPHA_PEN
1930
1931 ;----- GRAPHICS MODE
1932
1933 07DF B2 28      MOV     DL,40          ; DIVISOR FOR GRAPHICS
1934 07E1 F6 F2      DIV     DL          ; DETERMINE ROW(AL) AND COLUMN(AH)
1935 ; AL RANGE 0-99, AH RANGE 0-39
1936
1937 ;----- DETERMINE GRAPHIC ROW POSITION
1938
1939 07E3 8A E8      MOV     CH,AL          ; SAVE ROW VALUE IN CH
1940 07E5 02 ED      ADD     CH,CH          ; *2 FOR EVEN/ODD FIELD
1941 07E7 8A DC      MOV     BL,AH          ; COLUMN VALUE TO BX
1942 07E9 2A FF      SUB     BH,BH          ; MULTIPLY BY 8 FOR MEDIUM RES
1943 07EB 80 3E 0049 R 06 CMP     *CRT_MODE,6      ; DETERMINE MEDIUM OR HIGH RES
1944 07ED 75 04      JNE     V3          ; NOT HIGH RES
1945 07EF B1 04      MOV     CL,4          ; SHIFT VALUE FOR HIGH RES
1946 07F0 00 E4      SAL     AH,1          ; COLUMN VALUE TIMES 2 FOR HIGH RES
1947 07F2 00 E4      SAL     AH,1          ; NOT HIGH RES
1948 07F4 00 E4      SAL     AH,1          ; MULTIPLY *16 FOR HIGH RES
1949
1950 ;----- DETERMINE ALPHA CHAR POSITION
1951
1952 07F6 03 E3      SHL     BX,CL          ;
1953
1954 07F8 8A D4      MOV     DL,AH          ; COLUMN VALUE FOR RETURN
1955 07FA 8A F0      MOV     DH,AL          ; ROW VALUE
1956 07FC D0 EE      SHR     DH,1          ; DIVIDE BY 4
1957 07FE D0 EE      SHR     DH,1          ; FOR VALUE IN 0-24 RANGE
1958 0800 EB 12      JMP     SHORT V5          ; LIGHT_PEN_RETURN_SET
1959
1960 ;----- ALPHA MODE ON LIGHT PEN
1961
1962 0802          V4:
1963 0802 F6 36 004A R DIV     BYTE PTR *CRT_COLS ; ALPHA PEN
1964 0806 8A F0      MOV     DH,AL          ; DETERMINE ROW,COLUMN VALUE
1965 0808 8A D4      MOV     DL,AH          ; ROWS TO DH
1966 080A D2 E0      SAL     AL,CL          ; COLS TO DL
1967 080C 8A E8      MOV     CH,AL          ; MULTIPLY ROWS * 8
1968 080E 8A DC      MOV     BL,AH          ; GET RASTER VALUE TO RETURN REGISTER
1969 0810 32 FF      XOR     BH,BH          ; COLUMN VALUE
1970 0812 D3 E3      SAL     BX,CL          ; TO BX
1971 0814 B4 01      MOV     AH,1          ; LIGHT PEN RETURN SET
1972 0816 52          PUSH    DX          ; INDICATE EVERYTHING SET
1973 0817 8B 16 0063 R MOV     DX,*ADDR_6845 ; LIGHT PEN RETURN
1974 081E EE          OUT     DX,AL          ; SAVE RETURN VALUE (IN CASE)
1975 081F 5A          POP     DX          ; GET BASE ADDRESS
1976 0820          POP     DX          ; POINT TO RESET PARAM
1977 0820 5D          POP     BP          ; ADDRESS, NOT DATA, IS IMPORTANT
1978 0821 5F          POP     DI          ; RECOVER VALUE
1979 0822 5E          POP     SI          ; RETURN_NO_RESET
1980 0823 1F          POP     DS          ;
1981 0824 1F          POP     DS          ; DISCARD SAVED BX,CX,DX
1982 0825 1F          POP     DS          ;
1983 0826 1F          POP     DS          ;
1984 0827 07          POP     ES          ;
1985 0828 CF          IRET          ;
1986 0829          READ_LPEN      ENDP
1987 0829          CODE
1988          END

```

```

1      PAGE 118,121
2      TITLE BIOS1 ---- 01/10/86 INTERRUPT 15H BIOS ROUTINES
3      .LIST
4      CODE      SEGMENT BYTE PUBLIC
5
6      PUBLIC CASSETTE_IO_1
7
8      EXTRN     CONF_TBL:NEAR          ; SYSTEM/BIOS CONFIGURATION TABLE
9      EXTRN     DDS:NEAR              ; LOAD (DS) WITH DATA SEGMENT SELECTOR
10
11     INT 15 H
12     INPUT - CASSETTE I/O FUNCTIONS
13
14     (AH) = 00H
15     (AH) = 01H
16     (AH) = 02H
17     (AH) = 03H
18
19     RETURNS FOR THESE FUNCTIONS ALWAYS (AH) = 86H, CY = 1)
20     IF CASSETTE PORT NOT PRESENT
21
22     INPUT - UNUSED FUNCTIONS
23     (AH) = 04H THROUGH 7FH
24     RETURNS FOR THESE FUNCTIONS ALWAYS (AH) = 86H, CY = 1)
25     (UNLESS INTERCEPTED BY SYSTEM HANDLERS)
26     NOTE: THE KEYBOARD INTERRUPT HANDLER INTERRUPTS WITH AH=4FH
27
28     EXTENSIONS
29     (AH) = 80H  DEVICE OPEN (NULL)
30     (BX) = DEVICE ID
31     (CX) = PROCESS ID
32
33     (AH) = 81H  DEVICE CLOSE (NULL)
34     (BX) = DEVICE ID
35     (CX) = PROCESS ID
36
37     (AH) = 82H  PROGRAM TERMINATION (NULL)
38     (BX) = DEVICE ID
39
40     (AH) = 83H  EVENT WAIT (NULL)
41
42     (AH) = 84H  JOYSTICK SUPPORT
43     (DX) = 00H - READ THE CURRENT SWITCH SETTINGS
44     RETURNS AL = SWITCH SETTINGS (BITS 7-4)
45     (DX) = 01H - READ THE RESISTIVE INPUTS
46     RETURNS AX = A(x) VALUE
47     BX = A(y) VALUE
48     CX = B(x) VALUE
49     DX = B(y) VALUE
50
51     (AH) = 88H  EXTENDED MEMORY SIZE DETERMINE
52
53     (AH) = 91H  INTERRUPT COMPLETE FLAG SET
54     (AL) TYPE CODE
55     00H -> 7FH
56     SERIALY REUSABLE DEVICES
57     OPERATING SYSTEM MUST SERIALIZE ACCESS
58     80H -> BFH
59     REENTRANT DEVICES; ES:BX IS USED TO
60     DISTINGUISH DIFFERENT CALLS (MULTIPLE I/O
61     CALLS ARE ALLOWED SIMULTANEOUSLY)
62     COH -> FFH
63     WAIT ONLY CALLS -- THERE IS NO
64     COMPLEMENTARY 'POST' FOR THESE WAITS.
65     THESE ARE TIMEOUT ONLY. TIMES ARE
66     FUNCTION NUMBER DEPENDENT.
67
68     TYPE DESCRIPTION TIMEOUT
69
70     00H = DISK YES
71     01H = DISKETTE YES
72     02H = KEYBOARD NO
73     80H = NETWORK NO
74     ES:BX -> NCB
75     FDH = DISKETTE MOTOR START YES
76     FEH = PRINTER YES
77
78     (AH) = C0H  RETURN CONFIGURATION PARAMETERS POINTER
79     RETURNS
80     (AH) = 00H AND CY= 0 (IF PRESENT ELSE 86 AND CY= 1)
81     (ES:BX) = PARAMETER TABLE ADDRESS POINTER
82     WHERE:
83
84     DW 8 LENGTH OF FOLLOWING TABLE
85     DB MODEL_BYTE SYSTEM MODEL BYTE
86     DB TYPE_BYTE SYSTEM MODEL TYPE BYTE
87     DB BIOS_LEVEL BIOS REVISION LEVEL
88     10000000 = DMA CHANNEL 3 USE BY BIOS
89     01000000 = CASCADED INTERRUPT LEVEL 2
90     00100000 = REAL TIME CLOCK AVAILABLE
91     00010000 = KEYBOARD SCAN CODE HOOK (AH
92
93     DB 0
94     DB 0
95     DB 0
96     DB 0
97
98     ASSUME CS:CODE
99
100    CASSETTE_IO_1 PROC FAR
101    STI
102    CMP AH,080H ; ENABLE INTERRUPTS
103    JAE C1_G ; CHECK FOR RANGE OF 00-7FH
104    ; SKIP AND HANDLE, ELSE RETURN ERROR
105
106    C1: MOV AH,86H ; ERROR
107    STC ; SET BAD COMMAND
108    ; SET CARRY FLAG ON (CY=1)
109
110    C1_F: RET 2 ; COMMON EXIT
111    ; FAR RETURN EXIT FROM ROUTINES
112
113    C1_G: CMP AH,0C0H ; CONTINUE CHECKING FOR FUNCTION
114    JE CONF_PARMS ; CHECK FOR CONFIGURATION PARAMETERS

```



```

115 0011 80 EC 80      SUB     AH,080H      ; BASE ON 0
116 0014 74 25        JZ      DEV_OPEN    ; DEVICE OPEN      (80H)
117 0016 FE CC        DEC     AH          ;
118 0018 74 21        JZ      DEV_CLOSE   ; DEVICE CLOSE     (81H)
119 001A FE CC        DEC     AH          ;
120 001C 74 1D        JZ      PROG_TERM   ; PROGRAM TERMINATION (82H)
121 001E FE CC        DEC     AH          ; IGNORE EVENT WAIT (83H)
122 0020 FE CC        DEC     AH          ;
123 0022 74 27        JZ      JOY_STICK    ; JOYSTICK BIOS     (84H)
124 0024 FE CC        DEC     AH          ;
125 0026 74 13        JZ      SYS_REQ      ; SYSTEM REQUEST KEY (85H)
126 0028 FE CC        DEC     AH          ; IGNORE WAIT       (86H)
127 002A FE CC        DEC     AH          ; IGNORE BLOCK MOVE (87H)
128 002C FE CC        DEC     AH          ;
129 002E 74 18        JZ      EXT_MEMORY   ; EXTENDED MEMORY SIZE (88H)
130
131 0030 80 EC 08      SUB     AH,8        ; CHECK FOR FUNCTION (90H)
132 0033 74 06        JZ      DEVICE_BUSY  ;
133 0035 FE CC        DEC     AH          ; CHECK FOR FUNCTION (91H)
134 0037 74 05        JZ      INT_COMPLETE ; GO TO INTERRUPT COMPLETE RETURN
135 0039 EB CB        JMP     C1_F         ; EXIT IF NOT A VALID FUNCTION
136
137 003B              DEV_OPEN:           ; NULL HANDLERS
138 003B              DEV_CLOSE:
139 003B              PROG_TERM:
140 003B              SYS_REQ:
141 003B              DEVICE_BUSY:
142 003B F8           CLC
143 003C EB CB        JMP     C1_F         ; TURN CARRY OFF
144
145 003E              CASSETTE_IO_1      ENDP ; RETURN WITH (AH= 00) AND CY=0
146
147
148 ;----- INTERRUPT COMPLETE -----;
149 ; THIS ROUTINE IS A TEMPORARY HANDLER ;
150 ; FOR INTERRUPT COMPLETE             ;
151 ; INPUT - SEE PROLOGUE               ;
152 ;-----;
153
154
155 003E              INT_COMPLETE PROC NEAR ; RETURN
156 003E CF           IRET
157 003F              INT_COMPLETE ENDP
158
159 003F              CONF_PARMS PROC NEAR ; FUNCTION (C0H)
160 003F 0E           PUSH CS ; GET CODE SEGMENT
161 0040 07           POP ES ; PLACE IN SELECTOR POINTER
162 0041 BB 0000 E    MOV     BX,OFFSET CONF_TBL ; GET OFFSET OF PARAMETER TABLE
163 0044 32 E4        XOR     AH,AH ; CLEAR AH AND SET CARRY OFF
164 0046 EB C1        JMP     C1_F         ; EXIT THROUGH COMMON RETURN
165 0048              CONF_PARMS ENDP
166
167 ;--- INT 15H -- ( FUNCTION 85 H - I/O MEMORY SIZE DETERMINE ) -----;
168 ; EXT_MEMORY ;
169 ; THIS ROUTINE RETURNS THE AMOUNT OF MEMORY IN THE SYSTEM THAT IS ;
170 ; LOCATED STARTING AT THE 1024K ADDRESSING RANGE, AS DETERMINED BY ;
171 ; THE POST ROUTINES. ;
172 ; INPUT ;
173 ; AH = 88H ;
174 ; ;
175 ; OUTPUT ;
176 ; (AX) = 0 ;
177 ;-----;
178
179
180 0048              EXT_MEMORY PROC
181
182 0048 33 C0        XOR     AX,AX ; SET EXTENDED MEMORY SIZE TO ZERO
183
184 004A CF           IRET ; RETURN TO USER
185
186 004B              EXT_MEMORY ENDP

```

```

187 PAGE
188 ----- JOY STICK -----
189 THIS ROUTINE WILL READ THE JOYSTICK PORT
190 :
191 :
192 INPUT
193 (DX)=0 READ THE CURRENT SWITCHES
194 RETURNS (AL)= SWITCH SETTINGS IN BITS 7-4
195 :
196 (DX)=1 READ THE RESISTIVE INPUTS
197 RETURNS (BX)=A(x) VALUE
198 (CX)=B(x) VALUE
199 (DX)=B(y) VALUE
200 :
201 CY FLAG ON IF NO ADAPTER CARD OR INVALID CALL
202 -----
203
204 004B JOY_STICK PROC NEAR
205 004B FB STI ; INTERRUPTS BACK ON
206 004C BB C2 MOV AX,DX ; GET SUB FUNCTION CODE
207 004E BA 0201 MOV DX,201H ; ADDRESS OF PORT
208 0051 0A C0 OR AL,AL
209 0053 74 09 JZ JOY_2 ; READ SWITCHES
210 0055 FE C8 DEC AL
211 0057 74 0A JZ JOY_3 ; READ RESISTIVE INPUTS
212 0059 EB AB JMP C1 ; GO TO ERROR RETURN
213 005B JOY_1: STI ; GO TO COMMON RETURN
214 005B FB JMP C1_F
215 005C EB AB
216
217 005E JOY_2: IN AL,DX ; STRIP UNWANTED BITS OFF
218 005E EC AND AL,0F0H ; FINISHED
219 005F 24 F0 JMP JOY_1
220 0061 EB F8
221
222 0063 JOY_3: MOV BL,1
223 0063 B3 01 CALL TEST_CORD
224 0065 EB 0081 R PUSH CX ; SAVE A(X) VALUE
225 0068 51 MOV BL,2
226 0069 B3 02 CALL TEST_CORD
227 006B EB 0081 R PUSH CX ; SAVE A(Y) VALUE
228 006E 51 MOV BL,4
229 006F B3 04 CALL TEST_CORD
230 0071 EB 0081 R PUSH CX ; SAVE B(X) VALUE
231 0074 51 MOV BL,8
232 0075 B3 08 CALL TEST_CORD
233 0077 EB 0081 R MOV DX,CX
234 007A BB D1 MOV CX,CX ; SAVE B(Y) VALUE
235 007C 59 POP CX ; GET B(X) VALUE
236 007D 5B POP BX ; GET A(Y) VALUE
237 007E 58 POP AX ; GET A(X) VALUE
238 007F EB DA JMP JOY_1 ; FINISHED - RETURN
239
240 0081 TEST_CORD PROC NEAR
241 0081 52 PUSH DX ; SAVE
242 0082 FA CLD ; BLOCK INTERRUPTS WHILE READING
243 0083 B0 00 MOV AL,0 ; SET UP TO LATCH TIMER 0
244 0085 EB 43 OUT TIMER+3,AL
245 0087 EB 00 JMP $+2
246 0089 E4 40 IN AL,TIMER ; READ LOW BYTE OF TIMER 0
247 008B EB 00 JMP $+2
248 008D BA E0 MOV AH,AL
249 008F E4 40 IN AH,TIMER ; READ HIGH BYTE OF TIMER 0
250 0091 86 E0 XCHG AH,AL ; REARRANGE TO HIGH,LOW
251 0093 50 PUSH AX ; SAVE
252 0094 B9 04FF MOV CX,4FFH ; SET COUNT
253 0097 EE OUT DX,AL ; FIRE TIMER
254 0098 EB 00 JMP $+2
255 009A
256 009A EC TEST_CORD_1: IN AL,DX ; READ VALUES
257 009B 84 C3 TEST AL,BL ; HAS PULSE ENDED?
258 009D E0 FB LOOPNZ TEST_CORD_1
259 009F 83 F9 00 CMP CX,0
260 00A2 59 POP CX ; ORIGINAL COUNT
261 00A3 75 04 JNZ SHORT TEST_CORD_2
262 00A5 2B C9 SUB CX,CX ; SET 0 COUNT FOR RETURN
263 00A7 EB 2D JMP SHORT TEST_CORD_3 ; EXIT WITH COUNT = 0
264 00A9
265 00A9 B0 00 TEST_CORD_2: MOV AL,0 ; SET UP TO LATCH TIMER 0
266 00AB E6 43 OUT TIMER+3,AL
267 00AD EB 00 JMP $+2
268 00AF E4 40 IN AL,TIMER ; READ LOW BYTE OF TIMER 0
269 00B1 8A E0 MOV AH,AL
270 00B3 EB 00 JMP $+2
271 00B5 E4 40 IN AH,TIMER ; READ HIGH BYTE OF TIMER 0
272 00B7 86 E0 XCHG AH,AL ; REARRANGE TO HIGH,LOW
273
274 00B9 3B C8 CMP CX,AX ; CHECK FOR COUNTER WRAP
275 00BB 73 0B JAE TEST_CORD_4 ; GO IF NO
276 00BD 52 PUSH DX
277 00BE BA FFFF MOV DX,-1
278
279 00C1 2B D0 SUB DX,AX ; ADJUST FOR WRAP
280 00C3 03 CA ADD CX,DX
281 00C5 5A POP DX
282 00C6 EB 02 JMP SHORT TEST_CORD_5
283
284 00C8 TEST_CORD_4: SUB CX,AX
285 00C8 2B C8 TEST_CORD_5: AND CX,1FF0H ; ADJUST
286 00CA 81 E1 1FF0
287 00CC 01 E9 SHR CX,1
288 00CE 01 E9 SHR CX,1
289 00D0 01 E9 SHR CX,1
290 00D2 01 E9 SHR CX,1
291 00D4 01 E9 SHR CX,1
292
293 00D6 TEST_CORD_3: STI ; INTERRUPTS BACK ON
294 00D6 FB MOV DX,201H ; FLUSH OTHER INPUTS
295 00D7 BA 0201 PUSH CX
296 00DA 51 PUSH AX
297 00DB 50 MOV CX,4FFH ; COUNT
298 00DC B9 04FF
299 00DF TEST_CORD_6: IN AL,DX
300 00DF EC

```

```
301 00E0 A8 0F          TEST    AL,0FH
302 00E2 E0 FB          LOOPNZ  TEST_CORD_6
303
304 00E4 58             POP     AX
305 00E5 59             POP     CX
306 00E6 5A             POP     DX                ; SET COUNT
307
308 00E7 C3             RET                    ; RETURN
309
310 00E8                TEST_CORD             ENDP
311 00E8                JOY_STICK             ENDP
312
313 00E8                CODE                 ENDS
314                                END
```

```

1      PAGE 118,121
2      TITLE POST ----- 01/10/86 SYSTEM POST AND BIOS PROCEDURES
3
4      PUBLIC A1
5      PUBLIC BEEP
6      PUBLIC CONF_TBL
7      PUBLIC CRT_CHAR_GEN
8      PUBLIC DDS
9      PUBLIC DISK_BASE
10     PUBLIC M5
11     PUBLIC M6
12     PUBLIC M7
13     PUBLIC MD_TBL1
14     PUBLIC MD_TBL2
15     PUBLIC MD_TBL3
16     PUBLIC MD_TBL4
17     PUBLIC MD_TBL5
18     PUBLIC MD_TBL6
19     PUBLIC P_O_R
20     PUBLIC RESET
21     PUBLIC VIDEO_PARMS
22     PUBLIC WAITF
23
24     EXTRN CASSETTE_IO_1:NEAR
25     EXTRN DISKETTE_IO_1:NEAR
26     EXTRN DISK_INT_1:NEAR
27     EXTRN DISKETTE_SETUP:NEAR
28     EXTRN KB_INT_1:NEAR
29     EXTRN KEYBOARD_IO_1:NEAR
30     EXTRN NEC_OUTPUT:NEAR
31     EXTRN PRINTER_IO_1:NEAR
32     EXTRN RESULTS:NEAR
33     EXTRN RS232_IO_1:NEAR
34     EXTRN SEEK:NEAR
35     EXTRN VIDEO_IO_1:NEAR
36
37     EXTRN SET_MODE:NEAR
38     EXTRN SET_CTYPE:NEAR
39     EXTRN SET_OPS:NEAR
40     EXTRN READ_CURSOR:NEAR
41     EXTRN READ_LPEN:NEAR
42     EXTRN ACT_DISP_PAGE:NEAR
43     EXTRN SCROLL_UP:NEAR
44     EXTRN SCROLL_DOWN:NEAR
45     EXTRN READ_AC_CURRENT:NEAR
46     EXTRN WRITE_AC_CURRENT:NEAR
47     EXTRN WRITE_C_CURRENT:NEAR
48     EXTRN SET_COLOR:NEAR
49     EXTRN WRITE_DOT:NEAR
50     EXTRN READ_DOT:NEAR
51     EXTRN WRITE_TTY:NEAR
52     EXTRN VIDEO_STATE:NEAR
53     .LIST
54     -----
55     ; THE BIOS ROUTINES ARE MEANT TO BE ACCESSED THROUGH
56     ; SOFTWARE INTERRUPTS ONLY. ANY ADDRESSES PRESENT IN
57     ; THE LISTINGS ARE INCLUDED ONLY FOR COMPLETENESS,
58     ; NOT FOR REFERENCE. APPLICATIONS WHICH REFERENCE
59     ; ABSOLUTE ADDRESSES WITHIN THE CODE SEGMENT
60     ; VIOLATE THE STRUCTURE AND DESIGN OF BIOS.
61     -----
62     -----
63     ROM RESIDENT CODE
64     -----
65     CODE SEGMENT BYTE PUBLIC
66
67     0000 0000 1FFF [ CC ] DB 01FFFH DUP (0CCH) ; FILL UNUSED LOCATIONS WITH INTERRUPT 3
68
69     ;
70     ; ORG 0E000H
71     ; ORG 0
72     0000 0000 36 32 58 30 38 35 DB '62X0851 COPR. IBM 1986' ; COPYRIGHT NOTICE
73     0000 36 32 58 30 38 35
74     0000 31 20 43 4F 50 52
75     0000 2E 20 49 42 4D 20
76     0000 31 39 38 36
77
78     -----
79     ; INITIAL RELIABILITY TESTS -- PHASE 1
80     -----
81
82     ASSUME CS:CODE,SS:CODE,ES:ABS0,DS:DATA
83
84     0016 00D5 R C1 DW C11 ; RETURN ADDRESS
85     0018 0181 R C2 DW C24 ; RETURN ADDRESS FOR DUMMY STACK
86     001A 20 4B 42 20 4F 4B F3B DB 'KB OK',CR ; KB FOR MEMORY SIZE
87     0D
88
89     -----
90     ; LOAD A BLOCK OF TEST CODE THROUGH THE KEYBOARD PORT
91     ; FOR MANUFACTURING TEST
92     ; THIS ROUTINE WILL LOAD A TEST (MAX LENGTH=FAFFH) THROUGH
93     ; THE KEYBOARD PORT. CODE WILL BE LOADED AT LOCATION
94     ; 0000:0500. AFTER LOADING, CONTROL WILL BE TRANSFERRED
95     ; TO LOCATION 0000:0500. STACK WILL BE LOCATED JUST BELOW
96     ; THE TEST CODE. THIS ROUTINE ASSUMES THAT THE FIRST 2
97     ; BYTES TRANSFERED CONTAIN THE COUNT OF BYTES TO BE LOADED
98     ; (BYTE 1=COUNT LOW, BYTE 2=COUNT HI.)
99     -----
100    ;---- FIRST, GET THE COUNT
101
102    MFG_BOOT:
103    CALL SP_TEST ; GET COUNT LOW
104    MOV BH,BL ; SAVE IT
105    CALL SP_TEST ; GET COUNT HI
106    MOV CH,BL
107    MOV CL,BH ; CX NOW HAS COUNT
108    CLD ; SET DIR. FLAG TO INCRIMENT
109    CLI
110    MOV DI,0500H ; SET TARGET OFFSET (DS=0000)
111    MOV AL,0FDH ; UNMASK K/B INTERRUPT
112    OUT INTA0,AL
113    MOV AL,0AH ; SEND READ INT. REQUEST REG. CMD
114    OUT INTA0,AL
115    MOV DX,PORT_B ; SET UP PORT B ADDRESS

```

SECTION B

```

228 PAGE
229 -----
230 8237 DMA INITIALIZATION CHANNEL REGISTER TEST
231 DESCRIPTION
232 DISABLE THE 8237 DMA CONTROLLER. VERIFY THAT
233 TIMER 1 FUNCTIONS OK. WRITE/READ THE CURRENT
234 ADDRESS AND WORD COUNT REGISTERS FOR ALL
235 CHANNELS. INITIALIZE AND START DMA FOR MEMORY
236 REFRESH.
237 -----
238
239 1----- DISABLE DMA CONTROLLER
240
241 00D7 B0 02 MOV AL,02H
242 00D9 E6 60 OUT PORT_A,AL
243 00DB B0 04 MOV AL,04
244 00DD E6 08 OUT DMA08,AL
245
246
247 1----- VERIFY THAT TIMER 1 FUNCTIONS OK
248
249 00DF B0 54 MOV AL,54H
250 00E1 E6 43 OUT TIMER+3,AL
251 00E3 8A C1 MOV AL,C1
252 00E5 E6 41 OUT TIMER+1,AL
253
254 00E7 MOV AL,40H
255 00E9 E6 43 OUT TIMER+3,AL
256 00EB 80 FB FF CMP BL,OFFH
257 00ED 74 07 JE C13
258 00EF 0A 08 OR AL,TIMER+1
259 00F1 E2 F1 LOOP C12
260 00F3 F4 HLT
261
262 00F7 8A C3 MOV AL,BL
263 00F9 2B C9 SUB CX,CX
264 00FB E6 41 OUT TIMER+1,AL
265
266 00FD B0 40 MOV AL,40H
267 00FF E6 43 OUT TIMER+3,AL
268 0101 90 NOP
269 0103 90 NOP
270 0105 E4 41 IN AL,TIMER+1
271 0107 22 08 AND BL,AL
272 0109 74 03 JZ C15
273 010B E2 F2 LOOP C14
274 010D F4 HLT
275
276 1----- INITIALIZE TIMER 1 TO REFRESH MEMORY
277
278 010C B0 03 MOV AL,03H
279 010E E6 60 OUT PORT_A,AL
280
281 0110 E6 0D OUT DMA+0DH,AL
282
283 1----- WRAP DMA CHANNELS ADDRESS AND COUNT REGISTERS
284
285 0112 B0 FF MOV AL,OFFH
286 0114 8A D8 MOV BL,AL
287 0116 8A F8 MOV BH,AL
288 0118 B9 0008 MOV CX,8
289 011B BA 0000 MOV DX,DMA
290 011E EE OUT DX,AL
291 011F 50 PUSH AX
292 0120 EE OUT DX,AL
293 0121 B0 01 MOV AL,01H
294 0123 EC IN AL,DX
295 0124 8A E0 MOV AH,AL
296 0126 EC IN AL,DX
297 0127 3B D8 CMP BX,AX
298 0129 74 01 JE C18
299 012B HLT
300 012B F4 HLT
301 012C INC DX
302 012C 42 INC STC
303 012D F9 LOOP C17
304 012E E2 EE JNC C17A
305 0130 73 F9 JZ C17A
306 0132 FE C0 INC AL
307 0134 74 DE JZ C16
308
309 1----- INITIALIZE AND START DMA FOR MEMORY REFRESH.
310
311 0136 8E DB MOV DS,BX
312 0138 E6 C3 MOV ES,BX
313 ASSUME DS:ABS0,ES:ABS0
314 013A B0 FF MOV AL,0FFH
315 013C E6 01 OUT DMA+1,AL
316 013E 50 PUSH AX
317 013F E6 01 OUT DMA+1,AL
318 0141 B0 58 MOV AL,058H
319 0143 E6 0B OUT DMA+0BH,AL
320 0145 B0 00 MOV AL,0
321 0147 8A E8 MOV CH,AL
322 0149 E6 08 OUT DMA+8,AL
323 014B 50 PUSH AX
324 014C E6 0A OUT DMA+10,AL
325 014E B0 12 MOV AL,12H
326 0150 E6 41 OUT TIMER+1,AL
327 0152 B0 41 MOV AL,41H
328 0154 E6 0B OUT DMA+0BH,AL
329 0156 50 PUSH AX
330 0157 E4 08 IN AL,DMA+08
331 0159 24 10 AND AL,00010000B
332 015B 70 01 JZ C18C
333 015D F4 HLT
334 015E B0 42 MOV AL,42H
335 0160 E6 0B OUT DMA+0BH,AL
336 0162 B0 43 MOV AL,43H
337 0164 E6 0B OUT DMA+0BH,AL

```

```

338 PAGE
339 -----
340 : BASE 64K READ/WRITE STORAGE TEST :
341 : DESCRIPTION :
342 : WRITE/READ/VERIFY DATA PATTERNS :
343 : AA,55,FF,01, AND 00 TO 1ST 64K OF :
344 : STORAGE. VERIFY STORAGE ADDRESSABILITY. :
345 -----
346
347 0166 AD LODSW ; ALLOW RAM CHARGE TIME.
348 0167 AD LODSW
349 0168 AD LODSW
350 0169 AD LODSW
351
352 :----- DETERMINE MEMORY SIZE AND FILL MEMORY WITH DATA
353
354 016A 8B 1E 0472 R MOV BX,DATA_WORD[RESET_FLAG-DATA40] ; SAVE 'RESET_FLAG' IN BX
355 016E 8B 2E 0496 R MOV BP,DATA_WORD[KB_FLAG_3-DATA40] ; SAVE KEYBOARD TYPE
356 0172 B9 8000 MOV CX,08000H ; SET FOR 32K WORDS
357 0175 81 FB 1234 CMP BX,1234H ; WARM START?
358 0179 74 16 JE CLR_STG
359 017B CB 0018 R MOV SP,OFFSET C2
360 017E 59 00CF R JMP STG_STG_CNT
361 0181 74 12 JE HOW_BIG ; STORAGE OK, DETERMINE SIZE
362 0183 8A D8 MOV BL,AL ; SAVE FAILING BIT PATTERN
363 0185 B0 04 MOV AL,04H ;
364 0187 E6 60 OUT PORT_A,AL ; <---CHECKPOINT 4--->
365 0189 2B C9 SUB CX,CX ; BASE RAM FAILURE - HANG
366 018B E2 FE C24B: LOOP C24B ; FLIPPING BETWEEN 04 AND
367 018D 86 D8 XCHG BL,AL ; FAILING BIT PATTERN
368 018F EB F6 JMP C24A
369 0191 CLR_STG:
370 0191 2B C0 SUB AX,AX ; MAKE AX=0000
371 0193 F3/ AB REP STOSW ; STORE 32K WORDS OF 0000
372 0195
373 0195 89 1E 0472 R MOV BX,DATA_WORD[RESET_FLAG-DATA40],BX ; RESTORE RESET FLAG
374 0199 83 FD 10 CMP BP,KBX ; IS THE KBX BIT THE ONLY ONE ON?
375 019C 74 02 JE C24C ; IF NOT THEN THIS MUST BE A P.O.R.
376 019E 2B ED SUB BP,BP ; IF P.O.R. THEN INITIALIZE THIS TO ZERO
377 01A0
378 01A0 89 2E 0496 R MOV BP,SP ; RESTORE RESET FLAG
379 01A4 2B ED SUB BP,BP ; BP IS USED LATER AS AN ERROR INDICATOR
380 01A6 8A 0400 MOV DX,0400H ; SET POINTER TO JUST>16KB
381 01A9 8B 0010 MOV BX,16 ; BASIC COUNT OF 16K
382 01AC
383 01AC 8E C2 MOV ES,DX ; SET SEG. REG.
384 01AE 2B FF SUB DI,DI ;
385 01B0 B8 AA55 MOV AX,0AA55H ; TEST PATTERN
386 01B3 8B C8 MOV CX,AX ; PUT PATTERN
387 01B5 26 89 05 MOV ES:[DI],AX ; SEND PATTERN TO MEM.
388 01B8 B0 0F MOV AL,0FH ; PUT SOMETHING IN AL
389 01BA 26 8B 05 MOV AX,ES:[DI] ; GET PATTERN
390 01BD 33 C1 XOR AX,CX ; COMPARE PATTERNS
391 01BF 75 11 JNZ GO_END ; GO END IF NO COMPARE
392 01C1 B9 2000 MOV CX,2000H ; SET COUNT FOR 8K WORDS
393 01C4 F3/ AB REP STOSW ; FILL 8K WORDS
394 01C6 81 C2 0400 ADD DX,400H ; POINT TO NEXT 16KB BLOCK
395 01CA C3 C3 10 ADD DX,DX ; BUMP COUNT BY 16KB
396 01CD 80 FE A0 CMP DH,0A0H ; TOP OF RAM AREA YET? (A0000)
397 01D0 75 DA JNZ FILL_LOOP
398 01D2
399 01D2 89 1E 0413 R MOV DATA_WORD[MEMORY_SIZE-DATA40],BX ; SAVE MEMORY SIZE
400
401 :----- SETUP STACK SEG AND SP
402
403 01D6 B8 0030 MOV AX,STACK_SS ; GET STACK VALUE
404 01D9 8E D0 MOV SS,AX ; SET THE STACK UP
405 01DB BC 0100 MOV SP,TOS ; STACK IS READY TO GO
406
407 :----- INITIALIZE THE 8259 INTERRUPT CONTROLLER CHIP :
408 -----
409 01DE B0 13 C25: MOV AL,13H ; ICW1 - EDGE, SNGL, ICW4
410 01E0 E6 20 OUT INTA00,AL ;
411 01E2 B0 08 MOV AL,8 ; SETUP ICW2 - INT TYPE 8 (8-F)
412 01E4 E6 21 OUT INTA01,AL ;
413 01E6 B0 09 MOV AL,9 ; SETUP ICW4 - BUFFRD,8086 MODE
414 01E8 E6 21 OUT INTA01,AL ;
415 01EA B0 FF MOV AL,0FFH ; MASK ALL INTS. OFF
416 01EC E6 21 OUT INTA01,AL ; (VIDEO ROUTINE ENABLES INTS.)
417
418 :----- SET UP THE INTERRUPT VECTORS TO TEMP INTERRUPT
419
420 01EE 1E PUSH DS
421 01EF B9 0020 MOV CX,32 ; FILL ALL 32 INTERRUPTS
422 01F2 2B FF SUB DI,DI ; FIRST INTERRUPT LOCATION
423 01F4 8E C7 MOV ES,DI ; SET ES=0000 ALSO
424 01F6 B8 1F23 R D3: MOV AX,OFFSET D11 ; MOVE ADDR OF INTR PROC TO TBL
425 01F9 AB STOSW
426 01FA 8C C8 MOV AX,CX ; GET ADDR OF INTR PROC SEG
427 01FC AB STOSW
428 01FD E2 F7 LOOP D3 ; VECTBL0
429
430 :----- ESTABLISH BIOS SUBROUTINE CALL INTERRUPT VECTORS
431
432 01FF BF 0040 R MOV DI,OFFSET VIDEO_INT ; SETUP ADDR TO INTR AREA
433 0202 0E CS PUSH CS ;
434 0203 1F POP DS ; SETUP ADDR OF VECTOR TABLE
435 0204 BE 1F03 R MOV SI,OFFSET VECTOR_TABLE+16 ; START WITH VIDEO ENTRY
436 0207 B9 0010 MOV CX,16 ;
437 020A A5 MOVSW ; MOVE VECTOR TABLE TO RAM
438 020B 47 INC DI ; SKIP SEGMENT POINTER
439 020C 47 INC DI ;
440 020D E2 FB LOOP D3A ;
441
442 :----- DETERMINE CONFIGURATION AND MFG. MODE :
443 -----
444
445 020F 1F POP DS ;
446 0210 1E PUSH DS ; RECOVER DATA SEG
447 0211 E4 62 IN AL,PORT_C ; GET SWITCH INFO
448 0213 24 0F AND AL,00001111B ; ISOLATE SWITCHES
449 0215 8A ED MOV AH,AL ; SAVE
450 0217 B0 AD MOV AL,10101101B ;
451 0219 E6 61 OUT PORT_B,AL ; ENABLE OTHER BANK OF SWS.

```

```

452 021B 90                      NOP
453 021C E4 62                  IN     AL,PORT_C
454 021E B1 04                  MOV     CL,4
455 0220 D2 C0                  MOV     AL,CL          ; ROTATE TO HIGH NIBBLE
456 0222 24 F0                  AND     AL,11110000B
457 0224 0A C4                  OR      AL,AH          ; COMBINE WITH OTHER BANK
458 0226 2A E4                  SUB     AH,AH
459 0228 A3 0410 R              MOV     DATA_WORD[0EQUIP_FLAG-DATA40],AX      ; SAVE SWITCH INFO
460 022B B0 99                  MOV     AL,99H
461 022D E6 63                  OUT     CMD_PORT,AL
462 022F E8 19E3 R              CALL    KBD_RESET          ; SEE IF MFG. JUMPER IN
463 0232 B0 FB EA              CMP     BL,0EAH          ; IS THIS THE EXTENDED KEYBOARD?
464 0235 75 08                  JNE     KBX1              ; IF NOT THEN LEAVE THE FLAG ALONE
465 0237 C6 06 0496 R 10       MOV     DATA_AREA[0KB_FLAG_3-DATA40],KBX ; EXTENDED KEYBOARD
466 023C EB 22 90              JMP     E6                ; DONE WITH KEYBOARD HERE
467 023F                      ;
468 023F 80 FB AA              KBX1:  CMP     BL,0AAH          ; KEYBOARD PRESENT?
469 0242 74 1C                  JE      E6                ;
470 0244 80 FB 65              CMP     BL,065H          ; LOAD MFG. TEST REQUEST?
471 0247 75 03                  JNE     D3B1              ;
472 0249 E9 0021 R              JMP     MFG_BOOT          ; GO TO BOOTSTRAP IF 50
473 024C                      ;
474 024C 0A DB              D3B1:  OR      BL,BL          ; MFG PLUG IN?
475 024E 75 10                  JNZ     E6                ; NO
476 0250 B0 38                  MOV     AL,38H
477 0252 E6 61                  OUT     PORT_B,AL
478 0254 90                      NOP
479 0255 90                      NOP
480 0256 E4 60                  IN      AL,PORT_A
481 0258 24 FF                  AND     AL,0FFH          ; WAS DATA LINE GROUNDED
482 025A 75 04                  JNZ     E6                ;
483 025C FE 06 0412 R          INC     DATA_AREA[0MFG_TST-DATA40] ; SET MANUFACTURING TEST FLAG
484
485
486
487
488
489
490
491
492
493
494
495 0260                      ;-----
496 0260 A1 0410 R              E6:    MOV     AX,DATA_WORD[0EQUIP_FLAG-DATA40] ; GET SENSE SWITCH INFO
497 0263 57                      PUSH    AX                ; SAVE IT
498 0264 B0 30                  MOV     AL,30H
499 0266 A3 0410 R              MOV     DATA_WORD[0EQUIP_FLAG-DATA40],AX
500 0269 2A E4                  SUB     AH,AH
501 026B CD 10                  INT     10H              ; SEND INIT TO B/W CARD
502 026D B0 20                  MOV     AL,20H
503 026F A3 0410 R              MOV     DATA_WORD[0EQUIP_FLAG-DATA40],AX
504 0272 2A E4                  SUB     AH,AH            ; AND INIT COLOR CARD
505 0274 CD 10                  INT     10H
506 0276 58                      POP      AX              ; RECOVER REAL SWITCH INFO
507 0277 A3 0410 R              MOV     DATA_WORD[0EQUIP_FLAG-DATA40],AX ; RESTORE IT
508
509 027A 24 30                  AND     AL,30H           ; AND CONTINUE
510 027C 75 0A                  JNZ     E7                ; ISOLATE VIDEO SWS
511 027E BF 0040 R              MOV     DI,OFFSET 0VIDEO INT ; SET INT 10H TO DUMMY
512 0281 C7 05 1F49 R          MOV     WORD_PTR [DI],OFFSET DUMMY RETURN ; RET IF NO VIDEO CARD
513 0285 E9 033B R              JMP     E1B1              ; BYPASS VIDEO TEST
514 0288                      ;
515 0288 3C 30                  ET:    CMP     AL,30H           ; TEST VIDEO:
516 028A 74 08                  JE      E7                ; B/W CARD ATTACHED?
517 028C FE C4                  INC     AH                ; YES - SET MODE FOR B/W CARD
518 028E 3C 20                  CMP     AL,20H           ; SET COLOR MODE FOR COLOR CD
519 0290 75 02                  JNE     EB                ; 80X25 MODE SELECTED?
520 0292 B4 03                  MOV     AH,3             ; NO - SET MODE FOR 40X25
521 0294 86 E0                  XCHG    AH,AL            ; SET MODE:
522 0296 50                      PUSH    AX                ; SAVE VIDEO MODE ON STACK
523 0297 2A E4                  SUB     AH,AH            ; INITIALIZE TO ALPHANUMERIC MD
524 0299 CD 10                  INT     10H              ; CALL VIDEO IO
525 029B 58                      POP      AX              ; RESTORE VIDEO SENSE SWS IN AH
526 029C 50                      PUSH    AX                ; RESAVE VALUE
527 029D BB B000              MOV     BX,0B000H        ; BEG VIDEO RAM ADDR B/W CD
528 02A0 EB 24                  JMP     SHORT E8A         ;
529
530
531
532
533 02C3                      ;----- UNNATURAL ACT FOR ADDRESS COMPATIBILITY
534 02C3                      ;
535 02C3 E9 185C R              ; ORG 0E2C3H
536
537 02C6                      ; ORG 002C3H
538 02C6 BA 03BB              NMI_INT: JMP     NMI_INT_1
539 02C9 B9 0800              ;
540 02CC B0 01                  ;
541 02CE 80 FC 30              ;
542 02D1 74 09                  ;
543 02D3 B7 B8                  ;
544 02D5 BA 03DB              ;
545 02D8 B5 20                  ;
546 02DA FE C8                  ;
547 02DC                      ;
548 02DC EE                      ;
549 02DD 81 3E 0472 R 1234    ;
550 02E3 8E 09                  ;
551 02E5 74 07                  ;
552 02E7 8E DB              ;
553
554 02E9 E8 0CCF R              ;
555 02EC 75 33                  ;
556
557
558
559
560
561
562
563 02EE                      ;-----
564 02EE 58                      ; SETUP VIDEO DATA ON SCREEN FOR VIDEO
565 02EF 50                      ; LINE TEST.
566
567
568
569
570
571
572
573 02EE                      ; DESCRIPTION
574
575
576
577
578
579
580
581
582
583 02EE                      ; ENABLE VIDEO SIGNAL AND SET MODE.
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

```



```

566 02F0 B4 00      MOV     AH,0          ; ENABLE VIDEO AND SET MODE
567 02F2 CD 10      INT     10H         ; VIDEO
568 02F4 B8 7020    MOV     AX,7020H      ; WRT BLANKS IN REVERSE VIDEO
569
570 02F7 2B FF      SUB     DI,DI          ; SETUP STARTING LOC
571 02F9 B9 0028    MOV     CX,40         ; NO. OF BLANKS TO DISPLAY
572 02FC F3/ AB     REP     STOSW       ; WRITE VIDEO STORAGE
573
574
575
576
577
578
579
580 02FE 58          POP     AX          ; GET VIDEO SENSE SW INFO
581 02FF 50          PUSH    AX          ; SAVE IT
582 0300 80 FC 30    CMP     AH,30H        ; B/W CARD ATTACHED?
583 0303 B4 03BA     MOV     DX,03BAH      ; SETUP ADDR OF BW STATUS PORT
584 0306 74 03       JE      E11         ; YES - GO TEST LINES
585 0308 BA 03DA     MOV     DX,03DAH      ; COLOR CARD IS ATTACHED
586 030B             E11:      MOV     AH,8          ; LINE_TST
587 030B B4 08       MOV     AH,8          ;
588 030D             E12:      SUB     CX,CX          ; OFLOOP_CNT
589 030D 2B C9       SUB     CX,CX          ;
590 030F             E13:      IN      AL,DX          ; READ CRT STATUS PORT
591 030F EC          AND     AL,AH        ; CHECK VIDEO/HORIZ LINE
592 0310 22 C4       JNZ     E13         ; ITS ON - CHECK IF IT GOES OFF
593 0312 75 04       LOOP    E13         ; LOOP TILL ON OR TIMEOUT
594 0314 E2 F9       JMP     SHORT E17        ; GO PRINT ERROR MSG
595 0316 EB 09       E14:      SUB     CX,CX          ;
596 0318             E15:      IN      AL,DX          ; READ CRT STATUS PORT
597 0318 2B C9       AND     AL,AH        ; CHECK VIDEO/HORIZ LINE
598 031A             JZ      E16         ; ITS ON - CHECK NEXT LINE
599 031A EC          LOOP    E15         ; LOOP IF OFF TILL IT GOES ON
600 031B 22 C4       E17:      POP     DS          ; CRT_ERR
601 031D 74 11       PUSH    DS          ;
602 031F E2 F9       MOV     DS:0MFG_ERR_FLAG,06H ; <<<<<<CRT ERR CHKPT. 06<<<<<<
603 0321 IF          MOV     DX,102H      ;
604 0321 IF          CALL    ERR_BEEP      ; GO BEEP SPEAKER
605 0322 1E          JMP     SHORT E18        ;
606 0323 C6 06 0015 R 06 E16:      MOV     CL,3          ; NXT LINE
607 0328 BA 0102     SHR     AH,CL          ; GET NEXT BIT TO CHECK
608 032B E8 19A5 R   JNZ     E12         ;
609 032E EB 06       E18:      GO CHECK HORIZONTAL LINE
610 0330             DISPLAY CURSOR
611 0330 B1 03       POP     AX          ; GET VIDEO SENSE SWs (AH)
612 0332 D2 EC       MOV     AH,0          ; SET MODE AND DISPLAY CURSOR
613 0334 75 D7       INT     10H          ; CALL VIDEO I/O PROCEDURE
614 0336             E18B:      MOV     DX,0C000H      ; SEE IF ADVANCED VIDEO CARD
615 0338 58          E18A:      MOV     DS,DX          ; IS PRESENT
616 0337 B4 00       SUB     BX,BX          ;
617 0339 CD 10       MOV     AX,[BX]      ; GET FIRST 2 LOCATIONS
618 033B             POP     BX          ;
619 033B BA C000     CMP     AX,0AA55H      ; LET BUS SETTLE
620 033E             JNZ     E18B        ; PRESENT?
621 033E 5B          CALL    ROM_CHECK    ; NO? GO LOOK FOR OTHER MODULES
622 0340 3D AA55     JMP     SHORT E18C      ; GO SCAN MODULE
623 0342 8B 07       E18B:      ADD     DX,0080H      ; POINT TO NEXT 2K BLOCK
624 0344 53          E18C:      CMP     DX,0C800H      ; TOP OF VIDEO ROM AREA YET?
625 0345 5B          JL      E18A        ; GO SCAN FOR ANOTHER MODULE
626 0346 3D AA55     E18A:      MOV     DS,DX          ;
627 0349 75 05       SUB     BX,BX          ;
628 034B E8 1920 R   PUSH    BX          ;
629 034E EB 04       POP     BX          ;
630 0350             CMP     AX,0AA55H      ;
631 0350 81 C2 0080  JNZ     E18B        ;
632 0354             CALL    ROM_CHECK    ;
633 0354 81 FA C800  JMP     SHORT E18C      ;
634 0358 7C E4       E18B:      ADD     DX,0080H      ;
635
636
637
638
639
640
641
642
643
644 035A IF          E18C:      CMP     DX,0C800H      ;
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666 0374 A2 046B R   ; 8259 INTERRUPT CONTROLLER TEST
667 0377 FB          ; DESCRIPTION
668 037A             ; READ/WRITE THE INTERRUPT MASK REGISTER (IMR)
669 037A             ; WITH ALL ONES AND ZEROES, ENABLE SYSTEM
670 037A E2 FE       ; INTERRUPTS. MASK DEVICE INTERRUPTS OFF. CHECK
671 037C             ; FOR HOT INTERRUPTS (UNEXPECTED).
672 037C E2 FE       ;
673 037E 80 3E 046B R 00 C21:      ASSUME DS:AB50
674 0383 74 05       C21:      POP     DS
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

```



```

794 0448 B0 FE      MOV     AL,0FEH      ; ENABLE TIMER INTERRUPT
795 044A E6 21      OUT     INTA01,AL
796
797 ;-----
798 ; EXPANSION I/O BOX TEST
799 ; CHECK TO SEE IF EXPANSION BOX PRESENT - IF INSTALLED,
800 ; TEST DATA AND ADDRESS BUSES TO I/O BOX
801 ; ERROR='1801'
802 ;-----
803
804 ;----- DETERMINE IF BOX IS PRESENT
805
806 EXP_10:
807     MOV     DX,0210H      ; (CARD WAS ENABLED EARLIER)
808     MOV     AX,5555H      ; CONTROL PORT ADDRESS
809     OUT     DX,AL         ; SET DATA PATTERN
810     MOV     AL,01H      ; MAKE AL DIFFERENT
811     IN      AL,DX        ; RECOVER DATA
812     CMP     AL,AH        ; REPLY?
813     JNE     E19          ; NO RESPONSE, GO TO NEXT TEST
814     NOT     AX           ; MAKE DATA=AAAA
815     OUT     DX,AL
816     MOV     AL,01H
817     IN      AL,DX        ; RECOVER DATA
818     CMP     AL,AH
819     JNE     E19
820
821 ;----- CHECK ADDRESS BUS
822
823 EXP2:
824     MOV     BX,0001H
825     MOV     DX,0215H
826     MOV     CX,0016
827
828 EXP3:
829     MOV     CS:[BX],AL    ; WRITE ADDRESS F0000+BX
830     NOP
831     IN      AL,DX        ; READ ADDR. HIGH
832     CMP     AL,BH
833     JNE     EXP_ERR      ; GO ERROR IF MISCOMPARE
834     INC     DX           ; DX=216H (ADDR. LOW REG)
835     CMP     AL,BL
836     JNE     EXP_ERR      ; COMPARE TO LOW ADDRESS
837     DEC     DX           ; DX BACK TO 215H
838     SHL     BX,1
839     LOOP    EXP3         ; LOOP TILL '1' WALKS ACROSS BX
840
841 ;----- CHECK DATA BUS
842
843 EXP4:
844     MOV     CX,0008      ; DO 8 TIMES
845     MOV     AL,01
846     DEC     DX
847     MOV     AH,AL
848     OUT     DX,AL
849     MOV     AL,01H
850     IN      AL,DX
851     CMP     AL,AH
852     JNE     SHORT EXP_ERR
853     SHL     AL,1
854     LOOP    EXP4
855     JMP     SHORT E19
856
857 EXP_ERR:
858     MOV     SI,OFFSET F3C ; (1801)
859     CALL    E_MSG
860
861 ;-----
862 ; ADDITIONAL READ/WRITE STORAGE TEST
863 ; DESCRIPTION
864 ; WRITE/READ DATA PATTERNS TO ANY READ/WRITE
865 ; STORAGE AFTER THE FIRST 64K. STORAGE
866 ; ADDRESSABILITY IS CHECKED,
867 ;-----
868 ASSUME DS:DATA
869
870 E19:
871     CALL    DDS
872     PUSH    DS
873
874 E20:
875     CMP     0,RESET_FLAG,1234H ; WARM START?
876     JNE     E20A          ; CONTINUE TEST IF NOT
877     JMP     ROM_SCAN      ; GO TO NEXT ROUTINE IF 50
878
879 E20A:
880     MOV     AX,64
881     JMP     SHORT PRT_SIZ ; STARTING AMT. OF MEMORY OK
882
883 E20B:
884     MOV     BX,0MEMORY_SIZE ; GET MEM. SIZE WORD
885     SUB     BX,64          ; 1ST 64K ALREADY DONE
886     MOV     CL,4
887     SHR     BX,CL
888     MOV     CX,BX
889     MOV     BX,1000H
890
891 E21:
892     MOV     DS,BX
893     MOV     SI,5
894     ADD     BX,0400H
895     PUSH    DX
896     PUSH    CX
897     PUSH    AX
898     MOV     CX,02000H
899     CALL    STGTEST_CNT ; SET COUNT FOR 8K WORDS
900     JNZ     E21A          ; GO PRINT ERROR
901     POP     AX
902     POP     AX
903     ADD     AX,16
904
905 PRT_SIZ:
906     PUSH    AX
907     MOV     BX,10
908     MOV     CX,3
909
910 DECIMAL_LOOP:
911     XOR     DX,DX
912     DIV     BX
913     OR      DL,30H
914     PUSH    DX
915     LOOP    DECIMAL_LOOP ; DIVIDE BY 10
916
917 PRT_DEC_LOOP:
918     MOV     CX,3
919     MOV     DX,DX
920     OR      DL,30H
921     PUSH    DX
922     LOOP    PRT_DEC_LOOP ; MAKE INTO ASCII
923
924     POP     DX
925     POP     DX
926     POP     DX
927     POP     DX
928     POP     DX
929     POP     DX
930     POP     DX
931     POP     DX
932     POP     DX
933     POP     DX
934     POP     DX
935     POP     DX
936     POP     DX
937     POP     DX
938     POP     DX
939     POP     DX
940     POP     DX
941     POP     DX
942     POP     DX
943     POP     DX
944     POP     DX
945     POP     DX
946     POP     DX
947     POP     DX
948     POP     DX
949     POP     DX
950     POP     DX
951     POP     DX
952     POP     DX
953     POP     DX
954     POP     DX
955     POP     DX
956     POP     DX
957     POP     DX
958     POP     DX
959     POP     DX
960     POP     DX
961     POP     DX
962     POP     DX
963     POP     DX
964     POP     DX
965     POP     DX
966     POP     DX
967     POP     DX
968     POP     DX
969     POP     DX
970     POP     DX
971     POP     DX
972     POP     DX
973     POP     DX
974     POP     DX
975     POP     DX
976     POP     DX
977     POP     DX
978     POP     DX
979     POP     DX
980     POP     DX
981     POP     DX
982     POP     DX
983     POP     DX
984     POP     DX
985     POP     DX
986     POP     DX
987     POP     DX
988     POP     DX
989     POP     DX
990     POP     DX
991     POP     DX
992     POP     DX
993     POP     DX
994     POP     DX
995     POP     DX
996     POP     DX
997     POP     DX
998     POP     DX
999     POP     DX
1000    POP     DX

```

```

908 04ED 58          POP     AX                      ; RECOVER A NUMBER
909 04EE E8 1969 R   CALL    PRT_HEX
910 04F1 E2 FA       LOOP    PRT_DEC_LOOP
911 04F3 B9 0007      MOV     CX,7
912 04F6 BE 001A R   MOV     SI,OFFSET F3B          ; PRINT ' KB OK'
913 04F9
914 04F9 2E: 8A 04    KB_LOOP: MOV     AL,CS:[SI]
915 04FC 46           INC     SI
916 04FD E8 1969 R   CALL    PRT_HEX
917 0500 E2 F7       LOOP    KB_LOOP
918 0502 58          POP     AX                      ; RECOVER WORK REGS
919 0503 3D 0040      CMP     AX,64
920 0506 74 A9        JE      E20B                    ; FIRST PASS?
921 0508 58          POP     BX
922 0509 59          POP     CX
923 050A 5A          POP     DX
924 050B E2 B4       LOOP    E21
925 050D B0 0A       MOV     AL,10
926 050F E8 1969 R   CALL    PRT_HEX          ; LINE FEED
927
928
929
930 0512 E4 08        IN      AL,DMA+08H
931 0514 24 01        AND     AL,00000001B
932 0516 75 32        JNZ     ROM_SCAN
933 0518 1F           POP     DS
934 0519 C6 06 0015 R MOV     0MFG_ERR_FLAG,03H
935 051E E9 0385 R    JMP     D6
936
937
938
939 0521 8A E8        E21A: MOV     CH,AL
940 0523 B0 0D        MOV     AL,CR
941 0525 E8 1969 R   CALL    PRT_HEX
942 0528 B0 0A       MOV     AL,LF
943 052A E8 1969 R   CALL    PRT_HEX
944 052D 58          POP     AX
945 052E 83 C4 06    ADD     SP,6
946 0531 8C DA       MOV     DX,DS
947 0533 1F           POP     DS
948 0534 1E           PUSH    DS
949 0535 A3 0013 R   MOV     0MEMORY_SIZE,AX
950
951 0538 88 36 0015 R MOV     0MFG_ERR_FLAG,DH
952
953 053C E8 0CBA R   CALL    MOV_SEG
954 053F 8A C5       MOV     AL,CH
955 0541 E8 1958 R   CALL    XPC_BYTE
956 0544 BE 18D1 R   MOV     SI,OFFSET E1
957 0547 E8 1976 R   CALL    E_MSG
958
959
960
961
962
963
964 054A
965 054A BA C800      ROM_SCAN: MOV     DX,0C800H          ; SET BEGINNING ADDRESS
966 054D
967 054D BE DA       MOV     DS,DX
968 054F 2B DB       SUB     BX,BX
969 0551 B8 07       MOV     AX,[BX]
970 0553 53          PUSH    BX
971 0554 5B          POP     BX
972 0555 3D AA55     CMP     AX,0AA55H
973 0558 75 0A       JNZ     NEXT_ROM
974 055A E8 1920 R   CALL    ROM_CHECK
975 055D EB 05 90    JMP     ARE_WE_DONE
976 0560
977 0560 81 C2 0080  NEXT_ROM: ADD     DX,0080H          ; POINT TO NEXT 2K ADDRESS
978 0564
979 0564 81 FA F000  ARE_WE_DONE: CMP     DX,0F000H
980 0568 7C E3       JL      ROM_SCAN_1          ; AT F0000 YET?
981
982
983
984
985
986
987
988
989
990 056A
991 056A 1F           POP     DS
992 056B A0 0010 R   MOV     AL,BYTE PTR 0EQUIP_FLAG
993 056E 24 01        AND     AL,01H
994 0570 74 5E       JZ      F15
995 0572
996 0572 BA 03F1     F10: MOV     DX,3F1H
997 0575 EC          IN      AL,DX
998 0576 90          NOP
999 0577 B8 FFFF     MOV     BX,0FFFFH
1000 057A 24 F8      AND     AL,0F8H
1001 057C B0 26 008F R FE AND     0HF_CNTRL,11111110B
1002 0581 3C 50       CMP     AL,CARD_ID
1003 0583 75 05       JNE     NO_ID
1004 0585 B0 0E 008F R O1 AND     0HF_CNTRL,1
1005 058A
1006 058A E4 21       NO_ID: IN      AL,INTA01
1007 058C 24 BF      OUT     INTA01,AL
1008 058E E6 21      MOV     AH,0
1009 0590 B4 00      MOV     DL,AH
1010 0592 BA D4      INT     13H
1011 0594 CD 13      TEST    AH,0FFH
1012 0596 F6 C4 FF   JNZ     F13
1013 0599 75 19
1014
1015
1016
1017 059B BA 03F2     ;----- TURN DRIVE 0 MOTOR ON
1018 059E B0 1C       MOV     DX,03F2H
1019 05A0 EE          MOV     AL,1CH
1020 05A1 2B C9       OUT     DX,AL
1021 05A3             SUB     CX,CX
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3039
3040
3041
3042
3043
3044
3045
3046
3047
3048
3049
3050
3051
3052
3053
3054
3055
3056
3057
3058
3059
3060
3061
3062
3063
3064
3065
3066
3067
3068
3069
3070

```

```

1022 05A3 E2 FE          F12:  LOOP    F11          ; WAIT FOR 1 SECOND
1023 05A5                ; MOTOR WAIT!!
1024 05A5 E2 FE          LOOP    F12
1025 05A1 33 D2          XOR     DX,DX          ; SELECT DRIVE 0
1026 05A9 B5 22          MOV     CH,34          ; SELECT TRACK 34
1027 05AB B8 16 003E R   MOV     #SEEK_STATUS,DL
1028 05AF E8 0000 E       CALL    SEEK          ; RECALIBRATE DISKETTE AND SEEK TO 34
1029 05B2 73 05          JNC     F14          ; OK--> GO TURN OF MOTOR
1030 05B4                ; DISKETTE ERROR
1031 05B4 BE 0990 R       MOV     SI,OFFSET F3    ; GET ADDR OF MSG
1032 05B7 EB 02          JMP     SHORT F14A       ; DISPLAY MESSAGE AFTER DISKETTE SETUP
1033
1034                ;----- TURN DRIVE 0 MOTOR OFF
1035
1036 05B9                F14:    XOR     SI,SI          ; SEQUENCE END ENTRY IF NO ERROR
1037 05B9 33 F6          F14A:   MOV     AL,0CH          ; ZERO SI IF NO ERROR
1038 05BB                MOV     DX,03F2H          ; SEQUENCE END ENTRY IF ERROR
1039 05BB B0 0C          MOV     DX,03F2H          ; TURN DRIVE 0 MOTOR OFF
1040 05BD BA 03F2         OUT     DX,AL          ; FDC CTL ADDRESS
1041 05C0 EE
1042
1043                ;-----SETUP DISKETTE STATES
1044
1045 05C1 E8 0000 E       CALL    DSKETTE_SETUP
1046 05C4 72 04          JC      F14B          ; INITIALIZE DISKETTE PARMS
1047 05C6 0B F6          OR      SI,SI          ; CY-->DISKETTE SETUP ERROR
1048 05C8 74 06          JZ      F15          ; PREVIOUS DISKETTE ERROR
1049 05CA                ; NZ-->DISKETTE ERROR BEFORE SETUP
1050 05CA BE 0990 R       MOV     SI,OFFSET F3    ; GET ADDR OF MSG
1051 05CD E8 1976 R       CALL    E_MSG          ; GO PRINT ERROR MSG
1052
1053                ;----- SETUP PRINTER AND RS232 BASE ADDRESSES IF DEVICE ATTACHED
1054
1055 05D0                F15:    MOV     #INTR_FLAG,00H          ; SET STRAY INTERRUPT FLAG = 00
1056 05D0 C6 06 006B R 00 MOV     SI,OFFSET #KB_BUFFER ; SET KEYBOARD PARAMETERS
1057 05D5 BE 001E R       MOV     #BUFFER_HEAD,SI
1058 05D8 89 36 001A R   MOV     #BUFFER_TAIL,SI
1059 05DC 89 36 001C R   MOV     #BUFFER_START,SI
1060 05E0 89 36 0080 R   MOV     SI,32H          ; DEFAULT BUFFER OF 32 BYTES
1061 05E4 83 C6 20       ADD     SI,32H
1062 05E7 89 36 0082 R   MOV     #BUFFER_END,SI
1063 05EB BF 0078 R     MOV     DI,OFFSET #PRINT_TIM_OUT ; SET DEFAULT PRINTER TIMEOUT
1064 05EE 1E             PUSH    DS
1065 05EF 07             POP     ES
1066 05F0 B8 1414        MOV     AX,1414H          ; DEFAULT=20
1067 05F3 AB             STOSW
1068 05F4 AB             STOSW
1069 05F5 B8 0101        MOV     AX,0101H          ; RS232 DEFAULT=01
1070 05F8 AB             STOSW
1071 05F9 AB             STOSW
1072 05FA E4 21         IN      AL,INTA01
1073 05FC 24 FC         AND     AL,0FCH          ; ENABLE TIMER AND KB INTS
1074 05FE E6 21         OUT     INTA01,AL
1075
1076 0600 83 FD 00       CMP     BP,0000          ; CHECK FOR BP = NON ZERO
1077                ; (ERROR HAPPENED)
1078 0603 74 18          JE      F15A_0          ; CONTINUE IF NO ERROR
1079 0605 BA 0002        MOV     DX,2          ; 2 SHORT BEEPS (ERROR)
1080 0608 E8 19A5 R     CALL    ERR_BEEP
1081 060B BE 0769 R     MOV     SI,OFFSET F3D    ; LOAD ERROR MSG
1082 060E EB 1997 R     CALL    P_MSG
1083 0611                ERR_WAIT: MOV     AH,00
1084 0611 B4 00          INT     16H          ; WAIT FOR 'F1' KEY
1085 0613 CD 16          CMP     AH,3BH
1086 0615 8D 3B         JNE     ERR_WAIT
1087 0618 75 F7          JMP     F15A          ; BYPASS ERROR
1088 061A EB 0E 90       JMP     F15A_0
1089 061D
1090 061D B0 3E 0012 R 01 CMP     #MSG_TST,1
1091 0622 74 06          JE      F15A          ; MSG MODE
1092 0624 BA 0001        MOV     F15A          ; BYPASS BEEP
1093 0627 E8 19A5 R     CALL    DX,1          ; 1 SHORT BEEP (NO ERRORS)
1094 062A A0 0010 R     MOV     AL,BYTE PTR #EQUIP_FLAG ; GET SWITCHES
1095 062D 24 01         AND     AL,00000001B ; 'LOOP POST' SWITCH ON
1096 062F 75 03         JNZ     F15B          ; CONTINUE WITH BRING-UP
1097 0631 E9 005B R     JMP     START
1098 0634 2A E4         SUB     AH,AH
1099 0636 A0 0049 R     MOV     AL,#CRT_MODE
1100 0639 CD 10         INT     10H          ; CLEAR SCREEN
1101 063B
1102 063B BD 1970 R     F15C:   MOV     BP,OFFSET F4
1103 063E BE 0000        MOV     SI,0
1104 0641                F16:    MOV     DX,CS:[BP]          ; PRT_BASE
1105 0641 2E1 8B 56 00    MOV     AL,0AAH          ; GET PRINTER BASE ADDR
1106 0645 B0 AA          OUT     DX,AL          ; WRITE DATA TO PORT A
1107 0647 EE          OUT     DX,AL
1108 0648 EB          PUSH    DS
1109 0649 EC          IN      AL,DX          ; BUS SETTLEING
1110 064A 1F          POP     DS          ; READ PORT A
1111 064B 3C AA         CMP     AL,0AAH          ; DATA PATTERN SAME
1112 064D 75 05         JNE     F17          ; NO - CHECK NEXT PRT CD
1113 064F 89 54 08       MOV     [PTRINTR_BASE-DATA40][SI],DX ; YES - STORE PRT BASE ADDR
1114 0652 46             INC     SI          ; INCREMENT TO NEXT WORD
1115 0653 46             INC     SI
1116 0654                F17:    INC     BP          ; POINT TO NEXT BASE ADDR
1117 0654 45             INC     BP
1118 0655 45             INC     BP
1119 0656 81 FD 1976 R   CMP     BP,OFFSET F4E    ; ALL POSSIBLE ADDRS CHECKED?
1120 065A 75 E3         JNE     F16          ; PRT BASE
1121 065C BB 0000        MOV     BX,0          ; POINTER TO RS232 TABLE
1122 065F BA 03FA        MOV     DX,3FAH          ; CHECK IF RS232 CD 1 ATTCH?
1123 0662 EC          IN      AL,DX          ; READ INTR ID REG
1124 0663 A8 F8         TEST    AL,0F8H
1125 0665 75 06         JNZ     F18          ; SETUP RS232 CD #1 ADDR
1126 0667 C7 07 03F8    MOV     [#RS232_BASE-DATA40][BX],3F8H
1127 066B 43             INC     BX
1128 066D 43             INC     BX
1129 066D                F18:    MOV     DX,2FAH          ; CHECK IF RS232 CD 2 ATTCH
1130 066D BA 02FA        IN      AL,0F8H          ; READ INTERRUPT ID REG
1131 0670 EC          TEST    AL,0F8H
1132 0671 A8 F8         JNZ     F19          ; BASE_END
1133 0673 75 06         JNZ     F19
1134 0675 C7 07 02F8    MOV     [#RS232_BASE-DATA40][BX],2F8H ; SETUP RS232 CD #2
1135 0679 43             INC     BX

```

```

1136 067A 43          INC     BX
1137
1138          ;----- SET UP EQUIP FLAG TO INDICATE NUMBER OF PRINTERS AND RS232 CARDS
1139
1140 067B          F19:      MOV     AX,51          ; BASE END:
1141 067B 8B C6      MOV     CL,3          ; SI HAS 2* NUMBER OF RS232
1142 067D B1 03      ROR     AL,CL          ; SHIFT COUNT
1143 067F D2 C8      OR      AL,CL          ; ROTATE RIGHT 3 POSITIONS
1144 0681 0A C3      MOV     BYTE PTR #EQUIP_FLAG+1,AL ; OR IN THE PRINTER COUNT
1145 0683 A2 0011 R  ; STORE AS SECOND BYTE
1146 0686 A0 0201 R  ;
1147 0689 EC          MOV     DX,201H
1148 068A 90          NOP
1149 068B 90          NOP
1150 068C 90          NOP
1151 068D A8 0F      TEST    AL,0FH
1152 068F 75 05      JNZ     F20          ; NO_GAME_CARD
1153 0691 80 0E 0011 R 10 ;
1154 0696          F20:      MOV     BYTE PTR #EQUIP_FLAG+1,16 ; NO_GAME_CARD:
1155
1156          ;----- ENABLE NMI INTERRUPTS
1157
1158 0696 E4 61      IN      AL,PORT_B          ; RESET CHECK ENABLES
1159 0698 0C 30      OR      AL,30H
1160 069A E6 61      OUT     PORT_B,AL
1161 069C 24 CF      AND     AL,0CFH
1162 069E E6 61      OUT     PORT_B,AL
1163 06A0 B0 80      MOV     AL,80H          ; ENABLE NMI INTERRUPTS
1164 06A2 E6 A0      OUT     0A0H,AL
1165 06A4          F21:      INT     19H          ; LOAD BOOT STRAP:
1166 06A4 CD 19      ; GO TO THE BOOT LOADER
1167
1168          ;----- INT 19 -----
1169          ; BOOT STRAP LOADER
1170          ; TRACK 0, SECTOR 1 IS READ INTO THE
1171          ; BOOT LOCATION (SEGMENT 0, OFFSET 7C00)
1172          ; AND CONTROL IS TRANSFERRED THERE.
1173          ;
1174          ; IF THERE IS A HARDWARE ERROR CONTROL IS
1175          ; TRANSFERRED TO THE ROM BASIC ENTRY POINT.
1176          ;-----
1177          ASSUME CS:CODE,DS:ABS0
1178          ORG     0E6F2H
1179 06F2          ORG     006F2H
1180
1181 06F2          BOOT_STRAP PROC NEAR
1182 06F2 FB          STI
1183 06F3 2B C0      SUB     SI,AX          ; ENABLE INTERRUPTS
1184 06F5 8E D8      MOV     DS,AX          ; ESTABLISH ADDRESSING
1185
1186          ;----- RESET THE DISK PARAMETER TABLE VECTOR
1187
1188 06F7 C7 06 0078 R 0FC7 R  ;
1189 06FD 8C 0E 007A R  ; WORD PTR #DISK_POINTER,OFFSET DISK_BASE
1190          MOV     WORD PTR #DISK_POINTER+2,CS
1191
1192          ;----- LOAD SYSTEM FROM DISKETTE -- CX HAS RETRY COUNT
1193
1194 0701 B9 0004      MOV     CX,4          ; SET RETRY COUNT
1195 0704          H1:      PUSH    CX          ; IPL SYSTEM
1196 0705 B4 00      MOV     AH,0          ; SAVE RETRY COUNT
1197 0707 CD 13      INT     13H          ; RESET THE DISKETTE SYSTEM
1198 0709 72 0F      JC      H2          ; DISKETTE_IO
1199 070B B8 0201 R  ; IF ERROR, TRY AGAIN
1200 070E 2B D2      SUB     DX,DX          ; READ IN THE SINGLE SECTOR
1201 0710 8E C2      MOV     ES,DX          ; TO THE BOOT LOCATION
1202 0712 BB 7C00 R  ;
1203          MOV     BX,OFFSET #BOOT_LOCN
1204 0715 B9 0001      MOV     CX,1          ; DRIVE 0, HEAD 0
1205 0718 CD 13      INT     13H          ; SECTOR 1, TRACK 0
1206 071A          H2:      JNC     H4          ; DISKETTE_IO
1207 071A 59          POP     CX          ; RECOVER RETRY COUNT
1208 071B 73 04      JNC     H4          ; CF SET BY UNSUCCESSFUL READ
1209 071D E2 E5      LOOP    H1          ; DO IT FOR RETRY TIMES
1210
1211          ;----- UNABLE TO IPL FROM THE DISKETTE
1212
1213 071F          H3:      INT     18H          ; GO TO RESIDENT BASIC
1214 071F CD 18
1215
1216          ;----- IPL WAS SUCCESSFUL
1217
1218 0721          H4:      JMP     #BOOT_LOCN
1219 0721 EA 7C00 ---- R  ;
1220 0726          BOOT_STRAP ENDP
1221
1222          ;-----
1223 0729          ; ORG     0E729H
1224 0729 0417      ORG     00729H
1225 072B 0300      A1      DW     1047          ; 110 BAUD
1226 072D 0180      DW     384          ; 150
1227 072F 00C0      DW     192          ; 1300
1228 0731 0060      DW     96          ; 600
1229 0733 0030      DW     48          ; 1200
1230 0735 0018      DW     24          ; 2400
1231 0737 000C      DW     12          ; 4800
1232          ; 9600
1233 0739          RS232_IO:
1234 0739 E9 0000 E  ; JMP     RS232_IO_1
1235          RS232_IO_1
1236 073C          CONF_TBL:
1237 073C 0008      DW     CONF_E-CONF_TBL-2
1238 073E FB          DB     MODEL_BYTE
1239 073F 00          DB     SUB_MODEL_BYTE
1240 0740 01          DB     BIOS_LEVEL
1241 0741 50          DB     01010000B
1242          ; 10000000 = DMA CHANNEL 3 USE BY BIOS
1243          ; 01000000 = CASCADED INTERRUPT LEVEL 2
1244          ; 00100000 = REAL TIME CLOCK AVAILABLE
1245          ; 00010000 = KEYBOARD SCAN CODE HOOK 1AH
1246          ; RESERVED
1247          ; RESERVED
1248          ; RESERVED
1249 = 0746          CONF_E EQU $          ; RESERVED FOR EXPANSION

```

SECTION 5

```

1364 09CE      RTC_00  PROC    NEAR
1365 09CE      MOV      AL,0TIMER_OFL
1366 09D1      MOV      0TIMER_OFL,0
1367 09DA      MOV      CX,0TIMER_HIGH
1368 09DB      MOV      DX,0TIMER_LOW
1369 09DE      RET
1370
1371 09DF      RTC_10:  MOV      0TIMER_LOW,DX
1372 09DF      MOV      0TIMER_HIGH,CX
1373 09E3      MOV      0TIMER_OFL,0
1374 09E7      RET
1375 09EC      RTC_NS:  INVALID FUNCTION (NOT SUPPORTED)
1376
1377 09ED      STC
1378 09ED      RET
1379 09EE      RTC_AI:  MOV      CX,0DAY_COUNT
1380
1381 09EF      MOV      RET
1382 09EF      RTC_B0:  MOV      0DAY_COUNT,CX
1383 09F3      MOV      RET
1384
1385 09F4      RTC_00  ENDP
1386 09F4      MOV      0EC59H
1387 09F8      MOV      00C59H
1388 09F9      JMP      DISKETTE_IO_1
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404 0C5C      BEEP  PROC    NEAR
1405 0C5C      PUSHF
1406 0C5D      CLI
1407 0C5E      MOV      AL,10110110B
1408 0C60      OUT      TIMER-3,AL
1409 0C62      NOP
1410 0C63      MOV      AL,CL
1411 0C65      OUT      TIMER-2,AL
1412 0C67      NOP
1413 0C68      MOV      AL,CH
1414 0C6A      OUT      TIMER-2,AL
1415 0C6C      IN      AL,PORT_B
1416 0C6E      MOV      AH,AL
1417 0C70      OR      AL,GATE2+SPK2
1418 0C72      OUT      PORT_B,AL
1419 0C74      POPF
1420 0C75      GT:      MOV      CX,1035
1421 0C75      CALL     WAITF
1422 0C78      DEC      BL
1423 0C7B      JNZ      GT
1424 0C7D      JNZ      GT
1425
1426 0C7F      PUSHF
1427 0C80      CLI
1428 0C81      IN      AL,PORT_B
1429 0C83      OR      AL,NOT (GATE2+SPK2)
1430 0C85      AND      AH,AL
1431 0C87      MOV      AL,AH
1432 0C89      AND      AL,NOT (GATE2+SPK2)
1433 0C8B      OUT      PORT_B,AL
1434 0C8D      POPF
1435 0C8E      MOV      CX,1035
1436 0C91      CALL     WAITF
1437 0C94      PUSHF
1438 0C95      CLI
1439 0C96      IN      AL,PORT_B
1440 0C98      AND      AL,GATE2+SPK2
1441 0C9A      OR      AL,AH
1442 0C9C      OUT      PORT_B,AL
1443 0C9E      POPF
1444 0C9F      RET
1445
1446 0CA0      BEEP  ENDP
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460 0CA0      WAITF  PROC    NEAR
1461 0CA0      MOV      AX,0
1462 0CA1      SHR      CX,1
1463 0CA3      JCXZ     WAITF9
1464
1465 0CA5      OUT      DMA+12,AL
1466 0CA7      WAITF1:  PUSHF
1467 0CA7      CLI
1468 0CA8      WAITF3:  MOV      AL,DMA
1469 0CA9      AND      AL,11111110B
1470 0CA9      CMP      AH,AL
1471 0CAB      MOV      IN      AL,DMA
1472 0CAD      JE      WAITF3
1473 0CAF      POPF
1474 0CB1      MOV      AL,1
1475 0CB3      MOV      AL,1
1476
1477 0CB5      MOV      AL,1
1478
1479 0CB5      MOV      AL,1
1480
1481 0CB5      MOV      AL,1
1482
1483 0CB5      MOV      AL,1
1484
1485 0CB5      MOV      AL,1
1486
1487 0CB5      MOV      AL,1
1488
1489 0CB5      MOV      AL,1
1490
1491 0CB5      MOV      AL,1
1492
1493 0CB5      MOV      AL,1
1494
1495 0CB5      MOV      AL,1
1496
1497 0CB5      MOV      AL,1
1498
1499 0CB5      MOV      AL,1
1500
1501 0CB5      MOV      AL,1
1502
1503 0CB5      MOV      AL,1
1504
1505 0CB5      MOV      AL,1
1506
1507 0CB5      MOV      AL,1
1508
1509 0CB5      MOV      AL,1
1510
1511 0CB5      MOV      AL,1
1512
1513 0CB5      MOV      AL,1
1514
1515 0CB5      MOV      AL,1
1516
1517 0CB5      MOV      AL,1
1518
1519 0CB5      MOV      AL,1
1520
1521 0CB5      MOV      AL,1
1522
1523 0CB5      MOV      AL,1
1524
1525 0CB5      MOV      AL,1
1526
1527 0CB5      MOV      AL,1
1528
1529 0CB5      MOV      AL,1
1530
1531 0CB5      MOV      AL,1
1532
1533 0CB5      MOV      AL,1
1534
1535 0CB5      MOV      AL,1
1536
1537 0CB5      MOV      AL,1
1538
1539 0CB5      MOV      AL,1
1540
1541 0CB5      MOV      AL,1
1542
1543 0CB5      MOV      AL,1
1544
1545 0CB5      MOV      AL,1
1546
1547 0CB5      MOV      AL,1
1548
1549 0CB5      MOV      AL,1
1550
1551 0CB5      MOV      AL,1
1552
1553 0CB5      MOV      AL,1
1554
1555 0CB5      MOV      AL,1
1556
1557 0CB5      MOV      AL,1
1558
1559 0CB5      MOV      AL,1
1560
1561 0CB5      MOV      AL,1
1562
1563 0CB5      MOV      AL,1
1564
1565 0CB5      MOV      AL,1
1566
1567 0CB5      MOV      AL,1
1568
1569 0CB5      MOV      AL,1
1570
1571 0CB5      MOV      AL,1
1572
1573 0CB5      MOV      AL,1
1574
1575 0CB5      MOV      AL,1
1576
1577 0CB5      MOV      AL,1
1578
1579 0CB5      MOV      AL,1
1580
1581 0CB5      MOV      AL,1
1582
1583 0CB5      MOV      AL,1
1584
1585 0CB5      MOV      AL,1
1586
1587 0CB5      MOV      AL,1
1588
1589 0CB5      MOV      AL,1
1590
1591 0CB5      MOV      AL,1
1592
1593 0CB5      MOV      AL,1
1594
1595 0CB5      MOV      AL,1
1596
1597 0CB5      MOV      AL,1
1598
1599 0CB5      MOV      AL,1
1600
1601 0CB5      MOV      AL,1
1602
1603 0CB5      MOV      AL,1
1604
1605 0CB5      MOV      AL,1
1606
1607 0CB5      MOV      AL,1
1608
1609 0CB5      MOV      AL,1
1610
1611 0CB5      MOV      AL,1
1612
1613 0CB5      MOV      AL,1
1614
1615 0CB5      MOV      AL,1
1616
1617 0CB5      MOV      AL,1
1618
1619 0CB5      MOV      AL,1
1620
1621 0CB5      MOV      AL,1
1622
1623 0CB5      MOV      AL,1
1624
1625 0CB5      MOV      AL,1
1626
1627 0CB5      MOV      AL,1
1628
1629 0CB5      MOV      AL,1
1630
1631 0CB5      MOV      AL,1
1632
1633 0CB5      MOV      AL,1
1634
1635 0CB5      MOV      AL,1
1636
1637 0CB5      MOV      AL,1
1638
1639 0CB5      MOV      AL,1
1640
1641 0CB5      MOV      AL,1
1642
1643 0CB5      MOV      AL,1
1644
1645 0CB5      MOV      AL,1
1646
1647 0CB5      MOV      AL,1
1648
1649 0CB5      MOV      AL,1
1650
1651 0CB5      MOV      AL,1
1652
1653 0CB5      MOV      AL,1
1654
1655 0CB5      MOV      AL,1
1656
1657 0CB5      MOV      AL,1
1658
1659 0CB5      MOV      AL,1
1660
1661 0CB5      MOV      AL,1
1662
1663 0CB5      MOV      AL,1
1664
1665 0CB5      MOV      AL,1
1666
1667 0CB5      MOV      AL,1
1668
1669 0CB5      MOV      AL,1
1670
1671 0CB5      MOV      AL,1
1672
1673 0CB5      MOV      AL,1
1674
1675 0CB5      MOV      AL,1
1676
1677 0CB5      MOV      AL,1
1678
1679 0CB5      MOV      AL,1
1680
1681 0CB5      MOV      AL,1
1682
1683 0CB5      MOV      AL,1
1684
1685 0CB5      MOV      AL,1
1686
1687 0CB5      MOV      AL,1
1688
1689 0CB5      MOV      AL,1
1690
1691 0CB5      MOV      AL,1
1692
1693 0CB5      MOV      AL,1
1694
1695 0CB5      MOV      AL,1
1696
1697 0CB5      MOV      AL,1
1698
1699 0CB5      MOV      AL,1
1700
1701 0CB5      MOV      AL,1
1702
1703 0CB5      MOV      AL,1
1704
1705 0CB5      MOV      AL,1
1706
1707 0CB5      MOV      AL,1
1708
1709 0CB5      MOV      AL,1
1710
1711 0CB5      MOV      AL,1
1712
1713 0CB5      MOV      AL,1
1714
1715 0CB5      MOV      AL,1
1716
1717 0CB5      MOV      AL,1
1718
1719 0CB5      MOV      AL,1
1720
1721 0CB5      MOV      AL,1
1722
1723 0CB5      MOV      AL,1
1724
1725 0CB5      MOV      AL,1
1726
1727 0CB5      MOV      AL,1
1728
1729 0CB5      MOV      AL,1
1730
1731 0CB5      MOV      AL,1
1732
1733 0CB5      MOV      AL,1
1734
1735 0CB5      MOV      AL,1
1736
1737 0CB5      MOV      AL,1
1738
1739 0CB5      MOV      AL,1
1740
1741 0CB5      MOV      AL,1
1742
1743 0CB5      MOV      AL,1
1744
1745 0CB5      MOV      AL,1
1746
1747 0CB5      MOV      AL,1
1748
1749 0CB5      MOV      AL,1
1750
1751 0CB5      MOV      AL,1
1752
1753 0CB5      MOV      AL,1
1754
1755 0CB5      MOV      AL,1
1756
1757 0CB5      MOV      AL,1
1758
1759 0CB5      MOV      AL,1
1760
1761 0CB5      MOV      AL,1
1762
1763 0CB5      MOV      AL,1
1764
1765 0CB5      MOV      AL,1
1766
1767 0CB5      MOV      AL,1
1768
1769 0CB5      MOV      AL,1
1770
1771 0CB5      MOV      AL,1
1772
1773 0CB5      MOV      AL,1
1774
1775 0CB5      MOV      AL,1
1776
1777 0CB5      MOV      AL,1
1778
1779 0CB5      MOV      AL,1
1780
1781 0CB5      MOV      AL,1
1782
1783 0CB5      MOV      AL,1
1784
1785 0CB5      MOV      AL,1
1786
1787 0CB5      MOV      AL,1
1788
1789 0CB5      MOV      AL,1
1790
1791 0CB5      MOV      AL,1
1792
1793 0CB5      MOV      AL,1
1794
1795 0CB5      MOV      AL,1
1796
1797 0CB5      MOV      AL,1
1798
1799 0CB5      MOV      AL,1
1800
1801 0CB5      MOV      AL,1
1802
1803 0CB5      MOV      AL,1
1804
1805 0CB5      MOV      AL,1
1806
1807 0CB5      MOV      AL,1
1808
1809 0CB5      MOV      AL,1
1810
1811 0CB5      MOV      AL,1
1812
1813 0CB5      MOV      AL,1
1814
1815 0CB5      MOV      AL,1
1816
1817 0CB5      MOV      AL,1
1818
1819 0CB5      MOV      AL,1
1820
1821 0CB5      MOV      AL,1
1822
1823 0CB5      MOV      AL,1
1824
1825 0CB5      MOV      AL,1
1826
1827 0CB5      MOV      AL,1
1828
1829 0CB5      MOV      AL,1
1830
1831 0CB5      MOV      AL,1
1832
1833 0CB5      MOV      AL,1
1834
1835 0CB5      MOV      AL,1
1836
1837 0CB5      MOV      AL,1
1838
1839 0CB5      MOV      AL,1
1840
1841 0CB5      MOV      AL,1
1842
1843 0CB5      MOV      AL,1
1844
1845 0CB5      MOV      AL,1
1846
1847 0CB5      MOV      AL,1
1848
1849 0CB5      MOV      AL,1
1850
1851 0CB5      MOV      AL,1
1852
1853 0CB5      MOV      AL,1
1854
1855 0CB5      MOV      AL,1
1856
1857 0CB5      MOV      AL,1
1858
1859 0CB5      MOV      AL,1
1860
1861 0CB5      MOV      AL,1
1862
1863 0CB5      MOV      AL,1
1864
1865 0CB5      MOV      AL,1
1866
1867 0CB5      MOV      AL,1
1868
1869 0CB5      MOV      AL,1
1870
1871 0CB5      MOV      AL,1
1872
1873 0CB5      MOV      AL,1
1874
1875 0CB5      MOV      AL,1
1876
1877 0CB5      MOV      AL,1
1878
1879 0CB5      MOV      AL,1
1880
1881 0CB5      MOV      AL,1
1882
1883 0CB5      MOV      AL,1
1884
1885 0CB5      MOV      AL,1
1886
1887 0CB5      MOV      AL,1
1888
1889 0CB5      MOV      AL,1
1890
1891 0CB5      MOV      AL,1
1892
1893 0CB5      MOV      AL,1
1894
1895 0CB5      MOV      AL,1
1896
1897 0CB5      MOV      AL,1
1898
1899 0CB5      MOV      AL,1
1900
1901 0CB5      MOV      AL,1
1902
1903 0CB5      MOV      AL,1
1904
1905 0CB5      MOV      AL,1
1906
1907 0CB5      MOV      AL,1
1908
1909 0CB5      MOV      AL,1
1910
1911 0CB5      MOV      AL,1
1912
1913 0CB5      MOV      AL,1
1914
1915 0CB5      MOV      AL,1
1916
1917 0CB5      MOV      AL,1
1918
1919 0CB5      MOV      AL,1
1920
1921 0CB5      MOV      AL,1
1922
1923 0CB5      MOV      AL,1
1924
1925 0CB5      MOV      AL,1
1926
1927 0CB5      MOV      AL,1
1928
1929 0CB5      MOV      AL,1
1930
1931 0CB5      MOV      AL,1
1932
1933 0CB5      MOV      AL,1
1934
1935 0CB5      MOV      AL,1
1936
1937 0CB5      MOV      AL,1
1938
1939 0CB5      MOV      AL,1
1940
1941 0CB5      MOV      AL,1
1942
1943 0CB5      MOV      AL,1
1944
1945 0CB5      MOV      AL,1
1946
1947 0CB5      MOV      AL,1
1948
1949 0CB5      MOV      AL,1
1950
1951 0CB5      MOV      AL,1
1952
1953 0CB5      MOV      AL,1
1954
1955 0CB5      MOV      AL,1
1956
1957 0CB5      MOV      AL,1
1958
1959 0CB5      MOV      AL,1
1960
1961 0CB5      MOV      AL,1
1962
1963 0CB5      MOV      AL,1
1964
1965 0CB5      MOV      AL,1
1966
1967 0CB5      MOV      AL,1
1968
1969 0CB5      MOV      AL,1
1970
1971 0CB5      MOV      AL,1
1972
1973 0CB5      MOV      AL,1
1974
1975 0CB5      MOV      AL,1
1976
1977 0CB5      MOV      AL,1
1978
1979 0CB5      MOV      AL,1
1980
1981 0CB5      MOV      AL,1
1982
1983 0CB5      MOV      AL,1
1984
1985 0CB5      MOV      AL,1
1986
1987 0CB5      MOV      AL,1
1988
1989 0CB5      MOV      AL,1
1990
1991 0CB5      MOV      AL,1
1992
1993 0CB5      MOV      AL,1
1994
1995 0CB5      MOV      AL,1
1996
1997 0CB5      MOV      AL,1
1998
1999 0CB5      MOV      AL,1
2000

```



```

1478 0CB6 E2 EF      LOOP      WAITF1      ; DECREMENT CYCLES COUNT TILL COUNT END
1479 0CB8             WAITF9:      POP      AX      ; RESTORE (AH)
1480 0CB8 58             RET      ; RETURN (CX)= 0
1481 0CB9 C3
1482
1483 0CBA             WAITF      ENDP
1484
1485
1486 ;-----
1487 ; PRINT A SEGMENT VALUE TO LOOK LIKE A 20 BIT ADDRESS
1488 ; DX MUST CONTAIN SEGMENT VALUE TO BE PRINTED
1489 ;-----
1489 0CBA PROC      NEAR
1490 0CBA 8A C6      MOV      AL,DH      ;GET MSB
1491 0CB8 E8 1958 R  CALL      XPC_BYTE
1492 0CBF 8A C2      MOV      AL,D1      ;LSB
1493 0CC1 E8 1958 R  CALL      XPC_BYTE
1494 0CCA B0 30      MOV      AL,'0'      ; PRINT A '0 '
1495 0CC4 E8 1969 R  CALL      PRT_HEX
1496 0CC9 B0 20      MOV      AL,' '      ;SPACE
1497 0CCB E8 1969 R  CALL      PRT_HEX
1498 0CCE C3      RET
1499 0CCF      PRT_SEG      ENDP
1500
1501 ;-----
1502 ; THIS SUBROUTINE PERFORMS A READ/WRITE STORAGE TEST ON A BLOCK
1503 ; OF STORAGE
1504 ; ENTRY REQUIREMENTS:
1505 ; ES = ADDRESS OF STORAGE SEGMENT BEING TESTED
1506 ; DS = ADDRESS OF STORAGE SEGMENT BEING TESTED
1507 ; CX = WORD COUNT OF STORAGE BLOCK TO BE TESTED
1508 ; EXIT PARAMETERS:
1509 ; ZERO FLAG = 0 IF STORAGE ERROR (DATA COMPARE OR PARITY
1510 ; CHECK, AL=0 DENOTES A PARITY CHECK, ELSE AL=XOR'ED
1511 ; BIT PATTERN OF THE EXPECTED DATA PATTERN VS THE ACTUAL
1512 ; DATA READ.
1513 ; AX,BX,CX,DX,D1, AND SI ARE ALL DESTROYED.
1514 ;-----
1515
1516 0CCF      STGTST_CNT      PROC      NEAR
1517 0CCF BB D9      MOV      BX,CX      ; SAVE WORD COUNT OF BLOCK TO TEST
1518 0CD1 FC      CLD      ; SET DIR FLAG TO INCREMENT
1519 0CD2 2B FF      SUB      DI,D1      ; SET DI=OFFSET 0 REL TO ES REG
1520 0CD4 2B C0      SUB      AX,AX      ; SETUP FOR 0->FF PATTERN TEST
1521 0CD6
1522 0CD6 B8 05      MOV      [DI],AL      ; ON FIRST BYTE
1523 0CD8 B8 05      MOV      AL,[DI]
1524 0CDA 32 C4      XOR      AL,AH      ; O.K.?
1525 0CDC 75 79      JNZ      C7      ; GO ERROR IF NOT
1526 0CDE FE C4      INC      AH
1527 0CEE 0A C4      MOV      AL,AH
1528 0CE2 75 F2      JNZ      C2-1
1529 0CE4 B8 55AAH  MOV      AX,055AAH      ; LOOP TILL WRAP THROUGH FF
1530 0CE7 B8 D0      MOV      DX,AX      ; SET INITIAL COMPARE PATTERN.
1531 0CE9 F3 AB      REP      STOSW      ; FILL STORAGE LOCATIONS IN BLOCK
1532 0CEB E4 61      IN      AL,PORT_B
1533 0CED 0C 30      OR      AL,030H
1534 0CEF E6 61      OUT      PORT_B,AL      ; TOGGLE PARITY CHECK LATCHES
1535 0CF1 90      NOP
1536 0CF2 24 CF      AND      AL,0CFH
1537 0CF4 E6 61      OUT      PORT_B,AL
1538
1539 0CF6 4F      ; DEC      DI      ; POINT TO LAST WORD JUST WRITTEN
1540 0CF7 4F      DEC      DI
1541 0CF8 FD      STD      ; SET DIR FLAG TO GO BACKWARDS
1542 0CF9 B8 F7      MOV      SI,DI      ; INITIALIZE DESTINATION POINTER
1543 0CFB B8 CB      MOV      CX,BX      ; SETUP WORD COUNT FOR LOOP
1544 0CFD
1545 0CFD AD      C3:      LODSW      ; INNER TEST LOOP
1546 0CFE C3 C2      XOR      AX,DX      ; READ OLD TEST WORD FROM STORAGE
1547 0D00 75 57      JNE      C7X      ; DATA READ AS EXPECTED ?
1548 0D02 B8 AA55H  MOV      AX,0AA55H  ; NO - GO TO ERROR ROUTINE
1549 0D05 AB      STOSW      ; GET NEXT DATA PATTERN TO WRITE
1550 0D06 E2 F5      LOOP     C3      ; WRITE INTO LOCATION JUST READ
1551 ; DECREMENT WORD COUNT AND LOOP
1552 0D08 FC      ;
1553 0D09 47      ; SET DIR FLAG TO GO FORWARD
1554 0D0A 47      ; SET POINTER TO BEG LOCATION
1555 0D0B B8 F7      INC      DI
1556 0D0D B8 CB      MOV      SI,DI      ; INITIALIZE DESTINATION POINTER
1557 0D0F B8 CB      MOV      CX,BX      ; SETUP WORD COUNT FOR LOOP
1558 0D11
1559 0D11 AD      C4:      LODSW      ; INNER TEST LOOP
1560 0D12 C3 C2      XOR      AX,DX      ; READ OLD TEST WORD FROM STORAGE
1561 0D14 75 43      JNE      C7X      ; DATA READ AS EXPECTED ?
1562 0D16 B8 FFFFH  MOV      AX,0FFFFH  ; NO - GO TO ERROR ROUTINE
1563 0D19 AB      STOSW      ; GET NEXT DATA PATTERN TO WRITE
1564 0D1A E2 F5      LOOP     C4      ; WRITE INTO LOCATION JUST READ
1565 ; DECREMENT WORD COUNT AND LOOP
1566 0D1C 4F      ; DEC      DI      ; POINT TO LAST WORD JUST WRITTEN
1567 0D1D 4F      DEC      DI
1568 0D1E FD      STD      ; SET DIR FLAG TO GO BACKWARDS
1569 0D1F B8 F7      MOV      SI,DI      ; INITIALIZE DESTINATION POINTER
1570 0D21 B8 CB      MOV      CX,BX      ; SETUP WORD COUNT FOR LOOP
1571 0D23 B8 D0      MOV      DX,AX      ; SETUP COMPARE PATTERN "0FFFFH"
1572 0D25
1573 0D25 AD      C5:      LODSW      ; INNER TEST LOOP
1574 0D26 C3 C2      XOR      AX,DX      ; READ OLD TEST WORD FROM STORAGE
1575 0D28 75 2F      JNE      C7X      ; DATA READ AS EXPECTED ?
1576 0D2A B8 0101H  MOV      AX,00101H  ; NO - GO TO ERROR ROUTINE
1577 0D2D AB      STOSW      ; GET NEXT DATA PATTERN TO WRITE
1578 0D2E E2 F5      LOOP     C5      ; WRITE INTO LOCATION JUST READ
1579 ; DECREMENT WORD COUNT AND LOOP
1580 0D30 FC      ;
1581 0D31 47      ; SET DIR FLAG TO GO FORWARD
1582 0D32 47      ; SET POINTER TO BEG LOCATION
1583 0D33 B8 F7      INC      DI
1584 0D35 B8 CB      MOV      SI,DI      ; INITIALIZE DESTINATION POINTER
1585 0D37 B8 D0      MOV      CX,BX      ; SETUP WORD COUNT FOR LOOP
1586 0D39
1587 0D39 AD      C6:      LODSW      ; INNER TEST LOOP
1588 0D3A C3 C2      XOR      AX,DX      ; READ OLD TEST WORD FROM STORAGE
1589 0D3C 75 1B      JNE      C7X      ; DATA READ AS EXPECTED ?
1590 0D3E AB      STOSW      ; NO - GO TO ERROR ROUTINE
1591 0D3F E2 F8      LOOP     C6      ; WRITE ZERO INTO LOCATION READ
; DECREMENT WORD COUNT AND LOOP

```

```

1592          ;
1593 0041 4F          DEC     DI          ; POINT TO LAST WORD JUST WRITTEN
1594 0042 4F          DEC     DI
1595 0043 FD          STD
1596 0044 BB F7      MOV     SI,DI      ; SET DIR FLAG TO GO BACKWARDS
1597 0046 BB CB      MOV     CX,BX      ; INITIALIZE DESTINATION POINTER
1598 0048 BB D0      MOV     CX,AX      ; SETUP WORD COUNT FOR LOOP
1599 004A             MOV     CX,AX      ; SETUP COMPARE PATTERN "00000H"
1600 004A AD          C6X:  LODSW
1601 004B 33 C2      XOR     AX,DX      ; VERIFY MEMORY IS ZERO.
1602 004D 75 0A      JNE     C7X        ; DATA READ AS EXPECTED ?
1603 004F E2 F9      LOOP    C6X       ; NO - GO TO ERROR ROUTINE
1604             ;                   ; DECREMENT WORD COUNT AND LOOP
1605 0051 E4 62      IN      AL,PORT_C   ; DID A PARITY ERROR OCCUR ?
1606 0053 24 C0      AND     AL,0C0H    ; ZERO FLAG WILL BE OFF, IF PARITY ERROR
1607 0055 B0 00      MOV     AL,0       ; AL=0 DATA COMPARE OK
1608 0057             ;
1609 0057 FC          C7:  CLD          ; SET DIRECTION FLAG TO INC
1610 0058 C3         RET
1611 0059             ;
1612 0059 3C 00      C7X:  CMP     AL,0       ; FIND BYTE THAT FAILED.
1613 005B 75 FA      JNZ     C7         ;
1614 005D 8A 04      MOV     AL,AH      ;
1615 005F EB F6      JMP     SHORT C7    ;
1616 0061             STGTST_CNT      ENDP
1617             ;
1618             ; ORG     0EF57H
1619 0F57             ; ORG     00F57H
1620 0F57 E9 0000 E  DISK_INT:      JMP     DISK_INT_1
1621             ;
1622             ; ORG     0EF79H
1623 0F79             ; ORG     00F79H
1624             ;
1625 ----- MEDIA/DRIVE PARAMETER TABLES -----
1626 -----
1627             ;
1628             ; 40 TRACK LOW DATA RATE MEDIA IN 40 TRACK LOW DATA RATE DRIVE :
1629 -----
1630 0F79      MD_TBL1 LABEL BYTE
1631 0F79 DF      DB 1011111B ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
1632 0F7A 02      DB 2        ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
1633 0F7B 25      DB MOTOR_WAIT ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
1634 0F7C 02      DB 2        ; 512 BYTES/SECTOR
1635 0F7D 09      DB 09       ; EOT ( LAST SECTOR ON TRACK)
1636 0F7E 2A      DB 02AH     ; GAP LENGTH
1637 0F7F FF      DB 0FFH     ; DTL
1638 0F80 50      DB 050H     ; GAP LENGTH FOR FORMAT
1639 0F81 F6      DB 0F6H     ; FILL BYTE FOR FORMAT
1640 0F82 0F      DB 15       ; HEAD SETTLE TIME (MILLI/SECONDS)
1641 0F83 08      DB 8        ; MOTOR START TIME (1/8 SECONDS)
1642 0F84 27      DB 39       ; MAX. TRACK NUMBER
1643 0F85 80      DB RATE_250 ; DATA TRANSFER RATE
1644 -----
1645             ; 40 TRACK LOW DATA RATE MEDIA IN 80 TRACK HI DATA RATE DRIVE :
1646 -----
1647 0F86      MD_TBL2 LABEL BYTE
1648 0F86 DF      DB 1011111B ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
1649 0F87 02      DB 2        ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
1650 0F88 25      DB MOTOR_WAIT ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
1651 0F89 02      DB 2        ; 512 BYTES/SECTOR
1652 0F8A 09      DB 09       ; EOT ( LAST SECTOR ON TRACK)
1653 0F8B 2A      DB 02AH     ; GAP LENGTH
1654 0F8C FF      DB 0FFH     ; DTL
1655 0F8D 50      DB 050H     ; GAP LENGTH FOR FORMAT
1656 0F8E F6      DB 0F6H     ; FILL BYTE FOR FORMAT
1657 0F8F 0F      DB 15       ; HEAD SETTLE TIME (MILLI/SECONDS)
1658 0F90 08      DB 8        ; MOTOR START TIME (1/8 SECONDS)
1659 0F91 27      DB 39       ; MAX. TRACK NUMBER
1660 0F92 40      DB RATE_300 ; DATA TRANSFER RATE
1661 -----
1662             ; 80 TRACK HI DATA RATE MEDIA IN 80 TRACK HI DATA RATE DRIVE :
1663 -----
1664 0F93      MD_TBL3 LABEL BYTE
1665 0F93 DF      DB 1011111B ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
1666 0F94 02      DB 2        ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
1667 0F95 25      DB MOTOR_WAIT ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
1668 0F96 02      DB 2        ; 512 BYTES/SECTOR
1669 0F97 0F      DB 15       ; EOT ( LAST SECTOR ON TRACK)
1670 0F98 1B      DB 01BH     ; GAP LENGTH
1671 0F99 FF      DB 0FFH     ; DTL
1672 0F9A 54      DB 054H     ; GAP LENGTH FOR FORMAT
1673 0F9B F6      DB 0F6H     ; FILL BYTE FOR FORMAT
1674 0F9C 0F      DB 15       ; HEAD SETTLE TIME (MILLI/SECONDS)
1675 0F9D 08      DB 8        ; MOTOR START TIME (1/8 SECONDS)
1676 0F9E 4F      DB 79       ; MAX. TRACK NUMBER
1677 0F9F 00      DB RATE_500 ; DATA TRANSFER RATE
1678 -----
1679             ; 80 TRACK LOW DATA RATE MEDIA IN 80 TRACK LOW DATA RATE DRIVE :
1680 -----
1681 0FA0      MD_TBL4 LABEL BYTE
1682 0FA0 DF      DB 1011111B ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
1683 0FA1 02      DB 2        ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
1684 0FA2 25      DB MOTOR_WAIT ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
1685 0FA3 02      DB 2        ; 512 BYTES/SECTOR
1686 0FA4 09      DB 09       ; EOT ( LAST SECTOR ON TRACK)
1687 0FA5 2A      DB 02AH     ; GAP LENGTH
1688 0FA6 FF      DB 0FFH     ; DTL
1689 0FA7 50      DB 050H     ; GAP LENGTH FOR FORMAT
1690 0FA8 F6      DB 0F6H     ; FILL BYTE FOR FORMAT
1691 0FA9 0F      DB 15       ; HEAD SETTLE TIME (MILLI/SECONDS)
1692 0FAA 08      DB 8        ; MOTOR START TIME (1/8 SECONDS)
1693 0FAB 4F      DB 79       ; MAX. TRACK NUMBER
1694 0FAC 80      DB RATE_250 ; DATA TRANSFER RATE
1695 -----
1696             ; 80 TRACK LOW DATA RATE MEDIA IN 80 TRACK HI DATA RATE DRIVE :
1697 -----
1698 0FAD      MD_TBL5 LABEL BYTE
1699 0FAD DF      DB 1011111B ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
1700 0FAE 02      DB 2        ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
1701 0FAF 25      DB MOTOR_WAIT ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
1702 0FB0 02      DB 2        ; 512 BYTES/SECTOR
1703 0FB1 09      DB 09       ; EOT ( LAST SECTOR ON TRACK)
1704 0FB2 2A      DB 02AH     ; GAP LENGTH
1705 0FB3 FF      DB 0FFH     ; DTL

```

```

1706 0FB4 50          DB      050H          ; GAP LENGTH FOR FORMAT
1707 0FB5 F6          DB      0F6H          ; FILL BYTE FOR FORMAT
1708 0FB6 0F          DB      15           ; HEAD SETTLE TIME (MILLISECONDS)
1709 0FB7 0B          DB      8           ; MOTOR START TIME (1/8 SECONDS)
1710 0FB8 4F          DB      79          ; MAX. TRACK NUMBER
1711 0FB9 80          DB      RATE_250     ; DATA TRANSFER RATE
1712
1713 ;-----
1714 ; 80 TRACK HI DATA RATE MEDIA IN 80 TRACK HI DATA RATE DRIVE
1715 ;-----
1715 0FBA AF          MD_TBL6 LABEL BYTE
1716 0FBB 02          DB      10101111B    ; SRT=A, HD UNLOAD=0F - 1ST SPECIFY BYTE
1717 0FBC 02          DB      2           ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
1718 0FBD 25          DB      MOTOR_WAIT   ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
1719 0FBE 02          DB      2           ; 512 BYTES/SECTOR
1720 0FBF 12          DB      18          ; EOT (LAST SECTOR ON TRACK)
1721 0FC0 1B          DB      01BH        ; GAP LENGTH
1722 0FC0 FF          DB      0FFH        ; DTL
1723 0FC1 6C          DB      06CH        ; GAP LENGTH FOR FORMAT
1724 0FC2 F6          DB      0F6H        ; FILL BYTE FOR FORMAT
1725 0FC3 0F          DB      15          ; HEAD SETTLE TIME (MILLISECONDS)
1726 0FC4 08          DB      8           ; MOTOR START TIME (1/8 SECONDS)
1727 0FC5 4F          DB      79          ; MAX. TRACK NUMBER
1728 0FC6 00          DB      RATE_500    ; DATA TRANSFER RATE
1729
1730 ;-----
1731 ; DISK_BASE
1732 ; THIS IS THE SET OF PARAMETERS REQUIRED FOR DISKETTE OPERATION.
1733 ; THEY ARE POINTED AT BY THE DATA VARIABLE DISK_POINTER. TO
1734 ; MODIFY THE PARAMETERS, BUILD ANOTHER PARAMETER BLOCK AND POINT
1735 ; DISK_POINTER TO IT.
1736 ;-----
1737 0FC7              ORG      0EFC7H
1738 0FC7 CF          DISK_BASE LABEL BYTE
1739 0FC8 02          DB      10101111B    ; SRT=C, HD UNLOAD=0F - 1ST SPECIFY BYTE
1740 0FC8 02          DB      2           ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
1741 0FC9 25          DB      MOTOR_WAIT   ; WAIT AFTER OPN TIL MOTOR OFF
1742 0FCA 02          DB      2           ; 512 BYTES/SECTOR
1743 0FCB 0B          DB      8           ; EOT (LAST SECTOR ON TRACK)
1744 0FCC 2A          DB      02AH        ; GAP LENGTH
1745 0FCD FF          DB      0FFH        ; DTL
1746 0FCE 50          DB      050H        ; GAP LENGTH FOR FORMAT
1747 0FCF F6          DB      0F6H        ; FILL BYTE FOR FORMAT
1748 0FD0 19          DB      25          ; HEAD SETTLE TIME (MILLISECONDS)
1749 0FD1 04          DB      4           ; MOTOR START TIME (1/8 SECONDS)
1750
1751 ;-----
1752 0FD2              ORG      0EFD2H
1753 0FD2              ORG      0EFD2H
1754 0FD2 E9 0000 E   PRINTER_IO: JUMP     PRINTER_IO_1
1755
1756 ;-----
1757 1045              ORG      0F045H
1758 1045 0000 E     M1 ORG      01045H    ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
1759 1047 0000 E     DW      OFFSET SET_MODE ; TABLE OF ROUTINES WITHIN VIDEO I/O
1760 1049 0000 E     DW      OFFSET SET_CTYPE
1761 104B 0000 E     DW      OFFSET SET_CPOS
1762 104D 0000 E     DW      OFFSET READ_CURSOR
1763 104F 0000 E     DW      OFFSET READ_UPEN
1764 1051 0000 E     DW      OFFSET ACT_DISP_PAGE
1765 1053 0000 E     DW      OFFSET SCROLL_UP
1766 1055 0000 E     DW      OFFSET SCROLL_DOWN
1767 1057 0000 E     DW      OFFSET READ_AC_CURRENT
1768 1059 0000 E     DW      OFFSET WRITE_AC_CURRENT
1769 105B 0000 E     DW      OFFSET SET_COLOR
1770 105D 0000 E     DW      OFFSET WRITE_DOT
1771 105F 0000 E     DW      OFFSET READ_DOT
1772 1061 0000 E     DW      OFFSET WRITE_TTY
1773 1063 0000 E     DW      OFFSET VIDEO_STATE
1774 = 0020          M1L EQU      $-M1
1775
1776 ;-----
1777 1065              ORG      0F065H
1778 1065              ORG      01065H
1779 1065 E9 0000 E   VIDEO_IO: JUMP     VIDEO_IO_1
1780
1781 ;----- VIDEO PARAMETERS --- INIT_TABLE
1782
1783 ;-----
1784 10A4              ORG      0F0A4H
1785 10A4              ORG      010A4H
1786 10A4              VIDEO_PARMS LABEL BYTE
1787 10A4 38 28 2D 0A 1F 06 DB      38H,28H,2DH,0AH,1FH,6,19H ; SET UP FOR 40X25
1788 19
1789 10AB 1C 02 07 06 07 DB      1CH,2,7,6,7
1790 10B0 00 00 00 00 00 DB      0,0,0,0
1791 = 0010          M4 EQU      $-VIDEO_PARMS
1792
1793 10B4 71 50 5A 0A 1F 06 DB      71H,50H,5AH,0AH,1FH,6,19H ; SET UP FOR 80X25
1794 19
1795 10BB 1C 02 07 06 07 DB      1CH,2,7,6,7
1796 10C0 00 00 00 00 00 DB      0,0,0,0
1797
1798 10C4 38 28 2D 0A 7F 06 DB      38H,28H,2DH,0AH,7FH,6,64H ; SET UP FOR GRAPHICS
1799 19
1800 10CB 70 02 01 06 07 DB      70H,2,1,6,7
1801 10D0 00 00 00 00 00 DB      0,0,0,0
1802
1803 10D4 61 50 52 0F 19 06 DB      61H,50H,52H,0FH,19H,6,19H ; SET UP FOR 80X25 B&W CARD
1804 19
1805 10DB 19 02 0D 0B 0C DB      19H,2,0DH,0BH,0CH
1806 10E0 00 00 00 00 00 DB      0,0,0,0
1807
1808 10E4 0800          M5 DW      2048      ; TABLE OF REGEN LENGTHS
1809 10E6 1000          DW      4096      ; 40X25
1810 10E8 4000          DW      16384     ; 80X25
1811 10EA 4000          DW      16384     ; GRAPHICS
1812
1813 ;----- COLUMNS
1814 10EC 28 28 50 50 28 28 DB      40,40,80,80,40,40,80,80
1815 50
1816 ;----- C_REG_TAB
1817 10F4 2C 28 2D 29 2A 2E M7 DB      2CH,28H,2DH,29H,2AH,2EH,1EH,29H ; TABLE OF MODE SETS
1818 1E 29

```

```

1819 PAGE
1820 ---- INT 12 -----
1821 : MEMORY_SIZE_DET
1822 : THIS ROUTINE DETERMINES THE AMOUNT OF MEMORY IN THE SYSTEM
1823 : AS REPRESENTED BY THE SWITCHES ON THE PLANAR. NOTE THAT THE
1824 : SYSTEM MAY NOT BE ABLE TO USE I/O MEMORY UNLESS THERE IS A FULL
1825 : COMPLEMENT OF 64K BYTES ON THE PLANAR.
1826 :
1827 : INPUT
1828 : NO REGISTERS
1829 : THE MEMORY_SIZE_VARIABLE IS SET DURING POWER ON DIAGNOSTICS
1830 : ACCORDING TO THE FOLLOWING HARDWARE ASSUMPTIONS:
1831 : PORT 60 BITS 3,2 = 00 - 256K BASE RAM
1832 : 01 - 512K BASE RAM
1833 : 10 - 576K BASE RAM
1834 : 11 - 640K BASE RAM
1835 : PORT 62 BITS 3-0 INDICATE AMOUNT OF I/O RAM IN 32K INCREMENTS
1836 : E.G., 0000 - NO RAM IN I/O CHANNEL
1837 : 0010 - 64K RAM IN I/O CHANNEL, ETC.
1838 :
1839 : OUTPUT (AX) = NUMBER OF CONTIGUOUS 1K BLOCKS OF MEMORY
1840 :
1841 : ASSUME CS:CODE,DS:DATA
1842 : ORG 0F841H
1843 : ORG 01841H
1844 :
1845 : MEMORY_SIZE_DET PROC FAR
1846 : STI ; INTERRUPTS BACK ON
1847 : PUSH DS ; SAVE SEGMENT
1848 : CALL DDS
1849 : MOV AX, MEMORY_SIZE ; GET VALUE
1850 : POP DS ; RECOVER SEGMENT
1851 : IRET ; RETURN TO CALLER
1852 :
1853 : MEMORY_SIZE_DET ENDP
1854 :
1855 : ---- INT 11 -----
1856 : EQUIPMENT_DETERMINATION
1857 : THIS ROUTINE ATTEMPTS TO DETERMINE WHAT OPTIONAL
1858 : DEVICES ARE ATTACHED TO THE SYSTEM.
1859 :
1860 : INPUT
1861 : NO REGISTERS
1862 : THE EQUIP_FLAG_VARIABLE IS SET DURING THE POWER ON
1863 : DIAGNOSTICS USING THE FOLLOWING HARDWARE ASSUMPTIONS:
1864 : PORT 60 = LOW ORDER BYTE OF EQUIPMENT
1865 : PORT 3FA = INTERRUPT ID REGISTER OF 8250
1866 : BITS 7-3 ARE ALWAYS 0
1867 : PORT 378 = OUTPUT PORT OF PRINTER -- 8255 PORT THAT
1868 : CAN BE READ AS WELL AS WRITTEN
1869 :
1870 : OUTPUT (AX) IS SET, BIT SIGNIFICANT, TO INDICATE ATTACHED I/O
1871 : BIT 15,14 = NUMBER OF PRINTERS ATTACHED
1872 : BIT 13 NOT USED
1873 : BIT 12 = GAME I/O ATTACHED
1874 : BIT 11,10,9 = NUMBER OF RS232 CARDS ATTACHED
1875 : BIT 8 UNUSED
1876 : BIT 7,6 = NUMBER OF DISKETTE DRIVES
1877 : 00=1, 01=2, 10=3, 11=4 ONLY IF BIT 0 = 1
1878 : BIT 5,4 = INITIAL VIDEO MODE
1879 : 00 - UNUSED
1880 : 01 - 40X25 BW USING COLOR CARD
1881 : 10 - 80X25 BW USING COLOR CARD
1882 : 11 - 80X25 BW USING BW CARD
1883 : BIT 3,2 = PLANAR RAM SIZE (00=256K,01=512K,10=576K,11=640K)
1884 : BIT 1 = MATH COPROCESSOR
1885 : BIT 0 = IPL FROM DISKETTE -- THIS BIT INDICATES THAT
1886 : THERE ARE DISKETTE DRIVES ON THE SYSTEM
1887 :
1888 : NO OTHER REGISTERS AFFECTED
1889 :
1890 : ASSUME CS:CODE,DS:DATA
1891 : ORG 0F84DH
1892 : ORG 0184DH
1893 :
1894 : EQUIPMENT_DET PROC FAR
1895 : STI ; INTERRUPTS BACK ON
1896 : PUSH DS ; SAVE SEGMENT REGISTER
1897 : CALL DDS
1898 : MOV AX, EQUIP_FLAG ; GET THE CURRENT SETTINGS
1899 : POP DS ; RECOVER SEGMENT
1900 : IRET ; RETURN TO CALLER
1901 :
1902 : EQUIPMENT_DET ENDP
1903 :
1904 : ---- INT 15 -----
1905 : CASSETTE_IO:
1906 : ORG 0F859H
1907 : ORG 01859H
1908 :
1909 : CASSETTE_IO:
1910 : JMP CASSETTE_IO_1
1911 :
1912 :
1913 :
1914 : NON-MASKABLE INTERRUPT ROUTINE:
1915 : THIS ROUTINE WILL PRINT A "PARITY CHECK 1 OR 2" MESSAGE:
1916 : AND ATTEMPT TO FIND THE STORAGE LOCATION CONTAINING THE
1917 : BAD PARITY. IF FOUND, THE SEGMENT ADDRESS WILL BE
1918 : PRINTED. IF NO PARITY ERROR CAN BE FOUND (INTERMITTANT
1919 : READ PROBLEM) ?????<-WILL BE PRINTED WHERE THE ADDRESS
1920 : WOULD NORMALLY GO.
1921 :
1922 : NMI_INT_1 PROC NEAR
1923 : ASSUME DS:DATA
1924 : PUSH AX
1925 : IN AL,PORT_C ; SAVE ORIG CONTENTS OF AX
1926 : TEST AL,0COH ; PARITY CHECK?
1927 : JNZ NMI_1 ; NO, EXIT FROM ROUTINE
1928 : JMP D14
1929 :
1930 : NMI_1:
1931 : MOV DX,DATA
1932 : MOV SI,DX
1933 : MOV SI,OFFSET D1 ; ADDR OF ERROR MSG
1934 : TEST AL,40H ; I/O PARITY CHECK
1935 : JNZ D13 ; DISPLAY ERROR MSG
1936 : MOV SI,OFFSET D2 ; MUST BE PLANAR
1937 :
1938 : D13:
1939 : MOV AH,0
1940 : MOV AL,CRT_MODE ; INIT AND SET MODE FOR VIDEO
1941 : INT 0H
1942 : CALL P_MSG ; CALL VIDEO IO PROCEDURE
1943 : ; PRINT ERROR MSG

```

```

1933
1934
1935
1936 187F B0 00      MOV     AL,00H      ; DISABLE TRAP
1937 1881 E6 A0 00    OUT     AL,PORT_B
1938 1883 E4 61 00    IN      AL,PORT_B
1939 1885 0C 30 00    OR      AL,00110000B    ; TOGGLE PARITY CHECK ENABLES
1940 1887 E6 61 00    OR      PORT_B,AL
1941 1889 24 CF 00    AND     AL,1001111B
1942 188B E6 61 00    OUT     PORT_B,AL
1943 188D 8B 1E 0013 R MOV     BX,#MEMORY_SIZE    ; GET MEMORY SIZE WORD
1944 1891 FC 00 00    CLD      ; SET DIR FLAG TO INCREMENT
1945 1892 2B D2 00    SUB     DX,DX      ; POINT DX AT START OF MEM
1946 1894
1947 1894 8E DA 00    MOV     DS,DX
1948 1896 8E C2 00    MOV     ES,DX
1949 1898 B9 4A00 00 MOV     CX,4A00H
1950 189B 2B F6 00    SUB     SI,SI      ; SET FOR 16KB SCAN
1951
1952 189D F3/ AC 00    REP     LODSB      ; SET SI TO BE REALTIME TO
1953 189F E4 62 00    IN      AL,PORT_C    ; START OF ES
1954 18A1 24 C0 00    AND     AL,11000000B    ; READ 16KB OF MEMORY
1955 18A3 75 11 00    JNZ     PRT_NMI      ; SEE IF PARITY CHECK HAPPENED
1956 18A5 81 C2 0400 ADD     DX,0400H    ; GO PRINT ADDRESS IF IT DID
1957 18A9 83 EB 10 00 SUB     BX,160      ; POINT TO NEXT 16K BLOCK
1958 18AC 75 E6 00    JNZ     NMI_LOOP
1959 18AE BE 1902 R    MOV     SI,(OFFSET D2A) ; PRINT ROW OF ????? IF PARITY
1960 18B1 E8 1997 R    CALL    CL1         ; CHECK COULD NOT BE RE-CREATED
1961 18B4 FA 00 00    HLT      ; HALT SYSTEM
1962 18B5 F4
1963 18B6
1964 18B6 8C DA 00    MOV     DX,DS
1965 18B8 E8 0CBA R    CALL    PRT_SEG      ; PRINT SEGMENT VALUE
1966 18BB FA 00 00    CL1      ; HALT SYSTEM
1967 18BC F4
1968 18BD
1969 18BD 58 00 00    POP     AX      ; RESTORE ORIG CONTENTS OF AX
1970 18BE CF 00 00    IRET
1971 18BF
1972
1973
1974
1975
1976 18BF B9 0000 0000 ROS_CHECKSUM PROC NEAR ; NEXT ROS MODULE
1977 18BF B9 0000 0000 MOV     CX,0      ; NUMBER OF BYTES TO ADD
1978 18C2
1979 18C2 32 C0 00    ROS_CHECKSUM_CNT: XOR     CX,CX      ; ENTRY FOR OPTIONAL ROS TEST
1980 18C4
1981 18C4 02 07 00    C26: ADD     AL,DS:[BX] ; POINT TO NEXT BYTE
1982 18C6 43 00 00    INC     BX          ; ADD ALL BYTES IN ROS MODULE
1983 18C7 E2 FB 00    LOOP    C26         ; SUM = 0?
1984 18C9 0A C0 00    OR      AL,AL
1985 18CB C3 00 00    RET
1986 18CC
1987
1988
1989
1990 18CC 31 30 31 0D 0A E0 DB '101',CR,LF ; SYSTEM BOARD ERROR
1991 18D1 20 32 30 31 0D 0A E1 DB '201',CR,LF ; MEMORY ERROR
1992 18D7 52 4F 4D 0D 0A F3A DB 'ROM',CR,LF ; ROM CHECKSUM ERROR
1993 18DC 31 36 30 31 0D 0A F3C DB '1801',CR,LF ; EXPANSION IO BOX ERROR
1994 18E2 50 41 52 49 54 59 D1 DB 'PARITY CHECK 2',CR,LF
1995
1996
1997 18F2 50 41 52 49 54 59 D2 DB 'PARITY CHECK 1',CR,LF
1998
1999
2000 1902 3F 3F 3F 3F 0D D2A DB '?????',CR,LF
2001
2002
2003
2004
2005
2006
2007
2008 1909
2009 1909 FB 00 00    BLINK_INT PROC NEAR
2010 190A 50 00 00    STI
2011 190B E4 61 00    PUSH    AX      ; SAVE AX REG CONTENTS
2012 190D BA E0 00 00 MOV     AL,PORT_B ; READ CURRENT VAL OF PORT B
2013 190F F6 D0 00    NOT     AL
2014 1911 24 40 00    AND     AL,01000000B ; FLIP ALL BITS
2015 1913 80 E4 BF 00 AND     AH,10111111B ; ISOLATE CONTROL BIT
2016 1916 0A C4 00 00 OR      AL,AH      ; MASK OUT OF ORIGINAL VAL
2017 1918 E6 61 00    OUT     PORT_B,AL ; OR NEW CONTROL BIT IN
2018 191A B0 20 00    MOV     AL,E01
2019 191C E6 20 00    OUT     INTA00,AL
2020 191E 58 00 00    POP     AX
2021 191F CF 00 00    IRET      ; RESTORE AX REG
2022 1920
2023
2024
2025
2026
2027
2028 1920
2029 1920 B8 ---- R    PROC NEAR
2030 1923 9E C0 00    MOV     AX,DATA     ; POINT ES TO DATA AREA
2031 1925 2A E4 00 00 SUB     AH,AH        ; ZERO OUT AH
2032 1927 BA 47 02 00 MOV     AL,[BX+2]    ; GET LENGTH INDICATOR
2033 192A B1 09 00 00 MOV     CL,09H      ; MULTIPLY BY 512
2034 192C D3 04 00 00 SHL     AX,CL
2035 192E 8B C8 00 00 MOV     CX,AX
2036 1930 51 00 00    PUSH    CX          ; SET COUNT
2037 1931 B9 0004 0000 MOV     CX,4        ; SAVE COUNT
2038 1934 D3 E8 00 00 SHR     DX,AX      ; ADJUST
2039 1936 03 D0 00 00 ADD     DX,AX
2040 1938 59 00 00    POP     CX
2041 1939 E8 18C2 R    CALL    ROS_CHECKSUM_CNT ; SET POINTER TO NEXT MODULE
2042 193C 74 C4 00 00 JZ      ROM_CHECK_1    ; RETRIEVE COUNT
2043 193E E8 0746 R    CALL    ROM_ERR      ; DO CHECKSUM
2044 1941 EB 14 90 00 JMP     ROM_CHECK_END ; POST CHECKSUM ERROR
2045 1944
2046 1944 52 00 00    PUSH    DX          ; AND EXIT
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
3000

```

```

2047 1945 26: C7 06 0067 R 0003      MOV     ES:10 ROM_INIT,0003H ; LOAD OFFSET
2048 194C 26: 8C 1E 0069 R          MOV     ES:10 ROM_SEG,DS ; LOAD SEGMENT
2049 1951 26: FF 1E 0067 R          CALL    DWORD PTR ES:10 ROM_INIT ; CALL INIT./TEST ROUTINE
2050 1956 5A                        POP     DX
2051 1957                        ROM_CHECK_END:
2052 1957 C3                        RET             ; RETURN TO CALLER
2053 1958                        ROM_CHECK      ENDP
2054
2055      -----
2056      ; CONVERT AND PRINT ASCII CODE
2057      ; AL MUST CONTAIN NUMBER TO BE CONVERTED.
2058      ; AX AND BX DESTROYED.
2059      -----
2060 1958      PROC      NEAR
2061 1958 50      PUSH    AX ; SAVE FOR LOW NIBBLE DISPLAY
2062 1959 B1 04    MOV     CL,4 ; SHIFT COUNT
2063 195B D2 E8    SHR     AL,CL ; NYBBLE SWAP
2064 195D E8 16 3 R CALL    XLAT_PR ; DO THE HIGH NIBBLE DISPLAY
2065 1960 58      POP     AX ; RECOVER THE NIBBLE
2066 1961 24 0F    AND     AL,0FH ; ISOLATE TO LOW NIBBLE
2067 1962          ; FALL INTO LOW NIBBLE CONVERSION
2068 1963          ; CONVERT 00-0F TO ASCII CHARACTER
2069 1963 04 90    ADD     AL,090H ; ADD FIRST CONVERSION FACTOR
2070 1965 27      DAA      ; ADJUST FOR NUMERIC AND ALPHA RANGE
2071 1966 14 40    ADC     AL,040H ; ADD CONVERSION AND ADJUST LOW NIBBLE
2072 1968 27      DAA      ; ADJUST HIGH NIBBLE TO ASCII RANGE
2073 1969          ;
2074 1969 B4 0E    MOV     AH,14 ; DISPLAY CHARACTER IN AL
2075 196B B7 00    MOV     BH,0 ;
2076 196D CD 10    INT     10H ; CALL VIDEO_IO
2077 196F C3      RET
2078 1970          PRT_HEX_END:
2079 1970          XLAT_PR_END:
2080 1970          XPC_BYTE      ENDP
2081
2082 1970          F4      LABEL    WORD ; PRINTER SOURCE TABLE
2083 1970 03BC     DW      3BCH
2084 1972 0378     DW      378H
2085 1974 0278     DW      278H
2086 1976          F4E      LABEL    WORD
2087
2088      ; THIS SUBROUTINE WILL PRINT A MESSAGE ON THE DISPLAY
2089      ;
2090      ; ENTRY REQUIREMENTS:
2091      ; SI = OFFSET(ADDRESS) OF MESSAGE BUFFER
2092      ; CX = MESSAGE BYTE COUNT
2093      ; MAXIMUM MESSAGE LENGTH IS 36 CHARACTERS
2094      ;
2095      -----
2096 1976      E_MSG      PROC      NEAR
2097 1976 8B EE     MOV     BP,SI ; SET BP NON-ZERO TO FLAG ERR
2098 1978 E8 1997 R CALL    P_MSG ; PRINT MESSAGE
2099 197B 1E      PUSH    DS
2100 197C E8 1A12 R CALL    DDS
2101 197F A0 010 R MOV     AL,BYTE PTR 0EQUIP_FLAG ; LOOP/HALT ON ERROR
2102 1982 24 01    AND     AL,01H ; SWITCH ON?
2103 1984 75 0F    JNZ     G12 ; NO - RETURN
2104 1986          ; YES - HALT SYSTEM
2105 1986 FA      CLI
2106 1987 B0 89     MOV     AL,89H
2107 1989 E6 63     OUT     CMD_PORT,AL
2108 198B B0 85     MOV     AL,T0000101B ; DISABLE KB
2109 198D E6 61     OUT     PORT_B,AL
2110 198F A0 0015 R MOV     AL,0F8F
2111 1992 E6 60     OUT     PORT_A,AL ; RECOVER ERROR INDICATOR
2112 1994 F4      HLT ; SET INTO 8255 REG
2113 1995          ; HALT SYS
2114 1995 1F      POP     DS
2115 1996 C3      RET
2116 1997          E_MSG      ENDP
2117
2118 1997          P_MSG      PROC      NEAR
2119 1997          GT2A:
2120 1997 2E: 8A 04    MOV     AL,CS:[SI] ; PUT CHAR IN AL
2121 199A 46      INC     SI ; POINT TO NEXT CHAR
2122 199B 50      PUSH    AX ; SAVE PRINT CHAR
2123 199C E8 1969 R CALL    PRT_HEX ; CALL VIDEO_IO
2124 199F 58      POP     AX ; RECOVER PRINT CHAR
2125 19A0 3C 0A    CMP     AL,10 ; WAS IT LINE FEED?
2126 19A2 75 F3    JNE     G12A ; NO,KEEP PRINTING STRING
2127 19A4 C3      RET
2128 19A5          P_MSG      ENDP
2129          ASSUME     CS:CODE,DS:DATA
2130
2131      -----
2132      ; THIS PROCEDURE WILL ISSUE LONG TONES (1-3/4 SECONDS) AND ONE OR
2133      ; MORE SHORT TONES (9/32 SECOND) TO INDICATE A FAILURE ON THE
2134      ; PLANAR BOARD, A BAD MEMORY MODULE, OR A PROBLEM WITH THE CRT.
2135      ;
2136      ; ENTRY PARAMETERS:
2137      ; DL = NUMBER OF LONG TONES TO BEEP.
2138      ; DL = NUMBER OF SHORT TONES TO BEEP.
2139      ;
2140      -----
2141 19A5          ERR_BEEP      PROC      NEAR
2142 19A5 9C      PUSHF
2143 19A6 FA      CLI ; DISABLE SYSTEM INTERRUPTS
2144 19A7 0A F6    OR     DH,DH ; ANY LONG ONES TO BEEP
2145 19A9 74 1E    JZ     G1 ; NO, DO THE SHORT ONES
2146 19AB          ; LONG BEEPS
2147 19AB B3 70    MOV     BL,112 ; COUNTER FOR LONG BEEPS (1-3/4 SECONDS)
2148 19AD B9 05C0 R MOV     CX,1280 ; DIVISOR FOR 932 HZ
2149 19B0 E8 0C5C R CALL    BEEP ; DO THE BEEP
2150 19B3 B9 C233 MOV     CX,49715 ; 2/3 SECOND DELAY AFTER LONG BEEP
2151 19B6 E8 0CA0 R CALL    WAITF ; DELAY BETWEEN BEEPS
2152 19B9 FE CE    DEC     DH ; ANY MORE LONG BEEPS TO DO
2153 19BB 75 EE    JNZ     G1 ; LOOP TILL DONE
2154 19BD 1E      PUSH    DS ; SAVE DS REGISTER CONTENTS
2155 19BE E8 1A12 R CALL    DDS
2156 19C0 3E 0012 R CMP     0F8F,TST,01H ; MANUFACTURING TEST MODE?
2157 19C2 0F      JNE     MFG_HALT ; RESTORE ORIGINAL CONTENTS OF (DS)
2158 19C3 B3 12    MOV     BL,18 ; YES - STOP BLINKING LED
2159 19C5 0F      JNE     MFG_HALT ; SHORT BEEPS
2160 19C7 74 BD    JZ     G3 ; COUNTER FOR A SHORT BEEP (9/32)
2161 19C9 B3 12    MOV     BL,18 ; DIVISOR FOR 987 HZ
2162 19CB B9 04BB MOV     CX,1208 ; DO THE SOUND
2163 19CE E8 05C0 R CALL    BEEP

```

```
2161 19D1 B9 8178      MOV     CX,33144      ; 1/2 SECOND DELAY AFTER SHORT BEEP
2162 19D4 E8 0CA0 R    CALL    WAITF      ; DELAY BETWEEN BEEPS
2163 19D7 FE CA        DEC     DL          ; DONE WITH SHORT BEEPS COUNT
2164 19D9 75 EE        JNZ     G3          ; LOOP TILL DONE
2165 19DB B9 8178      MOV     CX,33144      ; 1/2 SECOND DELAY AFTER LAST BEEP
2166 19DE E8 0CA0 R    CALL    WAITF      ; MAKE IT ONE SECOND DELAY BEFORE RETURN
2167 19E1 9D          POPF     RET          ; RESTORE FLAGS TO ORIGINAL SETTINGS
2168 19E2 C3          RET          ; RETURN TO CALLER
2169 19E3          ERR_BEOP      ENDOP

2170
2171
2172
2173
2174
2175 19E3          KBD_RESET      PROC      NEAR
2176
2177 19E3 B0 08          ASSUME    DS:1A850
2178 19E5 E6 61          MOV     AL,08H      ; SET KBD CLK LINE LOW
2179 19E7 B9 2956      OUT     PORT_B,AL    ; WRITE 8255 PORT B
2180 19EA          MOV     CX,10582      ; HOLD KBD CLK LOW FOR 20 MS
2181 19EA E2 FE        LOOP     G8          ; LOOP FOR 20 MS
2182 19EC B0 C8        MOV     AL,0C8H    ; SET CLK, ENABLE LINES HIGH
2183 19EE E6 61        OUT     PORT_B,AL
2184 19F0          SP_TEST1      MOV     AL,48H      ; ENTRY FOR MANUFACTURING TEST 2
2185 19F0 B0 48        OUT     PORT_B,AL    ; SET KBD CLK HIGH, ENABLE LOW
2186 19F2 E6 61        MOV     AL,0F0H    ; ENABLE KEYBOARD INTERRUPTS
2187 19F4 B0 FD        OUT     INTA01,AL    ; WRITE 8259 IMR
2188 19F6 E6 21        MOV     DATA_AREA[*INTR_FLAG-DATA0],0 ; RESET INTERRUPT INDICATOR
2189 19F8 C6 06 046B R 00 STI          ; ENABLE INTERRUPTS
2190 19FD FB          SUB     CX,CX      ; SETUP INTERRUPT TIMEOUT CNT
2191 19FE 2B C9        G9:      TEST    DATA_AREA[*INTR_FLAG-DATA0],02H ; DID A KEYBOARD INTR OCCUR?
2192 1A00          JNZ     G10         ; YES - READ SCAN CODE RETURNED
2193 1A00 F6 06 046B R 02 G10:     LOOP   G9          ; NO - LOOP TILL TIMEOUT
2194 1A05 75 02
2195 1A07 E2 F7        IN     AL,PORT_A    ; READ KEYBOARD SCAN CODE
2196 1A09          MOV     BL,AL        ; SAVE SCAN CODE JUST READ
2197 1A09 B0 C8        MOV     AL,0C8H    ; CLEAR KEYBOARD
2198 1A0B BA D8        OUT     PORT_B,AL
2199 1A0D B0 C8        RET          ; RETURN TO CALLER
2200 1A0F E6 61
2201 1A11 C3          KBD_RESET      ENDOP
2202 1A12
2203
2204 1A12          DDS      PROC      NEAR      ; LOAD (DS) TO DATA AREA
2205 1A12 2E: 8E 1E 1A18 R MOV     DS,C5:DDSDATA    ; PUT SEGMENT VALUE OF DATA AREA INTO DS
2206 1A17 C3          RET          ; RETURN TO USER WITH (DS)= DATA
2207
2208 1A18 ---- R      DDSDATA DW      DATA      ; SEGMENT SELECTOR VALUE FOR DATA AREA
2209
2210 1A1A          DDS      ENDOP
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229 1A1A          ;--- HARDWARE INT 08 H -- ( IRQ LEVEL 0 )
2230 1A1A FB          ;
2231 1A1B 1E          ; THIS ROUTINE HANDLES THE TIMER INTERRUPT FROM FROM CHANNEL 0 OF
2232 1A1C 50          ; THE 8254 TIMER. INPUT FREQUENCY IS 1.19318 MHZ AND THE DIVISOR
2233 1A1D 52          ; IS 65536, RESULTING IN APPROXIMATELY 18.2 INTERRUPTS EVERY SECOND.
2234 1A1E B8 ---- R  ;
2235 1A21 BE D8        ; THE INTERRUPT HANDLER MAINTAINS A COUNT (4016C) OF INTERRUPTS SINCE
2236 1A23 FF 06 006C R MOV     AX,DATA    ; POWER ON TIME, WHICH MAY BE USED TO ESTABLISH TIME OF DAY.
2237 1A27 75 04        ; THE INTERRUPT HANDLER ALSO DECREMENTES THE MOTOR CONTROL COUNT (40140)
2238 1A29 FF 06 006E R INC     *TIMER_LOW    ; OF THE DISKETTE, AND WHEN IT EXPIRES, WILL TURN OFF THE
2239 1A2D          JNZ     T4          ; DISKETTE MOTOR(1), AND RESET THE MOTOR RUNNING FLAGS.
2240 1A2D 83 3E 006E R 18 CMP     *TIMER_HIGH,018H ; THE INTERRUPT HANDLER WILL ALSO INVOKE A USER ROUTINE THROUGH
2241 1A32 75 19        JNZ     T5          ; INTERRUPT ICH AT EVERY TIME TICK. THE USER MUST CODE A
2242 1A34 81 3E 006C R 00B0 CMP     *TIMER_LOW,0B0H ; ROUTINE AND PLACE THE CORRECT ADDRESS IN THE VECTOR TABLE.
2243 1A3A 75 11        JNZ     T5          ;
2244
2245
2246
2247 1A3C 2B C0        ASSUME    CS:CODE,DS:DATA
2248 1A3E A3 006E R    SUB     AX,AX
2249 1A41 A3 006C R    MOV     *TIMER_HIGH,AX    ; CLEAR TIMER COUNT HIGH
2250 1A44 C6 06 0070 R 01 MOV     *TIMER_LOW,AX      ; AND LOW
2251 1A49 FF 06 00CE R MOV     *TIMER_OFL,1      ; SET TIMER ELAPSED 24 HOURS FLAG
2252          INC     *DAY_COUNT    ; INCREMENT ELAPSED DAY COUNTER
2253
2254
2255 1A4D          ;----- TEST FOR DISKETTE TIME OUT
2256 1A4D FE 0E 0040 R T5:      DEC     *MOTOR_COUNT    ; DECREMENT DISKETTE MOTOR CONTROL
2257 1A51 75 08        JNZ     T6          ; RETURN IF COUNT NOT OUT
2258 1A53 B0 26 003F R F0 AND     *MOTOR_STATUS,0F0H ; TURN OFF MOTOR RUNNING BITS
2259 1A58 B0 0C        MOV     AL,0CH      ;
2260 1A5A BA 03F2      MOV     DX,03F2H    ; FDC CTL PORT
2261 1A5D EE          OUT     DX,AL        ; TURN OFF THE MOTOR
2262
2263 1A5E          T6:          INT     ICH          ; TIMER TICK INTERRUPT
2264 1A5E CD 1C        INT     ICH          ; TRANSFER CONTROL TO A USER ROUTINE
2265
2266 1A60 FA          CLI          ; DISABLE INTERRUPTS TILL STACK CLEARED
2267 1A61 B0 20        MOV     AL,E01      ; GET END OF INTERRUPT MASK
2268 1A63 E6 20        OUT     INTA00,AL    ; END OF INTERRUPT TO 8259 - 1
2269 1A65 5A          POP     DX          ; RESTORE (DX)
2270 1A66 58          POP     AX          ;
2271 1A67 1F          POP     DS          ; RESET MACHINE STATE
2272 1A68 CF          IRET         ; RETURN FROM INTERRUPT
2273
2274 1A69          TIMER_INT_1      ENDOP
```

2275	PAGE	
2276	CHARACTER GENERATOR GRAPHICS FOR 320X200 AND 640X200 GRAPHICS	
2277		
2278		
2279		
2280	1A6E	
2281	1A6E	CRT_CHAR GEN
2282	1A6E 00 00 00 00 00 00	000H,000H,000H,000H,000H,000H,000H,000H ; D_00 BLANK
2283	00 00	
2284	1A7E 7E 81 A5 81 BD 99	DB 07EH,081H,0A5H,081H,0BDH,099H,081H,07EH ; D_01 SMILING FACE
2285	81 7E	
2286	1A7E 7E FF DB FF C3 E7	DB 07EH,0FFH,0DBH,0FFH,0C3H,0ETH,0FFH,07EH ; D_02 SMILING FACE N
2287	FF 7E	
2288	1AB6 6C FE FE FE 7C 3B	DB 06CH,0FEH,0FEH,0FEH,07CH,03BH,010H,000H ; D_03 HEART
2289	10 00	
2290	1ABE 10 38 7C FE 7C 3B	DB 010H,03BH,07CH,0FEH,07CH,03BH,010H,000H ; D_04 DIAMOND
2291	10 00	
2292	1A96 38 7C 38 FE FE 7C	DB 03BH,07CH,03BH,0FEH,0FEH,07CH,03BH,07CH ; D_05 CLUB
2293	38 7C	
2294	1A9E 10 10 38 7C FE 7C	DB 010H,010H,03BH,07CH,0FEH,07CH,03BH,07CH ; D_06 SPADE
2295	38 7C	
2296	1AA6 00 00 18 3C 3C 18	DB 000H,000H,01BH,03CH,03CH,01BH,000H,000H ; D_07 BULLET
2297	00 00	
2298	1AAE FF FF E7 C3 C3 E7	DB 0FFH,0FFH,0ETH,0C3H,0C3H,0ETH,0FFH,0FFH ; D_08 BULLET NEG
2299	FF FF	
2300	1AB6 00 3C 66 42 42 66	DB 000H,03CH,066H,042H,042H,066H,03CH,000H ; D_09 CIRCLE
2301	3C 00	
2302	1ABE FF C3 99 BD BD 99	DB 0FFH,0C3H,099H,0BDH,0BDH,099H,0C3H,0FFH ; D_0A CIRCLE NEG
2303	C3 FF	
2304	1AC6 0F 07 0F 7D CC CC	DB 00FH,007H,00FH,07DH,0CCH,0CCH,00FH,00H ; D_0B MALE
2305	CC 78	
2306	1ACE 3C 66 66 66 3C 18	DB 03CH,066H,066H,066H,03CH,01BH,07EH,01BH ; D_0C FEMALE
2307	7E 18	
2308	1AD6 3F 33 3F 30 30 70	DB 03FH,033H,03FH,030H,030H,070H,0F0H,0E0H ; D_0D EIGHTH NOTE
2309	F0 E0	
2310	1ADE 7F 63 7F 63 63 67	DB 07FH,063H,07FH,063H,063H,067H,0E6H,0C0H ; D_0E TWO 1/16 NOTE
2311	E6 C0	
2312	1AE6 9A 3C E7 E7 3C	DB 099H,05AH,03CH,0ETH,0ETH,03CH,05AH,099H ; D_0F SUN
2313	5A 99	
2314		
2315	1AEE 80 E0 F8 FE F8 E0	DB 080H,0E0H,0F8H,0FEH,0F8H,0E0H,080H,000H ; D_10 R ARROWHEAD
2316	80 00	
2317	1AF6 02 0E 3E FE 3E 0E	DB 002H,00EH,03EH,0FEH,03EH,00EH,002H,000H ; D_11 L ARROWHEAD
2318	02 00	
2319	1AFE 18 3C 7E 18 18 7E	DB 01BH,03CH,07EH,01BH,01BH,07EH,03CH,01BH ; D_12 ARROW 2 VERT
2320	3C 18	
2321	1B06 66 66 66 66 66 00	DB 066H,066H,066H,066H,066H,000H,066H,000H ; D_13 2 EXCLAMATIONS
2322	66 00	
2323	1B0E 7F DB DB 7B 1B 1B	DB 07FH,0DBH,0DBH,07BH,01BH,01BH,01BH,000H ; D_14 PARAGRAPH
2324	1B 00	
2325	1B16 3E 63 38 6C 6C 3B	DB 03EH,063H,03BH,06CH,06CH,03BH,0CCH,07BH ; D_15 SECTION
2326	CC 78	
2327	1B1E 00 00 00 00 7E 7E	DB 000H,000H,000H,000H,07EH,07EH,07EH,000H ; D_16 RECTANGLE
2328	7E 00	
2329	1B26 18 3C 7E 18 7E 3C	DB 01BH,03CH,07EH,01BH,07EH,03CH,01BH,0FFH ; D_17 ARROW 2 VRT UP
2330	18 FF	
2331	1B2E 18 3C 7E 18 18 18	DB 01BH,03CH,07EH,01BH,01BH,01BH,01BH,000H ; D_18 ARROW VRT UP
2332	18 00	
2333	1B36 18 18 18 18 7E 3C	DB 01BH,01BH,01BH,01BH,07EH,03CH,01BH,000H ; D_19 ARROW VRT DOWN
2334	18 00	
2335	1B3E 00 18 0C FE 0C 1B	DB 000H,01BH,0CCH,0FEH,0CCH,01BH,000H,000H ; D_1A ARROW RIGHT
2336	00 00	
2337	1B46 00 30 60 FE 60 30	DB 000H,030H,060H,0FEH,060H,030H,000H,000H ; D_1B ARROW LEFT
2338	00 00	
2339	1B4E 00 C0 C0 C0 FE	DB 000H,000H,0C0H,0C0H,0C0H,0FEH,000H,000H ; D_1C NOT INVERTED
2340	00 00	
2341	1B56 00 24 66 FF 66 24	DB 000H,024H,066H,0FFH,066H,024H,000H,000H ; D_1D ARROW 2 HORZ
2342	00 00	
2343	1B5E 00 18 3C TE FF FF	DB 000H,01BH,03CH,07EH,0FFH,0FFH,000H,000H ; D_1E ARROWHEAD UP
2344	00 00	
2345	1B66 00 FF FF TE 3C 18	DB 000H,0FFH,0FFH,07EH,03CH,01BH,000H,000H ; D_1F ARROWHEAD DOWN
2346	00 00	
2347		
2348	1B6E 00 00 00 00 00 00	DB 000H,000H,000H,000H,000H,000H,000H,000H ; D_20 SPACE
2349	00 00	
2350	1B76 30 78 78 30 00 00	DB 030H,078H,078H,030H,030H,000H,030H,000H ; D_21 ! EXCLAMATION
2351	30 00	
2352	1B7E 6C 6C 6C 6C 00 00 00	DB 06CH,06CH,06CH,000H,000H,000H,000H,000H ; D_22 " QUOTATION
2353	00 00	
2354	1B86 6C 6C 6C 6C FE 6C	DB 06CH,06CH,0FEH,06CH,0FEH,06CH,06CH,000H ; D_23 # LB.
2355	6C 00	
2356	1B8E 30 7C C0 78 0C F8	DB 030H,07CH,0C0H,07BH,0CCH,0F8H,030H,000H ; D_24 \$ DOLLAR SIGN
2357	30 00	
2358	1B96 00 C6 CC 18 30 66	DB 000H,0C6H,0CCH,01BH,030H,066H,0C6H,000H ; D_25 % PERCENT
2359	C6 00	
2360	1B9E 38 6C 38 76 DC 00	DB 03BH,06CH,03BH,076H,0CCH,0CCH,076H,000H ; D_26 & AMPERSAND
2361	76 00	
2362	1BA6 60 60 C0 00 00 00	DB 060H,060H,0C0H,000H,000H,000H,000H,000H ; D_27 ' ASTROPHOSE
2363	00 00	
2364	1BAE 18 30 60 60 60 30	DB 01BH,030H,060H,060H,060H,030H,01BH,000H ; D_28 (L. PARENTHESIS
2365	18 00	
2366	1BB6 60 30 18 18 18 30	DB 060H,030H,01BH,01BH,01BH,030H,060H,000H ; D_29) R. PARENTHESIS
2367	60 00	
2368	1BBE 00 66 3C FF 3C 66	DB 000H,066H,03CH,0FFH,03CH,066H,000H,000H ; D_2A * ASTERISK
2369	00 00	
2370	1BC6 00 30 30 FC 30 30	DB 000H,030H,030H,0FCH,030H,030H,000H,000H ; D_2B + PLUS
2371	00 00	
2372	1BCE 00 00 00 00 00 30	DB 000H,000H,000H,000H,000H,030H,030H,060H ; D_2C , COMMA
2373	30 60	
2374	1BD6 00 00 00 FC 00 00	DB 000H,000H,000H,0FCH,000H,000H,000H,000H ; D_2D - DASH
2375	00 00	
2376	1BDE 00 00 00 00 00 30	DB 000H,000H,000H,000H,000H,030H,030H,000H ; D_2E . PERIOD
2377	30 00	
2378	1BE6 06 0C 18 30 60 C0	DB 066H,000H,01BH,030H,060H,0C0H,080H,000H ; D_2F / SLASH
2379	80 00	
2380		
2381	1BEE 7C C6 CE DE F6 E6	DB 07CH,0C6H,0CEH,0DEH,0F6H,0E6H,07CH,000H ; D_30 0
2382	7C 00	
2383	1BF6 30 70 30 30 30 30	DB 030H,070H,030H,030H,030H,030H,0FCH,000H ; D_31 1
2384	FC 00	
2385	1BFE 78 CC 0C 38 60 CC	DB 07BH,0CCH,0CCH,03BH,060H,0CCH,0FCH,000H ; D_32 2
2386	FC 00	
2387	1C06 78 CC 0C 38 0C CC	DB 07BH,0CCH,0CCH,03BH,0CCH,0CCH,07BH,000H ; D_33 3
2388	78 00	

2389	1C0E	1C	3C	6C	CC	FE	0C	DB	01CH,03CH,06CH,0CCH,0FEH,00CH,01EH,000H	D_34	4
2390		1E	00								
2391	1C16	FC	00	F8	0C	0C	CC	DB	0FCH,0C0H,0F8H,00CH,00CH,0CCH,078H,000H	D_35	5
2392		78	00								
2393	1C1E	38	60	00	F8	CC	CC	DB	038H,060H,0C0H,0F8H,0CCH,0CCH,078H,000H	D_36	6
2394		78	00								
2395	1C26	FC	CC	0C	18	30	30	DB	0FCH,0CCH,00CH,018H,030H,030H,030H,000H	D_37	7
2396		30	00								
2397	1C2E	78	CC	CC	78	CC	CC	DB	078H,0CCH,0CCH,078H,0CCH,0CCH,078H,000H	D_38	8
2398		78	00								
2399	1C36	78	CC	CC	7C	0C	18	DB	078H,0CCH,0CCH,07CH,00CH,018H,070H,000H	D_39	9
2400		70	00								
2401	1C3E	00	30	00	00	00	30	DB	000H,030H,030H,000H,000H,030H,030H,000H	D_3A	:
2402		30	00								COLON
2403	1C46	00	30	00	00	00	30	DB	000H,030H,030H,000H,000H,030H,030H,060H	D_3B	;
2404		30	00								SEMICOLON
2405	1C4E	18	30	60	C0	60	30	DB	018H,030H,060H,0C0H,060H,030H,018H,000H	D_3C	<
2406		18	00								LESS THAN
2407	1C56	00	00	FC	00	00	FC	DB	000H,000H,0FCH,000H,000H,0FCH,000H,000H	D_3D	=
2408		00	00								EQUAL
2409	1C5E	60	30	18	0C	18	30	DB	060H,030H,018H,00CH,018H,030H,060H,000H	D_3E	>
2410		60	00								GREATER THAN
2411	1C66	78	CC	0C	18	30	00	DB	078H,0CCH,00CH,018H,030H,000H,030H,000H	D_3F	?
2412		30	00								QUESTION MARK
2413											
2414	1C6E	7C	C6	DE	DE	DE	C0	DB	07CH,0C6H,0DEH,0DEH,0DEH,0C0H,078H,000H	D_40	@
2415		78	00								AT
2416	1C76	30	78	CC	CC	FC	CC	DB	030H,078H,0CCH,0CCH,0FCH,0CCH,0CCH,000H	D_41	A
2417		CC	00								
2418	1C7E	FC	66	66	7C	66	66	DB	0FCH,066H,066H,07CH,066H,066H,0FCH,000H	D_42	B
2419		FC	00								
2420	1C86	3C	66	C0	C0	C0	66	DB	03CH,066H,0C0H,0C0H,0C0H,066H,0FCH,000H	D_43	C
2421		3C	00								
2422	1C8E	F8	6C	66	66	66	6C	DB	0F8H,06CH,066H,066H,066H,06CH,0F8H,000H	D_44	D
2423		F8	00								
2424	1C96	FE	62	68	78	68	62	DB	0FEH,062H,068H,078H,068H,062H,0FEH,000H	D_45	E
2425		FE	00								
2426	1C9E	FE	62	68	78	68	60	DB	0FEH,062H,068H,078H,068H,060H,0F0H,000H	D_46	F
2427		F0	00								
2428	1CA6	3C	66	C0	C0	CE	66	DB	03CH,066H,0C0H,0C0H,0CEH,066H,03EH,000H	D_47	G
2429		3E	00								
2430	1CAE	CC	CC	CC	FC	CC	CC	DB	0CCH,0CCH,0CCH,0FCH,0CCH,0CCH,0CCH,000H	D_48	H
2431		CC	00								
2432	1CB6	78	30	30	30	30	30	DB	078H,030H,030H,030H,030H,030H,078H,000H	D_49	I
2433		78	00								
2434	1CBE	1E	0C	0C	0C	CC	CC	DB	01EH,00CH,0CCH,0CCH,0CCH,0CCH,078H,000H	D_4A	J
2435		78	00								
2436	1CC6	E6	6C	78	6C	66		DB	0E6H,066H,06CH,078H,06CH,066H,0E6H,000H	D_4B	K
2437		E6	00								
2438	1CCE	F0	60	60	60	62	66	DB	0F0H,060H,060H,060H,062H,066H,0FEH,000H	D_4C	L
2439		F0	00								
2440	1CD6	C6	E6	FE	FE	D6	C6	DB	0C6H,0EEH,0FEH,0FEH,0D6H,0C6H,0C6H,000H	D_4D	M
2441		C6	00								
2442	1CDE	C6	E6	F6	DE	CE	C6	DB	0C6H,0E6H,0F6H,0DEH,0CEH,0C6H,0C6H,000H	D_4E	N
2443		C6	00								
2444	1CE6	C6	C6	C6	C6	6C		DB	038H,06CH,0C6H,0C6H,0C6H,06CH,038H,000H	D_4F	O
2445		38	00								
2446											
2447	1CEE	FC	66	66	7C	60	60	DB	0FCH,066H,066H,07CH,060H,060H,0F0H,000H	D_50	P
2448		F0	00								
2449	1CF6	78	CC	CC	CC	DC	78	DB	078H,0CCH,0CCH,0CCH,0CCH,078H,01CH,000H	D_51	Q
2450		1C	00								
2451	1CFE	FC	66	66	7C	6C	66	DB	0FCH,066H,066H,07CH,06CH,066H,0E6H,000H	D_52	R
2452		E6	00								
2453	1D06	78	CC	E0	70	1C	CC	DB	078H,0CCH,0E0H,070H,01CH,0CCH,078H,000H	D_53	S
2454		78	00								
2455	1D0E	FC	64	30	30	30	30	DB	0FCH,0B4H,030H,030H,030H,030H,078H,000H	D_54	T
2456		78	00								
2457	1D16	CC	CC	CC	CC	CC	CC	DB	0CCH,0CCH,0CCH,0CCH,0CCH,0CCH,0FCH,000H	D_55	U
2458		FC	00								
2459	1D1E	CC	CC	CC	CC	78		DB	0CCH,0CCH,0CCH,0CCH,0CCH,078H,030H,000H	D_56	V
2460		30	00								
2461	1D26	C6	C6	C6	D6	FE	EE	DB	0C6H,0C6H,0C6H,0D6H,0FEH,0EEH,0C6H,000H	D_57	W
2462		C6	00								
2463	1D2E	C6	C6	6C	38	38	6C	DB	0C6H,0C6H,06CH,038H,038H,06CH,0C6H,000H	D_58	X
2464		C6	00								
2465	1D36	CC	CC	CC	78	30	30	DB	0CCH,0CCH,0CCH,078H,030H,030H,078H,000H	D_59	Y
2466		78	00								
2467	1D3E	FE	C6	8C	18	32	66	DB	0FEH,0C6H,08CH,018H,032H,066H,0FEH,000H	D_5A	Z
2468		FE	00								
2469	1D46	78	60	60	60	60	60	DB	078H,060H,060H,060H,060H,060H,078H,000H	D_5B	[
2470		78	00								LEFT BRACKET
2471	1D4E	C0	60	30	18	0C	06	DB	0C0H,060H,030H,018H,00CH,006H,002H,000H	D_5C	\
2472		02	00								BACKSLASH
2473	1D56	78	18	18	18	18		DB	078H,018H,018H,018H,018H,018H,078H,000H	D_5D	]
2474		78	00								RIGHT BRACKET
2475	1D5E	10	38	C6	C6	00	00	DB	010H,038H,06CH,0C6H,000H,000H,000H,000H	D_5E	^
2476		00	00								CIRCUMFLEX
2477	1D66	00	00	00	00	00	00	DB	000H,000H,000H,000H,000H,000H,00FH,000H	D_5F	_
2478		00	FF								UNDERSCORE
2479											
2480	1D6E	30	30	18	00	00	00	DB	030H,030H,018H,000H,000H,000H,000H,000H	D_60	'
2481		00	00								APOSTROPHE REV
2482	1D76	00	00	78	0C	7C	CC	DB	000H,000H,078H,00CH,07CH,0CCH,076H,000H	D_61	!
2483		76	00								
2484	1D7E	E0	60	7C	C6	66	66	DB	0E0H,060H,060H,07CH,066H,066H,0DCH,000H	D_62	b
2485		DC	00								
2486	1D86	00	00	78	CC	C0	CC	DB	000H,000H,078H,0CCH,0CCH,0CCH,078H,000H	D_63	c
2487		78	00								
2488	1D8E	1C	0C	0C	7C	CC	CC	DB	01CH,00CH,00CH,07CH,0CCH,0CCH,076H,000H	D_64	d
2489		78	00								
2490	1D96	00	00	78	CC	FC	C0	DB	000H,000H,078H,0CCH,0FCH,0C0H,078H,000H	D_65	e
2491		78	00								
2492	1D9E	38	60	F0	F0	60	60	DB	038H,06CH,060H,0F0H,060H,060H,0F0H,000H	D_66	f
2493		F0	00								
2494	1DA6	00	00	76	CC	CC	7C	DB	000H,000H,076H,0CCH,0CCH,07CH,00CH,0F8H	D_67	g
2495		0C	78								
2496	1DAE	E0	60	6C	76	66	66	DB	0E0H,060H,06CH,076H,066H,066H,0E6H,000H	D_68	h
2497		E6	00								
2498	1DB6	30	00	70	30	30	30	DB	030H,000H,070H,030H,030H,030H,078H,000H	D_69	i
2499		78	00								
2500	1DBE	0C	00	0C	0C	0C	CC	DB	00CH,000H,00CH,00CH,00CH,0CCH,0CCH,078H	D_6A	j
2501		CC	78								
2502	1DC6	E0	60	66	6C	78	6C	DB	0E0H,060H,066H,06CH,078H,06CH,0E6H,000H	D_6B	k

2503	E6 00		
2504	1DCE 70 30 30 30 30	DB	070H,030H,030H,030H,030H,078H,000H ; D_6C l
2505	78 00		
2506	1DD6 00 00 CC FE FE D6	DB	000H,000H,0CCH,0FEH,0FEH,0D6H,0C6H,000H ; D_6D m
2507	C6 00		
2508	1DDE 00 00 F8 CC CC CC	DB	000H,000H,0F8H,0CCH,0CCH,0CCH,0CCH,000H ; D_6E n
2509	CC 00		
2510	1DE6 00 00 78 CC CC CC	DB	000H,000H,078H,0CCH,0CCH,0CCH,078H,000H ; D_6F o
2511	78 00		
2512			
2513	1DEE 00 00 DC 66 66 7C	DB	000H,000H,0DCH,066H,066H,07CH,060H,0F0H ; D_70 p
2514	60 F0		
2515	1DF6 00 00 76 CC CC 7C	DB	000H,000H,076H,0CCH,0CCH,07CH,00CH,01EH ; D_71 q
2516	0C 1E		
2517	1DFE 00 00 DC 76 66 60	DB	000H,000H,0DCH,076H,066H,060H,0F0H,000H ; D_72 r
2518	F0 00		
2519	1E06 00 00 7C C0 78 0C	DB	000H,000H,07CH,0C0H,078H,00CH,0F8H,000H ; D_73 s
2520	F8 00		
2521	1E0E 10 30 7C 30 30 34	DB	010H,030H,07CH,030H,030H,034H,018H,000H ; D_74 t
2522	18 00		
2523	1E16 00 00 CC CC CC CC	DB	000H,000H,0CCH,0CCH,0CCH,0CCH,0CCH,000H ; D_75 u
2524	76 00		
2525	1E1E 00 00 CC CC CC 78	DB	000H,000H,0CCH,0CCH,0CCH,078H,030H,000H ; D_76 v
2526	30 00		
2527	1E26 00 00 C6 D6 FE FE	DB	000H,000H,0C6H,0D6H,0FEH,0FEH,06CH,000H ; D_77 w
2528	6C 00		
2529	1E2E 00 00 C6 6C 38 6C	DB	000H,000H,0C6H,06CH,038H,06CH,0C6H,000H ; D_78 x
2530	C6 00		
2531	1E36 00 00 CC CC CC 7C	DB	000H,000H,0CCH,0CCH,0CCH,07CH,00CH,0F8H ; D_79 y
2532	0C F8		
2533	1E3E 00 00 FC 98 30 64	DB	000H,000H,0FCH,098H,030H,064H,0FCH,000H ; D_7A z
2534	FC 00		
2535	1E46 1C 30 30 E0 30 30	DB	01CH,030H,030H,0E0H,030H,030H,01CH,000H ; D_7B { LEFT BRACE
2536	1C 00		
2537	1E4E 18 18 18 00 18 18	DB	018H,018H,018H,000H,018H,018H,018H,000H ; D_7C BROKEN STROKE
2538	18 00		
2539	1E56 E0 30 30 1C 30 30	DB	0E0H,030H,030H,01CH,030H,030H,0E0H,000H ; D_7D } RIGHT BRACE
2540	E0 00		
2541	1E5E 76 DC 00 00 00 00	DB	076H,0DCH,000H,000H,000H,000H,000H,000H ; D_7E ~ TILDE
2542	00 00		
2543	1E66 00 10 38 6C C6 C6	DB	000H,010H,038H,06CH,0C6H,0C6H,0FEH,000H ; D_7F DELTA
2544	FE 00		

```

2545 PAGE
2546 ;--- INT 1A -----
2547 ; TIME OF DAY
2548 ; THIS ROUTINE ALLOWS THE CLOCK TO BE SET/READ
2549 ;
2550 ; INPUT
2551 ; (AH) = 0 READ THE CURRENT CLOCK SETTING
2552 ; RETURNS CX = HIGH PORTION OF COUNT
2553 ; RETURNS DX = LOW PORTION OF COUNT
2554 ; AL = 0 IF TIMER HAS NOT PASSED
2555 ; 24 HOURS SINCE LAST READ
2556 ; >0 IF ON ANOTHER DAY
2557 ; (AH) = 1 SET THE CURRENT CLOCK
2558 ; CX = HIGH PORTION OF COUNT
2559 ; DX = LOW PORTION OF COUNT
2560 ; NOTE: COUNTS OCCUR AT THE RATE OF
2561 ; 119210/65536 COUNTS/SEC
2562 ; (OR ABOUT 18.2 PER SECOND -- SEE EQUATES BELOW)
2563 ;-----
2564 ; ASSUME CS:CODE,DS:DATA
2565 ; ORG 0FE6H
2566 IE6E ORG 01E6H
2567 IE6E TIME_OF_DAY:
2568 IE6E E9 0995 R JMP TIME_OF_DAY_11
2569
2570 ; ORG 0FEA5H
2571 IEA5 ORG 01EA5H
2572 IEA5 TIMER_INT:
2573 IEA5 E9 1A1A R JMP TIMER_INT_1
2574
2575 ;-----
2576 ; THESE ARE THE VECTORS WHICH ARE MOVED INTO
2577 ; THE 8086 INTERRUPT AREA DURING POWER ON.
2578 ; ONLY THE OFFSETS ARE DISPLAYED HERE, CODE
2579 ; SEGMENT WILL BE ADDED FOR ALL OF THEM, EXCEPT
2580 ; WHERE NOTED.
2581 ;-----
2582 ; ASSUME CS:CODE
2583 ; ORG 0FEF3H
2584 IEF3 ORG 01EF3H
2585 IEF3 VECTOR_TABLE:
2586 IEF3 LABEL WORD ; VECTOR TABLE VALUES FOR POST TESTS
2587 IEF3 DW OFFSET TIMER_INT ; INT 08H - HARDWARE TIMER 0 IRQ 0
2588 IEF3 DW OFFSET KB_INT ; INT 09H - KEYBOARD IRQ 1
2589 IEF3 DW OFFSET DT1 ; INT 0AH - IRQ 2
2590 IEF3 DW OFFSET D11 ; INT 0BH - IRQ 3
2591 IEF3 DW OFFSET D11 ; INT 0CH - IRQ 4
2592 IEF3 DW OFFSET D11 ; INT 0DH - IRQ 5
2593 IEF3 DW OFFSET DISK_INT ; INT 0EH - DISKETTE IRQ 6
2594 IEF3 DW OFFSET D11 ; INT 0FH - IRQ 7
2595
2596 ;----- SOFTWARE INTERRUPTS ( BIOS CALLS AND POINTERS ) -----
2597 IF03 1065 R DW OFFSET VIDEO_IO ; INT 10H -- VIDEO DISPLAY
2598 IF05 1840 R DW OFFSET EQUIPMENT ; INT 11H -- GET EQUIPMENT FLAG WORD
2599 IF07 1841 R DW OFFSET MEMORY_SIZE_DET ; INT 12H -- GET REAL MODE MEMORY SIZE
2600 IF09 0C59 R DW OFFSET DISKETTE_IO ; INT 13H -- DISKETTE
2601 IF0B 0739 R DW OFFSET RS232_IO ; INT 14H -- COMMUNICATION ADAPTER
2602 IF0D 1859 R DW OFFSET CASSETTE_IO ; INT 15H -- EXPANDED BIOS FUNCTION CALL
2603 IF0F 082E R DW OFFSET KEYBOARD_IO ; INT 16H -- KEYBOARD INPUT
2604 IF11 0FD2 R DW OFFSET PRINTER_IO ; INT 17H -- PRINTER OUTPUT
2605 IF13 0000 DW 0000H ; INT 18H -- 0F60H INSERTED FOR BASIC
2606 IF15 06F2 R DW OFFSET BOOT_STRAP ; INT 19H -- BOOT FROM SYSTEM MEDIA
2607 IF17 1E6E R DW OFFSET TIME_OF_DAY ; INT 1AH -- TIME OF DAY
2608 IF19 1F49 R DW OFFSET DUMMY_RETURN ; INT 1BH -- KEYBOARD BREAK ADDRESS
2609 IF1B 1F49 R DW OFFSET DUMMY_RETURN ; INT 1CH -- TIMER BREAK ADDRESS
2610 IF1D 10A4 R DW OFFSET VIDEO_PARAMS ; INT 1DH -- VIDEO PARAMETERS
2611 IF1F 0FC7 R DW OFFSET DISK_BASE ; INT 1EH -- DISKETTE PARAMETERS
2612 IF21 0000 DW 0000H ; INT 1FH -- POINTER TO VIDEO EXTENSION
2613
2614 ;-----
2615 ; TEMPORARY INTERRUPT SERVICE ROUTINE
2616 ; 1. THIS ROUTINE IS ALSO LEFT IN PLACE AFTER THE
2617 ; POWER ON DIAGNOSTICS TO SERVICE UNUSED
2618 ; INTERRUPT VECTORS. LOCATION 'INTR_FLAG' WILL
2619 ; CONTAIN EITHER: 1. LEVEL OF HARDWARE INT. THAT
2620 ; CAUSED CODE TO BE EXEC.
2621 ; 2. 'FF' FOR NON-HARDWARE INTERRUPTS THAT WAS
2622 ; EXECUTED ACCIDENTLY.
2623 ;-----
2624 IF23 PROC NEAR
2625 ASSUME DS:DATA
2626 PUSH DS
2627 IF24 E8 1A12 R CALL DOS ; SAVE REG AX CONTENTS
2628 IF27 50 PUSH AX ; READ IN-SERVICE REG
2629 IF28 B0 08 MOV AL,0BH ; (FIND OUT WHAT LEVEL BEING
2630 IF2A E6 20 OUT INTA00,AL ; SERVICED)
2631 IF2C 90 NOP ; GET LEVEL
2632 IF2D E4 20 IN AL,INTA00 ; SAVE IT
2633 IF2F 8A E0 MOV AH,AL ; 00? (NO HARDWARE ISR ACTIVE)
2634 IF31 0A C4 OR AL,AH
2635 IF33 75 04 JNZ HW_INT
2636 IF35 B4 FF MOV AH,0FFH
2637 IF37 EB 0A JMP SHORT SET_INTR_FLAG ; SET FLAG TO FF IF NON-HOWARE
2638 IF39
2639 IF39 E4 21 IN AL,INTA01 ; GET MASK VALUE
2640 IF3B 0A C4 OR AL,AH ; MASK OFF LVL BEING SERVICED
2641 IF3D E6 21 OUT INTA01,AL
2642 IF3F B0 20 MOV AL,E01
2643 IF41 E6 20 OUT INTA00,AL
2644 IF43
2645 IF43 88 26 006B R MOV @INTR_FLAG,AH ; SET FLAG
2646 IF47 58 POP AX ; RESTORE REG AX CONTENTS
2647 IF48 1F POP DS
2648 IF49
2649 IF49 CF DUMMY_RETURN: ; NEED IRET FOR VECTOR TABLE
2650 IF4A IRET
2651 IF4B ENDP
2652
2653 ;-----
2654 ; DUMMY RETURN FOR ADDRESS COMPATIBILITY
2655 ;
2656 IF53 ORG 0FF53H
2657 IF53 CF ORG 01F53H
2658 IF53 IRET

```

```

2658 PAGE
2659 ---- INT 05H -----
2660 PRINT_SCREEN
2661 THIS LOGIC WILL BE INVOKED BY INTERRUPT 05H TO PRINT THE SCREEN.
2662 THE CURSOR POSITION AT THE TIME THIS ROUTINE IS INVOKED WILL BE
2663 SAVED AND RESTORED UPON COMPLETION. THE ROUTINE IS INTENDED TO
2664 RUN WITH INTERRUPTS ENABLED. IF A SUBSEQUENT PRINT_SCREEN KEY
2665 IS DEPRESSED WHILE THIS ROUTINE IS PRINTING IT WILL BE IGNORED.
2666 THE BASE PRINTERS STATUS IS CHECKED FOR NOT BUSY AND NOT OUT OF
2667 PAPER. AN INITIAL STATUS ERROR WILL ABEND THE PRINT REQUEST.
2668 ADDRESS 0050:0000 CONTAINS THE STATUS OF THE PRINT SCREEN.
2669
2670 50:0 = 0 PRINT SCREEN HAS NOT BEEN CALLED OR UPON RETURN
2671 FROM A CALL THIS INDICATES A SUCCESSFUL OPERATION.
2672 = 1 PRINT SCREEN IS IN PROGRESS IGNORE THIS REQUEST.
2673 = 255 ERROR ENCOUNTERED DURING PRINTING.
2674
2675 ORG OFF54H
2676 IF54 ORG 01F54H
2677
2678 PRINT_SCREEN_1 PROC FAR
2679
2680 IF54 IE PUSH DS ; DELAY INTERRUPT ENABLE TILL FLAG SET
2681 IF55 E8 1A12 R CALL DDS ; USE 0040:0100 FOR STATUS AREA STORAGE
2682 IF56 80 3E 0100 R 01 CMP *STATUS_BYTE,1 ; GET STATUS_BYTE DATA SEGMENT
2683 IF5D 74 7C JE PRI10 ; SEE IF PRINT ALREADY IN PROGRESS
2684 IF5F C6 06 0100 R 01 MOV *STATUS_BYTE,1 ; EXIT IF PRINT ALREADY IN PROGRESS
2685 IF64 F8 STI ; INDICATE PRINT NOW IN PROGRESS
2686 IF65 50 PUSH AX ; MUST RUN WITH INTERRUPTS ENABLED
2687 IF66 53 PUSH BX ; SAVE WORK REGISTERS
2688 IF67 51 PUSH CX
2689 IF68 52 PUSH DX
2690 IF69 B4 0F MOV AH,0FH
2691 IF6B CD 10 INT 10H ; WILL REQUEST THE CURRENT SCREEN MODE
2692 ; (AL)= MODE
2693 ; (AH)= NUMBER COLUMNS/LINE
2694 IF6D 8A CC MOV CL,AH ; (BH)= VISUAL PAGE
2695 IF6F 8A 2E 0084 R MOV CH,*ROWS ; WILL MAKE USE OF (CX) REGISTER TO
2696 IF73 FE C5 INC CH ; CONTROL ROWS ON SCREEN & COLUMNS
2697 ; ADJUST ROWS ON DISPLAY COUNT
2698 ; (CL)= NUMBER COLUMNS/LINE
2699 ; (CH)= NUMBER OF ROWS ON DISPLAY
2700
2701 ;-----
2702 AT THIS POINT WE KNOW THE COLUMNS/LINE COUNT IS IN (CL) ;
2703 AND THE NUMBER OF ROWS ON THE DISPLAY IS IN (CH) ;
2704 THE PAGE IF APPLICABLE IS IN (BH). THE STACK HAS ;
2705 (DS),(AX),(BX),(CX),(DX) PUSHED. ;
2706
2707 IF75 33 D2 XOR DX,DX ; FIRST PRINTER
2708 IF77 B4 02 MOV AH,02H ; SET PRINTER STATUS REQUEST COMMAND
2709 IF79 CD 17 INT 17H ; REQUEST CURRENT PRINTER STATUS
2710 IF7B F8 74 80 XOR AH,080H ; CHECK FOR PRINTER BUSY (NOT CONNECTED)
2711 IF7E F6 C4 A0 TEST AH,0A0H ; OR OUT OF PAPER
2712 IF81 75 4E JNZ PR170 ; ERROR EXIT IF PRINTER STATUS ERROR
2713
2714 IF83 E8 1FDD R CALL CRLF ; CARRIAGE RETURN LINE FEED TO PRINTER
2715 IF86 51 PUSH CX ; SAVE SCREEN BOUNDS
2716 IF87 B4 03 MOV AH,03H ; NOW READ THE CURRENT CURSOR POSITION
2717 IF89 CD 10 INT 10H ; AND RESTORE AT END OF ROUTINE
2718 IF8B 59 POP CX ; RECALL SCREEN BOUNDS
2719 IF8C 52 PUSH DX ; PRESERVE THE ORIGINAL POSITION
2720 IF8D 33 D2 XOR DX,DX ; INITIAL CURSOR (0,0) AND FIRST PRINTER
2721
2722 ;-----
2723 THIS LOOP IS TO READ EACH CURSOR POSITION FROM THE ;
2724 SCREEN AND PRINT IT. (BH)= VISUAL PAGE (CH)= ROWS ;
2725
2726 IF8F PR110: IF8F MOV AH,02H ; INDICATE CURSOR SET REQUEST
2727 IF91 CD 10 INT 10H ; NEW CURSOR POSITION ESTABLISHED
2728 IF93 B4 08 MOV AH,08H ; INDICATE READ CHARACTER FROM DISPLAY
2729 IF95 CD 10 INT 10H ; CHARACTER NOW IN (AL)
2730 IF97 0A C0 OR AL,AL ; SEE IF VALID CHAR
2731 IF99 75 02 JNZ PR120 ; JUMP IF VALID CHAR
2732 IF9B 90 D0 MOV AL,' ' ; ELSE MAKE IT A BLANK
2733
2734 IF9D 52 PR120: IF9D PUSH DX ; SAVE CURSOR POSITION
2735 IF9E 33 D2 XOR DX,DX ; INDICATE FIRST PRINTER (DX= 0)
2736 IFA0 32 E4 XOR AH,AH ; INDICATE PRINT CHARACTER IN (AL)
2737 IFA2 CD 17 INT 17H ; PRINT THE CHARACTER
2738 IFA4 5A POP DX ; RECALL CURSOR POSITION
2739 IFA5 F6 C4 29 TEST AH,29H ; TEST FOR PRINTER ERROR
2740 IFA8 75 22 JNZ PR160 ; EXIT IF ERROR DETECTED
2741 IFAA FE C2 INC DL ; ADVANCE TO NEXT COLUMN
2742 IFAC 3A CA CMP CL,DL ; SEE IF AT END OF LINE
2743 IFAE 75 0F JNZ PR110 ; IF NOT LOOP FOR NEXT COLUMN
2744 IFB0 32 D2 XOR DL,DL ; BACK TO COLUMN 0
2745 IFB2 B4 E2 MOV AH,DL ; (AH)=0
2746 IFB4 52 PUSH DX ; SAVE NEW CURSOR POSITION
2747 IFB5 E8 1FDD R CALL CRLF ; LINE FEED CARRIAGE RETURN
2748 IFB8 5A POP DX ; RECALL CURSOR POSITION
2749 IFB9 FE C6 INC DH ; ADVANCE TO NEXT LINE
2750 IFBB 3A EE CMP CH,DH ; FINISHED?
2751 IFBD 75 D0 JNZ PR110 ; IF NOT LOOP FOR NEXT LINE
2752
2753 IFBF 5A POP DX ; GET CURSOR POSITION
2754 IFC0 B4 02 MOV AH,02H ; INDICATE REQUEST CURSOR SET
2755 IFC2 CD 10 INT 10H ; CURSOR POSITION RESTORED
2756 IFC4 F4 00 CLD ; BLOCK INTERRUPTS TILL STACK CLEARED
2757 IFC5 C6 06 0100 R 00 MOV *STATUS_BYTE,0 ; MOVE OK RESULTS FLAG TO STATUS_BYTE
2758 IFCA EB 0B JMP SHORT PR180 ; EXIT PRINTER ROUTINE

```

```

2757                                     PAGE
2758 IFCC                                     PR160:                                     ;
2759 IFCC 5A                                POP     DX                                     ; ERROR EXIT
2760 IFCD B4 02                            MOV     AH,02H                               ; GET CURSOR POSITION
2761 IFCF CD 10                            INT     10H                               ; INDICATE REQUEST CURSOR SET
2762 IFD1                                     PR170:                                     ; CURSOR POSITION RESTORED
2763 IFD1 FA                                CLI                                     ; BLOCK INTERRUPTS TILL STACK CLEARED
2764 IFD2 C6 06 0100 R FF                  MOV     *STATUS_BYTE,0FFH                ; SET ERROR FLAG
2765 IFD7                                     PR180:                                     ;
2766 IFD7 5A                                POP     DX                                     ; EXIT ROUTINE
2767 IFD8 59                                POP     CX                                     ; RESTORE ALL THE REGISTERS USED
2768 IFD9 58                                POP     BX
2769 IFDA 58                                POP     AX
2770 IFDB                                     PR190:                                     ;
2771 IFDB 1F                                POP     DS                                     ; ROUTINE BUSY EXIT
2772 IFDC CF                                IRET                                        ; RETURN WITH INITIAL INTERRUPT MASK
2773 IFDD                                     PRINT_SCREEN_1 ENOP
2774                                     ;----- CARRIAGE RETURN, LINE FEED SUBROUTINE
2775
2776                                     CRLF PROC NEAR
2777 IFDD                                     ;
2778                                     ; SEND CR,LF TO FIRST PRINTER
2779 IFDD 33 D2                            XOR     DX,DX                               ; ASSUME FIRST PRINTER (DX=0)
2780 IFDE B8 000D                          MOV     AX,CR                               ; GET THE PRINT CHARACTER COMMAND AND
2781 IFEE CD 17                            INT     17H                               ; THE CARRIAGE RETURN CHARACTER
2782 IFEF B8 000A                          MOV     AX,LF                               ; NOW GET THE LINE FEED AND
2783 IFEE CD 17                            INT     17H                               ; SEND IT TO THE BIOS PRINTER ROUTINE
2784 IFEE C3                                RET
2785 IFEA                                     CRLF ENOP
2786
2787                                     ;-----
2788                                     ; POWER ON RESET VECTOR ;
2789                                     ;-----
2790                                     ;
2791 IFFO                                     ORG     0FFFF0H
2792                                     ORG     01FF0H
2793
2794                                     ;----- POWER ON RESET
2795 IFFO P_O_R LABEL FAR
2796 IFFO EA                                DB     0EAH
2797 IFFF E05B                              DW     0E05BH
2798 IFFF F000                              DW     0F000H
2799 IFFF 30 31 2F 31 30 2F                DB     '01/10/86'
2800                                     ; RELEASE MARKER
2801
2802                                     ;
2803 IFFE                                     ORG     0FFFEH
2804 IFFE                                     ORG     01FFFEH
2805 IFFE FB                                MODEL:  DB     MODEL_BYTE
2806
2807 IFFF                                     CODE  ENDS
2808                                     END
  
```

Notes:



System BIOS Listing - 11/8/82

Quick Reference - 64/256K Board

Equates	5-113
8088 Interrupt 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:




```

1 $TITLE(BIOS FOR THE IBM PERSONAL COMPUTER XT)
2
3 -----
4 THE BIOS ROUTINES ARE MEANT TO BE ACCESSED THROUGH
5 SOFTWARE INTERRUPTS ONLY. ANY ADDRESSES PRESENT IN
6 THE LISTINGS ARE INCLUDED ONLY FOR COMPLETENESS.
7 NOT FOR REFERENCE. APPLICATIONS WHICH REFERENCE
8 ABSOLUTE ADDRESSES WITHIN THE CODE SEGMENT
9 VIOLATE THE STRUCTURE AND DESIGN OF BIOS.
10 -----
11
12 -----
13 EQUATES
14 -----
15 PORT_A EQU 60H ; 8255 PORT A ADDR
16 PORT_B EQU 61H ; 8255 PORT B ADDR
17 PORT_C EQU 62H ; 8255 PORT C ADDR
18 CMD_PORT EQU 63H
19 INTA00 EQU 20H ; 8259 PORT
20 INTA01 EQU 21H ; 8259 PORT
21 EOI EQU 20H
22 TIMER EQU 40H
23 TIM_CTL EQU 43H ; 8253 TIMER CONTROL PORT ADDR
24 TIMERO EQU 40H ; 8253 TIMER/COUNTER 0 PORT ADDR
25 TINT EQU 0 ; TIMER 0 INTR RECYD MASK
26 DMA08 EQU 08 ; DMA STATUS REG PORT ADDR
27 DMA EQU 00 ; DMA CH.0 ADDR. REG PORT ADDR
28 MAX_PERIOD EQU 540H
29 MIN_PERIOD EQU 410H
30 KBD_IN EQU 60H ; KEYBOARD DATA IN ADDR PORT
31 KBDINT EQU 02 ; KEYBOARD INTR MASK
32 KB_DATA EQU 60H ; KEYBOARD SCAN CODE PORT
33 KB_CTL EQU 61H ; CONTROL BITS FOR KEYBOARD SENSE DATA
34
35 -----
36 8088 INTERRUPT LOCATIONS
37 -----
38
39 ABS0 SEGMENT AT 0
40 STG_LOCO LABEL BYTE
41 ORG 2*4
42 NMI_PTR LABEL WORD
43 ORG 5*4
44 INT5_PTR LABEL WORD
45 ORG 8*4
46 INT_ADDR LABEL WORD
47 INT_PTR LABEL DWORD
48 ORG 10H*4
49 VIDEO_INT LABEL WORD
50 ORG 10H*4
51 PARM_PTR LABEL DWORD ; POINTER TO VIDEO PARMS
52 ORG 10H*4
53 BASIC_PTR LABEL WORD ; ENTRY POINT FOR CASSETTE BASIC
54 ORG 01EH*4 ; INTERRUPT IEH
55 DISK_POINTER LABEL DWORD
56 ORG 01FH*4 ; LOCATION OF POINTER
57 EXT_PTR LABEL DWORD ; POINTER TO EXTENSION
58 ORG 400H
59 DATA_AREA LABEL BYTE ; ABSOLUTE LOCATION OF DATA SEGMENT
60 DATA_WORD LABEL WORD
61 ORG 0500H
62 MFG_TEST_RTN LABEL FAR
63 ORG TC00H
64 BOOT_LOCN LABEL FAR
65 ABS0 ENDS
66
67 -----
68 ; STACK -- USED DURING INITIALIZATION ONLY
69 -----
70
71 STACK SEGMENT AT 30H
72 DW 128 DUP(?)
73
74 TOS LABEL WORD
75 STACK ENDS
76
77 -----
78 ROM BIOS DATA AREAS
79 -----
80
81 DATA SEGMENT AT 40H
82 RS232_BASE DW 4 DUP(?) ; ADDRESSES OF RS232 ADAPTERS
83
84 PRINTER_BASE DW 4 DUP(?) ; ADDRESSES OF PRINTERS
85
86 EQUIP_FLAG DW ? ; INSTALLED HARDWARE
87 MFG_TST DB ? ; INITIALIZATION FLAG
88 MEMORY_SIZE DW ? ; MEMORY SIZE IN K BYTES
89 MFG_ERR_FLAG DB ? ; SCRATCHPAD FOR MANUFACTURING
90 DB ? ; ERROR CODES
91
92 -----
93 KEYBOARD DATA AREAS
94 -----
95
96 KB_FLAG DB ?
97
98 -----
99 SHIFT FLAG EQUATES WITHIN KB_FLAG
100 -----
101
102 INS_STATE EQU 80H ; INSERT STATE IS ACTIVE
103 CAPS_STATE EQU 40H ; CAPS LOCK STATE HAS BEEN TOGGLED
104 NUM_STATE EQU 20H ; NUM LOCK STATE HAS BEEN TOGGLED
105 SCROLL_STATE EQU 10H ; SCROLL LOCK STATE HAS BEEN TOGGLED
106 ALT_SHIFT EQU 08H ; ALTERNATE SHIFT KEY DEPRESSED
107 CTL_SHIFT EQU 04H ; CONTROL SHIFT KEY DEPRESSED
108 LEFT_SHIFT EQU 02H ; LEFT SHIFT KEY DEPRESSED
109 RIGHT_SHIFT EQU 01H ; RIGHT SHIFT KEY DEPRESSED
110

```

```

LOC OBJECT          LINE  SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82

0018 ??            106  KB_FLAG_1      DB      ?          ; SECOND BYTE OF KEYBOARD STATUS
                   107
0080              108  INS_SHIFT     EQU    80H        ; INSERT KEY IS DEPRESSED
0040              109  CAPS_SHIFT    EQU    40H        ; CAPS LOCK KEY IS DEPRESSED
0020              110  NUM_SHIFT     EQU    20H        ; NUM LOCK KEY IS DEPRESSED
0010              111  SCROLL_SHIFT  EQU    10H        ; SCROLL LOCK KEY IS DEPRESSED
0008              112  HOLD_STATE    EQU    08H        ; SUSPEND KEY HAS BEEN TOGGLED
                   113
0019 ??            114  ALT_INPUT     DB      ?          ; STORAGE FOR ALTERNATE KEYPAD ENTRY
001A ????          115  BUFFER_HEAD  DW      ?          ; POINTER TO HEAD OF KEYBOARD BUFFER
001C ????          116  BUFFER_TAIL  DW      ?          ; POINTER TO TAIL OF KEYBOARD BUFFER
001E (16 ????      117  KB_BUFFER     DW      16 DUP(?)    ; ROOM FOR 15 ENTRIES
                   )

003E              118  KB_BUFFER_END LABEL WORD
                   119
                   120  ;----- HEAD = TAIL INDICATES THAT THE BUFFER IS EMPTY
                   121
0045              122  NUM_KEY       EQU    69          ; SCAN CODE FOR NUMBER LOCK
0046              123  SCROLL_KEY    EQU    70          ; SCROLL LOCK KEY
0038              124  ALT_KEY       EQU    56          ; ALTERNATE SHIFT KEY SCAN CODE
001D              125  CTL_KEY       EQU    29          ; SCAN CODE FOR CONTROL KEY
003A              126  CAPS_KEY     EQU    58          ; SCAN CODE FOR SHIFT LOCK
002A              127  LEFT_KEY     EQU    42          ; SCAN CODE FOR LEFT SHIFT
0036              128  RIGHT_KEY    EQU    54          ; SCAN CODE FOR RIGHT SHIFT
0052              129  INS_KEY      EQU    82          ; SCAN CODE FOR INSERT KEY
0053              130  DEL_KEY      EQU    83          ; SCAN CODE FOR DELETE KEY
                   131
                   132  ;-----
                   133  ; DISKETTE DATA AREAS
                   134  ;-----

003E ??            135  SEEK_STATUS  DB      ?          ; DRIVE RECALIBRATION STATUS
                   136                      ; BIT 3-0 = DRIVE 3-0 NEEDS RECAL
                   137                      ; BEFORE NEXT SEEK IF BIT 15 = 0
0080              138
003F ??            139  INT_FLAG      EQU    080H        ; INTERRUPT OCCURRENCE FLAG
                   140  MOTOR_STATUS DB      ?          ; MOTOR STATUS
                   141                      ; BIT 3-0 = DRIVE 3-0 IS CURRENTLY
                   142                      ; RUNNING
                   143                      ; BIT 7 = CURRENT OPERATION IS A WRITE,
                   144                      ; REQUIRES DELAY
                   145
0040 ??            146  MOTOR_COUNT  DB      ?          ; TIME OUT COUNTER FOR DRIVE TURN OFF
0025              147  MOTOR_WAIT   EQU    37          ; 2 SECS OF COUNTS FOR MOTOR TURN OFF
                   148
0041 ??            149  DISKETTE_STATUS DB      ?          ; RETURN CODE STATUS BYTE
0080              150  TIME_OUT     EQU    80H        ; ATTACHMENT FAILURE TO RESPOND
0040              151  BAD_SEEK     EQU    40H        ; SEEK OPERATION FAILED
0020              152  BAD_NEC      EQU    20H        ; NEC CONTROLLER HAS FAILED
0010              153  BAD_CRC      EQU    10H        ; BAD CRC ON DISKETTE READ
0009              154  DMA_BOUNDARY EQU    09H        ; ATTEMPT TO DMA ACROSS 64K BOUNDARY
0008              155  BAD_DMA      EQU    08H        ; DMA OVERRUN ON OPERATION
0004              156  RECORD_NOT_FND EQU    04H        ; REQUESTED SECTOR NOT FOUND
0003              157  WRITE_PROTECT EQU    03H        ; WRITE ATTEMPTED ON WRITE PROT DISK
0002              158  BAD_ADDR_MARK EQU    02H        ; ADDRESS MARK NOT FOUND
0001              159  BAD_CMD      EQU    01H        ; BAD COMMAND PASSED TO DISKETTE I/O
                   160
0042 ( ?           161  NEC_STATUS  DB      7 DUP(?)    ; STATUS BYTES FROM NEC
??              )
                   )

                   162
                   163  ;-----
                   164  ; VIDEO DISPLAY DATA AREA
                   165  ;-----

0049 ??            166  CRT_MODE     DB      ?          ; CURRENT CRT MODE
004A ????          167  CRT_COLS    DW      ?          ; NUMBER OF COLUMNS ON SCREEN
004C ????          168  CRT_LEN     DW      ?          ; LENGTH OF REGEN IN BYTES
004E ????          169  CRT_START   DW      ?          ; STARTING ADDRESS IN REGEN BUFFER
0050 (8 ????      170  CURSOR_POSN  DW      8 DUP(?)    ; CURSOR FOR EACH OF UP TO 8 PAGES
                   )

0060 ????          171  CURSOR_MODE  DW      ?          ; CURRENT CURSOR MODE SETTING
0062 ??            172  ACTIVE_PAGE  DB      ?          ; CURRENT PAGE BEING DISPLAYED
0063 ????          173  ADDR_6845   DW      ?          ; BASE ADDRESS FOR ACTIVE DISPLAY CARD
0065 ??            174  CRT_MODE_SET DB      ?          ; CURRENT SETTING OF THE 3X8 REGISTER
0066 ??            175  CRT_PALETTE  DB      ?          ; CURRENT PALETTE SETTING COLOR CARD
                   176
                   177  ;-----
                   178  ; POST DATA AREA
                   179  ;-----

0067 ????          180  IO_ROM_INIT  DW      ?          ; PNTR TO OPTIONAL I/O ROM INIT ROUTINE
0069 ????          181  IO_ROM_SEG  DW      ?          ; POINTER TO IO ROM SEGMENT
006B ??            182  INTR_FLAG   DB      ?          ; FLAG TO INDICATE AN INTERRUPT HAPPEND
                   183
                   184  ;-----
                   185  ; TIMER DATA AREA
                   186  ;-----

006C ????          187  TIMER_LOW   DW      ?          ; LOW WORD OF TIMER COUNT
006E ????          188  TIMER_HIGH  DW      ?          ; HIGH WORD OF TIMER COUNT
0070 ??            189  TIMER_OFL   DB      ?          ; TIMER HAS ROLLED OVER SINCE LAST READ
                   190  ; COUNTS_SEC EQU    18
                   191  ; COUNTS_MIN EQU    1092
                   192  ; COUNTS_HOUR EQU    65543
                   193  ; COUNTS_DAY EQU    1573040 = 1800B0H
                   194
                   195  ;-----
                   196  ; SYSTEM DATA AREA
                   197  ;-----

0071 ??            198  BIOS_BREAK  DB      ?          ; BIT 7=1 IF BREAK KEY HAS BEEN HIT
0072 ????          199  RESET_FLAG  DW      ?          ; WORDS=1234H IF KEYBOARD RESET UNDERWAY
                   200
                   201  ;-----
                   202  ; FIXED DISK DATA AREAS
                   203  ;-----

0074 ????          204                      DW      ?
0076 ????          205                      DW      ?
                   206
                   207  ;-----
                   208  ; PRINTER AND RS232 TIME-OUT VARIABLES
                   209  ;-----

0078 (4 ?          208  PRINT_TIM_OUT DB      4 DUP(?)
??              )
007C (4 ?          209  RS232_TIM_OUT DB      4 DUP(?)
??              )

```

```

LOC OBJECT          LINE SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82

0080 777?          210 ;-----
0082 777?          211 ; ADDITIONAL KEYBOARD DATA AREA :
-----          212 ;
213 BUFFER_START   DW    ?
214 BUFFER_END     DW    ?
215 DATA          ENDS
-----          216 ;-----
217 ; EXTRA DATA AREA :
218 ;-----
0000 ??            219 XDATA SEGMENT AT 50H
-----          220 STATUS_BYTE DB    ?
221 XDATA ENDS
-----          222 ;-----
223 ; VIDEO DISPLAY BUFFER :
224 ;-----
225 VIDEO_RAM       SEGMENT AT 0B00H
226 REGEN LABEL     BYTE
0000 0000          227 REGENW LABEL     WORD
0000 16384         228 DB    16384 DUP(?)
??
-----          229 VIDEO_RAM ENDS
230 ;-----
231 ; ROM RESIDENT CODE :
232 ;-----
0000 (57344       233 CODE SEGMENT AT 0F00H
??              234 DB    57344 DUP(?) ; FILL LOWEST 56K
-----          235 ;-----
E000 31353031353132 236 DB    '1501512 COPR. IBM 1982' ; COPYRIGHT NOTICE
20434F5052E20
49424D0313938
32
-----          237 ;-----
238 ;-----
239 ; INITIAL RELIABILITY TESTS -- PHASE I :
240 ;-----
241 ;
242 ; ASSUME CS:CODE,SS:CODE,ES:AB50,DS:DATA
243 ;-----
244 ;
245 ; DATA DEFINITIONS :
246 ;-----
247 ;-----
E016 D7E0          249 C1 DW    C11 ; RETURN ADDRESS
E018 TEE1          250 C2 DW    C24 ; RETURN ADDRESS FOR DUMMY STACK
-----          251 ;
E01A 204B42204F4B 252 F3B DB    ' KB OK',13 ; KB FOR MEMORY SIZE
E020 0D
-----          253 ;-----
254 ;
255 ; LOAD A BLOCK OF TEST CODE THROUGH THE KEYBOARD PORT :
256 ; FOR MANUFACTURING TEST. :
257 ; THIS ROUTINE WILL LOAD A TEST (MAX LENGTH=FAFFH) THROUGH :
258 ; THE KEYBOARD PORT. CODE WILL BE LOADED AT LOCATION :
259 ; 000010500. AFTER LOADING, CONTROL WILL BE TRANSFERRED :
260 ; TO LOCATION 000010500. STACK WILL BE LOCATED JUST BELOW :
261 ; THE TEST CODE. THIS ROUTINE ASSUMES THAT THE FIRST 2 :
262 ; BYTES TRANSFERRED CONTAIN THE COUNT OF BYTES TO BE LOADED :
263 ; (BYTE 1=COUNT LOW, BYTE 2=COUNT HI.) :
264 ;-----
265 ;-----
266 ;----- FIRST, GET THE COUNT
267 ;-----
E021 MFG_BOOT:      268
E021 E8131A        269 CALL SP_TEST ; GET COUNT LOW
E024 8AFB          270 MOV BH,BL ; SAVE IT
E026 E80E1A        271 CALL SP_TEST ; GET COUNT HI
E029 8AEB          272 MOV CH,BL ;
E02B 8ACF          273 MOV CL,BH ; CX NOW HAS COUNT
E02D FC           274 CLD ; SET DIR. FLAG TO INCRIMENT
E02E FA           275 CLI ;
E02F BF0005        276 MOV DI,0500H ; SET TARGET OFFSET (DS=00001)
E032 B0FD          277 MOV AL,0FDH ; UNMASK K/B INTERRUPT
E034 E621          278 OUT INTA01,AL ;
E036 B00A          279 MOV AL,0AH ; SEND READ INT. REQUEST REG. CMD
E038 E620          280 OUT INTA00,AL ;
E03A BA6100        281 MOV DX,61H ; SET UP PORT B ADDRESS
E03D BBCC4C        282 MOV BX,4CCCH ; CONTROL BITS FOR PORT B
E040 B402          283 MOV AH,02H ; K/B REQUEST PENDING MASK
E042
284 TST:
E042 8AC3          285 MOV AL,BL ;
E044 EE           286 OUT DX,AL ; TOGGLE K/B CLOCK
E045 8AC7          287 MOV AL,BH ;
E047 EE           288 OUT DX,AL ;
E048 4A           289 DEC DX ; POINT DX AT ADDR. 60 (KB DATA)
E049
290 TST1:
E049 E420          291 IN AL,INTA00 ; GET IRR REG
E04B 22C4          292 AND AL,AH ; KB REQUEST PENDING?
E04D T4FA          293 JZ TST1 ; LOOP TILL DATA PRESENT
E04F EC           294 IN AL,DX ; GET DATA
E050 AA           295 STOSB ; STORE IT
E051 42           296 INC DX ; POINT DX BACK AT PORT B (61)
E052 E2EE          297 LOOP TST ; LOOP TILL ALL BYTES READ
-----          298 ;
E054 EA00050000    299 JMP MFG_TEST_RTN ; FAR JUMP TO CODE THAT WAS JUST
300 ; LOADED
301

```

LINE SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82

SECTION 5

LOC	OBJECT	LINE	SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82
E18E	2BC0	531	CLR_STG:
E190	F3	532	SUB AX,AX
E191	AB	533	REP STOSW ; MAKE AX=0000
E192		534	HOW_BIG:
E192	891E7204	535	MOV DATA_WORD[OFFSET RESET_FLAG],BX ; RESTORE RESET FLAG
E196	BA0004	536	MOV DX,0400H ; SET POINTER TO JUST>16KB
E199	BB1000	537	MOV BX,16 ; BASIC COUNT OF 16K
E19C	8EC2	538	FILL_LOOP:
E19C	8EC2	539	MOV ES,DX ; SET SEG. REG.
E19E	2BFF	540	SUB DI,DI
E1A0	B855AA	541	MOV AX,0AA55H ; TEST PATTERN
E1A3	8BC8	542	MOV CX,AX ; SEND PATTERN
E1A5	268905	543	MOV ES:[DI],AX ; SEND PATTERN TO MEM.
E1A8	B00F	544	MOV AL,0FH ; PUT SOMETHING IN AL
E1AA	268B05	545	MOV AX,ES:[DI] ; GET PATTERN
E1AD	33C1	546	XOR AX,CX ; COMPARE PATTERNS
E1AF	7511	547	JNZ HOW_BIG_END ; GO END IF NO COMPARE
E1B1	B90020	548	MOV CX,2000H ; SET COUNT FOR 8K WORDS
E1B4	F3	549	REP STOSW ; FILL 8K WORDS
E1B5	AB		
E1B6	81C20004	550	ADD DX,400H ; POINT TO NEXT 16KB BLOCK
E1BA	83C310	551	ADD BX,16 ; BUMP COUNT BY 16KB
E1BD	80FEA0	552	CMP DH,0A0H ; TOP OF RAM AREA YET? (A0000)
E1C0	750A	553	JNZ FILL_LOOP
E1C2		554	HOW_BIG_END:
E1C2	891E1304	555	MOV DATA_WORD[OFFSET MEMORY_SIZE],BX ; SAVE MEMORY SIZE
E1C6	B83000	556	-----
E1C9	8E00	557	SETUP STACK SEG AND SP
E1CB	BC0001	558	
		559	MOV AX,STACK
		560	MOV SS,AX ; GET STACK VALUE
		561	MOV SP,OFFSET TOS ; SET THE STACK UP
		562	----- ; STACK IS READY TO GO
		563	INITIALIZE THE 8259 INTERRUPT CONTROLLER CHIP ;
		564	-----
E1CE	B013	565	MOV AL,13H
E1D0	E620	566	OUT INTA0,AL ; ICW1 - EDGE, SNGL, ICW4
E1D2	B008	567	MOV AL,8
E1D4	E621	568	OUT INTA0,AL ; SETUP ICW2 - INT TYPE 8 (8-F)
E1D6	B009	569	MOV AL,9
E1D8	E621	570	OUT INTA0,AL ; SETUP ICW4 - BUFFRD,8086 MODE
E1DA	B0FF	571	MOV AL,0FFH
E1DC	E621	572	OUT INTA0,AL ; MASK ALL INTS. OFF
		573	----- ; (VIDEO ROUTINE ENABLES INTS.)
		574	SET UP THE INTERRUPT VECTORS TO TEMP INTERRUPT
		575	
E1DE	1E	576	PUSH DS
E1DF	B92000	577	MOV CX,32
E1E2	2BFF	578	SUB DI,DI
E1E4	8EC7	579	MOV ES,DI
E1E6	B823FF	580	MOV AX,OFFSET D11
E1E9	AB	581	MOV AL,9
E1EA	8CC8	582	MOV AX,CS
E1EC	AB	583	STOSW ; GET ADDR OF INTR PROC SEG
E1ED	E2F7	584	LOOP D3 ; VECTBL0
		585	
		586	ESTABLISH BIOS SUBROUTINE CALL INTERRUPT VECTORS
E1EF	BF4000	587	MOV DI,OFFSET VIDEO_INT
E1F2	0E	588	PUSH CS ; SETUP ADDR TO INTR AREA
E1F3	1F	589	POP DS
E1F4	8CDB	590	MOV AX,DS
E1F6	BE03FF90	591	MOV SI,OFFSET VECTOR_TABLE+16
E1FA	B91000	592	MOV CX,16 ; START WITH VIDEO ENTRY
E1FD	A5	593	MOVSV
E1FE	47	594	INC DI
E1FF	47	595	INC DI
E200	E2FB	596	LOOP D3A ; MOVE VECTOR TABLE TO RAM
		597	----- ; SKIP SEGMENT POINTER
		598	-----
		599	DETERMINE CONFIGURATION AND MFG. MODE ;
		600	-----
E202	1F	601	POP DS
E203	1E	602	MOV DI,OFFSET VIDEO_INT
E204	E462	603	IN AL,PORT_C
E206	240F	604	AND AL,00001111B
E208	8AE0	605	MOV AH,AL
E20A	B0AD	606	MOV AL,10101101B
E20C	E661	607	OUT PORT_B,AL
E20E	90	608	NOP
E20F	E462	609	IN AL,PORT_C
E211	B104	610	CL,4
E213	D2C0	611	ROL AL,CL
E215	24F0	612	AND AL,11110000B
E217	0AC4	613	OR AL,AH
E219	2AE4	614	SUB AH,AH
E21B	A31004	615	MOV DATA_WORD[OFFSET EQUIP_FLAG],AX
E21E	B099	616	MOV AL,99H
E220	E663	617	CMD,PORT,AL
E222	E80518	618	CALL KBD_RESET
E225	80FBAA	619	CMP BL,0AAH
E228	7418	620	JE E6
E22A	80FB65	621	BL,065H
E22D	7503	622	JNE D3B
E22F	E9EFFF	623	JMP MFG_BOOT
E232	B036	624	MOV AL,36H
E234	E661	625	OUT PORT_B,AL
E236	90	626	NOP
E237	90	627	NOP
E238	E460	628	IN AL,PORT_A
E23A	24FF	629	AND AL,0FFH
E23C	7504	630	JNZ E6
E23E	FE061204	631	MOV DATA_AREA[OFFSET MFG_TST],AX
		632	----- ; SET MANUFACTURING TEST FLAG
		633	-----

```

634 :-----:
635 : INITIALIZE AND START CRT CONTROLLER (6845) :
636 : TEST VIDEO READ/WRITE STORAGE. :
637 : DESCRIPTION :
638 : RESET THE VIDEO ENABLE SIGNAL. :
639 : SELECT ALPHANUMERIC MODE, 40 * 25, B & W. :
640 : READ/WRITE DATA PATTERNS TO STG. CHECK STG :
641 : ADDRESSABILITY. :
642 : ERROR = 1 LONG AND 2 SHORT BEEPS :
643 :-----:
E242 E6:
E242 A11004 644 MOV AX,DATA_WORD[OFFSET EQUIP_FLAG] ; GET SENSE SWITCH INFO
E245 50 645 PUSH AX ; SAVE IT
E246 B030 646 MOV AL,30H
E248 A31004 647 MOV DATA_WORD[OFFSET EQUIP_FLAG],AX
E24B 2AE4 648 SUB AH,AH
E24D CD10 649 INT 10H ; SEND INIT TO B/W CARD
E24F B020 650 MOV AL,20H
E251 A31004 651 MOV DATA_WORD[OFFSET EQUIP_FLAG],AX
E254 2AE4 652 SUB AH,AH ; AND INIT COLOR CARD
E256 CD10 653 INT 10H
E258 58 654 POP AX
E259 A31004 655 MOV DATA_WORD[OFFSET EQUIP_FLAG],AX ; RECOVER REAL SWITCH INFO
656 : AND CONTINUE
E25C 2430 657 AND AL,30H
E25E 750A 658 JNZ E7
E260 BF4000 659 MOV DI,OFFSET VIDEO_INT ; ISOLATE VIDEO SWS
E263 C7054BFF 660 MOV [DI],OFFSET DUMMY_RETURN ; VIDEO SWS SET TO 0?
E267 E9A000 661 JMP E1B_1 ; SET INT 10H TO DUMMY
662 : BYPASS VIDEO TEST
E26A E7:
E26A 3C30 663 CMP AL,30H ; TEST VIDEO:
E26C 7408 664 JE E8 ; B/W CARD ATTACHED?
E26E FEC4 665 JNC AH ; YES - SET MODE FOR B/W CARD
E270 3C20 666 CMP AL,20H ; SET COLOR MODE FOR COLOR CD
E272 7502 667 JNE E8 ; 80X25 MODE SELECTED?
E274 B403 668 MOV AH,3 ; NO - SET MODE FOR 40X25
E276 86E0 669 XCHG AH,AL ; SET MODE:
E278 50 670 PUSH AX ; SAVE VIDEO MODE ON STACK
E279 2AE4 671 SUB AH,AH ; INITIALIZE TO ALPHANUMERIC MD
E27B CD10 672 INT 10H ; CALL VIDEO IO
E27D 58 673 POP AX ; RESTORE VIDEO SENSE SWS IN AH
E27E 50 674 PUSH AX ; RESAVE VALUE
E27F B800B0 675 MOV BX,0B000H ; BEG VIDEO RAM ADDR B/W CD
E282 BAB003 676 MOV CX,3B8H ; MODE REG FOR B/W
E285 B90008 677 MOV CX,2048 ; RAM WORD CNT FOR B/W CD
E288 B001 678 MOV AL,1 ; SET MODE FOR BW CARD
E28A 80FC30 679 CMDB AH,30H ; B/W VIDEO CARD ATTACHED?
E28D 7409 680 JE E9 ; YES - GO TEST VIDEO STG
E28F B7B8 681 MOV BH,0BBH ; BEG VIDEO RAM ADDR COLOR CD
E291 B4D803 682 MOV DX,3DBH ; MODE REG FOR COLOR CD
E294 B520 683 MOV CH,20H ; RAM WORD CNT FOR COLOR CD
E296 FEC8 684 DEC AL ; SET MODE TO 0 FOR COLOR CD
E298 E9:
E298 EE 685 OUT DX,AL ; TEST VIDEO STG:
E299 813E72043412 686 MOV DATA_WORD[OFFSET RESET_FLAG],234H ; DISABLE VIDEO FOR COLOR CD
E29F 8EC3 687 OR AX,E8 ; POD INIT BY KB0 RESET?
E2A1 7407 688 JE E10 ; POINT ES TO VIDEO RAM STG
E2A3 8ED8 689 MOV DS,BX ; YES - SKIP VIDEO RAM STG
E2A5 E8C703 690 ASSUME DS:NOTHING,ES:NOTHING ; POINT DS TO VIDEO RAM STG
E2A8 7546 691 CALL STGTST_CNT ; GO TEST VIDEO R/W STG
692 JNE E17 ; R/W STG FAILURE - BEEP SPK
693 :-----:
694 : SETUP VIDEO DATA ON SCREEN FOR VIDEO :
695 : LINE TEST. :
696 : DESCRIPTION :
697 : ENABLE VIDEO SIGNAL AND SET MODE. :
698 : DISPLAY A HORIZONTAL BAR ON SCREEN. :
699 :-----:
E2AA E10:
E2AA 58 700 POP AX ; GET VIDEO SENSE SWS (AH)
E2AB 50 701 PUSH AX ; SAVE IT
E2AC B400 702 MOV AH,0 ; ENABLE VIDEO AND SET MODE
E2AE CD10 703 INT 10H ; VIDEO
E2B0 B82070 704 MOV AX,7020H ; WRT BLANKS IN REVERSE VIDEO
705 :-----:
706 : UNNATURAL ACT FOR ADDRESS COMPATIBILITY :
707 :-----:
E2B3 EB11 708 JMP SHORT E10A
E2C3 709 ORG 0E2C3H
E2C3 E99915 710 JMP NM1_INT
711 :-----:
E2C6 E10A:
E2C6 2BFF 712 SUB DI,D1 ; SETUP STARTING LOC
E2C8 B92800 713 MOV CX,40 ; NO. OF BLANKS TO DISPLAY
E2CB F3 714 REP STOSW ; WRITE VIDEO STORAGE
715 :-----:
716 : CRT INTERFACE LINES TEST :
717 : DESCRIPTION :
718 : SENSE ON/OFF TRANSITION OF THE :
719 : VIDEO ENABLE AND HORIZONTAL :
720 : SYNC LINES. :
721 :-----:
E2CD 58 722 POP AX ; GET VIDEO SENSE SW INFO
E2CE 50 723 PUSH AX ; SAVE IT
E2CF 80FC30 724 MOV AH,30H ; B/W CARD ATTACHED?
E2D2 BABA03 725 MOV DX,03BAH ; SETUP ADDR OF BW STATUS PORT
E2D5 7403 726 JE E11 ; YES - GO TEST LINES
E2D7 BADA03 727 MOV DX,03DAH ; COLOR CARD IS ATTACHED
E2DA E11:
E2DA B408 728 MOV AH,8 ; LINE TEST:
E2DC E2C:
E2DC 2BC9 729 SUB CX,CX ; OFLOOP_CNT:
E2DE E3:
E2DE EC 730 IN AL,DX ; READ CRT STATUS PORT
E2DF 22C4 731 AND AL,AH ; CHECK VIDEO/HORZ LINE
E2E1 7504 732 JNZ E14 ; ITS ON - CHECK IF IT GOES OFF
E2E3 E2F9 733 LOOP E13 ; LOOP TILL ON OR TIMEOUT
E2E5 EB09 734 JMP SHORT E17 ; GO PRINT ERROR MSG
E2E7 E14:
E2E7 2BC9 735 SUB CX,CX
E2E9 E15:
E2E9 EC 736 IN AL,DX ; READ CRT STATUS PORT
E2EA 22C4 737 AND AL,AH ; CHECK VIDEO/HORZ LINE
E2EC 7411 738 JZ E16 ; ITS ON - CHECK NEXT LINE
E2EE E2F9 739 LOOP E15 ; LOOP IF OFF TILL IT GOES ON

```



```

861 ;----- SETUP TIMER 0 TO MODE 3
862
863 MOV AL,OFFH ; DISABLE ALL DEVICE INTERRUPTS
864 OUT INTA01,AL
865 MOV AL,36H ; SEL TIM 0,LSB,MSB,MODE 3
866 OUT TIMER+3,AL ; WRITE TIMER MODE REG
867 MOV AL,0
868 OUT TIMER,AL ; WRITE LSB TO TIMER 0 REG
869 OUT TIMER,AL ; WRITE MSB TO TIMER 0 REG
870 ;-----
871 ; KEYBOARD TEST
872 ; DESCRIPTION
873 ; RESET THE KEYBOARD AND CHECK THAT SCAN
874 ; CODE "AA" IS RETURNED TO THE CPU.
875 ; CHECK FOR STUCK KEYS.
876 ;-----
877 TST12:
878 MOV AL,99H ; SET 8255 MODE A,C=IN B=OUT
879 OUT CMD_PORT,AL
880 MOV AL,DATA_AREA[OFFSET EQUIP_FLAG]
881 AND AL,01 ; TEST CHAMBER?
882 ; BYPASS IF SO
883 CMP DATA_AREA[OFFSET MFG_TST],1 ; MANUFACTURING TEST MODE?
884 JE F7 ; YES - SKIP KEYBOARD TEST
885 CALL KBD_RESET ; ISSUE RESET TO KEYBRD
886 JCXZ F6 ; PRINT ERR MSG IF NO INTERRUPT
887 MOV AL,49H ; ENABLE KEYBOARD
888 OUT PORT_B,AL
889 CMP BL,0AAH ; SCAN CODE AS EXPECTED?
890 JNE F6 ; NO - DISPLAY ERROR MSG
891
892 ;----- CHECK FOR STUCK KEYS
893
894 MOV AL,0CBH ; CLR KBD, SET CLK LINE HIGH
895 OUT PORT_B,AL
896 MOV AL,48H ; ENABLE KBD,CLK IN NEXT BYTE
897 OUT PORT_B,AL
898 SUB CX,CX
899 F5: LOOP F5 ; KBD WAIT:
900 IN AL,KBD_IN ; DELAY FOR A WHILE
901 CMP AL,0 ; CHECK FOR STUCK KEYS
902 JE F7 ; SCAN CODE = 0?
903 CALL XPC_BYTE ; YES - CONTINUE TESTING
904 ; CONVERT AND PRINT
905 F6: MOV SI,OFFSET F1 ; GET MSG ADDR
906 CALL E_MSG ; PRINT MSG ON SCREEN
907 ;-----
908 ; SETUP HARDWARE INT. VECTOR TABLE
909 ;-----
910 F7: PUSH DS ; SETUP_INT_TABLE:
911 SUB AX,AX
912 MOV ES,AX
913 MOV CX,08 ; GET VECTOR CNT
914 PUSH CS ; SETUP DS SEG REG
915 POP DS
916 MOV SI,OFFSET VECTOR_TABLE
917 MOV DI,OFFSET INT_PTR
918 F7A: MOVSW
919 INC DI ; SKIP OVER SEGMENT
920 INC DI
921 LOOP F7A
922 POP DS
923
924 ;----- SET UP OTHER INTERRUPTS AS NECESSARY
925
926 MOV NMI_PTR,OFFSET NMI_INT ; NMI INTERRUPT
927 MOV INT5_PTR,OFFSET PRINT_SCREEN ; PRINT SCREEN
928 MOV BASIC_PTR+2,0F600H ; SEGMENT FOR CASSETTE BASIC
929
930 ;----- SETUP TIMER 0 TO BLINK LED IF MANUFACTURING TEST MODE
931
932 CMP DATA_AREA[OFFSET MFG_TST],01H ; MFG. TEST MODE?
933 JNZ EXP_TO
934 MOV WORD PTR(1CH*4),OFFSET BLINK_INT ; SETUP TIMER INTR TO BLINK LED
935 MOV AL,OFFH ; ENABLE TIMER INTERRUPT
936 OUT INTA01,AL

```

```

940 -----
941 : EXPANSION I/O BOX TEST :
942 : CHECK TO SEE IF EXPANSION BOX PRESENT - IF INSTALLED, :
943 : TEST DATA AND ADDRESS BUSES TO I/O BOX :
944 : ERROR="1801" :
945 -----
946
947 :----- DETERMINE IF BOX IS PRESENT
948
E418 949 EXP_IO: ; ICARD WAS ENABLED EARLIER)
E418 BA1002 950 MOV DX,0210H ; CONTROL PORT ADDRESS
E418 B85555 951 MOV AX,5555H ; SET DATA PATTERN
E41E EE 952 OUT DX,AL
E41F B001 953 MOV AL,01H ; MAKE AL DIFFERENT
E421 EC 954 IN AL,DX ; RECOVER DATA
E422 3AC4 955 CMP AL,AH ; REPLY?
E424 7544 956 JNE E19 ; NO RESPONSE, GO TO NEXT TEST
E426 F700 957 NOT AX ; MAKE DATA=AAAA
E428 EC 958 OUT DX,AL
E429 B001 959 MOV AL,01H
E42B EC 960 IN AL,DX ; RECOVER DATA
E42C 3AC4 961 CMP AL,AH
E42E 753A 962 JNE E19
963
964 :----- CHECK ADDRESS BUS
965
E430 966 EXP2:
E430 BB0100 967 MOV BX,0001H
E433 BA1502 968 MOV DX,0215H ; LOAD HI ADDR. REG ADDRESS
E436 B91000 969 MOV CX,0016 ; GO ACROSS 16 BITS
E439 970 EXP3:
E439 2E8807 971 MOV CS:[BX],AL ; WRITE ADDRESS F0000+BX
E43C 90 972 NOP
E43D EC 973 IN AL,DX ; READ ADDR. HIGH
E43E 3AC7 974 CMP AL,BH
E440 7521 975 JNE EXP_ERR ; GO ERROR IF MISCOMPARE
E442 42 976 INC DX ; DX=216H (ADDR. LOW REG)
E443 EC 977 IN AL,DX
E444 3AC3 978 CMP AL,BL ; COMPARE TO LOW ADDRESS
E446 751B 979 JNE EXP_ERR
E448 4A 980 DEC DX ; DX BACK TO 215H
E449 D1E3 981 SHL BX,1
E44B E2EC 982 LOOP EXP3 ; LOOP TILL 'I' WALKS ACROSS BX
983
984 :----- CHECK DATA BUS
985
E44D B90800 986 MOV CX,0008 ; DO 8 TIMES
E450 B001 987 MOV AL,01
E452 4A 988 DEC DX ; MAKE DX=214H (DATA BUS REG)
E453 989 EXP4:
E453 8AE0 990 MOV AH,AL
E455 EE 991 OUT DX,AL ; SAVE DATA BUS VALUE
E456 B001 992 MOV AL,01H ; SEND VALUE TO REG
E45B EC 993 IN AL,DX ; RETRIEVE VALUE FROM REG
E459 3AC4 994 CMP AL,AH ; = TO SAVED VALUE
E45B 7506 995 JNE SHORT_EXP_ERR
E45D D0E0 996 SHL AL,1 ; FORM NEW DATA PATTERN
E45F E2F2 997 LOOP EXP4 ; LOOP TILL BIT WALKS ACROSS AL
E461 EB07 998 JMP SHORT E19 ; GO ON TO NEXT TEST
E463 999 EXP_ERR:
E463 BE0FF990 1000 MOV SI,OFFSET F3C
E467 E83F15 1001 CALL E_M5G

```

```

1002 :-----
1003 : ADDITIONAL READ/WRITE STORAGE TEST :
1004 : DESCRIPTION :
1005 : WRITE/READ DATA PATTERNS TO ANY READ/WRITE :
1006 : STORAGE AFTER THE FIRST 32K. STORAGE :
1007 : ADDRESSABILITY IS CHECKED. :
1008 :-----
1009 : ASSUME DS:DATA
1010
E46A E19: 1010
E46A E8EC15 1011 CALL DD5
E46D 1E 1012 PUSH DS
E46E 1013
E46E 813E72003412 1014 CMP RESET_FLAG,1234H ; WARM START?
E474 7503 1015 JNE E20A ; CONTINUE TEST IF NOT
E476 E99F00 1016 JMP ROM_SCAN ; GO TO NEXT ROUTINE IF SO
E479 1017
E479 B81000 1018 MOV AX,16 ; STARTING AMT. OF MEMORY OK
E47C EB28 1019 JMP SHORT PRT_SIZ ; POST MESSAGE
E47E 1020
E47E 8B1E1300 1021 MOV BX,MEMORY_SIZE ; GET MEM. SIZE WORD
E482 83EB10 1022 SUB BX,16 ; 1ST 16K ALREADY DONE
E485 B104 1023 MOV CX,040H
E487 D3EB 1024 SHR CX,CL
E489 8BCB 1025 MOV CL,BX ; DIVIDE BY 16
E48B BB0004 1026 MOV BX,0400H ; SAVE COUNT OF 16K BLOCKS
E48E 1027
E48E 8EDB 1028 MOV DS,BX ; SET PTR. TO RAM SEGMENT>16K
E490 8EC3 1029 MOV ES,BX ; SET SEG. REG
E492 81C30004 1030 ADD BX,0400H ; POINT TO NEXT 16K
E496 52 1031 PUSH DX
E497 51 1032 PUSH CX ; SAVE WORK REGS
E498 53 1033 PUSH BX
E499 50 1034 PUSH AX
E49A B90020 1035 MOV CX,2000H ; SET COUNT FOR 8K WORDS
E49D E8CF01 1036 CALL STGST_CNT
E4A0 754C 1037 JNZ E21A ; GO PRINT ERROR
E4A2 58 1038 POP AX ; RECOVER TESTED MEM NUMBER
E4A3 051000 1039 ADD AX,16
E4A6 1040
E4A6 50 1041 PRT_SIZ: PUSH AX
E4A7 BB0A00 1042 MOV BX,10 ; SET UP FOR DECIMAL CONVERT
E4AA B90300 1043 MOV CX,3 ; OF 3 NIBBLES
E4AD 1044
E4AD 33D2 1045 XOR DX,DX
E4AF F7F3 1046 DIV BX ; DIVIDE BY 10
E4B1 80CA30 1047 OR DL,30H ; MAKE INTO ASCII
E4B4 52 1048 PUSH DX ; SAVE
E4B5 E2F6 1049 LOOP DECIMAL_LOOP
E4B7 B90300 1050 MOV CX,3
E4BA 1051
E4BA 58 1052 PRT_DEC_LOOP: POP AX ; RECOVER A NUMBER
E4BB E8DE14 1053 CALL PRT_HEX
E4BE E2FA 1054 LOOP PRT_DEC_LOOP
E4C0 B90700 1055 MOV CX,7
E4C3 BE1AE0 1056 MOV SI,OFFSET F3B ; PRINT ' KB OK'
E4C6 1057
E4C6 2E8A04 1058 MOV AL,CS:[SI]
E4C9 46 1059 INC SI
E4CA E8CF14 1060 CALL PRT_HEX
E4CD E2F7 1061 LOOP KB_LOOP
E4CF 58 1062 POP AX ; RECOVER WORK REGS
E4D0 3D1000 1063 CMP AX,16 ; FIRST PASS?
E4D3 74A9 1064 JE E20B
E4D5 5B 1065 POP BX
E4D6 59 1066 POP CX
E4D7 5A 1067 POP DX
E4D8 E2B4 1068 LOOP E21 ; LOOP TILL ALL MEM. CHECKED
E4DA B00A 1069 MOV AL,10
E4DC E8BD14 1070 CALL PRT_HEX ; LINE FEED
E4DE 1071
E4DF 1072
E4DF 1073
E4DF 1074
E4DF 1075
E4DF 1076
E4DF 1077
E4DF 1078
E4DF 1079
E4DF 1080
E4DF 1081
E4DF 1082
E4DF 1083
E4DF 1084
E4DF 1085
E4DF 1086
E4DF 1087
E4DF 1088
E4DF 1089
E4DF 1090
E4DF 1091
E4DF 1092
E4DF 1093
E4DF 1094
E4DF 1095
E4DF 1096
E4DF 1097
E4DF 1098
E4DF 1099
E4DF 1100
E4DF 1101
E4DF 1102
E4DF 1103
E4DF 1104
E4DF 1105
E4DF 1106
E4DF 1107
E4DF 1108
E4DF 1109
E4DF 1110
E4DF 1111
E4DF 1112
E4DF 1113
E4DF 1114
E4DF 1115
E4DF 1116
E4DF 1117
E4DF 1118
E4DF 1119
E4DF 1120
E4DF 1121
E4DF 1122
E4DF 1123
E4DF 1124
E4DF 1125
E4DF 1126
E4DF 1127
E4DF 1128
E4DF 1129
E4DF 1130
E4DF 1131
E4DF 1132
E4DF 1133
E4DF 1134
E4DF 1135
E4DF 1136
E4DF 1137
E4DF 1138
E4DF 1139
E4DF 1140
E4DF 1141
E4DF 1142
E4DF 1143
E4DF 1144
E4DF 1145
E4DF 1146
E4DF 1147
E4DF 1148
E4DF 1149
E4DF 1150
E4DF 1151
E4DF 1152
E4DF 1153
E4DF 1154
E4DF 1155
E4DF 1156
E4DF 1157
E4DF 1158
E4DF 1159
E4DF 1160
E4DF 1161
E4DF 1162
E4DF 1163
E4DF 1164
E4DF 1165
E4DF 1166
E4DF 1167
E4DF 1168
E4DF 1169
E4DF 1170
E4DF 1171
E4DF 1172
E4DF 1173
E4DF 1174
E4DF 1175
E4DF 1176
E4DF 1177
E4DF 1178
E4DF 1179
E4DF 1180
E4DF 1181
E4DF 1182
E4DF 1183
E4DF 1184
E4DF 1185
E4DF 1186
E4DF 1187
E4DF 1188
E4DF 1189
E4DF 1190
E4DF 1191
E4DF 1192
E4DF 1193
E4DF 1194
E4DF 1195
E4DF 1196
E4DF 1197
E4DF 1198
E4DF 1199
E4DF 1200
E4DF 1201
E4DF 1202
E4DF 1203
E4DF 1204
E4DF 1205
E4DF 1206
E4DF 1207
E4DF 1208
E4DF 1209
E4DF 1210
E4DF 1211
E4DF 1212
E4DF 1213
E4DF 1214
E4DF 1215
E4DF 1216
E4DF 1217
E4DF 1218
E4DF 1219
E4DF 1220
E4DF 1221
E4DF 1222
E4DF 1223
E4DF 1224
E4DF 1225
E4DF 1226
E4DF 1227
E4DF 1228
E4DF 1229
E4DF 1230
E4DF 1231
E4DF 1232
E4DF 1233
E4DF 1234
E4DF 1235
E4DF 1236
E4DF 1237
E4DF 1238
E4DF 1239
E4DF 1240
E4DF 1241
E4DF 1242
E4DF 1243
E4DF 1244
E4DF 1245
E4DF 1246
E4DF 1247
E4DF 1248
E4DF 1249
E4DF 1250
E4DF 1251
E4DF 1252
E4DF 1253
E4DF 1254
E4DF 1255
E4DF 1256
E4DF 1257
E4DF 1258
E4DF 1259
E4DF 1260
E4DF 1261
E4DF 1262
E4DF 1263
E4DF 1264
E4DF 1265
E4DF 1266
E4DF 1267
E4DF 1268
E4DF 1269
E4DF 1270
E4DF 1271
E4DF 1272
E4DF 1273
E4DF 1274
E4DF 1275
E4DF 1276
E4DF 1277
E4DF 1278
E4DF 1279
E4DF 1280
E4DF 1281
E4DF 1282
E4DF 1283
E4DF 1284
E4DF 1285
E4DF 1286
E4DF 1287
E4DF 1288
E4DF 1289
E4DF 1290
E4DF 1291
E4DF 1292
E4DF 1293
E4DF 1294
E4DF 1295
E4DF 1296
E4DF 1297
E4DF 1298
E4DF 1299
E4DF 1300
E4DF 1301
E4DF 1302
E4DF 1303
E4DF 1304
E4DF 1305
E4DF 1306
E4DF 1307
E4DF 1308
E4DF 1309
E4DF 1310
E4DF 1311
E4DF 1312
E4DF 1313
E4DF 1314
E4DF 1315
E4DF 1316
E4DF 1317
E4DF 1318
E4DF 1319
E4DF 1320
E4DF 1321
E4DF 1322
E4DF 1323
E4DF 1324
E4DF 1325
E4DF 1326
E4DF 1327
E4DF 1328
E4DF 1329
E4DF 1330
E4DF 1331
E4DF 1332
E4DF 1333
E4DF 1334
E4DF 1335
E4DF 1336
E4DF 1337
E4DF 1338
E4DF 1339
E4DF 1340
E4DF 1341
E4DF 1342
E4DF 1343
E4DF 1344
E4DF 1345
E4DF 1346
E4DF 1347
E4DF 1348
E4DF 1349
E4DF 1350
E4DF 1351
E4DF 1352
E4DF 1353
E4DF 1354
E4DF 1355
E4DF 1356
E4DF 1357
E4DF 1358
E4DF 1359
E4DF 1360
E4DF 1361
E4DF 1362
E4DF 1363
E4DF 1364
E4DF 1365
E4DF 1366
E4DF 1367
E4DF 1368
E4DF 1369
E4DF 1370
E4DF 1371
E4DF 1372
E4DF 1373
E4DF 1374
E4DF 1375
E4DF 1376
E4DF 1377
E4DF 1378
E4DF 1379
E4DF 1380
E4DF 1381
E4DF 1382
E4DF 1383
E4DF 1384
E4DF 1385
E4DF 1386
E4DF 1387
E4DF 1388
E4DF 1389
E4DF 1390
E4DF 1391
E4DF 1392
E4DF 1393
E4DF 1394
E4DF 1395
E4DF 1396
E4DF 1397
E4DF 1398
E4DF 1399
E4DF 1400
E4DF 1401
E4DF 1402
E4DF 1403
E4DF 1404
E4DF 1405
E4DF 1406
E4DF 1407
E4DF 1408
E4DF 1409
E4DF 1410
E4DF 1411
E4DF 1412
E4DF 1413
E4DF 1414
E4DF 1415
E4DF 1416
E4DF 1417
E4DF 1418
E4DF 1419
E4DF 1420
E4DF 1421
E4DF 1422
E4DF 1423
E4DF 1424
E4DF 1425
E4DF 1426
E4DF 1427
E4DF 1428
E4DF 1429
E4DF 1430
E4DF 1431
E4DF 1432
E4DF 1433
E4DF 1434
E4DF 1435
E4DF 1436
E4DF 1437
E4DF 1438
E4DF 1439
E4DF 1440
E4DF 1441
E4DF 1442
E4DF 1443
E4DF 1444
E4DF 1445
E4DF 1446
E4DF 1447
E4DF 1448
E4DF 1449
E4DF 1450
E4DF 1451
E4DF 1452
E4DF 1453
E4DF 1454
E4DF 1455
E4DF 1456
E4DF 1457
E4DF 1458
E4DF 1459
E4DF 1460
E4DF 1461
E4DF 1462
E4DF 1463
E4DF 1464
E4DF 1465
E4DF 1466
E4DF 1467
E4DF 1468
E4DF 1469
E4DF 1470
E4DF 1471
E4DF 1472
E4DF 1473
E4DF 1474
E4DF 1475
E4DF 1476
E4DF 1477
E4DF 1478
E4DF 1479
E4DF 1480
E4DF 1481
E4DF 1482
E4DF 1483
E4DF 1484
E4DF 1485
E4DF 1486
E4DF 1487
E4DF 1488
E4DF 1489
E4DF 1490
E4DF 1491
E4DF 1492
E4DF 1493
E4DF 1494
E4DF 1495
E4DF 1496
E4DF 1497
E4DF 1498
E4DF 1499
E4DF 1500
E4DF 1501
E4DF 1502
E4DF 1503
E4DF 1504
E4DF 1505
E4DF 1506
E4DF 1507
E4DF 1508
E4DF 1509
E4DF 1510
E4DF 1511
E4DF 1512
E4DF 1513
E4DF 1514
E4DF 1515
E4DF 1516
E4DF 1517
E4DF 1518
E4DF 1519
E4DF 1520
E4DF 1521
E4DF 1522
E4DF 1523
E4DF 1524
E4DF 1525
E4DF 1526
E4DF 1527
E4DF 1528
E4DF 1529
E4DF 1530
E4DF 1531
E4DF 1532
E4DF 1533
E4DF 1534
E4DF 1535
E4DF 1536
E4DF 1537
E4DF 1538
E4DF 1539
E4DF 1540
E4DF 1541
E4DF 1542
E4DF 1543
E4DF 1544
E4DF 1545
E4DF 1546
E4DF 1547
E4DF 1548
E4DF 1549
E4DF 1550
E4DF 1551
E4DF 1552
E4DF 1553
E4DF 1554
E4DF 1555
E4DF 1556
E4DF 1557
E4DF 1558
E4DF 1559
E4DF 1560
E4DF 1561
E4DF 1562
E4DF 1563
E4DF 1564
E4DF 1565
E4DF 1566
E4DF 1567
E4DF 1568
E4DF 1569
E4DF 1570
E4DF 1571
E4DF 1572
E4DF 1573
E4DF 1574
E4DF 1575
E4DF 1576
E4DF 1577
E4DF 1578
E4DF 1579
E4DF 1580
E4DF 1581
E4DF 1582
E4DF 1583
E4DF 1584
E4DF 1585
E4DF 1586
E4DF 1587
E4DF 1588
E4DF 1589
E4DF 1590
E4DF 1591
E4DF 1592
E4DF 1593
E4DF 1594
E4DF 1595
E4DF 1596
E4DF 1597
E4DF 1598
E4DF 1599
E4DF 1600
E4DF 1601
E4DF 1602
E4DF 1603
E4DF 1604
E4DF 1605
E4DF 1606
E4DF 1607
E4DF 1608
E4DF 1609
E4DF 1610
E4DF 1611
E4DF 1612
E4DF 1613
E4DF 1614
E4DF 1615
E4DF 1616
E4DF 1617
E4DF 1618
E4DF 1619
E4DF 1620
E4DF 1621
E4DF 1622
E4DF 1623
E4DF 1624
E4DF 1625
E4DF 1626
E4DF 1627
E4DF 1628
E4DF 1629
E4DF 1630
E4DF 1631
E4DF 1632
E4DF 1633
E4DF 1634
E4DF 1635
E4DF 1636
E4DF 1637
E4DF 1638
E4DF 1639
E4DF 1640
E4DF 1641
E4DF 1642
E4DF 1643
E4DF 1644
E4DF 1645
E4DF 1646
E4DF 1647
E4DF 1648
E4DF 1649
E4DF 1650
E4DF 1651
E4DF 1652
E4DF 1653
E4DF 1654
E4DF 1655
E4DF 1656
E4DF 1657
E4DF 1658
E4DF 1659
E4DF 1660
E4DF 1661
E4DF 1662
E4DF 1663
E4DF 1664
E4DF 1665
E4DF 1666
E4DF 1667
E4DF 1668
E4DF 1669
E4DF 1670
E4DF 1671
E4DF 1672
E4DF 1673
E4DF 1674
E4DF 1675
E4DF 1676
E4DF 1677
E4DF 1678
E4DF 1679
E4DF 1680
E4DF 1681
E4DF 1682
E4DF 1683
E4DF 1684
E4DF 1685
E4DF 1686
E4DF 1687
E4DF 1688
E4DF 1689
E4DF 1690
E4DF 1691
E4DF 1692
E4DF 1693
E4DF 1694
E4DF 1695
E4DF 1696
E4DF 1697
E4DF 1698
E4DF 1699
E4DF 1700
E4DF 1701
E4DF 1702
E4DF 1703
E4DF 1704
E4DF 1705
E4DF 1706
E4DF 1707
E4DF 1708
E4DF 1709
E4DF 1710
E4DF 1711
E4DF 1712
E4DF 1713
E4DF 1714
E4DF 1715
E4DF 1716
E4DF 1717
E4DF 1718
E4DF 1719
E4DF 1720
E4DF 1721
E4DF 1722
E4DF 1723
E4DF 1724
E4DF 1725
E4DF 1726
E4DF 1727
E4DF 1728
E4DF 1729
E4DF 1730
E4DF 1731
E4DF 1732
E4DF 1733
E4DF 1734
E4DF 1735
E4DF 1736
E4DF 1737
E4DF 1738
E4DF 1739
E4DF 1740
E4DF 1741
E4DF 1742
E4DF 1743
E4DF 1744
E4DF 1745
E4DF 1746
E4DF 1747
E4DF 1748
E4DF 1749
E4DF 1750
E4DF 1751
E4DF 1752
E4DF 1753
E4DF 1754
E4DF 1755
E4DF 1756
E4DF 1757
E4DF 1758
E4DF 1759
E4DF 1760
E4DF 1761
E4DF 1762
E4DF 1763
E4DF 1764
E4DF 1765
E4DF 1766
E4DF 1767
E4DF 1768
E4DF 1769
E4DF 1770
E4DF 1771
E4DF 1772
E4DF 1773
E4DF 1774
E4DF 1775
E4DF 1776
E4DF 1777
E4DF 1778
E4DF 1779
E4DF 1780
E4DF 1781
E4DF 1782
E4DF 1783
E4DF 1784
E4DF 1785
E4DF 1786
E4DF 1787
E4DF 1788
E4DF 1789
E4DF 1790
E4DF 1791
E4DF 1792
E4DF 1793
E4DF 1794
E4DF 1795
E4DF 1796
E4DF 1797
E4DF 1798
E4DF 1799
E4DF 1800
E4DF 1801
E4DF 1802
E4DF 1803
E4DF 1804
E4DF 1805
E4DF 1806
E4DF 1807
E4DF 1808
E4DF 1809
E4DF 1810
E4DF 1811
E4DF 1812
E4DF 1813
E4DF 1814
E4DF 1815
E4DF 1816
E4DF 1817
E4DF 1818
E4DF 1819
E4DF 1820
E4DF 1821
E4DF 1822
E4DF 1823
E4DF 1824
E4DF 1825
E4DF 1826
E4DF 1827
E4DF 1828
E4DF 1829
E4DF 1830
E4DF 1831
E4DF 1832
E4DF 1833
E4DF 1834
E4DF 1835
E4DF 1836
E4DF 1837
E4DF 1838
E4DF 1839
E4DF 1840
E4DF 1841
E4DF 1842
E4DF 1843
E4DF 1844
E4DF 1845
E4DF 1846
E4DF 1847
E4DF 1848
E4DF 1849
E4DF 1850
E4DF 1851
E4DF 1852
E4DF 1853
E4DF 1854
E4DF 1855
E4DF 1856
E4DF 1857
E4DF 1858
E4DF 1859
E4DF 1860
E4DF 1861
E4DF 1862
E4DF 1863
E4DF 1864
E4DF 1865
E4DF 1866
E4DF 1867
E4DF 1868
E4DF 1869
E4DF 1870
E4DF 1871
E4DF 1872
E4DF 1873
E4DF 1874
E4DF 1875
E4DF 1876
E4DF 1877
E4DF 1878
E4DF 1879
E4DF 1880
E4DF 1881
E4DF 1882
E4DF 1883
E4DF 1884
E4DF 1885
E4DF 1886
E4DF 1887
E4DF 1888
E4DF 1889
E4DF 1890
E4DF 1891
E4DF 1892
E4DF 1893
E4DF 1894
E4DF 1895
E4DF 1896
E4DF 1897
E4DF 1898
E4DF 1899
E4DF 1900
E4DF 1901
E4DF 1902
E4DF 1903
E4DF 1904
E4DF 1905
E4DF 1906
E4DF 1907
E4DF 1908
E4DF 1909
E4DF 1910
E4DF 1911
E4DF 1912
E4DF 1913
E4DF 1914
E4DF 1915
E4DF 1916
E4DF 1917
E4DF 1918
E4DF 1919
E4DF 1920
E4DF 1921
E4DF 1922
E4DF 1923
E4DF 1924
E4DF 1925
E4DF 1926
E4DF 1927
E4DF 1928
E4DF 1929
E4DF 1930
E4DF 1931
E4DF 1932
E4DF 1933
E4DF 1934
E4DF 1935
E4DF 1936
E4DF 1937
E4DF 1938
E4DF 1939
E4DF 1940
E4DF 1941
E4DF 1942
E4DF 1943
E4DF 1944
E4DF 1945
E4DF 1946
E4DF 1947
E4DF 1948
E4DF 1949
E4DF 1950
E4DF 1951
E4DF 1952
E4DF 1953
E4DF 1954
E4DF 1955
E4DF 1956
E4DF 1957
E4DF 1958
E4DF 1959
E4DF 1960
E4DF 1961
E4DF 1962
E4DF 1963
E4DF 1964
E4DF 1965
E4DF 1966
E4DF 1967
E4DF 1968
E4DF 1969
E4DF 1970
E4DF 1971
E4DF 1972
E4DF 1973
E4DF 1974
E4DF 1975
E4DF 1976
E4DF 1977
E4DF 1978
E4DF 1979
E4DF 1980
E4DF 1981
E4DF 1982
E4DF 1983
E4DF 1984
E4DF 1985
E4DF 1986
E4DF 1987
E4DF 1988
E4DF 1989
E4DF 1990
E4DF 1991
E4DF 1992
E4DF 1993
E4DF 1994
E4DF 1995
E4DF 1996
E4DF 1997
E4DF 1998
E4DF 1999
E4DF 2000
E4DF 2001
E4DF 2002
E4DF 2003
E4DF 2004
E4DF 2005
E4DF 2006
E4DF 2007
E4DF 2008
E4DF 2009
E4DF 2010
E4DF 2011
E4DF 2012
E4DF 2013
E4DF 2014
E4DF 2015
E4DF 2016
E4DF 2017
E4DF 2018
E4DF 2019
E4DF 2020
E4DF 2021
E4DF 2022
E4DF 2023
E4DF 2024
E4DF 2025
E4DF 2026
E4DF 2027
E4DF 2028
E4DF 2029
E4DF 2030
E4DF 2031
E4DF 2032
E4DF 2033
E4DF 2034
E4DF 2035
E4DF 2036
E4DF 2037
E4DF 2038
E4DF 2039
E4DF 2040
E4DF 2041
E4DF 2042
E4DF 2043
E4DF 2044
E4DF 2045
E4DF 2046
E4DF 2047
E4DF 2048
E4DF 2049
E4DF 2050
E4DF 2051
E4DF 2052
E4DF 2053
E4DF 2054
E4DF 2055
E4DF 2056
E4DF 2057
E4DF 2058
E4DF 2059
E4DF 2060
E4DF 2061
E4DF 2062
E4DF 2063
E4DF 2064
E4DF 2065
E4DF 2066
E4DF 2067
E4DF 2068
E4DF 2069
E4DF 2070
E4DF 2071
E4DF 2072
E4DF 2073
E4DF 2074
E4DF 2075
E4DF 2076
E4DF 2077
E4DF 2078
E4DF 2079
E4DF 2080
E4DF 2081
E4DF 2082
E4DF 2083
E4DF 2084
E4DF 2085
E4DF 2086
E4DF 2087
E4DF 2088
E4DF 2089
E4DF 2090
E4DF 2091
E4DF 2092
E4DF 2093
E4DF 2094
E4DF 2095
E4DF 2096
E4DF 2097
E4DF 2098
E4DF 2099
E4DF 2100
E4DF 2101
E4DF 2102
E4DF 2103
E4DF 2104
E4DF 2105
E4DF 2106
E4DF 2107
E4DF 2108
E4DF 2109
E4DF 2110
E4DF 2111
E4DF 2112
E4DF 2113
E4DF 2114
E4DF 2115
E4DF 2116
E4DF 2117
E4DF 2118
E4DF 2119
E4DF 2120
E4DF 2121
E4DF 2122
E4DF 2123
E4DF 2124
E4DF 2125
E4DF 2126
E4DF 2127
E4DF 2128
E4DF 2129
E4DF 2130
E4DF 2131
E4DF 2132
E4DF 2133
E4DF 2134
E4DF 2135
E4DF 2136
E4DF 2137
E4DF 2138
E4DF 2139
E4DF 2140
E4DF 2141
E4DF 2142
E4DF 2143
E4DF 2144
E4DF 2145
E4DF 2146
E4DF 2147
E4DF 2148
E4DF 2149
E4DF 2150
E4DF 2151
E4DF 2152
E4DF 2153
E4DF 2154
E4DF 2155
E4DF 2156
E4DF 2157
E4DF 2158
E4DF 2159
E4DF 2160
E4DF 2161
E4DF 2162
E4DF 2163
E4DF 2164
E4DF 2165
E4DF 2166
E4DF 2167
E4DF 2168
E4DF 2169
E4DF 2170
E4DF 2171
E4DF 2172
E4DF 2173
E4DF 2174
E4DF 2175
E4DF 2176
E4DF 2177
E4DF 2178
E4DF 2179
E4DF 2180
E4DF 2181
E4DF 2182
E4DF 2183
E4DF 2184
E4DF 2185
E4DF 2186
E4DF 2187
E4DF 2188
E4DF 2189
E4DF 2190
E4DF 2191
E4DF 2192
E4DF 2193
E4DF 2194
E4DF 2195
E4DF 2196
E4DF 2197
E4DF 2198
E4DF 2199
E4DF 2200
E4DF 2201
E4DF 2202
E4DF 2203
E4DF 2204
E4DF 2205
E4DF 2206
E4DF 2207
E4DF 2208
E4DF 2209
E4DF 2210
E4DF 2211
E4DF 2212
E4DF 2213
E4DF 2214
E4DF 2215
E4DF 2216
E4DF 2217
E4DF 2218
E4DF 2219
E4DF 2220
E4DF 2221
E4DF 2222
E4DF 2223
E4DF 2224
E4DF 2225
E4DF 2226
E4DF 2227
E4DF 2228
E4DF 2229
E4DF 2230
E4DF 2231
E4DF 2232
E4DF 2233
E4DF 2234
E4DF 2235
E4DF 2236
E4DF 2237
E4DF 2238
E4DF 2239
E4DF 2240
E4DF 2241
E4DF 2242
E4DF 2243
E4DF 2244
E4DF 2245
E4DF 2246
E4DF 2247
E4DF 2248
E4DF 2249
E4DF 2250
E4DF 2251
E4DF 2252
E4DF 2253
E4DF 2254
E4DF 2255
E4DF 2256
E4DF 2257
E4DF 2258
E4DF 2259
E4DF 2260
E4DF 2261
E4DF 2262
E4DF 2263
E4DF 2264
E4DF 2265
E4DF 2266
E4DF 2267
E4DF 2268
E4DF 2269
E4DF 2270
E4DF 2271
E4DF 2272
E4DF 2273
E4DF 2274
E4DF 2275
E4DF 2276
E4DF 2277
E4DF 2278
E4DF 2279
E4DF 2280
E4DF 2281
E4DF 2282
E4DF 22
```

```

1102 ;-----
1103 ; CHECK FOR OPTIONAL ROM FROM C8000->F4000 IN 2K BLOCKS :
1104 ; (A VALID MODULE HAS '55AA' IN THE FIRST 2 LOCATIONS, :
1105 ; LENGTH INDICATOR (LENGTH/512) IN THE 3D LOCATION AND :
1106 ; TEST/INIT. CODE STARTING IN THE 4TH LOCATION.) :
1107 ;-----
E518 ROM_SCAN: 1108 MOV DX,0C800H ; SET BEGINNING ADDRESS
E518 BA00C8 1109 ROM_SCAN_1: 1110 MOV DS,DX
E518 8EDA 1111 SUB BX,BX ; SET BX=0000
E510 2B0B 1112 MOV AX,[BX] ; GET 1ST WORD FROM MODULE
E51F 8B07 1113 PUSH BX
E521 53 1114 POP BX ; BUS SETTLING
E522 5B 1115 JNZ AX,0AA55H ; = TO ID WORD?
E523 3D55AA 1116 NEXT_ROM: 1117 CALL ROM_CHECK ; PROCEED TO NEXT ROM IF NOT
E526 7506 1118 CALL ARE_WE_DONE ; GO CHECK OUT MODULE
E528 E82814 1119 ARE_WE_DONE: 1120 JMP ARE_WE_DONE ; CHECK FOR END OF ROM SPACE
E52E EB0590 1121 NEXT_ROM: 1122 ADD DX,0080H ; POINT TO NEXT 2K ADDRESS
E52E 81C28000 1123 ARE_WE_DONE: 1124 CMP DX,0F600H ; AT F6000 YET?
E532 81FA00F6 1125 JMP BASE_ROM_CHK ; GO CHECK ANOTHER ADD. IF NOT
E536 TCE3 1126 BASE_ROM_CHK: 1127 JZ BASE_ROM_CHK ; GO CHECK BASIC ROM
E538 EB0190 1128 ;-----
1129 ; A CHECKSUM IS DONE FOR THE 4 ROS MODULES CONTAINING BASIC CODE :
1130 ;-----
E53B 1131 E4: 1132 MOV AH,4 ; NO. OF ROS MODULES TO CHECK
E53B B404 1133 SUB BX,BX ; SETUP STARTING ROS ADDR
E53D 2B0B 1134 MOV DS,DX ; CHECK ROS
E53F 8EDA 1135 CALL ROS_CHECKSUM ; CHECK ROS
E541 E8AE13 1136 JE E5 ; CONTINUE IF OK
E544 7403 1137 CALL ROM_ERR ; POST ERROR
E546 E88201 1138 E5: 1139 ADD DX,0200H ; POINT TO NEXT 8K MODULE
E549 81C20002 1140 DEC AH ; ANY MORE TO DO?
E54D FECC 1141 JNZ E4 ; YES - CONTINUE
E54F 75EC 1142 ;-----
1143 ; DISKETTE ATTACHMENT TEST :
1144 ; DESCRIPTION :
1145 ; CHECK IF IPL DISKETTE DRIVE IS ATTACHED TO SYSTEM. IF :
1146 ; ATTACHED, VERIFY STATUS OF NEC FDC AFTER A RESET. ISSUE :
1147 ; A RECAL AND SEEK CMD TO FDC AND CHECK STATUS, COMPLETE :
1148 ; SYSTEM INITIALIZATION THEN PASS CONTROL TO THE BOOT :
1149 ; LOADER PROGRAM. :
1150 ;-----
E551 F9: 1151 POP DS
E551 1F 1152 MOV AL,BYTE PTR EQUIP_FLAG ; DISKETTE PRESENT?
E552 A01000 1153 AND AL,01H ; NO - BYPASS DISKETTE TEST
E555 2401 1154 JZ F15 ; DISK_TEST:
E557 743E 1155 F10: 1156 IN AL,INTA01 ; ENABLE DISKETTE INTERRUPTS
E559 E421 1157 AND AL,0BFH ; RESET NEC FDC
E55B 24BF 1158 INTA01,AL ; SET FOR DRIVE 0
E55D E621 1159 MOV AH,0 ; VERIFY STATUS AFTER RESET
E55F B400 1160 MOV DL,AH ; STATUS OK?
E561 8AD4 1161 INT 3H ; NO - FDC FAILED
E563 CD13 1162 TEST AH,0FFH
E565 F6C4FF 1163 JNZ F13
E568 7520 1164 ;-----
1165 ; TURN DRIVE 0 MOTOR ON :
1166 ;-----
E56A BAF203 1167 MOV DX,03F2H ; GET ADDR OF FDC CARD
E56D B01C 1168 MOV AL,1CH ; TURN MOTOR ON, EN DMA/INT
E56F EE 1169 OUT DX,AL ; WRITE FDC CONTROL REG
E570 2BC9 1170 SUB CX,CX
E572 F11: 1171 LOOP F11 ; MOTOR WAIT:
E574 E2FE 1172 F12: 1173 LOOP F12 ; WAIT FOR 1 SECOND
E576 33D2 1174 MOV CH,1 ; MOTOR WAIT1:
E578 B501 1175 XOR DX,DX ; SELECT DRIVE 0
E57A 8B163E00 1176 MOV CH,1 ; SELECT TRACK 1
E57E E8FC08 1177 CALL SEEK_STATUS,DL ; RECALIBRATE DISKETTE
E581 7207 1178 JC F13 ; GO TO ERR SUBROUTINE IF ERR
E583 B522 1179 MOV CH,34 ; SELECT TRACK 34
E585 E8F508 1180 CALL JNC F14 ; SEEK TO TRACK 34
E588 7307 1181 JNC F14 ; OK, TURN MOTOR OFF
E58A F13: 1182 MOV SI,OFFSET F3 ; DSK_ERR:
E58A BE52EC90 1183 MOV SI,OFFSET F3 ; GET ADDR OF MSG
E58E E81814 1184 CALL E_MSG ; GO PRINT ERROR MSG
E589 1185 ;-----
1186 ; TURN DRIVE 0 MOTOR OFF :
1187 ;-----
E591 F14: 1188 MOV AL,0CH ; DR0_OFF:
E591 B00C 1189 MOV DX,03F2H ; TURN DRIVE 0 MOTOR OFF
E593 BAF203 1190 OUT DX,AL ; FDC CTL ADDRESS
E596 EE 1191 ;-----
1192 ; SETUP PRINTER AND RS232 BASE ADDRESSES IF DEVICE ATTACHED :
1193 ;-----
E597 F15: 1194 MOV INTR_FLAG,00H ; SET STRAY INTERRUPT FLAG = 00
E597 C606B0000 1195 MOV SI,OFFSET KB_BUFFER ; SETUP KEYBOARD PARAMETERS
E59C BE1E00 1196 MOV BUFFER_HEAD,SI
E59F 89361A00 1200 MOV BUFFER_TAIL,SI
E5A3 89361C00 1201 MOV BUFFER_START,SI
E5A7 89368000 1202 MOV SI,32 ; DEFAULT BUFFER OF 32 BYTES
E5AB 83C620 1203 MOV SI,32
E5AE 89368200 1204 MOV DI,OFFSET PRINT_TIM_OUT ; SET DEFAULT PRINTER TIMEOUT
E5B2 BF7800 1205 PUSH DS
E5B5 1E 1206 POP DS
E5B6 07 1207 MOV AX,1414H ; DEFAULT=20
E5B7 B81414 1208 STOSW
E5BA AB 1209 STOSW
E5BB AB 1210 MOV AX,0101H ; RS232 DEFAULT=01
E5BC B80101 1211 STOSW
E5BF AB 1212 STOSW
E5C0 AB 1213 STOSW
E5C1 E421 1214 IN AL,INTA01 ; ENABLE TIMER AND KB INTS
E5C3 24FC 1215 AND AL,0FCH
E5C5 E621 1216 OUT INTA01,AL

```

LOC	OBJECT	LINE	SOURCE	(BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82
E5C7	83FD00	1217	CMP	BP,0000H ; CHECK FOR BP= NON-ZERO
E5CA	7419	1218	Jc	F15A_0 ; (ERROR HAPPENED)
E5CC	BA0200	1219	DX,2	CONTINUE IF NO ERROR
E5CF	E80614	1220	MOV	ERR_BEEP ; 2 SHORT BEEPS (ERROR)
E5D2	BD09E890	1221	CALL	S1,OFFSET F3D ; LOAD ERROR MSG
E5D6	E8F113	1222	CALL	P_MSG
E5D9		1223	ERR_WAIT:	
E5D9	B400	1224	MOV	AH,00
E5DB	CD16	1225	INT	16H ; WAIT FOR 'F1' KEY
E5DD	80FC3B	1226	CMP	AH,3BH
E5E0	75F7	1227	JNE	ERR_WAIT
E5E2	E80E90	1228	JMP	F15A ; BYPASS ERROR
E5E5		1229	F15A_0:	
E5E5	803E120001	1230	CMP	MFG_TST,1
E5EA	7406	1231	JE	F15A ; MFG MODE
E5EC	BA0100	1232	MOV	DX,1 ; BYPASS BEEP
E5EF	E8E613	1233	CALL	ERR_BEEP ; 1 SHORT BEEP (NO ERRORS)
E5F2	A01000	1234	MOV	AL,BYTE PTR EQUIP_FLAG ; GET SWITCHES
E5F5	2401	1235	AND	AL,00000001B ; 'LOOP POST' SWITCH ON
E5F7	7503	1236	JNZ	F15B ; CONTINUE WITH BRING-UP
E5F9	E95FFA	1237	JMP	START
E5FC	2AE4	1238	SUB	AH,AH
E5FE	A04900	1239	MOV	AL,CRT_MODE
E601	CD10	1240	INT	10H ; CLEAR SCREEN
E603		1241	F15C:	
E603	BDA3F990	1242	MOV	BP,OFFSET F4
E607	BE0000	1243	MOV	SI,0 ; PRT_SRC_TBL
E60A		1244	F16:	
E60A	2E8B5600	1245	MOV	DX,CS:[BP] ; PRT_BASE1
E60E	B0AA	1246	MOV	AL,0AAH ; GET PRINTER BASE ADDR
E610	EE	1247	OUT	DX,AL ; WRITE DATA TO PORT A
E611	1E	1248	PUSH	DS ; BUS SETTLEING
E612	EC	1249	IN	AL,DX ; READ PORT A
E613	1F	1250	POP	DS
E614	3CAA	1251	CMP	AL,0AAH ; DATA PATTERN SAME
E616	7505	1252	JNE	F17 ; NO - CHECK NEXT PRT CD
E618	895408	1253	MOV	PRINTER_BASE[SI],DX ; YES - STORE PRT BASE ADDR
E61B	46	1254	INC	SI ; INCREMENT TO NEXT WORD
E61C	46	1255	INC	SI
E61D		1256	F17:	
E61D	45	1257	INC	BP ; POINT TO NEXT BASE ADDR
E61E	45	1258	INC	BP
E61F	81FDA9F9	1259	CMP	BP,OFFSET F4E ; ALL POSSIBLE ADDRS CHECKED?
E623	75E5	1260	JNE	F16 ; PRT BASE
E625	BB0000	1261	MOV	DX,0 ; POINTER TO RS232 TABLE
E628	BAFA03	1262	MOV	DX,3FAH ; CHECK IF RS232 CD 1 ATTCH?
E62B	EC	1263	IN	AL,DX ; READ INTR ID REG
E62C	A8F8	1264	TEST	AL,0F8H
E62E	7506	1265	JNZ	F18
E630	C707F803	1266	MOV	RS232_BASE[BX],3F8H ; SETUP RS232 CD #1 ADDR
E634	43	1267	INC	BX
E635	43	1268	INC	BX
E636		1269	F18:	
E636	BAFA02	1270	MOV	DX,2FAH ; CHECK IF RS232 CD 2 ATTCH
E639	EC	1271	IN	AL,DX ; READ INTERRUPT ID REG
E63A	A8F8	1272	TEST	AL,0F8H
E63C	7506	1273	JNZ	F19 ; BASE END
E63E	C707F802	1274	MOV	RS232_BASE[BX],2F8H ; SETUP RS232 CD #2
E642	43	1275	INC	BX
E643	43	1276	INC	BX
E644		1277		
E644		1278		
E644	8BC6	1279		
E646	B103	1280		
E648	D2C8	1281	F19:	
E64A	OAC3	1282	MOV	AX,SI ; BASE END:
E64C	A21100	1283	MOV	CL,3 ; SI HAS 2* NUMBER OF RS232
E64F	BA0102	1284	ROR	AL,CL ; SHIFT COUNT
E652	EC	1285	OR	AL,BL ; ROTATE RIGHT 3 POSITIONS
E653	90	1286	MOV	BYTE PTR EQUIP_FLAG+1,AL ; OR IN THE PRINTER COUNT
E654	90	1287	MOV	DX,201H ; STORE AS SECOND BYTE
E655	90	1288	IN	AL,DX
E656	A80F	1289	NOP	
E658	7505	1290	NOP	
E65A	800E110010	1291	NOP	
E65F		1292	TEST	AL,0FH
		1293	JNZ	F20 ; NO_GAME_CARD
		1294	OR	BYTE PTR EQUIP_FLAG+1,16 ; NO_GAME_CARD:
		1295	F20:	
		1296		
		1297		
		1298		
		1299		
E65F	E461	1300	IN	AL,PORT_B ; RESET CHECK ENABLES
E661	OC30	1301	OR	AL,30H
E663	E661	1302	OUT	PORT_B,AL
E665	24CF	1303	AND	PORT_B,AL
E667	E661	1304	OUT	PORT_B,AL
E669	B080	1305	MOV	AL,80H ; ENABLE NMI INTERRUPTS
E66B	E6A0	1306	MOV	0A0H,AL
E66D		1307	F21:	
E66D	CD19	1308	INT	19H ; LOAD BOOT_STRAP:
				GO TO THE BOOT LOADER

5-126 PC-XT System BIOS (11/08/82)

LOC OBJECT	LINE	SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82
	1408	;--- INT 19 -----
	1409	: BOOT STRAP LOADER :
	1410	: TRACK 0, SECTOR 1 IS READ INTO THE :
	1411	: BOOT LOCATION (SEGMENT 0, OFFSET 7C00) :
	1412	: AND CONTROL IS TRANSFERRED THERE. :
	1413	: :
	1414	: IF THERE IS A HARDWARE ERROR CONTROL IS :
	1415	: TRANSFERRED TO THE ROM BASIC ENTRY POINT. :
	1416	;------
E6F2	1417	ASSUME CS:CODE,DS:ABS0
	1418	ORG 0E6F2H
E6F2	1419	
E6F2 FB	1420	BOOT_STRAP PROC NEAR
E6F3 2B00	1421	STI ; ENABLE INTERRUPTS
E6F5 8ED8	1422	SUB AX,AX ; ESTABLISH ADDRESSING
	1423	MOV DS,AX
	1424	
	1425	;------ RESET THE DISK PARAMETER TABLE VECTOR
E6F7 C7067800C7EF	1426	
E6FD 8C0ETA00	1427	MOV WORD PTR DISK_POINTER, OFFSET DISK_BASE
	1428	MOV WORD PTR DISK_POINTER+2,CS
	1429	
	1430	;------ LOAD SYSTEM FROM DISKETTE -- CX HAS RETRY COUNT
	1431	
E701 B90400	1432	MOV CX,4 ; SET RETRY COUNT
E704	1433	H1: ; IPL SYSTEM
E704 51	1434	PUSH CX ; SAVE RETRY COUNT
E705 B400	1435	MOV AH,0 ; RESET THE DISKETTE SYSTEM
E707 CD13	1436	INT 13H ; DISKETTE_IO
E709 720F	1437	JC H2 ; IF ERROR, TRY AGAIN
E70B BB0102	1438	MOV AX,201H ; READ IN THE SINGLE SECTOR
E70E 2B02	1439	SUB DX,DX ; TO THE BOOT LOCATION
E710 8EC2	1440	MOV ES,DX
E712 BB007C	1441	MOV BX,OFFSET BOOT_LOCN
	1442	
E715 B90100	1443	MOV CX,1 ; DRIVE 0, HEAD 0
E718 CD13	1444	INT 13H ; SECTOR 1, TRACK 0
E71A	1445	H2: ; DISKETTE_IO
E71A 59	1446	POP CX ; RECOVER RETRY COUNT
E71B 7304	1447	JNC H4 ; CF SET BY UNSUCCESSFUL READ
E71D E2E5	1448	LOOP H1 ; DO IT FOR RETRY TIMES
	1449	
	1450	;------ UNABLE TO IPL FROM THE DISKETTE
	1451	
E71F	1452	H3: INT 18H ; GO TO RESIDENT BASIC
E71F CD18	1453	
	1454	
	1455	;------ IPL WAS SUCCESSFUL
	1456	
E721	1457	H4: ;
E721 EA007C0000	1458	JMP BOOT_LOCN
	1459	BOOT_STRAP ENDP
	1460	

```

1461: 1-----INT 14-----
1462: R5232_10
1463: THIS ROUTINE PROVIDES BYTE STREAM I/O TO THE COMMUNICATIONS
1464: PORT ACCORDING TO THE PARAMETERS:
1465: (AH)=0 INITIALIZE THE COMMUNICATIONS PORT
1466: (AL) HAS PARAMETERS FOR INITIALIZATION
1467:
1468: 7----- 6 5 4 3 2 1 0
1469: ----- BAUD RATE -- -PARITY-- STOPBIT --WORD LENGTH--
1470: 000 - 110 X0 - NONE 0 - 1 10 - 7 BITS
1471: 001 - 150 01 - ODD 1 - 2 11 - 8 BITS
1472: 010 - 300 11 - EVEN
1473: 011 - 600
1474: 100 - 1200
1475: 101 - 2400
1476: 110 - 4800
1477: 111 - 9600
1478:
1479: ON RETURN, CONDITIONS SET AS IN CALL TO COMMO STATUS (AH=3)
1480: (AH)=1 SEND THE CHARACTER IN (AL) OVER THE COMMO LINE
1481: (AL) REGISTER IS PRESERVED
1482: ON EXIT, BIT 7 OF AH IS SET IF THE ROUTINE WAS UNABLE
1483: TO TRANSMIT THE BYTE OF DATA OVER THE LINE.
1484: IF BIT 7 OF AH IS NOT SET, THE REMAINDER OF AH
1485: IS SET AS IN A STATUS REQUEST, REFLECTING THE
1486: CURRENT STATUS OF THE LINE.
1487: (AH)=2 RECEIVE A CHARACTER IN (AL) FROM COMMO LINE BEFORE
1488: RETURNING TO CALLER
1489: ON EXIT, AH HAS THE CURRENT LINE STATUS, AS SET BY THE
1490: THE STATUS ROUTINE, EXCEPT THAT THE ONLY BITS
1491: LEFT ON ARE THE ERROR BITS (7,4,3,2,1)
1492: IF AH HAS BIT 7 ON (TIME OUT) THE REMAINING
1493: BITS ARE NOT PREDICTABLE.
1494: THUS, AH IS NON ZERO ONLY WHEN AN ERROR
1495: OCCURRED.
1496: (AH)=3 RETURN THE COMMO PORT STATUS IN (AX)
1497: AH CONTAINS THE LINE STATUS
1498: BIT 7 = TIME OUT
1499: BIT 6 = TRANS SHIFT REGISTER EMPTY
1500: BIT 5 = TRAN HOLDING REGISTER EMPTY
1501: BIT 4 = BREAK DETECT
1502: BIT 3 = FRAMING ERROR
1503: BIT 2 = PARITY ERROR
1504: BIT 1 = OVERRUN ERROR
1505: BIT 0 = DATA READY
1506: AL CONTAINS THE MODEM STATUS
1507: BIT 7 = RECEIVED LINE SIGNAL DETECT
1508: BIT 6 = RING INDICATOR
1509: BIT 5 = DATA SET READY
1510: BIT 4 = CLEAR TO SEND
1511: BIT 3 = DELTA RECEIVE LINE SIGNAL DETECT
1512: BIT 2 = TRAILING EDGE RING DETECTOR
1513: BIT 1 = DELTA DATA SET READY
1514: BIT 0 = DELTA CLEAR TO SEND
1515:
1516: (DX) = PARAMETER INDICATING WHICH R5232 CARD (0,1 ALLOWED)
1517:
1518: DATA AREA R5232_BASE CONTAINS THE BASE ADDRESS OF THE 8250 ON THE
1519: CARD LOCATION 400H CONTAINS UP TO 4 R5232 ADDRESSES POSSIBLE.
1520: DATA AREA LABEL R5232_T1W_OUT (BYTE) CONTAINS OUTER LOOP COUNT
1521: VALUE FOR TIMEOUT (DEFAULT=1)
1522: OUTPUT
1523: AX MODIFIED ACCORDING TO PARMS OF CALL
1524: ALL OTHERS UNCHANGED
1525: -----
1526: ASSUME CS:CODE,DS:DATA
1527: ORG 0E729H
1528: A1 LABEL WORD ; TABLE OF INIT VALUES
1529: DW 1047 ; 110 BAUD
1530: DW 768 ; 150
1531: DW 384 ; 300
1532: DW 192 ; 600
1533: DW 96 ; 1200
1534: DW 48 ; 2400
1535: DW 24 ; 4800
1536: DW 12 ; 9600
1537:
1538: R5232_10 PROC FAR
1539:
1540: ;----- VECTOR TO APPROPRIATE ROUTINE
1541:
1542: STI ; INTERRUPTS BACK ON
1543: PUSH DS ; SAVE SEGMENT
1544: PUSH DX
1545: PUSH SI
1546: PUSH DI
1547: PUSH CX
1548: PUSH BX
1549: MOV SI,DX ; R5232 VALUE TO SI
1550: MOV DI,DX
1551: SHL SI,1 ; WORD OFFSET
1552: CALL DD5
1553: MOV DX,R5232_BASE[SI] ; GET BASE ADDRESS
1554: DX,DX ; TEST FOR 0 BASE ADDRESS
1555: JZ A3 ; RETURN
1556: OR AH,AH ; TEST FOR (AH)=0
1557: JZ A4 ; COMMUN INIT
1558: DEC AH ; TEST FOR (AH)=1
1559: JZ A5 ; SEND AL
1560: DEC AH ; TEST FOR (AH)=2
1561: JZ A12 ; RECEIVE INTO AL
1562: A2:
1563: DEC AH ; TEST FOR (AH)=3
1564: JNZ A3
1565: JMP A18 ; COMMUNICATION STATUS
1566: A3: ; RETURN FROM R5232
1567: POP BX
1568: POP CX
1569: POP DI
1570: POP SI
1571: POP DX
1572: POP DS
1573: IRET ; RETURN TO CALLER, NO ACTION
1574:

```



```

1575 ;----- INITIALIZE THE COMMUNICATIONS PORT
1576
E769 1577 A4: MOV AH,AL ; SAVE INIT PARMS IN AH
E769 8AE0 1578 MOV DX,3 ; POINT TO 8250 CONTROL REGISTER
E76B 83C203 1579 ADD AL,80H
E76E B080 1580 OUT DX,AL ; SET DLAB=1
E770 EE 1581
1582
1583 ;----- DETERMINE BAUD RATE DIVISOR
1584
E771 8AD4 1585 MOV DL,AH ; GET PARMS TO DL
E773 B104 1586 MOV CL,4
E775 D2C2 1587 ROL DL,CL
E777 81E20E00 1588 AND DX,0EH ; ISOLATE THEM
E77B BF29E7 1589 MOV DI,OFFSET A1 ; BASE OF TABLE
E77E 03FA 1590 MOV DI,DX ; PUT INTO INDEX REGISTER
E780 8B14 1591 MOV DX,RS232_BASE[SI] ; POINT TO HIGH ORDER OF DIVISOR
E782 42 1592 INC DX
E783 2E8A4501 1593 MOV AL,CS:[DI]+1 ; GET HIGH ORDER OF DIVISOR
E787 EE 1594 OUT DX,AL ; SET MS OF DIV TO 0
E788 4A 1595 DEC DX
E789 2E8A05 1596 MOV AL,SC:[DI] ; GET LOW ORDER OF DIVISOR
E78C EE 1597 OUT DX,AL ; SET LOW OF DIVISOR
E78D 83C203 1598 ADD DX,3
E790 8AC4 1599 MOV AL,AH ; GET PARMS BACK
E792 241F 1600 AND AL,01FH ; STRIP OFF THE BAUD BITS
E794 EE 1601 OUT DX,AL ; LINE CONTROL TO 8 BITS
E795 4A 1602 DEC DX
E796 4A 1603 DEC DX
E797 B000 1604 MOV AL,0
E799 EE 1605 OUT DX,AL ; INTERRUPT ENABLES ALL OFF
E79A EB49 1606 JMP SHORT A18 ; COM STATUS
1607
1608 ;----- SEND CHARACTER IN (AL) OVER COMMO LINE
1609
E79C 1610 A5:
E79C 50 1611 PUSH AX ; SAVE CHAR TO SEND
E79D 83C204 1612 ADD DX,4 ; MODEM CONTROL REGISTER
E7A0 B003 1613 MOV AL,3 ; DTR AND RTS
E7A2 EE 1614 OUT DX,AL ; DATA TERMINAL READY, REQUEST TO SEND
E7A3 42 1615 INC DX ; MODEM STATUS REGISTER
E7A4 42 1616 INC DX
E7A5 B730 1617 MOV BH,30H ; DATA SET READY & CLEAR TO SEND
E7A7 E84800 1618 CALL WAIT_FOR_STATUS ; ARE BOTH TRUE
E7AA 7408 1619 JE A9 ; YES, READY TO TRANSMIT CHAR
1620
E7AC 1621 A7: POP CX
E7AD 8AC1 1622 MOV AL,CL ; RELOAD DATA BYTE
E7AF 1623 A8: OR AH,80H ; INDICATE TIME OUT
E7AF 80CC80 1624 JMP A3 ; RETURN
E7B2 EBAE 1625
E7B4 1626 A9: DEC DX ; CLEAR TO SEND
E7B4 4A 1627 DEC DX ; LINE STATUS REGISTER
E7B5 1628 A10: MOV BH,20H ; WAIT SEND
E7B5 B720 1629 CALL WAIT_FOR_STATUS ; IS TRANSMITTER READY
E7B7 E83800 1630 CALL WAIT_FOR_STATUS ; TEST FOR TRANSMITTER READY
E7BA 75F0 1631 JNZ A7 ; RETURN WITH TIME OUT SET
1632
E7BC 83EA05 1633 SUB DX,5 ; OUT CHAR ; DATA PORT
E7BF 59 1634 POP CX ; RECOVER IN CX TEMPORARILY
E7C0 8AC1 1635 MOV AL,CL ; MOVE CHAR TO AL FOR OUT, STATUS IN AH
E7C2 EE 1636 OUT DX,AL ; OUTPUT CHARACTER
E7C3 EB9D 1637 JMP A3 ; RETURN
1638
1639 ;----- RECEIVE CHARACTER FROM COMMO LINE
1640
E7C5 1641 A12:
E7C5 83C204 1642 ADD DX,4 ; MODEM CONTROL REGISTER
E7C8 B001 1643 MOV AL,1 ; DATA TERMINAL READY
E7CA EE 1644 OUT DX,AL
E7CB 42 1645 INC DX ; MODEM STATUS REGISTER
E7CC 42 1646 INC DX
E7CD 1647 A13: MOV BH,20H ; WAIT DSR
E7CD B720 1648 CALL WAIT_FOR_STATUS ; DATA SET READY
E7CF E82000 1649 JNZ A8 ; TEST FOR DSR
E7D2 750B 1650 JNZ A8 ; RETURN WITH ERROR
E7D4 1651 A15: DEC DX ; WAIT DSR END
E7D4 4A 1652 DEC DX ; LINE STATUS REGISTER
E7D5 1653 A16: MOV BH,1 ; WAIT REC'V
E7D5 B701 1654 CALL WAIT_FOR_STATUS ; RECEIIVE BUFFER FULL
E7D7 E81800 1655 JNZ A8 ; TEST FOR REC. BUFF. FULL
E7DA 75D3 1656 JNZ A8 ; SET TIME OUT ERROR
E7DC 1657 A17: AND AH,0001110B ; GET CHAR
E7DC 80E41E 1658 MOV DX,RS232_BASE[SI] ; TEST FOR ERR CONDITIONS ON REC'V CHAR
E7DF 8B14 1659 MOV DI,DX ; DATA PORT
E7E1 EC 1660 IN AL,DX ; GET CHARACTER FROM LINE
E7E2 E97DFF 1661 JMP A3 ; RETURN
1662
1663 ;----- COMMO PORT STATUS ROUTINE
1664
E7E5 1665 A18: MOV DX,RS232_BASE[SI]
E7E5 8B14 1666 ADD DX,5 ; CONTROL PORT
E7E7 83C205 1667 INC AL,DX ; GET LINE CONTROL STATUS
E7EA EC 1668 MOV AH,AL ; PUT IN AH FOR RETURN
E7EB 8AE0 1669 INC DX ; POINT TO MODEM STATUS REGISTER
E7ED 42 1670 IN AL,DX ; GET MODEM CONTROL STATUS
E7EE EC 1671 JMP A3 ; RETURN
E7EF E970FF 1672

```

LOC OBJECT	LINE	SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82
	1673	-----
	1674	: WAIT FOR STATUS ROUTINE :
	1675	: :
	1676	: ENTRY: :
	1677	: BH=STATUS BIT(S) TO LOOK FOR, :
	1678	: DX=ADDR. OF STATUS REG :
	1679	: EXIT: :
	1680	: ZERO FLAG ON = STATUS FOUND :
	1681	: ZERO FLAG OFF = TIMEOUT. :
	1682	: AH=LAST STATUS READ :
	1683	-----
ETF2	1684	WAIT_FOR_STATUS PROC NEAR
ETF2 8A5D7C	1685	MOV BL,RS232_TIM_OUT[D1] ; LOAD OUTER LOOP COUNT
ETF5	1686	WFS0: SUB CX,CX
ETF5 2BC9	1687	
ETF7	1688	WFS1:
ETF7 EC	1689	IN AL,DX ; GET STATUS
ETF8 8AE0	1690	MOV AH,AL ; MOVE TO AH
ETF8 22C7	1691	AND AL,BH ; ISOLATE BITS TO TEST
ETF8 3AC7	1692	CMP AL,BH ; EXACTLY = TO MASK
ETF8 7408	1693	JE WFS_END ; RETURN WITH ZERO FLAG ON
E800 E2F5	1694	LODP WFS1 ; TRY AGAIN
E802 FECB	1695	DEC BL
E804 75EF	1696	JNZ WFS0
	1697	
E806 0AFF	1698	OR BH,BH ; SET ZERO FLAG OFF
E808	1699	WFS_END:
E808 C3	1700	RET
	1701	WAIT_FOR_STATUS ENDP
	1702	RS232_IO ENDP
	1703	
E809 4552524F522E20	1704	F3D DB 'ERROR. (RESUME = F1 KEY)',13,10 ; ERROR PROMPT
28524653554045		
203D2022463122		
204B455929		
E823 0D		
E824 0A		
	1705	

LOC OBJECT

LINE SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82

```

1706 1----- INT 16 -----
1707 : KEYBOARD I/O :
1708 : THESE ROUTINES PROVIDE KEYBOARD SUPPORT :
1709 : INPUT :
1710 : (AH)=0 READ THE NEXT ASCII CHARACTER STRUCK FROM THE KEYBOARD :
1711 : RETURN THE RESULT IN (AL), SCAN CODE IN (AH) :
1712 : (AH)=1 SET THE Z FLAG TO INDICATE IF AN ASCII CHARACTER IS :
1713 : AVAILABLE TO BE READ. :
1714 : (ZF)=1 -- NO CODE AVAILABLE :
1715 : (ZF)=0 -- CODE IS AVAILABLE :
1716 : IF ZF = 0, THE NEXT CHARACTER IN THE BUFFER TO BE READ :
1717 : IS IN AX, AND THE ENTRY REMAINS IN THE BUFFER :
1718 : (AH)=2 RETURN THE CURRENT SHIFT STATUS IN AL REGISTER :
1719 : THE BIT SETTINGS FOR THIS CODE ARE INDICATED IN THE :
1720 : THE EQUATES FOR KB_FLAG :
1721 : OUTPUT :
1722 : AS NOTED ABOVE, ONLY AX AND FLAGS CHANGED :
1723 : ALL REGISTERS PRESERVED :
1724 -----
1725 ASSUME CS:CODE,DS:DATA
1726 ORG 0E82EH
1727 KEYBOARD_10 PROC FAR
1728 STI ; INTERRUPTS BACK ON
1729 PUSH DS ; SAVE CURRENT DS
1730 PUSH BX ; SAVE BX TEMPORARILY
1731 CALL DDS
1732 OR AH,AH ; AH=0
1733 JZ K1 ; ASCII_READ
1734 DEC AH ; AH=1
1735 JZ K2 ; ASCII_STATUS
1736 DEC AH ; AH=2
1737 JZ K3 ; SHIFT_STATUS
1738 JMP SHORT INT10_END ; EXIT
1739
1740 1----- READ THE KEY TO FIGURE OUT WHAT TO DO
1741
1742 K1: ; ASCII READ
1743 STI ; INTERRUPTS BACK ON DURING LOOP
1744 NOP ; ALLOW AN INTERRUPT TO OCCUR
1745 CLI ; INTERRUPTS BACK OFF
1746 MOV BX,BUFFER_HEAD ; GET POINTER TO HEAD OF BUFFER
1747 CMP BX,BUFFER_TAIL ; TEST END OF BUFFER
1748 K1 ; LOOP UNTIL SOMETHING IN BUFFER
1749 MOV AX,[BX] ; GET SCAN CODE AND ASCII CODE
1750 CALL K4 ; MOVE POINTER TO NEXT POSITION
1751 MOV BUFFER_HEAD,BX ; STORE VALUE IN VARIABLE
1752 JMP SHORT INT10_END ; RETURN
1753
1754 1----- ASCII STATUS
1755
1756 K2: ; INTERRUPTS OFF
1757 CLI ; GET HEAD POINTER
1758 MOV BX,BUFFER_HEAD ; IF EQUAL (Z=1) THEN NOTHING THERE
1759 CMP BX,BUFFER_TAIL
1760 MOV AX,[BX]
1761 STI ; INTERRUPTS BACK ON
1762 POP BX ; RECOVER REGISTER
1763 POP DS ; RECOVER SEGMENT
1764 RET 2 ; THROW AWAY FLAGS
1765
1766 1----- SHIFT STATUS
1767
1768 K3: ; GET THE SHIFT STATUS FLAGS
1769 MOV AL,KB_FLAG
1770 INT10_END
1771 POP BX ; RECOVER REGISTER
1772 POP DS ; RECOVER REGISTERS
1773 IRET ; RETURN TO CALLER
1774 KEYBOARD_10 ENDP
1775
1776 1----- INCREMENT A BUFFER POINTER
1777
1778 K4 PROC NEAR
1779 INC BX ; MOVE TO NEXT WORD IN LIST
1780 INC BX
1781 CMP BX,BUFFER_END ; AT END OF BUFFER?
1782 JNE K5 ; NO, CONTINUE
1783 MOV BX,BUFFER_START ; YES, RESET TO BUFFER BEGINNING
1784 K5:
1785 RET
1786 K4 ENDP
1787
1788 1----- TABLE OF SHIFT KEYS AND MASK VALUES
1789
1790 K6 LABEL BYTE
1791 DB INS_KEY ; INSERT KEY
1792 DB CAPS_KEY,NUM_KEY,SCROLL_KEY,ALT_KEY,CTL_KEY
1793
1794 K6L EQU $-K6
1795
1796 1----- SHIFT_MASK_TABLE
1797
1798 K7 LABEL BYTE
1799 DB INS_SHIFT ; INSERT NODE SHIFT
1800 DB CAPS_SHIFT,NUM_SHIFT,SCROLL_SHIFT,ALT_SHIFT,CTL_SHIFT
1801
1802 DB LEFT_SHIFT,RIGHT_SHIFT
1803
1804 1----- SCAN CODE TABLES
1805 K8 DB 27,-1,0,-1,-1,-1,30,-1

```

LOC	OBJECT	LINE	SOURCE	BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82
E895	FF			
E896	FF			
E897	FF	1806	DB	-1,-1,-1,31,-1,127,-1,17
E898	FF			
E899	1F			
E89A	FF			
E89B	7F			
E89C	FF			
E89D	11			
E89E	17			
E89F	05	1807	DB	23,5,18,20,25,21,9,15
E8A0	12			
E8A1	14			
E8A2	19			
E8A3	15			
E8A4	09			
E8A5	0F			
E8A6	10			
E8A7	1B	1808	DB	16,27,29,10,-1,1,19
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			
E8BF	0D	1811	DB	14,13,-1,-1,-1,-1,-1
E8C0	FF			
E8C1	FF			
E8C2	FF			
E8C3	FF			
E8C4	FF			
E8C5	FF			
E8C6	20	1812	DB	' ','-1
E8C7	FF			
E8C8		1813	1----- CTL TABLE SCAN	
E8C8	5E	1814	K9 LABEL BYTE	
E8C9	5F	1815	DB	94,95,96,97,98,99,100,101
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
E8E1		1819	1----- LC TABLE	
E8E1	1B	1820	K10 LABEL BYTE	
E8E2	31323334353637	1821	DB	01BH,'1234567890='',08H,09H
E8E2	3839302D3D			
E8EE	08			
E8EF	09			
E8F0	71776572747975	1822	DB	'qwertyuiop[]',0DH,-1,'asdfghjkl:',027H
E8F0	696F705B5D			
E8FC	0D			
E8FD	FF			
E8FE	6173646667686A			
E8FE	6B6C3B			
E908	27			
E909	60	1823	DB	60H,-1,5CH,'zxcvbnm,./',-1,'*',-1,' '
E90A	FF			
E90B	5C			
E90C	7A786376626E6D			
E90C	2C2E2F			
E916	FF			
E917	2A			
E918	FF			
E919	20			
E91A	FF	1824	DB	-1
E91B		1825	1----- UC TABLE	
E91B	1B	1826	K11 LABEL BYTE	
E91C	21402324	1827	DB	27,'!@#&',37,05EH,'%*()*_+',08H,0
E920	25			
E921	5E			
E922	262A28295F2B			
E928	08			
E929	00			
E92A	51574552545955	1828	DB	'QWERTYUIOP{}',0DH,-1,'ASDFGHJKL:'
E92A	494F507B7D			
E936	0D			
E937	FF			
E938	4153444647484A			

LOC	OBJECT	LINE	SOURCE	(BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82
E943 7E	4B4C3A22	1829	DB	0'EH,-1,' ZXCVBNM<>?',-1,0,-1,' ', -1
E944 FF	TC5A584356424E			
E945	403C3E3F			
E950 FF				
E951 00				
E952 FF				
E953 20				
E954 FF				
E955		1830	UC TABLE	SCAN
E955 54		1831	K12	BYTE
E956 55		1832	LABEL	DB
E957 56				84,85,86,87,88,89,90
E958 57				
E959 58				
E95A 59				
E95B 5A				
E95C 5B		1833	DB	91,92,93
E95D 5C				
E95E 5D				
E95F		1834	ALT TABLE	SCAN
E95F 68		1835	K13	BYTE
E960 69		1836	LABEL	DB
E961 6A				104,105,106,107,108
E962 6B				
E963 6C				
E964 6D		1837	DB	109,110,111,112,113
E965 6E				
E966 6F				
E967 70				
E968 71				
E969		1838	NUM STATE TABLE	
E969 3738392D343536		1839	K14	BYTE
2B3132333302E		1840	LABEL	DB
				'789-456+1230.'
E976		1841	BASE CASE TABLE	
E976 47		1842	K15	BYTE
E977 48		1843	LABEL	DB
E978 49				71,72,73,-1,75,-1,77
E979 FF				
E97A 4B				
E97B FF				
E97C 4D				
E97D FF		1844	DB	-1,79,80,81,82,83
E97E 4F				
E97F 50				
E980 51				
E981 52				
E982 53				
		1845	KEYBOARD INTERRUPT ROUTINE	
E987		1846		
E987		1847		
E987 FB		1848	ORG	0E987H
E988 50		1849	KB_INT	FAR
E989 53		1850	STI	
E98A 51		1851	PUSH	AX
E98B 52		1852	PUSH	BX
E98C 56		1853	PUSH	CX
E98D 57		1854	PUSH	DX
E98E 1E		1855	PUSH	SI
E98F 06		1856	PUSH	DI
E990 FC		1857	PUSH	DS
E991 E8C510		1858	PUSH	ES
E994 E460		1859	CLD	
E996 50		1860	CALL	DDS
E997 E461		1861	IN	AL,KB_DATA
E998 8AE0		1862	PUSH	AX
E999 0C80		1863	IN	AL,KB_CTL
E99D E661		1864	MOV	AH,AL
E99F 86E0		1865	OR	AL,80H
E9A1 E661		1866	OUT	KB_CTL,AL
E9A3 58		1867	XCHG	AH,AL
E9A4 8AE0		1868	MOV	KB_CTL,AL
		1869	POP	AX
		1870	MOV	AH,AL
		1871		
		1872	TEST FOR OVERRUN SCAN CODE FROM KEYBOARD	
E9A6 3CFF		1873		
E9A8 7503		1874	CMP	AL,OFFH
E9AA E97A02		1875	JNZ	K16
		1876	JMP	K62
		1877		
		1878	TEST FOR SHIFT KEYS	
E9AD		1879		
E9AD 247F		1880	K16:	
E9AF 0E		1881	AND	AL,07FH
E9B0 07		1882	PUSH	CS
E9B1 BF7EE8		1883	POP	ES
E9B4 B90800		1884	MOV	DI,OFFSET K6
E9B7 F2		1885	MOV	CX,K6L
E9B8 AE		1886	REPNE	SCASB
E9B9 8AC4		1887	MOV	AL,AH
E9BB 7403		1888	JE	K17
E9BD E98500		1889	JMP	K25
		1890		
		1891	SHIFT KEY FOUND	
E9C0 81E7FE8		1892		
E9C4 2E8AA586E8		1893	K17:	
E9C9 A880		1894	SUB	DI,OFFSET K6+1
E9CB 7551		1895	MOV	AH,CS:K7[DI]
		1896	TEST	AL,80H
		1897	JNZ	K23
		1898	SHIFT MAKE FOUND, DETERMINE SET OR TOGGLE	
E9CD 80FC10		1899		
E9D0 7307		1900	CMP	AH,SCROLL_SHIFT
		1901	JAE	K18
		1902		
		1903	PLAIN SHIFT KEY, SET SHIFT ON	
E9D2 08261700		1904		
E9D6 E98000		1905	OR	KB_FLAG,AH
		1906	JMP	K26
				TURN ON SHIFT BIT
				INTERRUPT_RETURN

LOC	OBJECT	LINE	SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82
		1907	
		1908	
		1909	
		1910	1:----- TOGGLED SHIFT KEY, TEST FOR 1ST MAKE OR NOT
E9D9	F606170004	1911	K18: TEST KB_FLAG,CTL_SHIFT ; SHIFT-TOGGLE
E9DE	7565	1912	K25 ; CHECK CTL_SHIFT STATE
E9E0	3C52	1913	CMP AL,INS_KEY ; JUMP IF CTL_STATE
E9E2	7522	1914	JNZ K22 ; CHECK FOR INSERT KEY
E9E4	F606170008	1915	JNZ K25 ; JUMP IF NOT INSERT KEY
E9E7	755A	1916	TEST KB_FLAG,ALT_SHIFT ; CHECK FOR ALTERNATE SHIFT
E9EB	F606170020	1917	K19: TEST KB_FLAG,NUM_STATE ; CHECK FOR BASE STATE
E9F0	750D	1918	JNZ K21 ; JUMP IF NUM_LOCK IS ON
E9F2	F606170003	1919	TEST KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT ; JUMP IF BASE STATE
E9F7	740D	1920	JZ K22 ; JUMP IF BASE STATE
		1921	
E9F9		1922	K20: ; NUMERIC ZERO, NOT INSERT KEY
E9F9	B83052	1923	MOV AX,5230H ; PUT OUT AN ASCII ZERO
E9FC	E9D601	1924	JMP K57 ; BUFFER FILL
E9FF		1925	K21: ; MIGHT BE NUMERIC
E9FF	F606170003	1926	TEST KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT ; JUMP NUMERIC, NOT INSERT
EA04	74F3	1927	JZ K20
		1928	
EA06		1929	K22: ; SHIFT TOGGLE KEY HIT; PROCESS IT
EA06	84261800	1930	TEST AH,KB_FLAG_1 ; IS KEY ALREADY DEPRESSED
EA0A	754D	1931	JNZ K26 ; JUMP IF KEY ALREADY DEPRESSED
EA0C	08261800	1932	OR KB_FLAG_1,AH ; INDICATE THAT THE KEY IS DEPRESSED
EA10	30261700	1933	XOR KB_FLAG,AH ; TOGGLE THE SHIFT STATE
EA14	3C52	1934	CMP AL,INS_KEY ; TEST FOR 1ST MAKE OF INSERT KEY
EA16	7541	1935	JNZ K26 ; JUMP IF NOT INSERT KEY
EA18	B80052	1936	MOV AX,INS_KEY*256 ; SET SCAN CODE INTO AH, 0 INTO AL
EA1B	E9B701	1937	JMP K57 ; PUT INTO OUTPUT BUFFER
		1938	
		1939	1:----- BREAK SHIFT FOUND
		1940	
EA1E		1941	K23: ; BREAK-SHIFT-FOUND
EA1E	80FC10	1942	CMP JAE AH,SCROLL_SHIFT ; IS THIS A TOGGLE KEY
EA21	731A	1943	JAE K24 ; YES, HANDLE BREAK TOGGLE
EA23	F6D4	1944	NOT AH ; INVERT MASK
EA25	20261700	1945	AND KB_FLAG,AH ; TURN OFF SHIFT BIT
EA29	3CB8	1946	CMP AL,ALT_KEY+80H ; IS THIS ALTERNATE SHIFT RELEASE
EA2B	752C	1947	JNE K26 ; INTERRUPT_RETURN
		1948	1:----- ALTERNATE SHIFT KEY RELEASED, GET THE VALUE INTO BUFFER
		1950	
EA2D	A01900	1951	MOV AL,ALT_INPUT ; SCAN CODE OF 0
EA30	B400	1952	MOV AH,0 ; ZERO OUT THE FIELD
EA32	84261900	1953	MOV AL,INPUT,AH ; WAS THE INPUT=0
EA36	3C00	1954	CMP AL,0 ; INTERRUPT_RETURN
EA38	741F	1955	JE K26 ; IT WASN'T, SO PUT IN BUFFER
EA3A	E9A101	1956	JMP K58 ; BREAK-TOGGLE
EA3D		1957	K24: ; INVERT MASK
EA3D	F6D4	1958	NOT AH ; INDICATE NO LONGER DEPRESSED
EA3F	20261800	1959	AND KB_FLAG_1,AH ; INTERRUPT_RETURN
EA43	EB14	1960	JMP SHORT K26
		1961	
		1962	1:----- TEST FOR HOLD STATE
		1963	
EA45		1964	K25: ; NO-SHIFT-FOUND
EA45	3C80	1965	CMP AL,80H ; TEST FOR BREAK KEY
EA47	7310	1966	JAE K26 ; NOTHING FOR BREAK CHARS FROM HERE ON
EA49	F606180008	1967	TEST KB_FLAG_1,HOLD_STATE ; ARE WE IN HOLD STATE
EA4E	7417	1968	JZ K28 ; BRANCH AROUND TEST IF NOT
EA50	3C45	1969	CMP AL,NUM_KEY ; CAN'T END HOLD ON NUM_LOCK
EA52	7405	1970	JE K26 ; TURN OFF THE HOLD STATE BIT
EA54	80261800F7	1971	AND KB_FLAG_1,NOT HOLD_STATE ; INTERRUPT_RETURN
EA59	FA	1972	K26: ; TURN OFF INTERRUPTS
EA59	FA	1973	CLI ; TURN OFF INTERRUPTS
EA5A	B020	1974	MOV AL,E01 ; END OF INTERRUPT COMMAND
EA5C	E620	1975	OUT 020H,AL ; SEND COMMAND TO INT CONTROL PORT
EA5E		1976	K27: ; INTERRUPT-RETURN-NO-E01
EA5E	07	1977	POP E5
EA5F	1F	1978	POP D5
EA60	5F	1979	POP D1
EA61	5E	1980	POP S1
EA62	5A	1981	POP DX
EA63	59	1982	POP CX
EA64	5B	1983	POP BX
EA65	58	1984	POP AX
EA66	CF	1985	IRET ; RESTORE STATE
		1986	; RETURN, INTERRUPTS BACK ON
		1987	; WITH FLAG CHANGE
		1988	1:----- NOT IN HOLD STATE, TEST FOR SPECIAL CHARS
		1989	
EA67		1990	K28: ; NO-HOLD-STATE
EA67	F606170008	1991	TEST KB_FLAG,ALT_SHIFT ; ARE WE IN ALTERNATE SHIFT
EA6C	7503	1992	JNZ K29 ; JUMP IF ALTERNATE SHIFT
EA6E	E99100	1993	JMP K38 ; JUMP IF NOT ALTERNATE
		1994	
		1995	1:----- TEST FOR RESET KEY SEQUENCE (CTL ALT DEL)
		1996	
EA71		1997	K29: ; TEST-RESET
EA71	F606170004	1998	TEST KB_FLAG,CTL_SHIFT ; ARE WE IN CONTROL SHIFT ALSO
EA76	7433	1999	JZ K31 ; NO_RESET
EA78	3C53	2000	CMP AL,DEL_KEY ; SHIFT STATE IS THERE, TEST KEY
EA7A	752F	2001	JNE K31 ; NO_RESET
		2002	
		2003	1:----- CTL-ALT-DEL HAS BEEN FOUND, DO I/O CLEANUP
		2004	
EA7C	C70672003412	2005	MOV RESET_FLAG,1234H ; SET FLAG FOR RESET FUNCTION
EA82	EA5BE00F0	2006	JMP RESET ; JUMP TO POWER ON DIAGNOSTICS
		2007	
EA87		2008	1:----- ALT-INPUT-TABLE
EA87	52	2009	K30 LABEL BYTE
EA88	4F	2010	DB 82,79,80,81,75,76,77
EA89	50		
EA8A	51		
EA8B	4B		
EA8C	4C		
EA8D	4D		
EA8E	47	2011	DB 71,72,73 ; 10 NUMBERS ON KEYPAD
EA8F	48		
EA90	49		
		2012	1:----- SUPER-SHIFT-TABLE
EA91	10	2013	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	2014	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	2015	DB	36,37,38,44,45,46,47,48
EAA2	25			
EAA3	26			
EAA4	2C			
EAA5	2D			
EAA6	2E			
EAA7	2F			
EAA8	30	2016	DB	49,50
EAA9	31			
EAAA	32			
		2017		
		2018	1-----	IN ALTERNATE SHIFT, RESET NOT FOUND
		2019		
EAAB		2020	K31:	
EAAB	3C39	2021	CMP	AL,57 ; NO-RESET
EAD0	7505	2022	JNE	K32 ; TEST FOR SPACE KEY
EAF7	B020	2023	MOV	AL,' ' ; NOT THERE
EAB1	E92101	2024	JMP	K57 ; SET SPACE CHAR
		2025		; BUFFER_FILL
		2026	1-----	LOOK FOR KEY PAD ENTRY
		2027		
EB4		2028	K32:	
EB4	BF87EA	2029	MOV	DI,OFFSET K30 ; ALT-KEY-PAD
EAB7	B90A00	2030	MOV	CX,10 ; ALT-INPUT-TABLE
EAB8	F2	2031	REPNE	SCASB ; LOOK FOR ENTRY USING KEYPAD
EAB8	AE			; LOOK FOR MATCH
EAB8	7512	2032	JNE	K33 ; NO ALT-KEYPAD
EAB8	81EF88EA	2033	SUB	DI,OFFSET K30+1 ; DI NOW HAS ENTRY VALUE
EAC2	A01900	2034	MOV	AL,ALT_INPUT ; GET THE CURRENT BYTE
EAC5	B40A	2035	MOV	AH,10 ; MULTIPLY BY 10
EAC7	F6E4	2036	MUL	AH
EAC9	03C7	2037	ADD	AX,DI ; ADD IN THE LATEST ENTRY
EACB	A21900	2038	MOV	ALT_INPUT,AL ; STORE IT AWAY
EACE	EB89	2039	JMP	K26 ; THROW AWAY THAT KEYSTROKE
		2040		
		2041	1-----	LOOK FOR SUPERSHIFT ENTRY
		2042		
EAD0		2043	K33:	
EAD0	C066190000	2044	MOV	ALT_INPUT,0 ; NO-ALT-KEYPAD
EAD5	B91A00	2045	MOV	CX,26 ; ZERO ANY PREVIOUS ENTRY INTO INPUT
EAD8	F2	2046	REPNE	SCASB ; DI,ES ALREADY POINTING
EAD9	AE			; LOOK FOR MATCH IN ALPHABET
EADA	7505	2047	JNE	K34 ; NOT FOUND, FUNCTION KEY OR OTHER
EADC	B000	2048	MOV	AL,0 ; ASCII CODE OF ZERO
EADE	E9F400	2049	JMP	K57 ; PUT IT IN THE BUFFER
		2050		
		2051	1-----	LOOK FOR TOP ROW OF ALTERNATE SHIFT
		2052		
EAE1		2053	K34:	
EAE1	3C02	2054	CMP	AL,2 ; ALT-TOP-ROW
EAE3	120C	2055	JB	K35 ; KEY WITH '1' ON IT
EAE5	3C0E	2056	CMP	AL,14 ; NOT ONE OF INTERESTING KEYS
EAE7	7308	2057	JAE	K35 ; IS IT IN THE REGION
EAE7	80C476	2058	ADD	AH,118 ; ALT-FUNCTION
EAE8	B000	2059	MOV	AL,0 ; CONVERT PSEUDO SCAN CODE TO RANGE
EAE8	E9E400	2060	JMP	K57 ; INDICATE AS SUCH
		2061		; BUFFER_FILL
		2062	1-----	TRANSLATE ALTERNATE SHIFT PSEUDO SCAN CODES
		2063		
EAF1		2064	K35:	
EAF1	3C3B	2065	CMP	AL,59 ; ALT-FUNCTION
EAF3	7303	2066	JAE	K37 ; TEST FOR IN TABLE
EAF5		2067		; ALT-CONTINUE
EAF5	E961FF	2068	JMP	K26 ; CLOSE-RETURN
EAF8		2069	K37:	
EAF8	3C47	2070	CMP	AL,71 ; ALT-CONTINUE
EAF8	73F9	2071	JAE	K36 ; IN KEYPAD REGION
EAF8	BB5FE9	2072	MOV	BX,OFFSET K13 ; IF SO, IGNORE
EAF8	E91B01	2073	JMP	K63 ; ALT SHIFT PSEUDO SCAN TABLE
		2074		; TRANSLATE THAT
		2075	1-----	NOT IN ALTERNATE SHIFT
		2076		
EB02		2077	K38:	
EB02	F066170004	2078	TEST	KB_FLAG,CTL_SHIFT ; NOT-ALT-SHIFT
EB07	7458	2079	JZ	K44 ; ARE WE IN CONTROL SHIFT
		2080		; NOT-CTL-SHIFT
		2081	1-----	CONTROL SHIFT, TEST SPECIAL CHARACTERS
		2082	1-----	TEST FOR BREAK AND PAUSE KEYS
		2083		
EB09	3C46	2084	CMP	AL,SCROLL_KEY ; TEST FOR BREAK
EB0B	7518	2085	JNE	K39 ; NO-BREAK
EB0D	8B1E8000	2086	MOV	BX,BUFFER_START ; RESET BUFFER TO EMPTY
EB11	891E1A00	2087	MOV	BUFFER_HEAD,BX
EB15	891E1C00	2088	MOV	BUFFER_TAIL,BX
EB19	C606710080	2089	INT	BIOS_BREAK,80H ; TURN ON BIOS BREAK BIT
EB1E	CD1B	2090	INT	IBH ; BREAK INTERRUPT VECTOR
EB20	28C0	2091	SUB	AX,AX ; PUT OUT DUMMY CHARACTER
EB22	E98000	2092	JMP	K57 ; BUFFER_FILL
EB25		2093	K39:	
EB25	3C45	2094	CMP	AL,NUM_KEY ; NO-BREAK
EB27	7521	2095	JNE	K41 ; LOOK FOR PAUSE KEY
EB29	8B0E180008	2096	OR	KB_FLAG_1,HOLD_STATE ; NO-PAUSE
EB2E	B020	2097	MOV	AL,E01 ; TURN ON THE HOLD FLAG
EB30	E620	2098	OUT	020H,AL ; END OF INTERRUPT TO CONTROL PORT
		2099		; ALLOW FURTHER KEYSTROKE INTS
		2100	1-----	DURING PAUSE INTERVAL, TURN CRT BACK ON
		2101		
EB32	803E490007	2102	CMP	CRT_MODE,7 ; IS THIS BLACK AND WHITE CARD
EB37	7407	2103	JE	K40 ; YES, NOTHING TO DO
EB39	BAD003	2104	MOV	DX,03DBH ; PORT FOR COLOR CARD
EB3C	A06500	2105	MOV	AL,CRT_MODE_SET ; GET THE VALUE OF THE CURRENT MODE
EB3F	EE	2106	OUT	DX,AL ; SET THE CRT MODE, SO THAT CRT IS ON

LOC	OBJECT	LINE	SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82
EB40		2107	K40: : PAUSE-LOOP
EB40 F606180008		2108	TEST KB_FLAG_1,HOLD_STATE : SET UP TO TRANSLATE CTL
EB45 75F9		2109	JNZ K40 : LOOP UNTIL FLAG TURNED OFF
EB47 E914FF		2110	JMP K27 : INTERRUPT_RETURN_NO_EOI
EB4A		2111	K41: : NO-PAUSE
		2112	
		2113	1----- TEST SPECIAL CASE KEY 55
		2114	
EB4A 3C37		2115	CMP AL,55
EB4C 7506		2116	JNE K42 : NOT-KEY-55
EB4E B80072		2117	MOV AX,114*256 : START/STOP PRINTING SWITCH
EB51 E91100		2118	JMP K57 : BUFFER_FILL
		2119	
		2120	1----- SET UP TO TRANSLATE CONTROL SHIFT
		2121	
EB54		2122	K42: : NOT-KEY-55
EB54 B88EE8		2123	MOV BX,OFFSET K8 : SET UP TO TRANSLATE CTL
EB57 3C3B		2124	CMP AL,59 : IS IT IN TABLE
		2125	: CTL-TABLE-TRANSLATE
EB59 7276		2126	JB K56 : YES, GO TRANSLATE CHAR
EB5B		2127	K43: : CTL-TABLE-TRANSLATE
EB5B B8C8E8		2128	MOV BX,OFFSET K9 : CTL TABLE SCAN
EB5E E9BC00		2129	JMP K63 : TRANSLATE_SCAN
		2130	
		2131	1----- NOT IN CONTROL SHIFT
		2132	
EB61		2133	K44: : NOT-CTL-SHIFT
EB61 3C47		2134	CMP AL,71 : TEST FOR KEYPAD REGION
EB63 732C		2135	JAE K48 : HANDLE KEYPAD REGION
EB65 F606170003		2136	TEST KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT
EB6A 745A		2137	JZ K54 : TEST FOR SHIFT STATE
		2138	
		2139	1----- UPPER CASE, HANDLE SPECIAL CASES
		2140	
EB6C 3C0F		2141	CMP AL,15 : BACK TAB KEY
EB6E 7505		2142	JNE K45 : NOT-BACK-TAB
EB70 B8000F		2143	MOV AX,15*256 : SET PSEUDO SCAN CODE
EB73 EB60		2144	JMP SHORT K57 : BUFFER_FILL
EB75		2145	K45: : NOT-BACK-TAB
EB75 3C37		2146	CMP AL,55 : PRINT SCREEN KEY
EB77 7509		2147	JNE K46 : NOT-PRINT-SCREEN
		2148	
		2149	1----- ISSUE INTERRUPT TO INDICATE PRINT SCREEN FUNCTION
		2150	
EB79 B020		2151	MOV AL,E01 : END OF CURRENT INTERRUPT
EB7D E620		2152	OUT 020H,AL : SO FURTHER THINGS CAN HAPPEN
EB7D C0D5		2153	INT 5H : ISSUE PRINT SCREEN INTERRUPT
EB7F E9D0FE		2154	JMP K27 : GO BACK WITHOUT EOI OCCURRING
EB82		2155	K46: : NOT-PRINT-SCREEN
EB82 3C3B		2156	CMP AL,59 : FUNCTION KEYS
EB84 7206		2157	JB K47 : NOT-UPPER-FUNCTION
EB86 B855E9		2158	MOV BX,OFFSET K12 : UPPER CASE PSEUDO SCAN CODES
EB89 E99100		2159	JMP K63 : TRANSLATE_SCAN
EB8C		2160	K47: : NOT-UPPER-FUNCTION
EB8C B81BE9		2161	MOV BX,OFFSET K11 : POINT TO UPPER CASE TABLE
EB8F EB40		2162	JMP SHORT K56 : OK, TRANSLATE THE CHAR
		2163	
		2164	1----- KEYPAD KEYS, MUST TEST NUM LOCK FOR DETERMINATION
		2165	
EB91		2166	K48: : KEYPAD-REGION
EB91 F606170020		2167	TEST KB_FLAG,NUM_STATE : ARE WE IN NUM_LOCK
EB96 7520		2168	JNZ K52 : TEST FOR SURE
EB98 F606170003		2169	TEST KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT : I ARE WE IN SHIFT STATE
EB9D 7520		2170	JNZ K53 : IF SHIFTED, REALLY NUM STATE
		2171	
		2172	1----- BASE CASE FOR KEYPAD
		2173	
EB9F		2174	K49: : BASE-CASE
EB9F 3C4A		2175	CMP AL,74 : SPECIAL CASE FOR A COUPLE OF KEYS
EBA1 740B		2176	JLE K50 : MINUS
EBA3 3C4E		2177	CMP AL,78
EBA5 740C		2178	JE K51
EBA7 2C47		2179	SUB AL,71
EBA9 B876E9		2180	MOV BX,OFFSET K15 : CONVERT ORIGIN
EBAE EB71		2181	JMP SHORT K64 : CONVERT TO PSEUDO SCAN
EBAE		2182	K50: : BASE CASE TABLE
EBAE B82D4A		2183	MOV AX,74*256+1 : MINUS
EBB1 EB22		2184	JMP SHORT K57 : BUFFER_FILL
EBB3		2185	K51: : PLUS
EBB3 B82B4E		2186	MOV AX,78*256+1 : BUFFER_FILL
EBB6 EB1D		2187	JMP SHORT K57
		2188	
		2189	1----- MIGHT BE NUM LOCK, TEST SHIFT STATUS
		2190	
EBB8		2191	K52: : ALMOST-NUM-STATE
EBB8 F606170003		2192	TEST KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT
EBBD 75E0		2193	JNZ K49 : SHIFTED TEMP OUT OF NUM STATE
EBBF		2194	K53: : REALLY NUM STATE
EBBF 2C46		2195	SUB AL,70 : CONVERT ORIGIN
EBD1 B869E9		2196	MOV BX,OFFSET K14 : NUM STATE TABLE
EBD4 EB0B		2197	JMP SHORT K56 : TRANSLATE_CHAR
		2198	
		2199	1----- PLAIN OLD LOWER CASE
		2200	
EBD6		2201	K54: : NOT-SHIFT
EBD6 3C3B		2202	CMP AL,59 : TEST FOR FUNCTION KEYS
EBD8 7204		2203	JB K55 : NOT-LOWER-FUNCTION
EBDA B000		2204	MOV AL,0 : SCAN CODE IN AH ALREADY
EBDC EB07		2205	JMP SHORT K57 : BUFFER_FILL
EBDE		2206	K55: : NOT-LOWER-FUNCTION
EBDE BBE1E8		2207	MOV BX,OFFSET K10 : LC TABLE
		2208	
		2209	1----- TRANSLATE THE CHARACTER
		2210	
EBD1		2211	K56: : TRANSLATE-CHAR
EBD1 FEC8		2212	DEC AL : CONVERT ORIGIN
EBD3 2ED7		2213	XLAT CS:K11 : CONVERT THE SCAN CODE TO ASCII
		2214	
		2215	1----- PUT CHARACTER INTO BUFFER
		2216	
EBD5		2217	K57: : BUFFER-FILL
EBD5 3CFF		2218	CMP AL,-1 : IS THIS AN IGNORE CHAR
EBD7 741F		2219	JE K59 : YES, DO NOTHING WITH IT
EBD9 80FCFF		2220	CMP AH,-1 : LOOK FOR -1 PSEUDO SCAN
EBDC 741A		2221	JE K59 : NEAR_INTERRUPT_RETURN
		2222	

LOC	OBJECT	LINE	SOURCE	(BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82
		2223	;	----- HANDLE THE CAPS LOCK PROBLEM
EBDE		2224		
EBDE F606170040		2225	K58:	;
EBE3 7420		2226	TEST	KB FLAG,CAPS_STATE ; BUFFER-FILL-NOTEST
		2227	JZ	K61 ; ARE WE IN CAPS LOCK STATE
		2228		;
		2229		SKIP IF NOT
		2230		
EBE5 F606170003		2231	TEST	KB FLAG,LEFT_SHIFT+RIGHT_SHIFT ; TEST FOR SHIFT STATE
EBEA 740F		2232	JZ	K60 ; IF NOT SHIFT, CONVERT LOWER TO UPPER
		2233		
		2234	;	----- CONVERT ANY UPPER CASE TO LOWER CASE
EBEC 3C41		2235		
EBEE 7215		2236	CMP	AL,'A' ; FIND OUT IF ALPHABETIC
EBF0 3C5A		2237	JB	K61 ; NOT_CAPS_STATE
EBF2 7111		2238	CMP	AL,'Z' ;
EBF4 0A20		2239	JA	K61 ; NOT_CAPS_STATE
EBF6 EB0D		2240	ADD	AL,'a'-'A' ; CONVERT TO LOWER CASE
EBF8		2241	JMP	SHORT K61 ; NOT_CAPS_STATE
EBF8 E95EFE		2242	K59:	;
		2243	JMP	K26 ; NEAR-INTERRUPT-RETURN
		2244		;
		2245	;	----- CONVERT ANY LOWER CASE TO UPPER CASE
EBFB		2246		
EBFB 3C61		2247	K60:	;
EBFD 7206		2248	CMP	AL,'a' ; LOWER-TO-UPPER
EBFF 3C7A		2249	JB	K61 ; FIND OUT IF ALPHABETIC
EC01 7102		2250	CMP	AL,'z' ; NOT_CAPS_STATE
EC03 2C20		2251	JA	K61 ;
EC05		2252	SUB	AL,'a'-'A' ; CONVERT TO UPPER CASE
EC05 8B1E1C00		2253	K61:	;
EC09 8BF3		2254	MOV	BX,BUFFER_TAIL ; NOT_CAPS-STATE
EC0B E863FC		2255	MOV	SI,BX ; GET THE END POINTER TO THE BUFFER
EC0E 3B1E1A00		2256	CALL	K4 ; SAVE THE VALUE
EC12 7413		2257	CMP	BX,BUFFER_HEAD ; ADVANCE THE TAIL
EC14 8904		2258	JE	K62 ; HAS THE BUFFER WRAPPED AROUND
EC16 891E1C00		2259	MOV	[SI],AX ; BUFFER_FULL_BEEP
EC1A E93CFE		2260	MOV	BUFFER_TAIL,BX ; STORE THE VALUE
		2261	JMP	K26 ; MOVE THE POINTER UP
		2262		;
		2263	;	----- TRANSLATE SCAN FOR PSEUDO SCAN CODES
EC1D		2264		
EC1D 2C3B		2265	K63:	;
EC1F		2266	SUB	AL,59 ; TRANSLATE-SCAN
EC1F 2ED7		2267	K64:	;
EC21 8AE0		2268	XLAT	CS:K9 ; CONVERT ORIGIN TO FUNCTION KEYS
EC23 B000		2269	MOV	AH,AL ; TRANSLATE-SCAN-ORGD
EC25 EBAE		2270	MOV	AL,0 ; CTL TABLE SCAN
		2271	JMP	K57 ; PUT VALUE INTO AH
		2272		;
		2273	KB_INT	ENDP ; ZERO ASCII CODE
		2274		;
		2275	;	----- BUFFER IS FULL, SOUND THE BEEPER
EC27		2276		
EC27 B020		2277	K62:	;
EC29 E620		2278	MOV	AL,E01 ; BUFFER-FULL-BEEP
EC2B BB8000		2279	OUT	20H,AL ; END OF INTERRUPT COMMAND
EC2E E461		2280	MOV	BX,080H ; SEND COMMAND TO INT CONTROL PORT
EC30 50		2281	IN	AL,KB_CTL ; NUMBER OF CYCLES FOR 1/12 SECOND TONE
EC31		2282	PUSH	AX ; GET CONTROL INFORMATION
EC31 24FC		2283	K65:	;
EC33 E661		2284	AND	AL,0FCH ; SAVE
EC35 B94800		2285	MOV	KB_CTL,AL ; BEEP-CYCLE
EC38		2286	MOV	CX,48H ; TURN OFF TIMER GATE AND SPEAKER DATA
EC38 E2FE		2287	K66:	;
EC3A 0C02		2288	LOOP	K66 ; SPEAKER OFF
EC3C E661		2289	OR	AL,2 ; TURN ON SPEAKER BIT
EC3E B94800		2290	OUT	KB_CTL,AL ; OUTPUT TO CONTROL
EC41		2291	MOV	CX,48H ; SET UP COUNT
EC41 E2FE		2292	K67:	;
EC43 4B		2293	LOOP	K67 ; ANOTHER HALF CYCLE
EC44 75EB		2294	DEC	BX ; TOTAL TIME COUNT
EC46 5B		2295	JNZ	K65 ; DO ANOTHER CYCLE
EC47 E661		2296	POP	AX ; RECOVER CONTROL
EC49 E912FE		2297	OUT	KB_CTL,AL ; OUTPUT THE CONTROL
		2298	JMP	K27 ;
		2299		
EC4C 20333031		2300	F1	DB ' 301',13,10 ; KEYBOARD ERROR
EC50 0D				
EC51 0A				
EC52 363031		2301	F3	DB '601',13,10 ; DISKETTE ERROR
EC55 0D				
EC56 0A				

```

2302
2303 ;-- INT 13 -----
2304 ; DISKETTE I/O
2305 ; THIS INTERFACE PROVIDES ACCESS TO THE 5 1/4 DISKETTE DRIVES
2306 ; INPUT
2307 ; (AH)=0 RESET DISKETTE SYSTEM
2308 ; HARD RESET TO NEC, PREPARE COMMAND, RECAL REQUIRED
2309 ; ON ALL DRIVES
2310 ; (AH)=1 READ THE STATUS OF THE SYSTEM INTO (AL)
2311 ; DISKETTE_STATUS FROM LAST OPERATION IS USED
2312 ;
2313 ; REGISTERS FOR READ/WRITE/VERIFY/FORMAT
2314 ; (DL) - DRIVE NUMBER (0-3 ALLOWED, VALUE CHECKED)
2315 ; (DH) - HEAD NUMBER (0-1 ALLOWED, NOT VALUE CHECKED)
2316 ; (CH) - TRACK NUMBER (0-39, NOT VALUE CHECKED)
2317 ; (CL) - SECTOR NUMBER (1-8, NOT VALUE CHECKED,
2318 ; NOT USED FOR FORMAT)
2319 ; (AL) - NUMBER OF SECTORS ( MAX = 8, NOT VALUE CHECKED, NOT USED
2320 ; FOR FORMAT)
2321 ; (ES:BX) - ADDRESS OF BUFFER ( NOT REQUIRED FOR VERIFY)
2322 ;
2323 ; (AH)=2 READ THE DESIRED SECTORS INTO MEMORY
2324 ; (AH)=3 WRITE THE DESIRED SECTORS FROM MEMORY
2325 ; (AH)=4 VERIFY THE DESIRED SECTORS
2326 ; (AH)=5
2327 ; FORMAT THE DESIRED TRACK
2328 ; FOR THE FORMAT OPERATION, THE BUFFER POINTER (ES,BX)
2329 ; MUST POINT TO THE COLLECTION OF DESIRED ADDRESS FIELDS
2330 ; FOR THE TRACK. EACH FIELD IS COMPOSED OF 4 BYTES,
2331 ; (C,H,R,N), WHERE C = TRACK NUMBER, H=HEAD NUMBER,
2332 ; R = SECTOR NUMBER, N= NUMBER OF BYTES PER SECTOR
2333 ; (00=128, 01=256, 02=512, 03=1024). THERE MUST BE ONE
2334 ; ENTRY FOR EVERY SECTOR ON THE TRACK. THIS INFORMATION
2335 ; IS USED TO FIND THE REQUESTED SECTOR DURING READ/WRITE
2336 ; ACCESS.
2337 ; DATA VARIABLE -- DISK POINTER
2338 ; DOUBLE WORD POINTER TO THE CURRENT SET OF DISKETTE PARAMETERS
2339 ; OUTPUT
2340 ; AH = STATUS OF OPERATION
2341 ; STATUS BITS ARE DEFINED IN THE EQUATES FOR
2342 ; DISKETTE_STATUS VARIABLE IN THE DATA SEGMENT OF THIS
2343 ; MODULE.
2344 ; CY = 0 SUCCESSFUL OPERATION (AH=0 ON RETURN)
2345 ; CY = 1 FAILED OPERATION (AH HAS ERROR REASON)
2346 ; FOR READ/WRITE/VERIFY
2347 ; DS,BX,DX,CH,CL PRESERVED
2348 ; AL = NUMBER OF SECTORS ACTUALLY READ
2349 ; ***** AL MAY NOT BE CORRECT IF TIME OUT ERROR OCCURS
2350 ; NOTE: IF AN ERROR IS REPORTED BY THE DISKETTE CODE, THE
2351 ; APPROPRIATE ACTION IS TO RESET THE DISKETTE, THEN RETRY
2352 ; THE OPERATION. ON READ ACCESSES, NO MOTOR START DELAY
2353 ; IS TAKEN, SO THAT THREE RETRIES ARE REQUIRED ON READS
2354 ; TO ENSURE THAT THE PROBLEM IS NOT DUE TO MOTOR
2355 ; START-UP.
2356 ;-----
2357 ; ASSUME CS:CODE,DS:DATA,ES:DATA
2358 ; ORG DEC59H
2359 ; DISKETTE_10 PROC FAR
2360 ; STI ; INTERRUPTS BACK ON
2361 ; PUSH BX ; SAVE ADDRESS
2362 ; PUSH CX
2363 ; PUSH MS
2364 ; PUSH SI ; SAVE SEGMENT REGISTER VALUE
2365 ; PUSH DI ; SAVE ALL REGISTERS DURING OPERATION
2366 ; PUSH BP
2367 ; PUSH DX
2368 ; MOV BP,SP ; SET UP POINTER TO HEAD PARM
2369 ; CALL DOS
2370 ; CALL J1
2371 ; MOV BX,4 ; GET THE MOTOR WAIT PARAMETER
2372 ; CALL GET_PARM
2373 ; MOV MOTOR_COUNT,AH ; SET THE TIMER COUNT FOR THE MOTOR
2374 ; MOV AH,DISKETTE_STATUS ; GET STATUS OF OPERATION
2375 ; CMP AH,1 ; SET THE CARRY FLAG TO INDICATE
2376 ; CMC ; SUCCESS OR FAILURE
2377 ; POP DX ; RESTORE ALL REGISTERS
2378 ; POP BP
2379 ; POP DI
2380 ; POP SI
2381 ; POP DS
2382 ; POP CX
2383 ; POP BX ; RECOVER ADDRESS
2384 ; RET 2 ; THROW AWAY SAVED FLAGS
2385 ; DISKETTE_10 ENDP
2386 ;
2387 ; J1 PROC NEAR
2388 ; MOV DH,AL ; SAVE # SECTORS IN DH
2389 ; AND MOTOR_STATUS,07FH ; INDICATE A READ OPERATION
2390 ; OR AH,AH ; AH=0
2391 ; JZ DISK_RESET ; AH=1
2392 ; DEC AH ; AH=2
2393 ; JZ DISK_STATUS
2394 ; MOV DISKETTE_STATUS,0 ; RESET THE STATUS INDICATOR
2395 ; CMP DL,4 ; TEST FOR DRIVE IN 0-3 RANGE
2396 ; JAE J3 ; ERROR IF ABOVE
2397 ; DEC AH ; AH=2
2398 ; JZ DISK_READ
2399 ; DEC AH ; AH=3
2400 ; JNZ J2 ; TEST_DISK_VERF
2401 ; JMP DISK_WRITE ; TEST_DISK_VERF
2402 ; J2: ; TEST_DISK_VERF
2403 ; DEC AH ; AH=4
2404 ; DISK_VERF
2405 ; DEC AH ; AH=5
2406 ; JZ DISK_FORMAT
2407 ; J3: ; BAD COMMAND
2408 ; MOV DISKETTE_STATUS,BAD_CMD ; ERROR CODE, NO SECTORS TRANSFERRED
2409 ; RET ; UNDEFINED OPERATION
2410 ; J1 ENDP
2411 ;

```

LOC	OBJECT	LINE	SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82
		2412	;----- RESET THE DISKETTE SYSTEM
		2413	
ECB7		2414	DISK_RESET PROC NEAR
ECB7 BAF203		2415	MOV DX,03F2H ; ADAPTER CONTROL PORT
ECBA FA		2416	CLI ; NO INTERRUPTS
ECBB A03F00		2417	MOV AL,MOTOR_STATUS ; WHICH MOTOR IS ON
ECBE B104		2418	MOV CL,4 ; SHIFT COUNT
ECCE D2E0		2419	SAL AL,CL ; MOVE MOTOR VALUE TO HIGH NYBBLE
ECCE A820		2420	TEST J5, 20H ; SELECT CORRESPONDING DRIVE
ECCE 150C		2421	JNZ J5 ; JUMP IF MOTOR ONE IS ON
ECCE A840		2422	TEST AL, 40H ;
ECCE 1506		2423	JNZ J4 ; JUMP IF MOTOR TWO IS ON
ECCE A880		2424	TEST AL, 80H ;
ECCE 7406		2425	JZ ; JUMP IF MOTOR ZERO IS ON
ECCE FEC0		2426	INC AL ;
ECDD		2427	J4; ;
ECDD FEC0		2428	J5; INC AL ;
ECDD		2429	J5; INC AL ;
ECDD FEC0		2430	J6; INC AL ;
ECDD		2431	J6; ;
ECDD 0C08		2432	OR AL,8 ; TURN ON INTERRUPT ENABLE
ECDE EE		2433	OUT DX,AL ; RESET THE ADAPTER
ECDE C6063E0000		2434	MOV MOV SEEK_STATUS,0 ; SEEK REQUIRED ON ALL DRIVES
ECDE C606410000		2435	MOV DISKETTE_STATUS,0 ; SET OK STATUS FOR DISKETTE
ECDE 0C04		2436	OR AL,4 ; TURN OFF RESET
ECDE EE		2437	OUT DX,AL ; TURN OFF THE RESET
ECDE FB		2438	STI ; REENABLE THE INTERRUPTS
ECDE E82A02		2439	CALL CHK_STAT_2 ; DO SENSE INTERRUPT STATUS
		2440	; FOLLOWING RESET
ECDE A04200		2441	MOV AL,NEC_STATUS ; IGNORE ERROR RETURN AND DO OWN TEST
ECDE 3CC0		2442	CMP AL,0C0H ; TEST FOR DRIVE READY TRANSITION
ECDE 7406		2443	JJ ; EVERYTHING OK
ECDE 800E410020		2444	OR DISKETTE_STATUS,BAD_NEC ; SET ERROR CODE
ECDE C3		2445	RET ;
		2446	
		2447	;----- SEND SPECIFY COMMAND TO NEC
		2448	
ECF5		2449	J7; ; DRIVE READY
ECF5 B403		2450	MOV AH,03H ; SPECIFY COMMAND
ECF7 E84701		2451	CALL NEC_OUTPUT ; OUTPUT THE COMMAND
ECFA B80100		2452	MOV BX,1 ; FIRST BYTE PARM IN BLOCK
ECFD E86C01		2453	CALL GET_PARM ; TO THE NEC CONTROLLER
ED00 B80300		2454	MOV BX,3 ; SECOND BYTE PARM IN BLOCK
ED03 E86601		2455	CALL GET_PARM ; TO THE NEC CONTROLLER
ED06		2456	J8; ; RESET RET
ED06 C3		2457	RET ; RETURN TO CALLER
		2458	DISK_RESET ENDP
		2459	
		2460	;----- DISKETTE STATUS ROUTINE
		2461	
ED07		2462	DISK_STATUS PROC NEAR
ED07 A04100		2463	MOV AL,DISKETTE_STATUS
ED0A C3		2464	RET
		2465	DISK_STATUS ENDP
		2466	
		2467	;----- DISKETTE READ
		2468	
ED0B		2469	DISK_READ PROC NEAR
ED0B B046		2470	MOV AL,046H ; READ COMMAND FOR DMA
ED0D		2471	J9; ; DISK_READ CONT
ED0D E8B801		2472	CALL DMA_SETUP ; SET UP THE DMA
ED10 B4E6		2473	MOV AH,0E6H ; SET UP RD COMMAND FOR NEC CONTROLLER
ED12 E836		2474	JMP SHORT RW_OPN ; GO DO THE OPERATION
		2475	DISK_READ ENDP
		2476	
		2477	;----- DISKETTE VERIFY
		2478	
ED14		2479	DISK_VERIFY PROC NEAR
ED14 B042		2480	MOV AL,042H ; VERIFY COMMAND FOR DMA
ED16 E8F5		2481	JMP J9 ; DO AS IF DISK READ
		2482	DISK_VERIFY ENDP
		2483	
		2484	;----- DISKETTE FORMAT
		2485	
ED18		2486	DISK_FORMAT PROC NEAR
ED18 800E3F0080		2487	OR MOTOR_STATUS,80H ; INDICATE WRITE OPERATION
ED1D B04A		2488	MOV AL,04AH ; WILL WRITE TO THE DISKETTE
ED1F E8A601		2489	CALL DMA_SETUP ; SET UP THE DMA
ED22 B44D		2490	MOV AH,04DH ; ESTABLISH THE FORMAT COMMAND
ED24 E824		2491	JMP SHORT RW_OPN ; DO THE OPERATION
ED26		2492	J10; ; CONTINUATION OF RW_OPN FOR FMT
ED26 B80700		2493	MOV BX,7 ; GET THE
ED29 E84001		2494	CALL GET_PARM ; BYTES/SECTOR VALUE TO NEC
ED2C B80900		2495	MOV BX,9 ; GET THE
ED2F E83A01		2496	CALL GET_PARM ; SECTORS/TRACK VALUE TO NEC
ED32 B80F00		2497	MOV BX,15 ; GET THE
ED35 E83401		2498	CALL GET_PARM ; GAP LENGTH VALUE TO NEC
ED38 B81100		2499	MOV BX,17 ; GET THE FILLER BYTE
ED3B E9AB00		2500	J16 ; TO THE CONTROLLER
		2501	DISK_FORMAT ENDP
		2502	
		2503	;----- DISKETTE WRITE ROUTINE
		2504	
ED3E		2505	DISK_WRITE PROC NEAR
ED3E 800E3F0080		2506	OR MOTOR_STATUS,80H ; INDICATE WRITE OPERATION
ED43 B04A		2507	MOV AL,04AH ; DMA WRITE COMMAND
ED45 E88001		2508	CALL DMA_SETUP ;
ED48 B4C5		2509	MOV AH,0C5H ; NEC COMMAND TO WRITE TO DISKETTE
		2510	DISK_WRITE ENDP
		2511	
		2512	;----- ALLOW WRITE ROUTINE TO FALL INTO RW_OPN
		2513	

```

LOC OBJECT      LINE  SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82

2514  ;-----
2515  ; RW_OPN
2516  ; THIS ROUTINE PERFORMS THE READ/WRITE/VERIFY OPERATION
2517  ;-----
ED4A  2518  RW_OPN  PROC    NEAR
ED4A  2519      JNC     J11      ; TEST FOR DMA ERROR
ED4C  C060410009 2520      MOV     DISKETTE_STATUS,DMA_BOUNDARY SET ERROR
ED51  B000      2521      MOV     AL,0      ; NO SECTORS TRANSFERRED
ED53  C3        2522      RET              ; RETURN TO MAIN ROUTINE
ED54  2523      J11:      ; DO RW_OPN
ED54  2524      PUSH    AX      ; SAVE THE COMMAND
ED54  2525
ED54  2526  ;----- TURN ON THE MOTOR AND SELECT THE DRIVE
ED54  2527
ED55  2528      PUSH    CX      ; SAVE THE T/S PARMS
ED56  8ACA      2529      MOV     CL,DL      ; GET DRIVE NUMBER AS SHIFT COUNT
ED56  B001      2530      MOV     AL,1      ; MASK FOR DETERMINING MOTOR BIT
ED5A  D2E0      2531      AL,CL      ; SHIFT THE MASK BIT
ED5C  FA        2532      CLI              ; NO INTERRUPTS WHILE DETERMINING
ED5C  2533      MOV     MOTOR_COUNT,OFFH ; SET LARGE COUNT DURING OPERATION
ED5D  C0604000FF 2534      TEST    AL,MOTOR_STATUS ; TEST THAT MOTOR FOR OPERATING
ED5E  8A063F00 2535      JNZ     J14      ; IF RUNNING, SKIP THE WAIT
ED66  7531      2536      AND     MOTOR_STATUS,0F0H ; TURN OFF ALL MOTOR BITS
ED68  80263F00F0 2537      OR      MOTOR_STATUS,AL ; TURN ON THE CURRENT MOTOR
ED6D  80663F00 2538      STI      ; INTERRUPTS BACK ON
ED71  FB        2539      MOV     AL,10H ; MASK BIT
ED72  B010      2540      SAL     AL,CL      ; DEVELOP BIT MASK FOR MOTOR ENABLE
ED74  D2E0      2541      OR      AL,DL      ; GET DRIVE SELECT BITS IN
ED76  0AC2      2542      OR      AL,0CH      ; NO RESET, ENABLE DMA/INT
ED78  0C0C      2543      PUSH    DX      ; SAVE REG
ED7A  52        2544      MOV     DX,03F2H ; CONTROL PORT ADDRESS
ED7B  BAF203    2545      OUT     DX,AL      ; RECOVER REGISTERS
ED7E  EE        2546      POP     DX
ED7F  5A        2547
ED7F  2548
ED7F  2549  ;----- WAIT FOR MOTOR IF WRITE OPERATION
ED7F  2550
ED80  F6063F00B0 2551      TEST    MOTOR_STATUS,80H ; IS THIS A WRITE
ED85  7412      2552      JZ      J14      ; NO, CONTINUE WITHOUT WAIT
ED87  BB1400    2553      MOV     BX,20      ; GET THE MOTOR WAIT
ED8A  E8D0F000 2554      CALL    GET_PARM ; PARAMETER
ED8D  0AE4      2555      OR      AH,AH      ; TEST FOR NO WAIT
ED8F  2556      J12:      ; TEST WAIT TIME
ED8F  7408      2557      JZ      J14      ; EXIT WITH TIME EXPIRED
ED91  2BC9      2558      SUB     CX,CX      ; SET UP 1/8 SECOND LOOP TIME
ED93  2559      J13:      ; WAIT FOR THE REQUIRED TIME
ED93  E2FE      2560      LOOP    J13      ; DECREMENT TIME VALUE
ED95  FECC      2561      DEC     AH          ; ARE WE DONE YET
ED97  EBF6      2562      JMP     J14:      ; MOTOR RUNNING
ED99  FB        2563      STI      ; INTERRUPTS BACK ON FOR BYPASS WAIT
ED9A  59        2564      POP     CX
ED9A  2565
ED9A  2566  ;----- DO THE SEEK OPERATION
ED9A  2567
ED9B  E8DF00    2568      CALL    SEEK
ED9E  58        2569      POP     AX
ED9F  8AFC      2570      MOV     BH,AH      ; MOVE TO CORRECT TRACK
ED9F  8AFC      2571      MOV     BH,AH      ; RECOVER COMMAND
EDA1  B600      2572      MOV     DH,0      ; SAVE COMMAND IN BH
EDA3  724B      2573      JC      J17      ; SET NO SECTORS READ IN CASE OF ERROR
EDA5  BEF0ED9D 2574      MOV     SI,OFFSET J17 ; IF ERROR, THEN EXIT AFTER MOTOR OFF
EDA5  59        2575      PUSH    SI          ; DUMMY RETURN ON STACK FOR NEC_OUTPUT
EDA5  2576      ; SO THAT IT WILL RETURN TO MOTOR OFF
EDA5  2577      ; LOCATION
EDA5  2578  ;----- SEND OUT THE PARAMETERS TO THE CONTROLLER
EDA5  2579
EDA5  2580      CALL    NEC_OUTPUT ; OUTPUT THE OPERATION COMMAND
EDA5  2581      MOV     AH,[BP+1] ; GET THE CURRENT HEAD NUMBER
EDA5  2582      SAL     AH,1      ; MOVE IT TO BIT 2
EDA5  2583      SAL     AH,1      ;
EDA5  2584      AND     AH,4      ; ISOLATE THAT BIT
EDA5  2585      OR      AH,DL      ; OR IN THE DRIVE NUMBER
EDA5  2586      CALL    NEC_OUTPUT ;
EDA5  2587
EDA5  2588  ;----- TEST FOR FORMAT COMMAND
EDA5  2589
EDBC  80FF4D    2590      CMP     BH,04DH      ; IS THIS A FORMAT OPERATION
EDBF  7503      2591      JNE     J15      ; NO, CONTINUE WITH R/W/V
EDC1  E962FF    2592      JMP     J15:      ; IF SO, HANDLE SPECIAL
EDC4  2593
EDC4  8AE5      2594      MOV     AH,CH      ; CYLINDER NUMBER
EDC6  E87800    2595      CALL    NEC_OUTPUT ; HEAD NUMBER FROM STACK
EDC9  8A6601    2596      MOV     AH,[BP+1] ;
EDCC  E87200    2597      CALL    NEC_OUTPUT ; SECTOR NUMBER
EDCF  8AE1      2598      MOV     AH,CL      ;
EDD1  E86000    2599      CALL    NEC_OUTPUT ; BYTES/SECTOR PARM FROM BLOCK
EDD4  BB0700    2600      MOV     BX,7      ; TO THE NEC
EDD7  E89200    2601      CALL    GET_PARM ; EOT PARM FROM BLOCK
EDDA  BB0900    2602      MOV     BX,9      ; TO THE NEC
EDDD  E88C00    2603      CALL    GET_PARM ; CAP LENGTH PARM FROM BLOCK
EDE0  BB0800    2604      MOV     BX,11     ; TO THE NEC
EDE3  E88600    2605      CALL    GET_PARM ; DTL PARM FROM BLOCK
EDE6  BB0D00    2606      MOV     BX,13     ; RW_OPN FINISH
EDE9  2607      ; TO THE NEC
EDE9  E88000    2608      CALL    GET_PARM ; CAN NOW DISCARD THAT DUMMY
EDEB  5E        2609      POP     SI          ; RETURN ADDRESS
EDEB  2610
EDEB  2611
EDEB  2612  ;----- LET THE OPERATION HAPPEN
EDEB  2613
EDEB  2614      CALL    WAIT_INT ; WAIT FOR THE INTERRUPT
EDEB  2615      J17:      ; MOTOR OFF
EDF0  7245      2616      JC      J21      ; LOOK FOR ERROR
EDF2  E87401    2617      CALL    RESULTS ; GET THE NEC STATUS
EDF5  723F      2618      JC      J20      ; LOOK FOR ERROR
EDF5  2619
EDF5  2620  ;----- CHECK THE RESULTS RETURNED BY THE CONTROLLER
EDF5  2621
EDF7  FC        2622      CLD
EDF8  BE4200    2623      MOV     SI,OFFSET NEC_STATUS ; SET THE CORRECT DIRECTION
EDFB  AC        2624      LODS    NEC_STATUS ; POINT TO STATUS FIELD
EDFC  24C0      2625      AND     AL,0C0H      ; GET ST0
EDFE  743B      2626      JZ      J22      ; TEST FOR NORMAL TERMINATION
EE00  3C40      2627      CMP     AL,040H      ; TEST FOR ABNORMAL TERMINATION
EE02  7529      2628      JNZ     J18      ; NOT ABNORMAL, BAD NEC
EE02  2629

```

LOC	OBJECT	LINE	SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82
		2630	;----- ABNORMAL TERMINATION, FIND OUT WHY
		2631	
EE04 AC		2632	LOADS NEC_STATUS ; GET ST1
EE05 D0E0		2633	SAL AL,T ; TEST FOR EOT FOUND
EE07 B404		2634	MOV AH,RECORD_NOT_FND ;
EE09 7224		2635	JC J19 ; RW_FAIL
EE0B D0E0		2636	SAL AL,I ;
EE0D D0E0		2637	SAL AL,I ; TEST FOR CRC ERROR
EE0F B410		2638	MOV AH,BAD_CRC ;
EE11 721C		2639	J19 ; RW_FAIL
EE13 D0E0		2640	SAL AL,I ; TEST FOR DMA OVERRUN
EE15 B408		2641	MOV AH,BAD_DMA ;
EE17 7216		2642	J19 ; RW_FAIL
EE19 D0E0		2643	SAL AL,I ;
EE1B D0E0		2644	SAL AL,I ; TEST FOR RECORD NOT FOUND
EE1D B404		2645	MOV AH,RECORD_NOT_FND ;
EE1F 720E		2646	JC J19 ; RW_FAIL
EE21 D0E0		2647	SAL AL,I ;
EE23 B403		2648	MOV AH,WRITE_PROTECT ; TEST FOR WRITE_PROTECT
EE25 7208		2649	JC J19 ; RW_FAIL
EE27 D0E0		2650	SAL AL,I ; TEST MISSING ADDRESS MARK
EE29 B402		2651	MOV AH,BAD_ADDR_MARK ;
EE2B 7202		2652	JC J19 ; RW_FAIL
		2653	
		2654	;----- NEC MUST HAVE FAILED
		2655	
EE2D B420		2656	J1B: MOV AH,BAD_NEC ; RW-NEC-FAIL
EE2F		2657	
EE2F 08264100		2658	J19: OR DISKETTE_STATUS,AH ; RW-FAIL
EE33 E87801		2659	CALL NUM_TRANS ; HOW MANY WERE REALLY TRANSFERRED
EE36		2660	
EE36 C3		2661	J20: RET ; RW_ERR
EE37		2662	
EE37 E82F01		2663	J21: CALL RESULTS ; RW_ERR RES
EE3A C3		2664	RET ; FLUSH THE RESULTS BUFFER
		2665	
		2666	;----- OPERATION WAS SUCCESSFUL
		2667	
EE3B		2668	
EE3B E87001		2669	J22: ; OPN_OK
EE3E 32E4		2670	CALL NUM_TRANS ; HOW MANY GOT MOVED
EE40 C3		2671	XOR AH,AH ; NO ERRORS
		2672	RET
		2673	RW_OPN ENDP
		2674	;
		2675	; NEC_OUTPUT
		2676	;
		2677	THIS ROUTINE SENDS A BYTE TO THE NEC CONTROLLER AFTER TESTING
		2678	FOR CORRECT DIRECTION AND CONTROLLER READY THIS ROUTINE WILL
		2679	TIME OUT IF THE BYTE IS NOT ACCEPTED WITHIN A REASONABLE
		2680	AMOUNT OF TIME, SETTING THE DISKETTE STATUS ON COMPLETION.
		2681	INPUT (AH) BYTE TO BE OUTPUT
		2682	OUTPUT
		2683	CY = 0 SUCCESS
		2684	CY = 1 FAILURE -- DISKETTE STATUS UPDATED
		2685	IF A FAILURE HAS OCCURRED, THE RETURN IS MADE ONE LEVEL
		2686	HIGHER THAN THE CALLER OF NEC_OUTPUT.
		2687	THIS REMOVES THE REQUIREMENT OF TESTING AFTER EVERY
		2688	CALL OF NEC_OUTPUT.
		2689	(AL) DESTROYED
		2690	;
EE41		2691	NEC_OUTPUT PROC NEAR
EE41 52		2692	PUSH DX ; SAVE REGISTERS
EE42 51		2693	PUSH CX ;
EE43 BAF403		2694	MOV DX,03F4H ; STATUS PORT
EE46 33C9		2695	XOR CX,CX ; COUNT FOR TIME OUT
EE48		2696	J23: ;
EE48 EC		2697	IN AL,DX ; GET STATUS
EE49 A840		2698	TEST AL,040H ; TEST DIRECTION BIT
EE4B 740C		2699	JZ J25 ; DIRECTION OK
EE4D E2F9		2700	LOOP J23 ;
EE4F		2701	J24: ; TIME_ERROR
EE4F 800E410080		2702	OR DISKETTE_STATUS,TIME_OUT
EE54 59		2703	POP CX ;
EE55 5A		2704	POP DX ; SET ERROR CODE AND RESTORE REGS
EE56 58		2705	POP AX ; DISCARD THE RETURN ADDRESS
EE57 F9		2706	STC ; INDICATE ERROR TO CALLER
EE58 C3		2707	RET ;
EE59		2708	J25: ;
EE59 33C9		2709	XOR CX,CX ; RESET THE COUNT
EE5B		2710	J26: ;
EE5B EC		2711	IN AL,DX ; GET THE STATUS
EE5C A880		2712	TEST AL,080H ; IS IT READY
EE5E 7504		2713	JNZ J27 ; YES, GO OUTPUT
EE60 E2F9		2714	LOOP J26 ; COUNT DOWN AND TRY AGAIN
EE62 EBEB		2715	JMP J24 ; ERROR CONDITION
EE64		2716	J27: ; OUTPUT
EE64 8AC4		2717	MOV AL,AH ; GET BYTE TO OUTPUT
EE66 B2F5		2718	MOV DL,OF5H ; DATA PORT (3F5)
EE68 EE		2719	OUT DX,AL ; OUTPUT THE BYTE
EE69 59		2720	POP CX ; RECOVER REGISTERS
EE6A 5A		2721	POP DX ;
EE6B C3		2722	RET ; CY = 0 FROM TEST INSTRUCTION
		2723	NEC_OUTPUT ENDP

LINE SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82

5-142 PC-XT System BIOS (11/08/82)

LINE SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82

SECTION 5

```

LOC OBJECT      LINE  SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82

2904 :-----
2905 : WAIT_INT
2906 : THIS ROUTINE WAITS FOR AN INTERRUPT TO OCCUR. A TIME OUT
2907 : ROUTINE TAKES PLACE DURING THE WAIT, SO THAT AN ERROR MAY BE
2908 : RETURNED IF THE DRIVE IS NOT READY.
2909 : INPUT
2910 : NONE
2911 : OUTPUT
2912 : CY = 0 SUCCESS
2913 : CY = 1 FAILURE -- DISKETTE_STATUS IS SET ACCORDINGLY
2914 : (AX) DESTROYED
2915 :-----
EF33 2916 WAIT_INT      PROC      NEAR
EF33 FB 2917 STI             BX
EF34 53 2918 PUSH        CX
EF35 51 2919 MOV          BL,2
EF36 B302 2920 XOR          CX,CX
EF38 33C9 2921 J36:
EF3A F6063E0080 2922 TEST        SEEK_STATUS,INT_FLAG
EF3F 750C 2923 JNZ          J37
EF41 E2F7 2924 LOOP        J36
EF43 F6C8 2925 BL
EF45 75F3 2926 JNZ          J36
EF47 800E410080 2927 OR          DISKETTE_STATUS,TIME_OUT
EF4C F9 2928 STC
EF4D 2929 J37:
EF4D 9C 2930 J37:
EF4E 80263E007F 2931 PUSHF
EF4F 5D 2932 AND          SEEK_STATUS,NOT INT_FLAG
EF54 59 2933 POP         CX
EF55 5B 2934 POP         CX
EF56 C3 2935 RET
2936 : RECOVER REGISTERS
2937 : GOOD RETURN CODE COMES
2938 : FROM TEST INST
2939 :-----
2940 : DISK_INT
2941 : THIS ROUTINE HANDLES THE DISKETTE INTERRUPT
2942 : INPUT
2943 : NONE
2944 : OUTPUT
2945 : THE INTERRUPT FLAG IS SET IS SEEK_STATUS
2946 :-----
EF57 2947 ORG          0EF57H
EF57 FB 2948 DISK_INT      PROC      FAR
EF58 1E 2949 STI             DS
EF59 50 2950 PUSH        AX
EF5A E8FC0A 2951 CALL        DDS
EF5D 800E3E0080 2952 OR          SEEK_STATUS,INT_FLAG
EF62 B020 2953 MOV          AL,20H
EF64 E620 2954 OUT          20H,AL
EF66 58 2955 POP         AX
EF67 1F 2956 POP         DS
EF68 CF 2957 IRET
2958 : RECOVER SYSTEM
2959 : RETURN FROM INTERRUPT
2960 :-----
2961 : RESULTS
2962 : THIS ROUTINE WILL READ ANYTHING THAT THE NEC CONTROLLER HAS
2963 : TO SAY FOLLOWING AN INTERRUPT.
2964 : INPUT
2965 : NONE
2966 : OUTPUT
2967 : CY = 0 SUCCESSFUL TRANSFER
2968 : CY = 1 FAILURE -- TIME OUT IN WAITING FOR STATUS
2969 : NEC_STATUS AREA HAS STATUS BYTE LOADED INTO IT
2970 : (AH) DESTROYED
2971 :-----
EF69 2972 RESULTS      PROC      NEAR
EF69 FC 2973 CLD
EF6A BF4200 2974 MOV          DI,OFFSET NEC_STATUS
EF6D 51 2975 PUSH        CX
EF6E 52 2976 DX
EF6F 53 2977 PUSH        BX
EF70 B307 2978 MOV          BL,7
2979 : MAX STATUS BYTES
2980 :-----
2981 : WAIT FOR REQUEST FOR MASTER
2982 J38:
2983 XOR          CX,CX
2984 MOV          DX,03F4H
2985 J39:
2986 IN          AL,DX
2987 TEST        AL,080H
2988 JNZ          J40A
2989 LOOP        J39
2990 OR          DISKETTE_STATUS,TIME_OUT
2991 J40:
2992 STC
2993 POP         BX
2994 POP         DX
2995 POP         CX
2996 RET
2997 : RESULTS_ERROR
2998 : SET ERROR RETURN
2999 :-----
3000 : TEST THE DIRECTION BIT
3001 J40A:
3002 IN          AL,DX
3003 TEST        AL,040H
3004 J41:
3005 JNZ          J42
3006 OR          DISKETTE_STATUS,BAD_NEC
3007 JMP          J40
3008 : RESULTS_ERROR
3009 :-----
3010 : READ IN THE STATUS
3011 J42:
3012 INC         DX
3013 IN          AL,DX
3014 MOV          [DI],AL
3015 INC         DI
3016 MOV          CX,10
3017 LOOP        J43
3018 DEC         DX
3019 IN          AL,DX
3020 TEST        AL,010H
3021 : INPUT_STAT
3022 : POINT AT DATA PORT
3023 : GET THE DATA
3024 : STORE THE BYTE
3025 : INCREMENT THE POINTER
3026 : LOOP TO KILL TIME FOR NEC
3027 : POINT AT STATUS PORT
3028 : GET STATUS
3029 : TEST FOR NEC STILL BUSY

```


LOC OBJECT	LINE	SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82
EFA2 7406	3020	JZ J44 ; RESULTS DONE
EFA4 FECB	3021	DEC BL ; DECREMENT THE STATUS COUNTER
EFA6 75CA	3022	JNZ J38 ; GO BACK FOR MORE
EFA8 EBE3	3023	JMP J41 ; CHIP HAS FAILED
	3024	
	3025	;----- RESULT OPERATION IS DONE
	3026	
EFAA	3027	J44:
EFAA 5B	3028	POP BX
EFA4 5A	3029	POP DX
EFA6 59	3030	POP CX ; RECOVER REGISTERS
EFA8 C3	3031	RET ; GOOD RETURN CODE FROM TEST INST
	3032	;
	3033	NUM_TRANS ;
	3034	; THIS ROUTINE CALCULATES THE NUMBER OF SECTORS THAT ;
	3035	WERE ACTUALLY TRANSFERRED TO/FROM THE DISKETTE ;
	3036	INPUT ;
	3037	(CH) = CYLINDER OF OPERATION ;
	3038	(CL) = START SECTOR OF OPERATION ;
	3039	OUTPUT ;
	3040	(AL) = NUMBER ACTUALLY TRANSFERRED ;
	3041	NO OTHER REGISTERS MODIFIED ;
	3042	;
EFAE	3043	NUM_TRANS PROC NEAR
EFAE A04500	3044	MOV AL,NEC_STATUS+3 ; GET CYLINDER ENDED UP ON
EFB1 3AC5	3045	MOV AL,CH ; SAME AS WE STARTED
EFB3 A04700	3046	MOV AL,NEC_STATUS+5 ; GET ENDING SECTOR
EFB6 740A	3047	JZ J45 ; IF ON SAME CYL, THEN NO ADJUST
EFB8 BB0B00	3048	MOV BX,8
EFB8 EBAEFE	3049	CALL GET_PARM ; GET EOT VALUE
EFB8 8AC4	3050	MOV AL,AH ; INTO AL
EFC0 FEC0	3051	INC AL ; USE EOT+1 FOR CALCULATION
EFC2	3052	
EFC2 2AC1	3053	J45: SUB AL,CL ; SUBTRACT START FROM END
EFC4 C3	3054	RET
	3055	NUM_TRANS ENDP
	3056	RESULTS ENDP
	3057	;
	3058	DISK_BASE ;
	3059	THIS IS THE SET OF PARAMETERS REQUIRED FOR DISKETTE OPERATION. ;
	3060	THEY ARE POINTED AT BY THE DATA VARIABLE DISK_POINTER. TO ;
	3061	MODIFY THE PARAMETERS, BUILD ANOTHER PARAMETER BLOCK AND POINT ;
	3062	DISK_POINTER TO IT. ;
	3063	;
EFCT	3064	ORG 0EFC7H
EFCT	3065	DISK_BASE LABEL BYTE
EFCT CF	3066	DB 11001111B ; SRT=C, HD UNLOAD=0F - 1ST SPECIFY BYTE
EFCE 02	3067	DB 2 ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
EFC9 25	3068	DB MOTOR_WAIT ; WAIT AFTER OPN TIL MOTOR OFF
EFC4 02	3069	DB 2 ; 512 BYTES/SECTOR
EFCE 08	3070	DB 8 ; EOT (LAST SECTOR ON TRACK)
EFC2 2A	3071	DB 02AH ; GAP LENGTH
EFC0 FF	3072	DB 0FFH ; OTL
EFC2 2A	3073	DB 050H ; GAP LENGTH FOR FORMAT
EFCF F6	3074	DB 0F6H ; FILL BYTE FOR FORMAT
EFD0 19	3075	DB 25 ; HEAD SETTLE TIME (1/1111 SECONDS)
EFD1 04	3076	DB 4 ; MOTOR START TIME (1/8 SECONDS)
	3077	

```

3078 :--- INT 17 -----
3079 : PRINTER_IO
3080 : THIS ROUTINE PROVIDES COMMUNICATION WITH THE PRINTER
3081 : INPUT
3082 : (AH)=0 PRINT THE CHARACTER IN (AL)
3083 : ON RETURN, AH=1 IF CHARACTER COULD NOT BE PRINTED
3084 : (TIME OUT). OTHER BITS SET AS ON NORMAL STATUS CALL
3085 : (AH)=1 INITIALIZE THE PRINTER PORT
3086 : RETURNS WITH (AH) SET WITH PRINTER STATUS
3087 : (AH)=2 READ THE PRINTER STATUS INTO (AH)
3088 :
3089 : 7 6 5 4 3 2-1 0 TIME OUT
3090 : | | | | | | | UNUSED
3091 : | | | | | | | ERROR
3092 : | | | | | | |
3093 : | | | | | | |
3094 : | | | | | | |
3095 : | | | | | | |
3096 : | | | | | | |
3097 : | | | | | | |
3098 : | | | | | | |
3099 : | | | | | | |
3100 : | | | | | | |
3101 : | | | | | | |
3102 : | | | | | | |
3103 : | | | | | | |
3104 : | | | | | | |
3105 : | | | | | | |
3106 : | | | | | | |
3107 : | | | | | | |
3108 : | | | | | | |
3109 : | | | | | | |
3110 : | | | | | | |
3111 : | | | | | | |
3112 : | | | | | | |
3113 : | | | | | | |
3114 : | | | | | | |
3115 : | | | | | | |
3116 : | | | | | | |
3117 : | | | | | | |
3118 : | | | | | | |
3119 : | | | | | | |
3120 : | | | | | | |
3121 : | | | | | | |
3122 : | | | | | | |
3123 : | | | | | | |
3124 : | | | | | | |
3125 : | | | | | | |
3126 : | | | | | | |
3127 : | | | | | | |
3128 : | | | | | | |
3129 : | | | | | | |
3130 : | | | | | | |
3131 : | | | | | | |
3132 : | | | | | | |
3133 : | | | | | | |
3134 : | | | | | | |
3135 : | | | | | | |
3136 : | | | | | | |
3137 : | | | | | | |
3138 : | | | | | | |
3139 : | | | | | | |
3140 : | | | | | | |
3141 : | | | | | | |
3142 : | | | | | | |
3143 : | | | | | | |
3144 : | | | | | | |
3145 : | | | | | | |
3146 : | | | | | | |
3147 : | | | | | | |
3148 : | | | | | | |
3149 : | | | | | | |
3150 : | | | | | | |
3151 : | | | | | | |
3152 : | | | | | | |
3153 : | | | | | | |
3154 : | | | | | | |
3155 : | | | | | | |
3156 : | | | | | | |
3157 : | | | | | | |
3158 : | | | | | | |
3159 : | | | | | | |
3160 : | | | | | | |
3161 : | | | | | | |
3162 : | | | | | | |
3163 : | | | | | | |
3164 : | | | | | | |
3165 : | | | | | | |
3166 : | | | | | | |
3167 : | | | | | | |
3168 : | | | | | | |
3169 : | | | | | | |
3170 : | | | | | | |
3171 : | | | | | | |
3172 : | | | | | | |
3173 : | | | | | | |
3174 : | | | | | | |
3175 : | | | | | | |
3176 : | | | | | | |
3177 : | | | | | | |
3178 : | | | | | | |
3179 : | | | | | | |
3180 : | | | | | | |
3181 : | | | | | | |
3182 : | | | | | | |

```

(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, 40BH 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

PRINTER_IO PROC FAR

STI ; INTERRUPTS BACK ON

PUSH DS ; SAVE SEGMENT

PUSH DX

PUSH SI

PUSH CX

PUSH BX

DDI

MOV SI,DX ; GET PRINTER PARM

MOV BL,PRINT_TIM_OUT[SI] ; LOAD TIME-OUT PARM

SHL SI,1 ; WORD OFFSET INTO TABLE

MOV DX,PRINTER_BASE[SI] ; GET BASE ADDRESS FOR PRINTER CARD

OR DX,DX ; TEST DX FOR ZERO

JZ B1 ; INDICATING NO PRINTER

JZ B1 ; RETURN

OR AH,AH ; TEST FOR (AH)=0

JZ B2 ; PRINT AL

DEC AH ; TEST FOR (AH)=1

JZ B3 ; INIT_PRT

DEC AH ; TEST FOR (AH)=2

JZ B5 ; PRINTER STATUS

JZ B5 ; RETURN

B1: POP BX ; RECOVER REGISTERS

POP CX ; RECOVER REGISTERS

POP SI

POP DX

DS

IRET

PRINTER STATUS

B2: PUSH AX ; SAVE VALUE TO PRINT

OUT DX,AL ; OUTPUT CHAR TO PORT

INC DX ; POINT TO STATUS PORT

B3: SUB CX,CX ; WAIT_BUSY

B3_1: IN AL,DX ; GET STATUS

MOV AH,AL ; STATUS TO AH ALSO

TEST AL,80H ; IS THE PRINTER CURRENTLY BUSY

JNZ B3 ; OUT_STROBE

DEC BL ; TRY AGAIN

OR AH,1 ; DROP LOOP COUNT

JNZ B3 ; GO TILL TIMEOUT ENDS

AND AH,0F9H ; SET ERROR FLAG

JMP SHORT B7 ; TURN OFF THE OTHER BITS

B4: MOV AL,0DH ; RETURN WITH ERROR FLAG SET

INC DX ; OUT_STROBE

OUT DX,AL ; SET THE STROBE HIGH

POP AX ; STROBE IS BIT 0 OF PORT C OF 8255

PRINTER STATUS

B5: PUSH AX ; SAVE AL REG

B6: MOV DX,PRINTER_BASE[SI] ; GET PRINTER STATUS

INC DX ; STATUS SET

MOV AL,DX ; GET CHARACTER INTO AL

AND AH,0F8H ; FLIP A COUPLE OF BITS

B7: POP DX ; RETURN FROM ROUTINE

MOV AL,DL

XOR AH,40H

JMP B1

LOC OBJECT

LINE SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82

```

3183
3184 ;----- INITIALIZE THE PRINTER PORT
3185
3186 B8:
3187     PUSH    AX                ; SAVE AL
3188     INC     DX                ; POINT TO OUTPUT PORT
3189     INC     DX
3190     MOV     AL,8              ; SET INIT LINE LOW
3191     OUT     DX,AL
3192     MOV     AX,1000
3193 B9:
3194     DEC     AX                ; INIT LOOP
3195     JNZ     B9                ; LOOP FOR RESET TO TAKE
3196     MOV     AL,0CH            ; INIT LOOP
3197                                ; NO INTERRUPTS, NON AUTO LF,
3198                                ; INIT HIGH
3199     OUT     DX,AL
3200     JMP     B6                ; PRT_STATUS_1
3201     PRINTER_10
3202     ENDP

```

```

3202
3203 --- INT 10 ---
3204 VIDEO_10
3205 THESE ROUTINES PROVIDE THE CRT INTERFACE
3206 THE FOLLOWING FUNCTIONS ARE PROVIDED:
3207 (AH)=0 SET MODE (AL) CONTAINS MODE VALUE
3208 (AL)=0 40X25 BW (POWER ON DEFAULT)
3209 (AL)=1 40X25 COLOR
3210 (AL)=2 80X25 BW
3211 (AL)=3 80X25 COLOR
3212 GRAPHICS MODES
3213 (AL)=4 320X200 COLOR
3214 (AL)=5 320X200 BW
3215 (AL)=6 640X200 BW
3216 CRT MODE=7 80X25 BW CARD (USED INTERNAL TO VIDEO ONLY)
3217 *** NOTE BW MODES OPERATE SAME AS COLOR MODES, BUT
3218 COLOR BURST IS NOT ENABLED
3219 (AH)=1 SET CURSOR TYPE
3220 (CH) = BITS 4-0 = START LINE FOR CURSOR
3221 ** HARDWARE WILL ALWAYS CAUSE BLIN
3222 ** SETTING BIT 5 OR 6 WILL CAUSE ERRATIC
3223 BLINKING OR NO CURSOR AT ALL
3224 (CL) = BITS 4-0 END LINE FOR CURSOR
3225 (AH)=2 SET CURSOR POSITION
3226 (DH,DL) = ROW,COLUMN (0,0) IS UPPER LEFT
3227 (BH) = PAGE NUMBER (MUST BE 0 FOR GRAPHICS MODES)
3228 (AH)=3 READ CURSOR POSITION
3229 (BH) = PAGE NUMBER (MUST BE 0 FOR GRAPHICS MODES)
3230 ON EXIT (DH,DL) = ROW,COLUMN OF CURRENT CURSOR
3231 (CH,CL) CURSOR MODE CURRENTLY SET
3232 (AH)=4 READ LIGHT PEN POSITION
3233 ON EXIT:
3234 (AH) = 0 -- LIGHT PEN SWITCH NOT DOWN/NOT TRIGGERED
3235 (AH) = 1 -- VALID LIGHT PEN VALUE IN REGISTERS
3236 (DH,DL) = ROW,COLUMN OF CHARACTER LP POSN
3237 (CH) = RASTER LINE (0-199)
3238 (BX) = PIXEL COLUMN (0-319,639)
3239 (AH)=5 SELECT ACTIVE DISPLAY PAGE (VALID ONLY FOR ALPHA MODES)
3240 (AL)=NEW PAGE VAL (0-7 FOR MODES 0&1, 0-3 FOR MODES 2&3)
3241 (AH)=6 SCROLL ACTIVE PAGE UP
3242 (AL) = NUMBER OF LINES, INPUT LINES BLANKED AT BOTTOM
3243 OF WINDOW
3244 AL = 0 MEANS BLANK ENTIRE WINDOW
3245 (CH,CL) = ROW,COLUMN OF UPPER LEFT CORNER OF SCROLL
3246 (DH,DL) = ROW,COLUMN OF LOWER RIGHT CORNER OF SCROLL
3247 (BH) = ATTRIBUTE TO BE USED ON BLANK LINE
3248 (AH)=7 SCROLL ACTIVE PAGE DOWN
3249 (AL) = NUMBER OF LINES, INPUT LINES BLANKED AT TOP
3250 OF WINDOW
3251 AL = 0 MEANS BLANK ENTIRE WINDOW
3252 (CH,CL) = ROW,COLUMN OF UPPER LEFT CORNER OF SCROLL
3253 (DH,DL) = ROW,COLUMN OF LOWER RIGHT CORNER OF SCROLL
3254 (BH) = ATTRIBUTE TO BE USED ON BLANK LINE
3255
3256 CHARACTER HANDLING ROUTINES
3257
3258 (AH) = 8 READ ATTRIBUTE/CHARACTER AT CURRENT CURSOR POSITION
3259 (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY)
3260 ON EXIT:
3261 (AL) = CHAR READ
3262 (AH) = ATTRIBUTE OF CHARACTER READ (ALPHA MODES ONLY)
3263 (AH) = 9 WRITE ATTRIBUTE/CHARACTER AT CURRENT CURSOR POSITION
3264 (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY)
3265 (CX) = COUNT OF CHARACTERS TO WRITE
3266 (AL) = CHAR TO WRITE
3267 (BL) = ATTRIBUTE OF CHARACTER (ALPHA)/COLOR OF CHAR
3268 (GRAPHICS)
3269 SEE NOTE ON WRITE DOT FOR BIT 7 OF BL = 1.
3270 (AH) = 10 WRITE CHARACTER ONLY AT CURRENT CURSOR POSITION
3271 (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY)
3272 (CX) = COUNT OF CHARACTERS TO WRITE
3273 (AL) = CHAR TO WRITE
3274 FOR READ/WRITE CHARACTER INTERFACE WHILE IN GRAPHICS MODE, THE
3275 CHARACTERS ARE FORMED FROM A CHARACTER GENERATOR IMAGE
3276 MAINTAINED IN THE SYSTEM ROM. ONLY THE 1ST 128 CHARS
3277 ARE CONTAINED THERE. TO READ/WRITE THE SECOND 128
3278 CHARS, THE USER MUST INITIALIZE THE POINTER AT
3279 INTERRUPT 1FH (LOCATION 0007CH) TO POINT TO THE 1K BYTE
3280 TABLE CONTAINING THE CODE POINTS FOR THE SECOND
3281 128 CHARS (128-255).
3282 FOR WRITE CHARACTER INTERFACE IN GRAPHICS MODE, THE REPLICATION
3283 FACTOR CONTAINED IN (CX) ON ENTRY WILL PRODUCE VALID
3284 RESULTS ONLY FOR CHARACTERS CONTAINED ON THE SAME ROW.
3285 CONTINUATION TO SUCCEEDING LINES WILL NOT PRODUCE
3286 CORRECTLY.
3287
3288 GRAPHICS INTERFACE
3289 (AH) = 11 SET COLOR PALETTE
3290 (BH) = PALETTE COLOR ID BEING SET (0-127)
3291 (BL) = COLOR VALUE TO BE USED WITH THAT COLOR ID
3292 NOTE: FOR THE CURRENT COLOR CARD, THIS ENTRY POINT
3293 HAS MEANING ONLY FOR 320X200 GRAPHICS.
3294 COLOR ID = 0 SELECTS THE BACKGROUND COLOR (0-15)
3295 COLOR ID = 1 SELECTS THE PALETTE TO BE USED:
3296 0 = GREEN(1)/RED(2)/YELLOW(3)
3297 1 = CYAN(1)/MAGENTA(2)/WHITE(3)
3298 IN 40X25 OR 80X25 ALPHA MODES, THE VALUE SET
3299 FOR PALETTE COLOR 0 INDICATES THE
3300 BORDER COLOR TO BE USED (VALUES 0-31,
3301 WHERE 16-31 SELECT THE HIGH INTENSITY
3302 BACKGROUND SET.
3303
3304 (AH) = 12 WRITE DOT
3305 (DX) = ROW NUMBER
3306 (CX) = COLUMN NUMBER
3307 (AL) = COLOR VALUE
3308 IF BIT 7 OF AL = 1, THEN THE COLOR VALUE IS
3309 EXCLUSIVE OR'D WITH THE CURRENT CONTENTS OF
3310 THE DOT
3311 (AH) = 13 READ DOT
3312 (DX) = ROW NUMBER
3313 (CX) = COLUMN NUMBER
3314 (AL) RETURNS THE DOT READ

```

```

3314 :
3315 : ASCII TELETYPE ROUTINE FOR OUTPUT
3316 :
3317 : (AH) = 14 WRITE TELETYPE TO ACTIVE PAGE
3318 : (AL) = CHAR TO WRITE
3319 : (BL) = FOREGROUND COLOR IN GRAPHICS MODE
3320 : NOTE -- SCREEN WIDTH IS CONTROLLED BY PREVIOUS MODE SET
3321 :
3322 : (AH) = 15 CURRENT VIDEO STATE
3323 : RETURNS THE CURRENT VIDEO STATE
3324 : (AL) = MODE CURRENTLY SET 1 SEE AH=0 FOR EXPLANATION
3325 : (AH) = NUMBER OF CHARACTER COLUMNS ON SCREEN
3326 : (BH) = CURRENT ACTIVE DISPLAY PAGE
3327 :
3328 : CS,SS,DS,ES,BX,CX,DX PRESERVED DURING CALL
3329 : ALL OTHERS DESTROYED
3330 : -----
3331 : ASSUME CS:CODE,DS:DATA,ES:VIDEO_RAM
3332 : ORG 0F045H
3333 M1 LABEL WORD ; TABLE OF ROUTINES WITHIN VIDEO 1/0
3334 DW OFFSET SET_MODE
3335 DW OFFSET SET_CTYPE
3336 DW OFFSET SET_CPOS
3337 DW OFFSET READ_CURSOR
3338 DW OFFSET READ_LPEN
3339 DW OFFSET ACT_DISP_PAGE
3340 DW OFFSET SCROLL_UP
3341 DW OFFSET SCROLL_DOWN
3342 DW OFFSET READ_AC_CURRENT
3343 DW OFFSET WRITE_AC_CURRENT
3344 DW OFFSET WRITE_C_CURRENT
3345 DW OFFSET SET_COLOR
3346 DW OFFSET WRITE_DOT
3347 DW OFFSET READ_DOT
3348 DW OFFSET WRITE_TTY
3349 DW OFFSET VIDEO_STATE
3350 MIL EQU $-M1
3351
3352 : ORG 0F065H
3353 VIDEO_10 PROC NEAR
3354 STI
3355 CLD ; INTERRUPTS BACK ON
3356 PUSH ES ; SET DIRECTION FORWARD
3357 PUSH DS ; SAVE SEGMENT REGISTERS
3358 PUSH DX
3359 PUSH CX
3360 PUSH BX
3361 PUSH SI
3362 PUSH DI
3363 PUSH AX
3364 MOV AL,AH ; SAVE AX VALUE
3365 XOR AH,AH ; GET INTO LOW BYTE
3366 SAL AX,1 ; ZERO TO HIGH BYTE
3367 MOV SI,AX ; *2 FOR TABLE LOOKUP
3368 CMP AX,MIL ; PUT INTO SI FOR BRANCH
3369 JB M2 ; TEST FOR WITHIN RANGE
3370 POP AX ; BRANCH AROUND BRANCH
3371 JMP VIDEO_RETURN ; THROW AWAY THE PARAMETER
3372 M2: ; DO NOTHING IF NOT IN RANGE
3373 CALL DDS
3374 MOV AX,0B800H ; SEGMENT FOR COLOR CARD
3375 MOV DI,EQUIP_FLAG ; GET EQUIPMENT SETTING
3376 AND DI,30H ; ISOLATE CRT SWITCHES
3377 CMP DI,30H ; IS SETTING FOR BW CARD?
3378 JNE M3
3379 MOV AH,0B0H ; SEGMENT FOR BW CARD
3380 M3:
3381 MOV ES,AX ; SET UP TO POINT AT VIDEO RAM AREAS
3382 POP AX ; RECOVER VALUE
3383 MOV AH,CRT_MODE ; GET CURRENT MODE INTO AH
3384 JMP WORD PTR CS:[SI+OFFSET M1]
3385 VIDEO_10 ENDP

```

```

LOC OBJECT      LINE  SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82

3386 ;-----
3387 ; SET_MODE
3388 ; THIS ROUTINE INITIALIZES THE ATTACHMENT TO
3389 ; THE SELECTED MODE. THE SCREEN IS BLANKED.
3390 ; INPUT
3391 ; (AL) = MODE SELECTED (RANGE 0-9)
3392 ; OUTPUT
3393 ; NONE
3394 ;-----
3395
3396 ;----- TABLES FOR USE IN SETTING OF MODE
3397
3398 ; ORG 0F0A4H
3399 VIDEO_PARMs LABEL BYTE
3400 ;----- INIT_TABLE
3401 DB 38H,28H,20H,0AH,1FH,6,19H ; SET UP FOR 40X25

FOA4 38
FOA5 28
FOA6 2D
FOA7 0A
FOA8 1F
FOA9 06
FOAA 19
FOAB 1C
FOAC 02
FOAD 07
FOAE 06
FOAF 07
FOB0 00
FOB1 00
FOB2 00
FOB3 00
FOB4 71
FOB5 50
FOB6 5A
FOB7 0A
FOB8 1F
FOB9 06
FOBA 19
FOBB 1C
FOBC 02
FOBD 07
FOBE 06
FOBF 07
FOC0 00
FOC1 00
FOC2 00
FOC3 00
FOC4 38
FOC5 28
FOC6 2D
FOC7 0A
FOC8 7F
FOC9 06
FOCA 64
FOCB 70
FOCC 02
FOCD 01
FOCE 06
FOCF 07
FOD0 00
FOD1 00
FOD2 00
FOD3 00
FOD4 61
FOD5 50
FOD6 52
FOD7 0F
FOD8 19
FOD9 06
FODA 19
FODB 19
FODC 02
FODD 0D
FODE 0B
FODF 0C
FOE0 00
FOE1 00
FOE2 00
FOE3 00
FOE4 0008
FOE6 0010
FOE8 0040
FOEA 0040
FOEC 28
FOED 28
FOEE 50
FOEF 50
FOF0 28
FOF1 28
FOF2 50
FOF3 50
FOF4 2C
FOF5 28
FOF6 2D
FOF7 29
FOF8 2A
FOF9 2E
FOFA 1E
FOFB 29

3402 DB 1CH,2,7,6,7
3403 DB 0,0,0,0
3404 M4 EQU %-VIDEO_PARMs
3405
3406 DB 71H,50H,5AH,0AH,1FH,6,19H ; SET UP FOR 80X25
3407 DB 1CH,2,7,6,7
3408 DB 0,0,0,0
3409
3410 DB 38H,28H,20H,0AH,7FH,6,64H ; SET UP FOR GRAPHICS
3411 DB 70H,2,1,6,7
3412 DB 0,0,0,0
3413
3414 DB 61H,50H,52H,0FH,19H,6,19H ; SET UP FOR 80X25 B&W CARD
3415 DB 19H,2,0DH,0BH,0CH
3416 DB 0,0,0,0
3417
3418 M5 LABEL WORD ; TABLE OF REGEN LENGTHS
3419 DW 2048 ; 40X25
3420 DW 4096 ; 80X25
3421 DW 16384 ; GRAPHICS
3422 DW 16384
3423
3424 ;----- COLUMNS
3425
3426 M6 LABEL BYTE
3427 DB 40,40,80,80,40,40,80,80

3428 ;----- C_REG_TAB
3429
3430 M7 LABEL BYTE ; TABLE OF MODE SETS
3431 DB 2CH,28H,2DH,29H,2AH,2EH,1EH,29H
3432

```

```

3433
3434 SET_MODE PROC NEAR
3435 MOV DX,03D4H ; ADDRESS OF COLOR CARD
3436 MOV BL,0 ; MODE SET FOR COLOR CARD
3437 CMP DI,30H ; IS BW CARD INSTALLED
3438 JNE M8 ; OK WITH COLOR
3439 MOV AL,7 ; INDICATE BW CARD MODE
3440 MOV DI,0B4H ; ADDRESS OF BW CARD (3B4)
3441 INC BL ; MODE SET FOR BW CARD
3442 M8:
3443 MOV AH,AL ; SAVE MODE IN AH
3444 CRT_MODE,AL ; SAVE IN GLOBAL VARIABLE
3445 MOV ADDR_6845,DX ; SAVE ADDRESS OF BASE
3446 PUSH DS ; SAVE POINTER TO DATA SEGMENT
3447 PUSH AX ; SAVE MODE
3448 PUSH DX ; SAVE OUTPUT PORT VALUE
3449 ADD DX,4 ; POINT TO CONTROL REGISTER
3450 MOV AL,BL ; GET MODE SET FOR CARD
3451 OUT DX,AL ; RESET VIDEO
3452 POP DX ; BACK TO BASE REGISTER
3453 SUB AX,AX ; SET UP FOR ABSO SEGMENT
3454 MOV DS,AX ; ESTABLISH VECTOR TABLE ADDRESSING
3455 ASSUME DS:ABSO
3456 LDS BX,PARAM_PTR ; GET POINTER TO VIDEO PARMS
3457 POP AX ; RECOVER PARMS
3458 ASSUME DS:CODE
3459 MOV CX,M4
3460 CMP AH,2 ; LENGTH OF EACH ROW OF TABLE
3461 JC M9 ; DETERMINE WHICH ONE TO USE
3462 ADD BX,CX ; MODE IS 0 OR 1
3463 CMP AH,4 ; MOVE TO NEXT ROW OF INIT TABLE
3464 JC M9 ; MODE IS 2 OR 3
3465 ADD BX,CX ; MOVE TO GRAPHICS ROW OF INIT_TABLE
3466 CMP AH,7 ; MODE IS 4,5, OR 6
3467 JC M9 ; MOVE TO BW CARD ROW OF INIT_TABLE
3468 ADD BX,CX
3469
3470 ;----- BX POINTS TO CORRECT ROW OF INITIALIZATION TABLE
3471
3472 M9:
3473 PUSH AX ; OUT INIT
3474 XOR AH,AH ; SAVE MODE IN AH
3475 ; AH WILL SERVE AS REGISTER
3476 ; NUMBER DURING LOOP
3477 ;----- LOOP THROUGH TABLE, OUTPUTTING REG ADDRESS, THEN VALUE FROM TABLE
3478
3479 M10:
3480 MOV AL,AH ; INIT LOOP
3481 OUT DX,AL ; GET 6845 REGISTER NUMBER
3482 INC DX ; POINT TO DATA PORT
3483 INC AH ; NEXT REGISTER VALUE
3484 MOV AL,[BX] ; GET TABLE VALUE
3485 OUT DX,AL ; OUT TO CHIP
3486 INC BX ; NEXT IN TABLE
3487 DEC DX ; BACK TO POINTER REGISTER
3488 LOOP M10 ; DO THE WHOLE TABLE
3489 POP AX ; GET MODE BACK
3490 POP DS ; RECOVER SEGMENT VALUE
3491 ASSUME DS:DATA
3492
3493 ;----- FILL REGEN AREA WITH BLANK
3494
3495 XOR DI,D1 ; SET UP POINTER FOR REGEN
3496 MOV CRT_START,D1 ; START ADDRESS SAVED IN GLOBAL
3497 MOV ACTIVE_PAGE,0 ; SET PAGE VALUE
3498 MOV CX,B192 ; NUMBER OF WORDS IN COLOR CARD
3499 CMP AH,4 ; TEST FOR GRAPHICS
3500 JC M12 ; NO GRAPHICS_INIT
3501 CMP AH,7 ; TEST FOR BW CARD
3502 JE M11 ; BW CARD INIT
3503 XOR AX,AX ; FILL FOR GRAPHICS MODE
3504 JMP SHORT M13 ; CLEAR BUFFER
3505 M11:
3506 MOV CH,08H ; BUFFER SIZE ON BW CARD
3507 M12:
3508 MOV AX,' '*7*256 ; NO GRAPHICS_INIT
3509 M13:
3510 REP STOSW ; FILL CHAR FOR ALPHA
3511 ; FILL THE REGEN BUFFER WITH BLANKS
3512
3513 ;----- ENABLE VIDEO AND CORRECT PORT SETTING
3514
3515 MOV CURSOR_MODE,607H ; SET CURRENT CURSOR MODE
3516 MOV AL,CRT_MODE ; GET THE MODE
3517 XOR AH,AH ; INTO AX REGISTER
3518 MOV SI,AX ; TABLE POINTER, INDEXED BY MODE
3519 MOV DX,ADDR_6845 ; PREPARE TO OUTPUT TO
3520 ; VIDEO ENABLE PORT
3521 ADD DX,4
3522 MOV AL,CS:[SI+OFFSET M7] ; SET VIDEO ENABLE PORT
3523 OUT DX,AL ; SAVE THAT VALUE
3524 CRT_MODE_SET,AL
3525
3526 ;----- DETERMINE NUMBER OF COLUMNS, BOTH FOR ENTIRE DISPLAY
3527 ;----- AND THE NUMBER TO BE USED FOR TTY INTERFACE
3528
3529 MOV AL,CS:[SI+OFFSET M6]
3530 XOR AH,AH
3531 MOV CRT_COLS,AX ; NUMBER OF COLUMNS IN THIS SCREEN
3532
3533 ;----- SET CURSOR POSITIONS
3534
3535 AND SI,0EH
3536 MOV CX,CS:[SI+OFFSET M5] ; WORD OFFSET INTO CLEAR LENGTH TABLE
3537 CRT_LEN,CX ; LENGTH TO CLEAR
3538 MOV DI,OFFSET CURSOR_POSN ; SAVE LENGTH OF CRT -- NOT USED FOR BW
3539 PUSH DS ; CLEAR ALL CURSOR POSITIONS
3540 POP DS
3541 XOR AX,AX ; ESTABLISH SEGMENT
3542 REP STOSW ; ADDRESSING
3543 ; FILL WITH ZEROS
3544

```

```

3543
3544 ;----- SET UP OVERSCAN REGISTER
3545
F1B5 42      3546      INC      DX              ; SET OVERSCAN PORT TO A DEFAULT
F1B6 B030    3547      MOV      AL,30H          ; VALUE OF 30H FOR ALL MODES
3548          3548          ; EXCEPT 640X200
3549          3549      CMP      CRT_MODE,6      ; SEE IF THE MODE IS 640X200 BW
F1BD 7502    3550      JNZ      M14              ; IF IT ISNT 640X200, THEN GOTO REGULAR
F1BF B03F    3551      MOV      AL,3FH          ; IF IT IS 640X200, THEN PUT IN 3FH
F1C1          3552          ;
F1C1 EE      3553      OUT      DX,AL          ; OUTPUT THE CORRECT VALUE TO 3D9 PORT
F1C2 A26600  3554      MOV      CRT_PALETTE,AL    ; SAVE THE VALUE FOR FUTURE USE
3555
3556 ;----- NORMAL RETURN FROM ALL VIDEO RETURNS
3557
F1C5          3558      VIDEO_RETURN:
F1C5 5F      3559      POP      DI
F1C6 5E      3560      POP      SI
F1C7 5B      3561      POP      BX
3562          3562          ; VIDEO_RETURN_C
F1C8 59      3563      POP      CX
F1C9 5A      3564      POP      DX
F1CA 1F      3565      POP      DS
F1CB 07      3566      POP      ES
F1CC CF      3567      IRET             ; RECOVER SEGMENTS
3568          3568          ; ALL DONE
3569          3569      SET_MODE      ENDP
3570          3570      ;-----
3571          3571      ; SET_CTYPE
3572          3572      ; THIS ROUTINE SETS THE CURSOR VALUE
3573          3573      ; INPUT
3574          3574      ; (CX) HAS CURSOR VALUE CH-START LINE, CL-STOP LINE
3575          3575      ; OUTPUT
3576          3576      ; NONE
3577          3577      ;-----
F1CD          3578      SET_CTYPE      PROC      NEAR
F1CD B40A    3579      MOV      AH,10              ; 6845 REGISTER FOR CURSOR SET
F1CF 890E000 3580      MOV      CURSOR_MODE,CX      ; SAVE IN DATA AREA
F1D3 E80200  3581      CALL     M16              ; OUTPUT CX REG
F1D6 EBED    3582      JMP      VIDEO_RETURN
3583          3583      ;----- THIS ROUTINE OUTPUTS THE CX REGISTER TO THE 6845 REGS NAMED IN AH
3584          3584      M16:
F1D8          3585      MOV      DX,ADDR_6845      ; ADDRESS REGISTER
F1D8 8B166300 3586      MOV      AL,AH              ; GET VALUE
F1DC 8AC4    3587      OUT      DX,AL              ; REGISTER SET
F1DE EE      3588      INC      DX              ; DATA REGISTER
F1DF 42      3589      MOV      AL,CH          ; DATA
F1E0 8AC5    3590      OUT      DX,AL          ; DATA
F1E2 EE      3591      DEC      DX
F1E3 4A      3592      MOV      AL,AH
F1E4 8AC4    3593      INC      AL
F1E6 FEC0    3594      OUT      DX,AL          ; POINT TO OTHER DATA REGISTER
F1E8 EE      3595      INC      DX              ; SET FOR SECOND REGISTER
F1E9 42      3596      MOV      AL,CL          ; SECOND DATA VALUE
F1EA 8AC1    3597      OUT      DX,AL          ; SECOND DATA VALUE
F1EC EE      3598      RET
F1ED C3      3599      ; ALL DONE
3600          3600      SET_CTYPE      ENDP
3601          3601      ;-----
3602          3602      ; SET_CPOS
3603          3603      ; THIS ROUTINE SETS THE CURRENT CURSOR
3604          3604      ; POSITION TO THE NEW X-Y VALUES PASSED
3605          3605      ; INPUT
3606          3606      ; DX - ROW,COLUMN OF NEW CURSOR
3607          3607      ; BH - DISPLAY PAGE OF CURSOR
3608          3608      ; OUTPUT
3609          3609      ; CURSOR IS SET AT 6845 IF DISPLAY PAGE
3610          3610      ; IS CURRENT DISPLAY
3611          3611      ;-----
F1EE          3612      SET_CPOS      PROC      NEAR
F1EE 8ACF    3613      MOV      CL,BH
F1F0 32ED    3614      XOR      CH,CH          ; ESTABLISH LOOP COUNT
F1F2 DIE1    3615      SAL      CX,1              ; WORD OFFSET
F1F4 8BF1    3616      MOV      SI,CX          ; USE INDEX REGISTER
F1F6 895450  3617      MOV      [SI+OFFSET_CURSOR_POSN],DX ; SAVE THE POINTER
F1F9 383E6200 3618      CMP      ACTIVE_PAGE,BH      ; SET_CPOS_RETURN
F1FD 1505    3619      JNZ      M17              ; GET_ROW/COLUMN TO AX
F1FF 8BC2    3620      MOV      AX,DX          ; CURSOR SET
F201 E80200  3621      CALL     M18              ; SET_CP05_RETURN
F204          3622      M17:
F204 EBBF    3623      JMP      VIDEO_RETURN
3624          3624      SET_CPOS      ENDP
3625          3625      ;----- SET CURSOR POSITION, AX HAS ROW/COLUMN FOR CURSOR
3626          3626      M18
F206          3627      PROC      NEAR
F206 E87C00  3628      CALL     POSITION          ; DETERMINE LOCATION IN REGEN BUFFER
F209 8BC8    3629      MOV      CX,AX
F20B 030E4E00 3630      ADD      CX,CRT_START
F20F D1F9    3631      ADD      CX,CRT_START
F211 B40E    3632      SAR      CX,1
F213 E8C2FF  3633      MOV      AH,14
F216 C3      3634      CALL     M16              ; REGISTER NUMBER FOR CURSOR
3635          3635      RET
3636          3636      M18      ENDP

```



```

3637 :-----
3638 : ACT_DISP_PAGE
3639 : THIS ROUTINE SETS THE ACTIVE DISPLAY PAGE, ALLOWING THE
3640 : FULL USE OF THE RAM SET ASIDE FOR THE VIDEO ATTACHMENT
3641 : INPUT
3642 : AL HAS THE NEW ACTIVE DISPLAY PAGE
3643 : OUTPUT
3644 : THE 6845 IS RESET TO DISPLAY THAT PAGE
3645 :-----
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 9B CBW ; CONVERT AL TO WORD
F21F 50 PUSH AX ; SAVE PAGE VALUE
F220 F7E1 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 8B4700 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
F238 :-----
3663 ACT_DISP_PAGE ENDP
3664 :-----
3665 : READ_CURSOR
3666 : THIS ROUTINE READS THE CURRENT CURSOR VALUE FROM THE
3667 : 6845, FORMATS IT, AND SENDS IT BACK TO THE CALLER
3668 : INPUT
3669 : BH - PAGE OF CURSOR
3670 : OUTPUT
3671 : DX - ROW, COLUMN OF THE CURRENT CURSOR POSITION
3672 : CX - CURRENT CURSOR MODE
3673 :-----
F239 READ_CURSOR PROC NEAR
F239 8ADF MOV BL,BH
F23B 32FF XOR BH,BH
F23D D1E3 SAL BX,1 ; WORD OFFSET
F23F 8B5700 MOV DX,[BX+OFFSET CURSOR_POSN]
F242 8B0E0000 MOV CX,CURSOR_MODE
F246 5F POP DI
F247 5E POP SI
F248 5B POP BX
F249 5B POP AX ; DISCARD SAVED CX AND DX
F24A 5B POP AX
F24B 1F POP DS
F24C 07 POP ES
F24D CF IRET
F24E :-----
3688 READ_CURSOR ENDP
3689 :-----
3690 : SET COLOR
3691 : THIS ROUTINE WILL ESTABLISH THE BACKGROUND COLOR, THE OVERSCAN
3692 : COLOR, AND THE FOREGROUND COLOR SET FOR MEDIUM RESOLUTION
3693 : GRAPHICS
3694 : INPUT
3695 : (BH) HAS COLOR ID
3696 : IF BH=0, THE BACKGROUND COLOR VALUE IS SET
3697 : FROM THE LOW BITS OF BL (0-31)
3698 : IF BH=1, THE PALETTE SELECTION IS MADE
3699 : BASED ON THE LOW BIT OF BL
3700 : 0=GREEN, RED, YELLOW FOR COLORS 1,2,3
3701 : 1=BLUE, CYAN, MAGENTA FOR COLORS 1,2,3
3702 : (BL) HAS THE COLOR VALUE TO BE USED
3703 : OUTPUT
3704 : THE COLOR SELECTION IS UPDATED
3705 :-----
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
F25B :-----
3713 :----- HANDLE COLOR 0 BY SETTING THE BACKGROUND COLOR
3714 :-----
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: OUT DX,AL ; 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
F268 :-----
3723 :----- HANDLE COLOR 1 BY SELECTING THE PALETTE TO BE USED
3724 :-----
F26A 24D0 M20: 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
F273 :-----
3731 SET_COLOR ENDP
3732 :-----
3733 : VIDEO STATE
3734 : RETURNS THE CURRENT VIDEO STATE IN AX
3735 : AH = NUMBER OF COLUMNS ON THE SCREEN
3736 : AL = CURRENT VIDEO MODE
3737 : BH = CURRENT ACTIVE PAGE
3738 :-----
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
F283 :-----
3747 VIDEO_STATE ENDP

```

5-154 PC-XT System BIOS (11/08/82)

LOC	OBJECT	LINE	SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82
F323 1F		3864	POP DS ; FOR BOTH POINTERS
F324 80FB00		3865	CMP BL,0 ; 0 SCROLL MEANS BLANK FIELD
F327 C3		3866	RET ; RETURN WITH FLAGS SET
		3867	SCROLL_POSITION ENDP
		3868	
		3869	----- MOVE_ROW
		3870	
F328		3871	N10 PROC NEAR
F328 8ACA		3872	MOV CL,DL ; GET # OF COLS TO MOVE
F32A 56		3873	PUSH SI
F32B 57		3874	PUSH DI
F32C F3		3875	REP MOVSW ; SAVE START ADDRESS
F32D A5			
F32E 5F		3876	POP DI ; MOVE THAT LINE ON SCREEN
F32F 5E		3877	POP SI
F330 C3		3878	RET ; RECOVER ADDRESSES
		3879	N10 ENDP
		3880	
		3881	----- CLEAR_ROW
		3882	
F331		3883	N11 PROC NEAR
F331 8ACA		3884	MOV CL,DL ; GET # COLUMNS TO CLEAR
F333 57		3885	PUSH DI
F334 F3		3886	REP STOSW ; STORE THE FILL CHARACTER
F335 AB			
F336 5F		3887	POP DI
F337 C3		3888	RET
		3889	N11 ENDP
		3890	-----
		3891	SCROLL_DOWN
		3892	THIS ROUTINE MOVES THE CHARACTERS WITHIN A
		3893	DEFINED BLOCK DOWN ON THE SCREEN, FILLING THE
		3894	TOP LINES WITH A DEFINED CHARACTER
		3895	INPUT
		3896	(AH) = CURRENT CRT MODE
		3897	(AL) = NUMBER OF LINES TO SCROLL
		3898	(CX) = UPPER LEFT CORNER OF REGION
		3899	(DX) = LOWER RIGHT CORNER OF REGION
		3900	(BH) = FILL CHARACTER
		3901	(DS) = DATA SEGMENT
		3902	(ES) = REGEN SEGMENT
		3903	OUTPUT
		3904	NONE -- SCREEN IS SCROLLED
		3905	-----
F338		3906	SCROLL_DOWN PROC NEAR
F338 FD		3907	STD
F339 8AD8		3908	MOV BL,AL ; DIRECTION FOR SCROLL DOWN
F338 80FC04		3909	CMP AH,4 ; LINE COUNT TO BL
F33E 7208		3910	JC N12 ; TEST FOR GRAPHICS
F340 80FC07		3911	CMP AH,7 ; TEST FOR BW CARD
F343 7403		3912	JE N12
F345 E9A601		3913	JMP GRAPHICS_DOWN
F348		3914	
F348 53		3915	N12: PUSH BX ; CONTINUE DOWN
F349 8BC2		3916	MOV AX,DX ; SAVE ATTRIBUTE IN BH
F34B E894FF		3917	CALL SCROLL_POSITION ; LOWER RIGHT CORNER
F34E 7420		3918	JZ N16 ; GET REGEN LOCATION
F350 2BF0		3919	SUB SI,AX ; SI IS FROM ADDRESS
F352 8AE6		3920	MOV AH,0H ; GET TOTAL # ROWS
F354 2AE3		3921	SUB AH,BL ; COUNT TO MOVE IN SCROLL
F356		3922	
F356 E8CFFF		3923	N13: CALL N10 ; MOVE ONE ROW
F359 2BF5		3924	SUB SI,BP
F35B 2BFD		3925	SUB DI,BP
F35D FECC		3926	DEC AH
F35F 75F5		3927	JNZ N13
F361		3928	
F361 58		3929	N14: POP AX ; RECOVER ATTRIBUTE IN AH
F362 B020		3930	MOV AL,' '
F364		3931	N15: CALL N11 ; CLEAR ONE ROW
F364 E8CAFF		3932	SUB DI,BP ; GO TO NEXT ROW
F367 2BFD		3933	DEC BL
F369 FECB		3934	JNZ N15
F36B 75F7		3935	JMP N5 ; SCROLL_END
F36D E95AFF		3936	
F370		3937	N16: MOV BL,DH
F370 8ADE		3938	JMP N14
F372 EBED		3939	
		3940	SCROLL_DOWN ENDP

```

LOC OBJECT          LINE  SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82

3941 :-----
3942 : READ_AC_CURRENT :
3943 : THIS ROUTINE READS THE ATTRIBUTE AND CHARACTER :
3944 : AT THE CURRENT CURSOR POSITION AND RETURNS THEM :
3945 : TO THE CALLER :
3946 : INPUT :
3947 : (AH) = CURRENT CRT MODE :
3948 : (BH) = DISPLAY PAGE ( ALPHA MODES ONLY ) :
3949 : (DS) = DATA SEGMENT :
3950 : (ES) = REGEN SEGMENT :
3951 : OUTPUT :
3952 : (AL) = CHAR READ :
3953 : (AH) = ATTRIBUTE READ :
3954 :-----
3955 : ASSUME CS:CODE,DS:DATA,ES:DATA
F374 :
F374 80FC04 3956 READ_AC_CURRENT PROC NEAR
F377 7208 3957 CMP AH,4 : IS THIS GRAPHICS
F379 80FC07 3958 JC P1 :
F37C 7403 3959 CMP AH,7 : IS THIS BW CARD
F37E E9A802 3960 JE P1 :
F381 E81A00 3961 JMP GRAPHICS_READ :
F384 8BF3 P1: 3962 CALL FIND_POSITION : READ_AC_CONTINUE
3963 :
3964 MOV SI,BX : ESTABLISH ADDRESSING IN SI
3965 :
3966 :----- WAIT FOR HORIZONTAL RETRACE
3967 :
F386 8B166300 3968 MOV DX,ADDR_6845 : GET BASE ADDRESS
F38A 83C206 3969 ADD DX,6 : POINT AT STATUS PORT
F38D 06 3970 PUSH ES :
F38E 1F 3971 POP DS :
F38F :
F38F EC 3972 P2: IN AL,DX : WAIT FOR RETRACE LOW
F390 A801 3973 TEST AL,1 : GET STATUS
F392 75FB 3974 JNZ P2 : IS HORIZ RETRACE LOW
F394 FA 3975 CLJ P2 : WAIT UNTIL IT IS
F395 :
F395 EC 3976 P3: IN AL,DX : NO MORE INTERRUPTS
F396 A801 3977 TEST AL,1 : WAIT FOR RETRACE HIGH
F398 74FB 3978 JZ P3 : GET STATUS
F39A AD 3979 LODSW : IS IT HIGH
F39B E927FE 3980 JMP VIDEO_RETURN : WAIT UNTIL IT IS
3981 :
3982 READ_AC_CURRENT ENDP : GET THE CHAR/ATTR
3983 :
3984 :
3985 FIND_POSITION PROC NEAR
F39E 8ACF 3986 MOV CL,BH : DISPLAY PAGE TO CX
F3A0 32ED 3987 XOR CH,CH :
F3A2 8BF1 3988 MOV SI,CX : MOVE TO SI FOR INDEX
F3A4 D1E6 3989 SAL SI,1 : * 2 FOR WORD OFFSET
F3A6 8B4450 3990 MOV AX,[SI+OFFSET CURSOR_POSN] : GET ROW/COLUMN OF THAT PAGE
F3A9 33DB 3991 XOR BX,BX : SET START ADDRESS TO ZERO
F3AB E306 3992 JCXZ P5 : NO PAGE
F3AD 031E4C00 3993 P4: ADD BX,CRT_LEN : PAGE LOOP
F3B1 E2FA 3994 LOOP P4 : LENGTH OF BUFFER
F3B3 E8CFFE 3995 P5: : NO PAGE
F3B6 03D8 3996 CALL POSITION : DETERMINE LOCATION IN REGEN
F3B8 C3 3997 ADD BX,AX : ADD TO START OF REGEN
3998 :
3999 RET
4000 FIND_POSITION ENDP
4001 :-----
4002 : WRITE_AC_CURRENT :
4003 : THIS ROUTINE WRITES THE ATTRIBUTE :
4004 : AND CHARACTER AT THE CURRENT CURSOR :
4005 : POSITION :
4006 : INPUT :
4007 : (AH) = CURRENT CRT MODE :
4008 : (BH) = DISPLAY PAGE :
4009 : (CX) = COUNT OF CHARACTERS TO WRITE :
4010 : (AL) = CHAR TO WRITE :
4011 : (BL) = ATTRIBUTE OF CHAR TO WRITE :
4012 : (DS) = DATA SEGMENT :
4013 : (ES) = REGEN SEGMENT :
4014 : OUTPUT :
4015 : NONE :
4016 :-----
4017 WRITE_AC_CURRENT PROC NEAR
F3B9 80FC04 4018 CMP AH,4 : IS THIS GRAPHICS
F3BC 7208 4019 JC P6 :
F3BE 80FC07 4020 CMP AH,7 : IS THIS BW CARD
F3C1 7403 4021 JE P6 :
F3C3 E9B201 4022 JMP GRAPHICS_WRITE :
F3C6 8AE3 4023 P6: MOV AH,BL : WRITE AC_CONTINUE
F3C8 50 4024 PUSH AX : GET ATTRIBUTE TO AH
F3C9 51 4025 PUSH CX : SAVE ON STACK
F3CA E8D1FF 4026 CALL FIND_POSITION : SAVE WRITE COUNT
F3CD 8BF8 4027 MOV DI,BX : ADDRESS TO DI REGISTER
F3CF 59 4028 POP CX : WRITE COUNT
F3D0 5B 4029 POP BX : CHARACTER IN BX REG
F3D1 :
4030 PT: 4031 : WRITE_LOOP
4032 :
4033 :----- WAIT FOR HORIZONTAL RETRACE
4034 :
F3D1 8B166300 4035 MOV DX,ADDR_6845 : GET BASE ADDRESS
F3D5 83C206 4036 ADD DX,6 : POINT AT STATUS PORT
F3D8 :
F3D8 EC 4037 P8: IN AL,DX : GET STATUS
F3D9 A801 4038 TEST AL,1 : IS IT LOW
F3DB 75FB 4039 JNZ P8 : WAIT UNTIL IT IS
F3DD FA 4040 CLJ P8 : NO MORE INTERRUPTS
F3DE :
F3DE EC 4041 P9: IN AL,DX : GET STATUS
F3DF A801 4042 TEST AL,1 : IS IT HIGH
F3E1 74FB 4043 JZ P9 : WAIT UNTIL IT IS
F3E3 8BC3 4044 MOV AX,BX : RECOVER THE CHAR/ATTR
F3E5 AB 4045 STOSW : PUT THE CHAR/ATTR
F3E6 FB 4046 STI : INTERRUPTS BACK ON
F3E7 E2EB 4047 LOOP P7 : AS MANY TIMES AS REQUESTED
F3E9 E9D9FD 4048 JMP VIDEO_RETURN
4049 :
4050 WRITE_AC_CURRENT ENDP
4051 :

```

```

4052 :-----
4053 : WRITE_C_CURRENT
4054 : THIS ROUTINE WRITES THE CHARACTER AT
4055 : THE CURRENT CURSOR POSITION, ATTRIBUTE
4056 : UNCHANGED
4057 : INPUT
4058 : (AH) = CURRENT CRT MODE
4059 : (BH) = DISPLAY PAGE
4060 : (CX) = COUNT OF CHARACTERS TO WRITE
4061 : (AL) = CHAR TO WRITE
4062 : (DS) = DATA SEGMENT
4063 : (ES) = REGEN SEGMENT
4064 : OUTPUT
4065 : NONE
4066 :-----
F3EC 4067 WRITE_C_CURRENT PROC NEAR
F3EC 80FC04 4068 CMP AH,4 ; IS THIS GRAPHICS
F3EF 7208 4069 JC P10
F3F1 80FC07 4070 CMP AH,7 ; IS THIS BW CARD
F3F4 7403 4071 JE P10
F3F6 E97F01 4072 JMP GRAPHICS_WRITE
F3F9 4073
F3F9 50 4074 PUSH AX ; SAVE ON STACK
F3FA 51 4075 PUSH CX ; SAVE WRITE COUNT
F3FB E8A0FF 4076 CALL FIND_POSITION
F3FE 8BF8 4077 MOV DI,BX ; ADDRESS TO DI
F400 59 4078 POP CX ; WRITE COUNT
F401 5B 4079 POP BX ; BL HAS CHAR TO WRITE
F402 4080 P11: WRITE_LOOP
4081
4082 :----- WAIT FOR HORIZONTAL RETRACE
4083
4084
F402 8B166300 4085 MOV DX,ADDR_6845 ; GET BASE ADDRESS
F406 83C206 4086 ADD DX,6 ; POINT AT STATUS PORT
F409 4087
F409 EC 4088 P12: IN AL,DX ; GET STATUS
F40A A801 4089 TEST AL,1 ; IS IT LOW
F40C 75FB 408A JNZ P12 ; WAIT UNTIL IT IS
F40E FA 408B CLI ; NO MORE INTERRUPTS
F40F 408C
F40F EC 408D P13: IN AL,DX ; GET STATUS
F410 A801 408E TEST AL,1 ; IS IT HIGH
F412 74FB 408F JZ P13 ; WAIT UNTIL IT IS
F414 8AC3 4090 MOV AL,BL ; RECOVER CHAR
F416 AA 4091 STOSB ; PUT THE CHAR/ATTR
F417 FB 4092 STI ; INTERRUPTS BACK ON
F418 47 4093 INC DI ; BUMP POINTER PAST ATTRIBUTE
F419 E2E7 4094 LOOP P11 ; AS MANY TIMES AS REQUESTED
F41B E9A7FD 4095 JMP VIDEO_RETURN
4096
4097 WRITE_C_CURRENT ENDP
4098
4102 :-----
4103 : READ DOT -- WRITE DOT
4104 : THESE ROUTINES WILL WRITE A DOT, OR READ THE DOT AT
4105 : THE INDICATED LOCATION
4106 : ENTRY --
4107 : DX = ROW (0-199) (THE ACTUAL VALUE DEPENDS ON THE MODE)
4108 : CX = COLUMN (0-639) (THE VALUES ARE NOT RANGE CHECKED)
4109 : AL = DOT VALUE TO WRITE (1,2 OR 4 BITS DEPENDING ON MODE,
4110 : REQ'D FOR WRITE DOT ONLY, RIGHT JUSTIFIED)
4111 : BIT 7 OF AL=1 INDICATES XOR THE VALUE INTO THE LOCATION
4112 : DS = DATA SEGMENT
4113 : ES = REGEN SEGMENT
4114 :
4115 : EXIT
4116 : AL = DOT VALUE READ, RIGHT JUSTIFIED, READ ONLY
4117 :-----
4118 ASSUME CS:CODE,DS:DATA,ES:DATA
4119 READ_DOT PROC NEAR
4120 CALL R3
4121 MOV AL,ES:[SI] ; DETERMINE BYTE POSITION OF DOT
4122 AND AL,AH ; GET THE BYTE
4123 SHL AL,CL ; MASK OFF THE OTHER BITS IN THE BYTE
4124 MOV CL,DH ; LEFT JUSTIFY THE VALUE
4125 ROL AL,CL ; GET NUMBER OF BITS IN RESULT
4126 JMP VIDEO_RETURN ; RIGHT JUSTIFY THE RESULT
4127 READ_DOT ENDP
4128
4129 WRITE_DOT PROC NEAR
4130 PUSH AX ; SAVE DOT VALUE
4131 PUSH AX ; TWICE
4132 CALL R3 ; DETERMINE BYTE POSITION OF THE DOT
4133 SHR AL,CL ; SHIFT TO SET UP THE BITS FOR OUTPUT
4134 AND AL,AH ; STRIP OFF THE OTHER BITS
4135 MOV CL,ES:[SI] ; GET THE CURRENT BYTE
4136 POP BX ; RECOVER XOR FLAG
4137 TEST BL,80H ; IS IT ON
4138 JNZ R2 ; YES, XOR THE DOT
4139 NOT AH ; SET THE MASK TO REMOVE THE
4140 AND CL,AH ; INDICATED BITS
4141 OR AL,AH ; OR IN THE NEW VALUE OF THOSE BITS
4142 R1: FINISH_DOT
4143 MOV ES:[SI],AL ; RESTORE THE BYTE IN MEMORY
4144 POP AX
4145 JMP VIDEO_RETURN ; RETURN FROM VIDEO IO
4146 R2: XOR_DOT
4147 XOR AL,CL ; EXCLUSIVE OR THE DOTS
4148 JMP R1 ; FINISH UP THE WRITING
4149 WRITE_DOT ENDP

```

LOC OBJECT	LINE	SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82
	4150	-----
	4151	: THIS SUBROUTINE DETERMINES THE REGEN BYTE LOCATION :
	4152	: OF THE INDICATED ROW COLUMN VALUE IN GRAPHICS MODE. :
	4153	: ENTRY -- :
	4154	: DX = ROW VALUE (0-199) :
	4155	: CX = COLUMN VALUE (0-639) :
	4156	: EXIT -- :
	4157	: SI = OFFSET INTO REGEN BUFFER FOR BYTE OF INTEREST :
	4158	: AH = MASK TO STRIP OFF THE BITS OF INTEREST :
	4159	: CL = BITS TO SHIFT TO RIGHT JUSTIFY THE MASK IN AH :
	4160	: DH = # BITS IN RESULT :
	4161	-----
F452	4162	R3 PROC NEAR
F452 53	4163	PUSH BX ; SAVE BX DURING OPERATION
F453 50	4164	PUSH AX ; WILL SAVE AL DURING OPERATION
	4165	
	4166	----- DETERMINE 1ST BYTE IN IDICATED ROW BY MULTIPLYING ROW VALUE BY 40
	4167	----- (LOW BIT OF ROW DETERMINES EVEN/ODD, 80 BYTES/ROW
	4168	
F454 B028	4169	MOV AL,40
F456 52	4170	PUSH DX ; SAVE ROW VALUE
F457 80E2FE	4171	AND DL,0FEH ; STRIP OFF ODD/EVEN BIT
F45A F6E2	4172	MUL DL ; AX HAS ADDRESS OF 1ST BYTE
	4173	; OF INDICATED ROW
F45C 5A	4174	POP DX ; RECOVER IT
F45D F6C201	4175	TEST DL,1 ; TEST FOR EVEN/ODD
F460 7403	4176	JZ R4 ; JUMP IF EVEN ROW
F462 050020	4177	ADD AX,2000H ; OFFSET TO LOCATION OF ODD ROWS
F465	4178	R4: ; EVEN ROW
F465 8BF0	4179	MOV SI,AX ; MOVE POINTER TO SI
F467 58	4180	POP AX ; RECOVER AL VALUE
F468 8BD1	4181	MOV DX,CX ; COLUMN VALUE TO DX
	4182	
	4183	----- DETERMINE GRAPHICS MODE CURRENTLY IN EFFECT
	4184	
	4185	-----
	4186	: SET UP THE REGISTERS ACCORDING TO THE MODE :
	4187	: CH = MASK FOR LOW OF COLUMN ADDRESS (1/3 FOR HIGH/MED RES) :
	4188	: CL = # OF ADDRESS BITS IN COLUMN VALUE (3/2 FOR H/M) :
	4189	: BL = MASK TO SELECT BITS FROM POINTED BYTE (80H/COH FOR H/M) :
	4190	: BH = NUMBER OF VALID BITS IN POINTED BYTE (1/2 FOR H/M) :
	4191	-----
F46A BBC002	4192	MOV BX,2C0H
F46D B90203	4193	MOV CX,302H ; SET PARMS FOR MED RES
F470 803E490006	4195	CMP CRT_MODE,6
F475 7206	4196	JC R5 ; HANDLE IF MED ARES
F477 BB8001	4197	MOV BX,180H
F47A B90307	4198	MOV CX,703H ; SET PARMS FOR HIGH RES
	4199	
	4200	----- DETERMINE BIT OFFSET IN BYTE FROM COLUMN MASK
	4201	
F47D	4202	R5: ;
F47D 22EA	4203	AND CH,DL ; ADDRESS OF PEL WITHIN BYTE TO CH
	4204	
	4205	----- DETERMINE BYTE OFFSET FOR THIS LOCATION IN COLUMN
	4206	
F47F D3EA	4207	SHR DX,CL ; SHIFT BY CORRECT AMOUNT
F481 03F2	4208	ADD SI,DX ; INCREMENT THE POINTER
F483 8AF7	4209	MOV DH,BH ; GET THE # OF BITS IN RESULT TO DH
	4210	
	4211	----- MULTIPLY BH (VALID BITS IN BYTE) BY CH (BIT OFFSET)
	4212	
F485 2AC9	4213	R6: SUB CL,CL ; ZERO INTO STORAGE LOCATION
F487	4214	
F487 D0C8	4215	ROR AL,1 ; LEFT JUSTIFY THE VALUE
	4216	; IN AL (FOR WRITE)
F489 02CD	4217	ADD CL,CH ; ADD IN THE BIT OFFSET VALUE
F48B FECF	4218	DEC BH ; LOOP CONTROL
F48D 75F8	4219	JNZ R6 ; ON EXIT, CL HAS SHIFT COUNT
	4220	; TO RESTORE BITS
F48F 8AE3	4221	MOV AH,BL ; GET MASK TO AH
F491 D2EC	4222	SHR AH,CL ; MOVE THE MASK TO CORRECT LOCATION
F493 5B	4223	POP BX ; RECOVER REG
F494 C3	4224	RET ; RETURN WITH EVERYTHING SET UP
	4225	R3 ENDP
	4226	-----
	4227	: SCROLL UP :
	4228	: THIS ROUTINE SCROLLS UP THE INFORMATION ON THE CRT :
	4229	: ENTRY :
	4230	: CH,CL = UPPER LEFT CORNER OF REGION TO SCROLL :
	4231	: DH,DL = LOWER RIGHT CORNER OF REGION TO SCROLL :
	4232	: BOTH OF THE ABOVE ARE IN CHARACTER POSITIONS :
	4233	: BH = FILL VALUE FOR BLANKED LINES :
	4234	: AL = # LINES TO SCROLL (AL=0 MEANS BLANK THE ENTIRE :
	4235	: FIELD) :
	4236	: DS = DATA SEGMENT :
	4237	: ES = REGEN SEGMENT :
	4238	: EXIT :
	4239	: NOTHING, THE SCREEN IS SCROLLED :
	4240	-----
F495	4241	GRAPHICS UP PROC NEAR
F495 8AD8	4242	MOV BL,AL ; SAVE LINE COUNT IN BL
F497 8BC1	4243	MOV AX,CX ; GET UPPER LEFT POSITION INTO AX REG
	4244	
	4245	----- USE CHARACTER SUBROUTINE FOR POSITIONING
	4246	----- ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE
	4247	
F499 E86902	4248	CALL GRAPH_POSN
F49C 8BF8	4249	MOV DI,AX ; SAVE RESULT AS DESTINATION ADDRESS
	4250	
	4251	----- DETERMINE SIZE OF WINDOW
	4252	
F49E 2BD1	4253	SUB DX,CX
F4A0 81C20101	4254	ADD DX,101H ; ADJUST VALUES
F4A4 D0E6	4255	SAL DH,1 ; MULTIPLY # ROWS BY 4
	4256	; SINCE 8 VERT DOTS/CHAR
F4A6 D0E6	4257	SAL DH,1 ; AND EVEN/ODD ROWS
	4258	
	4259	----- DETERMINE CRT MODE
	4260	
F4A8 803E490006	4261	CMP CRT_MODE,6 ; TEST FOR MEDIUM RES
F4AD 7304	4262	JNC R7 ; FIND_SOURCE

```

4263
4264 ;----- MEDIUM RES UP
4265
F4AF D0E2      4266     SAL     DL,I           ; # COLUMNS * 2, SINCE 2 BYTES/CHAR
F4B1 D1E7      4267     SAL     DI,I           ; OFFSET *2 SINCE 2 BYTES/CHAR
4268
4269 ;----- DETERMINE THE SOURCE ADDRESS IN THE BUFFER
4270
F4B3           4271 RT:           ; FIND_SOURCE
F4B3 06        4272     PUSH    ES             ; GET SEGMENTS BOTH POINTING TO REGEN
F4B4 1F        4273     POP     DS             ;
F4B5 2AED      4274     SUB     CH,CH         ; ZERO TO HIGH OF COUNT REG
F4B7 D0E3      4275     SAL     BL,I         ; MULTIPLY NUMBER OF LINES BY 4
F4B9 D0E3      4276     JZ      BL,I         ;
F4BB 742D      4277     R11     ; IF ZERO, THEN BLANK ENTIRE FIELD
F4BB 8AC3      4278     MOV     AL,BL       ; GET NUMBER OF LINES IN AL
F4BF B450      4279     MOV     AH,80       ; 80 BYTES/ROW
F4C1 F6E4      4280     MUL     AH           ; DETERMINE OFFSET TO SOURCE
F4C3 8BF7      4281     MOV     SI,DI       ; SET UP SOURCE
F4C5 03F0      4282     ADD     SI,AX         ; ADD IN OFFSET TO IT
F4C7 8AE6      4283     MOV     AH,DH       ; NUMBER OF ROWS IN FIELD
F4C9 2AE3      4284     SUB     AH,BL       ; DETERMINE NUMBER TO MOVE
4285
4286 ;----- LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD FIELDS
4287
F4CB           4288 R8:           ; ROW LOOP
F4CB E88000    4289     CALL    R17           ; MOVE ONE ROW
F4CE 81EEB01F  4290     SUB     SI,2000H-80 ; MOVE TO NEXT ROW
F4D0 81EFB01F  4291     SUB     DI,2000H-80 ;
F4D6 FECC      4292     DEC     AH           ; NUMBER OF ROWS TO MOVE
F4D8 75F1      4293     JNZ     R8          ; CONTINUE TILL ALL MOVED
4294
4295 ;----- FILL IN THE VACATED LINE(S)
4296
F4DA           4297 R9:           ; CLEAR ENTRY
F4DA 8AC7      4298     MOV     AL,BH         ; ATTRIBUTE TO FILL WITH
F4DC          4299 R10:          ;
F4DC E88800    4300     CALL    R18           ; CLEAR THAT ROW
F4DF 81EFB01F  4301     SUB     DI,2000H-80 ; POINT TO NEXT LINE
F4E3 FECB      4302     DEC     BL           ; NUMBER OF LINES TO FILL
F4E5 75F5      4303     JNZ     R10          ; CLEAR_LOOP
F4E7 E9DBFC    4304     JMP     VIDEO_RETURN ; EVERYTHING DONE
F4EA          4305 R11:          ; BLANK FIELD
F4EA 8ADE      4306     MOV     BL,DH         ; SET BLANK COUNT TO
4307                                     ; EVERYTHING IN FIELD
F4EC EBEC      4308     JMP     R9           ; CLEAR THE FIELD
4309
4310 GRAPHICS_UP ENDP
4311 ; SCROLL DOWN
4312 ; THIS ROUTINE SCROLLS DOWN THE INFORMATION ON THE CRT
4313 ; ENTRY
4314 ; CH,CL = UPPER LEFT CORNER OF REGION TO SCROLL
4315 ; DH,DL = LOWER RIGHT CORNER OF REGION TO SCROLL
4316 ; BOTH OF THE ABOVE ARE IN CHARACTER POSITIONS
4317 ; BH = FILL VALUE FOR BLANKED LINES
4318 ; AL = # LINES TO SCROLL (AL=0 MEANS BLANK THE ENTIRE
4319 ; FIELD)
4320 ; DS = DATA SEGMENT
4321 ; ES = REGEN SEGMENT
4322 ; EXIT
4323 ; NOTHING, THE SCREEN IS SCROLLED
4324
4325 GRAPHICS_DOWN PROC NEAR
4326     STD
4327     MOV     BL,AL
4328     MOV     AX,DX
4329
4330 ;----- USE CHARACTER SUBROUTINE FOR POSITIONING
4331 ;----- ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE
4332
F4F3 E80F02    4333     CALL    GRAPH_POSN
F4F6 8BF8      4334     MOV     DI,AX         ; SAVE RESULT AS DESTINATION ADDRESS
4335
4336 ;----- DETERMINE SIZE OF WINDOW
4337
F4F8 2BD1      4338     SUB     DX,CX
F4FA 81C20101  4339     ADD     DX,101H      ; ADJUST VALUES
F4FE D0E6      4340     SAL     DH,I         ; MULTIPLY # ROWS BY 4
4341                                     ; SINCE 8 VERT DOTS/CHAR
4342     SAL     DH,I         ; AND EVEN/ODD ROWS
4343
4344 ;----- DETERMINE CRT MODE
4345
F502 803E490006 4346     CMP     CRT_MODE,6   ; TEST FOR MEDIUM RES
F507 7305      4347     JNC     R12          ; FIND_SOURCE_DOWN
4348
4349 ;----- MEDIUM RES DOWN
4350
F509 D0E2      4351     SAL     DL,I         ; # COLUMNS * 2, SINCE
4352                                     ; 2 BYTES/CHAR (OFFSET OK)
F50B D1E7      4353     SAL     DI,I         ; OFFSET *2 SINCE 2 BYTES/CHAR
F50D 47        4354     INC     DI           ; POINT TO LAST BYTE
4355
4356 ;----- DETERMINE THE SOURCE ADDRESS IN THE BUFFER
4357
F50E           4358 R12:          ; FIND_SOURCE_DOWN
F50E 06        4359     PUSH    ES             ; BOTH SEGMENTS TO REGEN
F50F 1F        4360     POP     DS             ;
F510 2AED      4361     SUB     CH,CH         ; ZERO TO HIGH OF COUNT REG
F512 81C7F000  4362     ADD     DI,240       ; POINT TO LAST ROW OF PIXELS
F516 D0E3      4363     SAL     BL,I         ; MULTIPLY NUMBER OF LINES BY 4
F518 D0E3      4364     JZ      BL,I         ;
F51A 742E      4365     R16     ; IF ZERO, THEN BLANK ENTIRE FIELD
F51C 8AC3      4366     MOV     AL,BL       ; GET NUMBER OF LINES IN AL
F51E B450      4367     MOV     AH,80       ; 80 BYTES/ROW
F520 F6E4      4368     MUL     AH           ; DETERMINE OFFSET TO SOURCE
F522 8BF7      4369     MOV     SI,DI       ; SET UP SOURCE
F524 2BF0      4370     SUB     SI,AX         ; SUBTRACT THE OFFSET
F526 8AE6      4371     MOV     AH,DH       ; NUMBER OF ROWS IN FIELD
F528 2AE3      4372     SUB     AH,BL       ; DETERMINE NUMBER TO MOVE

```

```

4373
4374 ;----- LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD FIELDS
4375
F52A      4376 R13:      CALL      R17      ; ROW LOOP DOWN
F52A E82100 4377      MOV     ONE ROW
F52D 81EE5020 4378      SUB     SI,2000H+80      ; MOVE TO NEXT ROW
F531 81EF5020 4379      SUB     DI,2000H+80
F535 FECC 4380      DEC     AH              ; NUMBER OF ROWS TO MOVE
F537 75F1 4381      JNZ     R13          ; CONTINUE TILL ALL MOVED
4382
4383 ;----- FILL IN THE VACATED LINE(S)
4384
F539      4385 R14:      MOV     AL,BH      ; CLEAR ENTRY_DOWN
F539 8AC7 4386      MOV     AL,BH      ; ATTRIBUTE TO FILL WITH
F53B E82900 4387 R15:      CALL      R18      ; CLEAR LOOP_DOWN
F53B 81EF5020 4388      CALL      R18      ; CLEAR A ROW
F542 FECC 4389      SUB     DI,2000H+80      ; POINT TO NEXT LINE
F544 75F5 4390      DEC     BL              ; NUMBER OF LINES TO FILL
F546 FC 4391      JNZ     R15          ; CLEAR LOOP_DOWN
F547 E97BFC 4392      CLD      ; RESET THE DIRECTION FLAG
F54A 8ADE 4393      JMP     VIDEO_RETURN ; EVERYTHING DONE
F54A 8ADE 4394 R16:      MOV     BL,DH      ; BLANK FIELD_DOWN
F54C EBEB 4395      MOV     BL,DH      ; SET BLANK COUNT TO EVERYTHING
4396      ; IN FIELD
4397      JMP     R14              ; CLEAR THE FIELD
4398      GRAPHICS_DOWN  ENDP
4400 ;----- ROUTINE TO MOVE ONE ROW OF INFORMATION
4401
F54E      4402 R17      PROC     NEAR
F54E 8ACA 4403      MOV     CL,DL      ; NUMBER OF BYTES IN THE ROW
F550 56 4404      PUSH    SI
F551 57 4405      PUSH    DI              ; SAVE POINTERS
F552 F3 4406      REP     MOVSB      ; MOVE THE EVEN FIELD
F553 A4 4407      POP     DI
F554 5F 4408      POP     SI
F555 5E 4409      ADD     SI,2000H
F556 81C60020 4410      ADD     DI,2000H      ; POINT TO THE ODD FIELD
F55E 56 4411      PUSH    SI
F55F 57 4412      PUSH    DI              ; SAVE THE POINTERS
F560 8ACA 4413      MOV     CL,DL      ; COUNT BACK
F562 F3 4414      REP     MOVSB      ; MOVE THE ODD FIELD
F563 A4 4415      POP     DI
F564 5F 4416      POP     SI              ; POINTERS BACK
F565 5E 4417      RET     ; RETURN TO CALLER
F566 C3 4418 R17      ENDP
4419
4420 ;----- CLEAR A SINGLE ROW
4421
F567      4422 R18      PROC     NEAR
F567 8ACA 4423      MOV     CL,DL      ; NUMBER OF BYTES IN FIELD
F569 57 4424      PUSH    DI              ; SAVE POINTER
F56A F3 4425      REP     STOSB      ; STORE THE NEW VALUE
F56B AA 4426      POP     DI
F56C 5F 4427      POP     DI,2000H      ; POINTER BACK
F56D 81C70020 4428      PUSH    DI              ; POINT TO ODD FIELD
F571 57 4429      MOV     CL,DL
F572 8ACA 4430      REP     STOSB      ; FILL THE ODD FIELD
F574 F3 4431      POP     DI
F575 AA 4432      RET     ; RETURN TO CALLER
F576 5F 4433 R18      ENDP
F577 C3 4434 ;-----
4435 ; GRAPHICS WRITE
4436 ; THIS ROUTINE WRITES THE ASCII CHARACTER TO THE
4437 ; CURRENT POSITION ON THE SCREEN.
4438 ; ENTRY
4439 ; AL = CHARACTER TO WRITE
4440 ; BL = COLOR ATTRIBUTE TO BE USED FOR FOREGROUND COLOR
4441 ; IF BIT 7 IS SET, THE CHAR IS XOR'D INTO THE REGEN
4442 ; BUFFER (0 IS USED FOR THE BACKGROUND COLOR)
4443 ; CX = NUMBER OF CHARS TO WRITE
4444 ; DS = DATA SEGMENT
4445 ; ES = REGEN SEGMENT
4446 ; EXIT
4447 ; NOTHING IS RETURNED
4448 ;
4449 ; GRAPHICS READ
4450 ; THIS ROUTINE READS THE ASCII CHARACTER AT THE CURRENT
4451 ; CURSOR POSITION ON THE SCREEN BY MATCHING THE DOTS ON
4452 ; THE SCREEN TO THE CHARACTER GENERATOR CODE POINTS
4453 ; ENTRY
4454 ; NONE (0 IS ASSUMED AS THE BACKGROUND COLOR)
4455 ; EXIT
4456 ; AL = CHARACTER READ AT THAT POSITION (0 RETURNED IF
4457 ; NONE FOUND)
4458 ;
4459 ; FOR BOTH ROUTINES, THE IMAGES USED TO FORM CHARS ARE
4460 ; CONTAINED IN ROWS FOR THE 1ST 128 CHARS. TO ACCESS CHARS
4461 ; IN THE SECOND HALF, THE USER MUST INITIALIZE THE VECTOR AT
4462 ; INTERRUPT 1FH (LOCATION 0007CH) TO POINT TO THE USER
4463 ; SUPPLIED TABLE OF GRAPHIC IMAGES (8X8 BOXES).
4464 ; FAILURE TO DO SO WILL CAUSE IN STRANGE RESULTS.
4465 ;-----
4466      ASSUME  CS:CODE,DS:DATA,ES:DATA
4467      GRAPHICS_WRITE  PROC     NEAR
4468      MOV     AH,0
4469      PUSH    AX
4470      ; ZERO TO HIGH OF CODE POINT
4471      ; SAVE CODE POINT VALUE
4472 ;----- DETERMINE POSITION IN REGEN BUFFER TO PUT CODE POINTS
4473      CALL    S26
4474      MOV     DI,AX
4475      ; REGEN POINTER IN DI
4476 ;----- DETERMINE REGION TO GET CODE POINTS FROM
4477
F57B E88401 4478      POP     AX
F57E 8BF8 4479      CMP     AL,80H
F580 58 4480      JAE     SI
F581 3C80
F583 7306 4481      JAE     SI
4482      ; YES

```


LOC OBJECT

LINE SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82

```

4481
4482
4483 ;----- IMAGE 15 IN FIRST HALF, CONTAINED IN ROM
F585 BE6EFA 4484      MOV     SI,0FA6EH      ; CRT_CHAR_GEN (OFFSET OF IMAGES)
F588 0E      4485      PUSH    CS           ; SAVE SEGMENT ON STACK
F589 EB0F    4486      JMP     SHORT S2      ; DETERMINE_MODE
4487
4488 ;----- IMAGE 15 IN SECOND HALF, IN USER RAM
4489
F58B        4490 S1:      SUB     AL,80H      ; EXTEND_CHAR
F58B 2C80    4491      PUSH    DS           ; ZERO ORIGIN FOR SECOND HALF
F58D 1E      4492      SUB     SI,SI         ; SAVE DATA POINTER
F58E 2BF6    4493      MOV     DS,SI         ; ESTABLISH VECTOR ADDRESSING
F590 8EDE    4494      ASSUME  DS:ABS0
4495      4495      LDS     SI,EXT_PTR  ; GET THE OFFSET OF THE TABLE
F592 C5367C00 4496      MOV     DX,DS         ; GET THE SEGMENT OF THE TABLE
F596 8CDA    4497      ASSUME  DS:DATA
F598 1F      4498      POP     DS           ; RECOVER DATA SEGMENT
F599 52      4499      PUSH    DX           ; SAVE TABLE SEGMENT ON STACK
4500
4501
4502 ;----- DETERMINE GRAPHICS MODE IN OPERATION
4503
F59A        4504 S2:      SAL     AX,1       ; DETERMINE_MODE
F59A D1E0    4505      SAL     AX,1       ; MULTIPLY CODE POINT
F59C D1E0    4506      SAL     AX,1       ; VALUE BY 8
F59E D1E0    4507      ADD     SI,AX        ; SI HAS OFFSET OF DESIRED CODES
F5A0 03F0    4508      CMP     CRT_MODE,6
F5A2 803E490006 4509      POP     DS           ; RECOVER TABLE POINTER SEGMENT
F5A7 1F      4510      JC      S7         ; TEST FOR MEDIUM RESOLUTION MODE
F5A8 722C    4511
4512
4513 ;----- HIGH RESOLUTION MODE
4514
F5AA        4515 S3:      PUSH    DI         ; HIGH CHAR
F5AA 57      4516      PUSH    SI         ; SAVE REGEN POINTER
F5AB 56      4517      MOV     DH,4       ; SAVE CODE POINTER
F5AC B604    4518      S4:      LODSB      ; NUMBER OF TIMES THROUGH LOOP
F5AE AC      4519      TEST     BL,80H     ; GET BYTE FROM CODE POINTS
F5AF F6C380 4520      JNZ     S6         ; SHOULD WE USE THE FUNCTION
F5B2 7516    4521      JNZ     S6         ; TO PUT CHAR IN
4522
F5B4 AA      4523      STOSB      ; STORE IN REGEN BUFFER
F5B5 AC      4524
F5B6        4525 S5:      MOV     ES:[DI+2000H-1],AL ; STORE IN SECOND HALF
F5B6 268885FF1F 4526      ADD     DI,79       ; MOVE TO NEXT ROW IN REGEN
F5BB 83C74F 4527      DEC     DH         ; DONE WITH LOOP
F5BE FECE    4528      JNZ     S4         ; RECOVER REGEN POINTER
F5C0 75EC    4529      POP     DI         ; POINT TO NEXT CHAR POSITION
F5C2 5E      4530      POP     DI         ; MORE CHARS TO WRITE
F5C3 5F      4531      INC     DI         ; POINT TO NEXT CHAR POSITION
F5C4 47      4532      LOOP    S3         ; MORE CHARS TO WRITE
F5C5 E2E3    4533      JMP     VIDEO_RETURN
F5C7 E9FBFB 4534
F5CA        4535 S6:      XOR     AL,ES:[DI]    ; EXCLUSIVE OR WITH CURRENT
F5CA 263205 4536      STOSB      ; STORE THE CODE POINT
F5CD AA      4537      XOR     AL,ES:[DI+2000H-1] ; AGAIN FOR ODD FIELD
F5CE AC      4538      XOR     AL,ES:[DI+2000H-1] ; BACK TO MAINSTREAM
F5CF 263285FF1F 4539      JMP     S5
F5D4 EBE0    4540
4541
4542 ;----- MEDIUM RESOLUTION WRITE
4543
F5D6        4544 S7:      MOV     DL,BL      ; MED.RES WRITE
F5D6 8AD3    4545      SAL     DI,1       ; SAVE HIGH COLOR BIT
F5D8 D1E7    4546      CALL    S19        ; OFFSET*2 SINCE 2 BYTES/CHAR
F5DA E8D100 4547      CALL    S19        ; EXPAND BL TO FULL WORD OF COLOR
F5DD        4548 S8:      PUSH    DI         ; MED.CHAR
F5DD 57      4549      PUSH    SI         ; SAVE REGEN POINTER
F5DE 56      4550      MOV     DH,4       ; SAVE THE CODE POINTER
F5DF B604    4551      MOV     DH,4       ; NUMBER OF LOOPS
4552
F5E1 AC      4552 S9:      LODSB      ; GET CODE POINT
F5E2 E8DE00 4553      CALL    S21        ; DOUBLE UP ALL THE BITS
F5E5 23C3    4554      AND     AX,BX       ; CONVERT THEM TO FOREGROUND
4555      4555      COLOR ( 0 BACK )
F5E7 F6C280 4556      TEST     DL,80H     ; IS THIS XOR FUNCTION
F5EA 7407    4557      JZ      S10        ; NO, STORE IT IN AS IT IS
F5EC 263225 4558      XOR     AH,ES:[DI]   ; DO FUNCTION WITH HALF
F5EF 26324501 4559      XOR     AL,ES:[DI+1] ; AND WITH OTHER HALF
F5F3        4560
4561
F5F3 268825 4562      MOV     ES:[DI],AH   ; STORE FIRST BYTE
F5F6 26884501 4563      MOV     ES:[DI+1],AL ; STORE SECOND BYTE
F5FA AC      4564      LODSB      ; GET CODE POINT
F5FB E8C500 4565      CALL    S21
F5FE 23C3    4566      AND     AX,BX       ; CONVERT TO COLOR
F600 F6C280 4567      TEST     DL,80H     ; AGAIN, IS THIS XOR FUNCTION
F603 740A    4568      JZ      S11        ; NO, JUST STORE THE VALUES
F605 2632A50020 4569      XOR     AH,ES:[DI+2000H] ; FUNCTION WITH FIRST HALF
F60A 2632850120 4570      XOR     AL,ES:[DI+2001H] ; AND WITH SECOND HALF
F60F        4571
F60F 2688A50020 4572      MOV     ES:[DI+2000H,AH]
F614 2688850120 4573      MOV     ES:[DI+2000H+1],AL ; STORE IN SECOND PORTION OF BUFFER
F619 83C750 4574      ADD     DI,80        ; POINT TO NEXT LOCATION
F61C FECE    4575      DEC     DH         ; KEEP GOING
F61E 75C1    4576      JNZ     S9         ; RECOVER CODE POINTER
F620 5E      4577      POP     SI         ; RECOVER REGEN POINTER
F621 5F      4578      POP     DI         ; POINT TO NEXT CHAR POSITION
F622 47      4579      INC     DI         ; MORE TO WRITE
F623 47      4580      INC     DI
F624 E2B7    4581      LOOP    S8
F626 E99CFB 4582      JMP     VIDEO_RETURN
4583
4583 GRAPHICS_WRITE ENDP

```

LOC	OBJECT	LINE	SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82	
		4584	-----	
		4585	: GRAPHICS_READ :	
		4586	GRAPHICS_READ PROC NEAR	
F629		4587	CALL S26	: CONVERTED TO OFFSET IN REGEN
F629 E8D600		4588	MOV SI,AX	: SAVE IN SI
F62C 8BF0		4589	SUB SP,8	: ALLOCATE SPACE TO SAVE THE
F62E 83EC08		4590		: READ CODE POINT
		4591		: POINTER TO SAVE AREA
F631 8BEC		4592	MOV BP,SP	
		4593		
		4594	----- DETERMINE GRAPHICS MODES	
		4595		
F633 803E490006		4596	CMP CRT_MODE,6	
F638 06		4597	PUSH ES	
F639 1F		4598	POP D5	: POINT TO REGEN SEGMENT
F63A 721A		4599	JC S13	: MEDIUM RESOLUTION
		4600		
		4601	----- HIGH RESOLUTION READ	
		4602		
		4603	----- GET VALUES FROM REGEN BUFFER AND CONVERT TO CODE POINT	
		4604		
F63C B604		4605	MOV DH,4	: NUMBER OF PASSES
F63E	S12:	4606	MOV AL,[SI]	: GET FIRST BYTE
F640 884600		4607	[BP],AL	: SAVE IN STORAGE AREA
F643 45		4608	INC BP	: NEXT LOCATION
F644 8A840020		4609	MOV AL,[SI+2000H]	: GET LOWER REGION BYTE
F648 884600		4610	[BP],AL	: ADJUST AND STORE
F64B 45		4611	INC BP	
F64C 83C650		4612	ADD SI,80	: POINTER INTO REGEN
F64F FECE		4613	DEC DH	: LOOP CONTROL
F651 75E9		4614	JNZ S12	: DO IT SOME MORE
F653 EB1790		4615	JMP S15	: GO MATCH THE SAVED CODE POINTS
		4616		
		4617	----- MEDIUM RESOLUTION READ	
		4618		
		4619		
F656		4620	S13: SAL SI,1	: MED RES READ
F656 DIE6		4621	MOV DH,4	: OFFSET*2 SINCE 2 BYTES/CHAR
F658 B604		4622		: NUMBER OF PASSES
F65A	S14:	4623	CALL S23	: GET PAIR BYTES FROM REGEN
F65A E88800		4624		: INTO SINGLE SAVE
		4625		
F65D 81C60020		4626	ADD SI,2000H	: GO TO LOWER REGION
F661 E88100		4627	CALL S23	: GET THIS PAIR INTO SAVE
F664 81EEB01F		4628	SI,2000H-80	: ADJUST POINTER BACK INTO UPPER
F668 FECE		4629	DEC DH	
F66A 75EE		4630	JNZ S14	: KEEP GOING UNTIL ALL 8 DONE
		4631		
		4632	----- SAVE AREA HAS CHARACTER IN IT, MATCH IT	
		4633		
F66C	S15:	4634	MOV DI,OFFSET CRT_CHAR_GEN	: FIND CHAR
F66C BF6EFA90		4635	PUSH CS	: ESTABLISH ADDRESSING
F670 0E		4636	POP ES	: CODE POINTS IN CS
F671 07		4637	SUB BP,8	: ADJUST POINTER TO BEGINNING
F672 83ED08		4638		: OF SAVE AREA
		4639		
F675 8BF5		4640	MOV SI,BP	: ENSURE DIRECTION
F677 FC		4641	CLD	: CURRENT CODE POINT BEING MATCHED
F678 B000		4642	MOV AL,0	
F67A	S16:	4643	PUSH SS	: ESTABLISH ADDRESSING TO STACK
F67B 1F		4644	POP D5	: FOR THE STRING COMPARE
F67C BA8000		4645	MOV DX,128	: NUMBER TO TEST AGAINST
F67F	S17:	4646		
F67F 56		4647	PUSH SI	: SAVE SAVE AREA POINTER
F680 57		4648	PUSH DI	: SAVE CODE POINTER
F681 B90800		4649	MOV CX,8	: NUMBER OF BYTES TO MATCH
F684 F3		4650	REPE CMPSB	: COMPARE THE 8 BYTES
F685 A6		4651		
F686 5F		4652	POP DI	: RECOVER THE POINTERS
F687 5E		4653	POP SI	
F688 741E		4654	JZ S18	: IF ZERO FLAG SET, THEN MATCH OCCURRED
F68A FEC0		4655	INC AL	: NO MATCH, MOVE ON TO NEXT
F68C 83C708		4656	ADD DI,8	: NEXT CODE POINT
F68F 4A		4657	DEC DX	: LOOP CONTROL
F690 75ED		4658	JNZ S17	: DO ALL OF THEM
		4659		
		4660	----- CHAR NOT MATCHED, MIGHT BE IN USER SUPPLIED SECOND HALF	
		4661		
F692 3C00		4662	CMP AL,0	: AL <> 0 IF ONLY 1ST HALF SCANNED
F694 7412		4663	JE S18	: IF = 0, THEN ALL HAS BEEN SCANNED
F696 2BC0		4664	SUB AX,AX	
F698 8ED8		4665	MOV DS,AX	: ESTABLISH ADDRESSING TO VECTOR
		4666	ASSUME DS:ABS0	
F69A C43E7C00		4667	LES DI,EXT_PTR	: GET POINTER
F69E 8CC0		4668	MOV AX,ES	: SEE IF THE POINTER REALLY EXISTS
F6A0 0BC7		4669	OR AX,D1	: IF ALL 0, THEN DOESN'T EXIST
F6A2 7404		4670	JZ S18	: NO SENSE LOOKING
F6A4 B080		4671	MOV AL,128	: ORIGIN FOR SECOND HALF
F6A6 EBD2		4672	JMP S16	: GO BACK AND TRY FOR IT
		4673	ASSUME DS:DATA	
		4674		
		4675	----- CHARACTER IS FOUND (AL=0 IF NOT FOUND)	
		4676		
F6A8	S18:	4677		
F6A8 83C408		4678	ADD SP,8	: READJUST THE STACK, THROW AWAY SAVE
F6AB E917FB		4679	VIDEO_RETURN	: ALL DONE
		4680	GRAPHICS_READ ENDP	

```

4681 -----
4682 : EXPAND_MED_COLOR :
4683 : THIS ROUTINE EXPANDS THE LOW 2 BITS IN BL TO :
4684 : FILL THE ENTIRE BX REGISTER :
4685 : ENTRY :
4686 : BL = COLOR TO BE USED ( LOW 2 BITS ) :
4687 : EXIT :
4688 : BX = COLOR TO BE USED ( 8 REPLICATIONS OF THE :
4689 : 2 COLOR BITS ) :
4690 : -----
F6AE S19 4691 PROC NEAR :
F6AE 80E303 4692 AND BL,3 : ISOLATE THE COLOR BITS
F6B1 8AC3 4693 MOV AL,BL : COPY TO AL
F6B3 51 4694 PUSH : SAVE REGISTER
F6B4 B90300 4695 MOV CX,3 : NUMBER OF TIMES TO DO THIS
F6B7 4696 S20: :
F6B7 D0E0 4697 SAL AL,1 :
F6B9 D0E0 4698 SAL AL,1 : LEFT SHIFT BY 2
F6BB 0AD8 4699 OR BL,AL : ANOTHER COLOR VERSION INTO BL
F6BD E2F8 4700 LOOP S20 : FILL ALL OF BL
F6BF 8AF8 4701 MOV BH,BL : FILL UPPER PORTION
F6C1 59 4702 POP CX : REGISTER BACK
F6C2 C3 4703 RET : ALL DONE
4704 S19 4704 ENDP
4705 : -----
4706 : EXPAND_BYTE :
4707 : THIS ROUTINE TAKES THE BYTE IN AL AND DOUBLES :
4708 : ALL OF THE BITS, TURNING THE 8 BITS INTO :
4709 : 16 BITS. THE RESULT IS LEFT IN AX :
4710 : -----
F6C3 S21 4711 PROC NEAR :
F6C3 52 4712 PUSH DX : SAVE REGISTERS
F6C4 51 4713 CX :
F6C5 53 4714 PUSH BX :
F6C6 2BD2 4715 SUB DX,DX : RESULT REGISTER
F6CB B90400 4716 MOV CX,1 : MASK REGISTER
F6CB 4717 S22: :
F6CB 8BD8 4718 MOV BX,AX : BASE TO TEMP
F6CD 23D9 4719 AND BX,CX : USE MASK TO EXTRACT A BIT
F6CF 0BD3 4720 OR DX,BX : PUT INTO RESULT REGISTER
F6D1 D1E0 4721 SHL AX,1 :
F6D3 D1E1 4722 SHL CX,1 : SHIFT BASE AND MASK BY 1
F6D5 8BD8 4723 MOV BX,AX : BASE TO TEMP
F6D7 23D9 4724 AND BX,CX : EXTRACT THE SAME BIT
F6D9 0BD3 4725 OR DX,BX : PUT INTO RESULT
F6DB D1E1 4726 SHL CX,1 : SHIFT ONLY MASK NOW,
4727 : MOVING TO NEXT BASE
F6DD 73EC 4728 JNC S22 : USE MASK BIT COMING OUT TO TERMINATE
F6DF 8BC2 4729 AX,DX : RESULT TO PARAM REGISTER
F6E1 5B 4730 POP BX :
F6E2 59 4731 POP CX : RECOVER REGISTERS
F6E3 5A 4732 POP DX :
F6E4 C3 4733 RET : ALL DONE
4734 S21 4734 ENDP
4735 : -----
4736 : MED_READ_BYTE :
4737 : THIS ROUTINE WILL TAKE 2 BYTES FROM THE REGEN :
4738 : BUFFER, COMPARE AGAINST THE CURRENT FOREGROUND :
4739 : COLOR, AND PLACE THE CORRESPONDING ON/OFF BIT :
4740 : PATTERN INTO THE CURRENT POSITION IN THE SAVE :
4741 : AREA :
4742 : ENTRY :
4743 : SI,DS = POINTER TO REGEN AREA OF INTEREST :
4744 : CX = EXPANDED FOREGROUND COLOR :
4745 : BP = POINTER TO SAVE AREA :
4746 : EXIT :
4747 : BP IS INCREMENT AFTER SAVE :
4748 : -----
F6E5 S23 4749 PROC NEAR :
F6E5 8A24 4750 MOV AH,[SI] : GET FIRST BYTE
F6E7 8A4401 4751 MOV AL,[SI+1] : GET SECOND BYTE
F6EA B90C00 4752 MOV CX,0C000H : 2 BIT MASK TO TEST THE ENTRIES
F6ED B200 4753 MOV DL,0 : RESULT REGISTER
F6EF 4754 S24: :
F6EF 85C1 4755 TEST AX,CX : IS THIS SECTION BACKGROUND?
F6F1 F8 4756 CLC : CLEAR CARRY IN HOPES THAT IT IS
F6F2 7401 4757 JZ S25 : IF ZERO, IT IS BACKGROUND
F6F4 F9 4758 STC : WASN'T, SO SET CARRY
F6F5 D0D2 4759 RCL DL,1 : MOVE THAT BIT INTO THE RESULT
F6F7 D1E9 4760 SHR CX,1 :
F6F9 D1E9 4761 SHR CX,1 :
F6FB 73F2 4762 JNC S24 : MOVE THE MASK TO THE RIGHT BY 2 BITS
F6FD 8B5600 4763 MOV [BP],DL : DO IT AGAIN IF MASK DIDN'T FALL OUT
F700 45 4764 INC BP : STORE RESULT IN SAVE AREA
F701 C3 4765 RET : ADJUST POINTER
4766 S23 4766 ENDP : ALL DONE
4767 : -----
4768 : V4_POSITION :
4769 : THIS ROUTINE TAKES THE CURSOR POSITION :
4770 : CONTAINED IN THE MEMORY LOCATION, AND :
4771 : CONVERTS IT INTO AN OFFSET INTO THE :
4772 : REGEN BUFFER, ASSUMING ONE BYTE/CHAR. :
4773 : FOR MEDIUM RESOLUTION GRAPHICS, :
4774 : THE NUMBER MUST BE DOUBLED. :
4775 : ENTRY :
4776 : NO REGISTERS, MEMORY LOCATION :
4777 : CURSOR_POSN IS USED :
4778 : EXIT :
4779 : AX CONTAINS OFFSET INTO REGEN BUFFER :
4780 : -----
F702 S26 4781 PROC NEAR :
F702 A15000 4782 MOV AX,CURSOR_POSN : GET CURRENT CURSOR
F705 4783 GRAPH_POSN LABEL NEAR :
F705 53 4784 PUSH BX : SAVE REGISTER
F706 8B08 4785 MOV BX,AX : SAVE A COPY OF CURRENT CURSOR
F708 8AC4 4786 MOV AL,AH : GET ROWS TO AL
F70A F624A00 4787 MUL BYTE PTR CRT_COLS : MULTIPLY BY BYTES/COLUMN
F70E D1E0 4788 SHL AX,1 : MULTIPLY * 4 SINCE 4 ROWS/BYTE
F710 D1E0 4789 SHL AX,1 :
F712 2AFF 4790 SUB BH,BH : ISOLATE COLUMN VALUE
F714 03C3 4791 ADD AX,BX : DETERMINE OFFSET
F716 5B 4792 POP BX : RECOVER POINTER
F717 C3 4793 RET : ALL DONE
4794 S26 4794 ENDP

```

```

4795 :-----
4796 : WRITE_TTY
4797 : THIS INTERFACE PROVIDES A TELETYPE LIKE INTERFACE TO THE VIDEO
4798 : CARD. THE INPUT CHARACTER IS WRITTEN TO THE CURRENT CURSOR
4799 : POSITION, AND THE CURSOR IS MOVED TO THE NEXT POSITION. IF THE
4800 : CURSOR LEAVES THE LAST COLUMN OF THE FIELD, THE COLUMN IS SET
4801 : TO ZERO, AND THE ROW VALUE IS INCREMENTED. IF THE ROW VALUE
4802 : LEAVES THE FIELD, THE CURSOR IS PLACED ON THE LAST ROW, FIRST
4803 : COLUMN, AND THE ENTIRE SCREEN IS SCROLLED UP ONE LINE. WHEN
4804 : THE SCREEN IS SCROLLED UP, THE ATTRIBUTE FOR FILLING THE NEWLY
4805 : BLANKED LINE IS READ FROM THE CURSOR POSITION ON THE PREVIOUS
4806 : LINE BEFORE THE SCROLL, IN CHARACTER MODE. IN GRAPHICS MODE,
4807 : THE 0 COLOR IS USED.
4808 : ENTRY
4809 : (AH) = CURRENT CRT MODE
4810 : (AL) = CHARACTER TO BE WRITTEN
4811 : NOTE THAT BACK SPACE, CAR RET, BELL AND LINE FEED ARE HANDLED
4812 : AS COMMANDS RATHER THAN AS DISPLAYABLE GRAPHICS
4813 : (BL) = FOREGROUND COLOR FOR CHAR WRITE IF CURRENTLY IN A
4814 : GRAPHICS MODE
4815 : EXIT
4816 : ALL REGISTERS SAVED
4817 :-----
4818 : ASSUME CS:CODE,DS:DATA
4819 : WRITE_TTY PROC NEAR
4820 :     PUSH AX
4821 :     PUSH AX
4822 :     MOV AH,3
4823 :     MOV BH,ACTIVE_PAGE
4824 :     INT 10H
4825 :     POP AX
4826 :
4827 : ;----- DX NOW HAS THE CURRENT CURSOR POSITION
4828 :
4829 :     CMP AL,8
4830 :     JE U8
4831 :     CMP AL,DDH
4832 :     JE U9
4833 :     CMP AL,0AH
4834 :     JE U10
4835 :     CMP AL,07H
4836 :     JE U11
4837 :
4838 : ;----- WRITE THE CHAR TO THE SCREEN
4839 :
4840 :
4841 :     MOV AH,10
4842 :     MOV CX,1
4843 :     INT 10H
4844 :
4845 : ;----- POSITION THE CURSOR FOR NEXT CHAR
4846 :
4847 :     INC DL
4848 :     CMP DL,BYTE PTR CRT_COLS
4849 :     JNZ U7
4850 :     MOV DL,0
4851 :     CMP DH,24
4852 :     JNZ U6
4853 :
4854 : ;----- SCROLL REQUIRED
4855 :
4856 : U1:     MOV AH,2
4857 :     INT 10H
4858 :
4859 : ;----- DETERMINE VALUE TO FILL WITH DURING SCROLL
4860 :
4861 :     MOV AL,CRT_MODE
4862 :     CMP AL,4
4863 :     JC U2
4864 :     MOV AH,7
4865 :     MOV BH,0
4866 :     JNE U3
4867 :
4868 : U2:     MOV AH,8
4869 :     INT 10H
4870 :     MOV BH,AH
4871 :
4872 : U3:     MOV AX,601H
4873 :     SUB CX,CX
4874 :     MOV DH,24
4875 :     MOV DL,BYTE PTR CRT_COLS
4876 :     DEC DL
4877 :
4878 : U4:     INT 10H
4879 :
4880 : U5:     POP AX
4881 :     JMP VIDEO_RETURN
4882 :
4883 : U6:     INC DH
4884 :
4885 : U7:     MOV AH,2
4886 :     JMP U4
4887 :
4888 : ;----- BACK SPACE FOUND
4889 :
4890 :
4891 : U8:     CMP DL,0
4892 :     JE U7
4893 :     DEC DL
4894 :     JMP U7
4895 :
4896 : ;----- CARRIAGE RETURN FOUND
4897 :
4898 :
4899 : U9:     MOV DL,0
4900 :     JMP U7
4901 :
4902 : ;----- LINE FEED FOUND
4903 :
4904 :
4905 : U10:    CMP DH,24
4906 :     JE U5
4907 :     JMP U1
4908 :

```

LOC	OBJECT	LINE	SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82
4909		4909	
4910		4910	;----- BELL FOUND
4911		4911	
F78D		4912	UII:
F78D B302		4913	MOV BL,2 ; SET UP COUNT FOR BEEP
F78F E87602		4914	CALL BEEP ; SOUND THE POD BELL
F792 E8DB		4915	JMP US ; TTY_RETURN
		4916	WRITE_TTY ENDP
		4917	;-----
		4918	; LIGHT PEN
		4919	; THIS ROUTINE TESTS THE LIGHT PEN SWITCH AND THE LIGHT
		4920	; PEN TRIGGER. IF BOTH ARE SET, THE LOCATION OF THE LIGHT
		4921	; PEN IS DETERMINED. OTHERWISE, A RETURN WITH NO
		4922	; INFORMATION IS MADE.
		4923	; ON EXIT
		4924	; (AH) = 0 IF NO LIGHT PEN INFORMATION IS AVAILABLE
		4925	; BX,CX,DX ARE DESTROYED
		4926	; (AH) = 1 IF LIGHT PEN IS AVAILABLE
		4927	; (DH,DL) = ROW,COLUMN OF CURRENT LIGHT PEN
		4928	; POSITION
		4929	; (CH) = RASTER POSITION
		4930	; (BX) = BEST GUESS AT PIXEL HORIZONTAL POSITION
		4931	;-----
		4932	ASSUME CS:CODE,DS:DATA
F794		4933	;----- SUBTRACT TABLE
F794 03		4934	VI LABEL BYTE
F795 03		4935	DB 3,3,5,5,3,3,3,4 ;
F796 05			
F797 05			
F798 03			
F799 03			
F79A 03			
F79B 04			
F79C		4936	READ_LPEN PROC NEAR
		4937	
		4938	;----- WAIT FOR LIGHT PEN TO BE DEPRESSED
		4939	
F79C B400		4940	MOV AH,0 ; SET NO LIGHT PEN RETURN CODE
F79E 8B166300		4941	MOV DX,ADDR_6845 ; GET BASE ADDRESS OF 6845
FA2 83C206		4942	ADD DX,6 ; POINT TO STATUS REGISTER
FA5 EC		4943	IN AL,DX ; GET STATUS REGISTER
FA6 A804		4944	TEST AL,4 ; TEST LIGHT PEN SWITCH
FA8 757E		4945	JNZ V6 ; NOT SET, RETURN
		4946	
		4947	;----- NOW TEST FOR LIGHT PEN TRIGGER
		4948	
F7AA A802		4949	TEST AL,2 ; TEST LIGHT PEN TRIGGER
F7AC 7503		4950	V7A ; RETURN WITHOUT RESETTING TRIGGER
F7AE E98100		4951	JMP V7
		4952	
		4953	;----- TRIGGER HAS BEEN SET, READ THE VALUE IN
F7B1		4954	V7A:
F7B1 B410		4955	MOV AH,16 ; LIGHT PEN REGISTERS ON 6845
		4956	
		4957	;----- INPUT REGS POINTED TO BY AH, AND CONVERT TO ROW COLUMN IN DX
		4958	
F7B3 8B166300		4959	
F7B7 8AC4		4960	MOV DX,ADDR_6845 ; ADDRESS REGISTER FOR 6845
F7B9 EE		4961	MOV AL,AH ; REGISTER TO READ
F7BA 42		4962	OUT DX,AL ; SET IT UP
F7BB EC		4963	INC DX ; DATA REGISTER
F7BC 8AE8		4964	IN AL,DX ; GET THE VALUE
F7BE 4A		4965	MOV CH,AL ; SAVE IN CX
F7BF FEC4		4966	DEC DX ; ADDRESS REGISTER
F7C1 8AC4		4967	INC AH
F7C3 EE		4968	MOV AL,AH ; SECOND DATA REGISTER
F7C4 42		4969	OUT DX,AL
F7C5 EC		4970	INC DX ; POINT TO DATA REGISTER
F7C6 8AE5		4971	IN AL,DX ; GET SECOND DATA VALUE
		4972	MOV AH,CH ; AX HAS INPUT VALUE
		4973	
		4974	;----- AX HAS THE VALUE READ IN FROM THE 6845
		4975	
F7C8 8A1E4900		4976	MOV BL,CRT_MODE
F7CC 2AFF		4977	SUB BH,BH ; MODE VALUE TO BX
F7CE 2E8A9F94F7		4978	BL,CS:V1[BX] ; DETERMINE AMOUNT TO SUBTRACT
F7D3 2BC3		4979	SUB AX,BX ; TAKE IT AWAY
F7D5 8B1E4E00		4980	MOV BX,CRT_START
F7D9 D1EB		4981	SHR BX,1
F7DB 2BC3		4982	SUB AX,BX
F7DD 7902		4983	JNS V2 ; IF POSITIVE, DETERMINE MODE
F7DF 2BC0		4984	SUB AX,AX ; <0 PLAYS AS 0
		4985	
		4986	;----- DETERMINE MODE OF OPERATION
		4987	
F7E1		4988	V2:
F7E1 B103		4989	MOV CL,3 ; DETERMINE MODE
F7E3 803E490004		4990	CMP CRT_MODE,4 ; SET *8 SHIFT COUNT
F7E8 722A		4991	JB V4 ; DETERMINE IF GRAPHICS OR ALPHA
F7EA 803E490007		4992	CMP CRT_MODE,7 ; ALPHA_PEN
F7EF 7423		4993	JE V4 ; ALPHA_PEN
		4994	
		4995	;----- GRAPHICS MODE
		4996	
F7F1 B228		4997	MOV DL,40 ; DIVISOR FOR GRAPHICS
F7F3 F6F2		4998	DIV DL ; DETERMINE ROW(AL) AND COLUMN(AH)
		4999	; AL RANGE 0-99, AH RANGE 0-39
		5000	
		5001	;----- DETERMINE GRAPHIC ROW POSITION
		5002	
F7F5 8AE8		5003	MOV CH,AL ; SAVE ROW VALUE IN CH
F7F7 02ED		5004	ADD CH,CH ; *2 FOR EVEN/ODD FIELD
F7F9 8ADC		5005	MOV BL,AH ; COLUMN VALUE TO BX
F7FB 2AFF		5006	SUB BH,BH ; MULTIPLY BY 8 FOR MEDIUM RES
F7FD 803E490006		5007	CMP CRT_MODE,6 ; DETERMINE MEDIUM OR HIGH RES
F802 7504		5008	JNE V3 ; NOT HIGH RES
F804 B104		5009	MOV CL,4 ; SHIFT VALUE FOR HIGH RES
F806 D0E4		5010	SAL AH,1 ; COLUMN VALUE TIMES 2 FOR HIGH RES
F808		5011	V3: ; NOT HIGH RES
F808 D3E3		5012	SHL BX,CL ; MULTIPLY *16 FOR HIGH RES

LOC	OBJECT	LINE	SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82
		5013	
		5014	;----- DETERMINE ALPHA CHAR POSITION
		5015	
F80A	8AD4	5016	MOV DL,AH ; COLUMN VALUE FOR RETURN
F80C	8AF0	5017	MOV DH,AL ; ROW VALUE
F80E	D0EE	5018	SHR DH,1 ; DIVIDE BY 4
F810	D0EE	5019	SHR DH,1 ; FOR VALUE IN 0-24 RANGE
F812	EB12	5020	JMP SHORT V5 ; LIGHT_PEN_RETURN_SET
		5021	
		5022	;----- ALPHA MODE ON LIGHT PEN
		5023	
F814		5024	V4: ; ALPHA_PEN
F814	F6364A00	5025	DIV BYTE PTR CRT_COLS ; DETERMINE ROW,COLUMN VALUE
F818	8AF0	5026	MOV DH,AL ; ROWS TO DH
F81A	8AD4	5027	MOV DL,AH ; COLS TO DL
F81C	D2E0	5028	SAL AL,CL ; MULTIPLY ROWS * 8
F81E	8AE8	5029	MOV CH,AL ; GET RASTER VALUE TO RETURN REG
F820	8ADC	5030	MOV BL,AH ; COLUMN VALUE
F822	32FF	5031	XOR BH,BH ; TO BX
F824	D3E3	5032	SAL BX,CL
F826		5033	V5: ; LIGHT_PEN_RETURN_SET
F826	B401	5034	MOV AH,1 ; INDICATE EVERYTHING SET
F828		5035	V6: ; LIGHT_PEN_RETURN
F828	52	5036	PUSH DX ; SAVE RETURN VALUE (IN CASE)
F829	8B166300	5037	MOV DX,ADDR_6845 ; GET BASE ADDRESS
F82D	83C207	5038	ADD DX,7 ; POINT TO RESET PARM
F830	EE	5039	OUT DX,AL ; ADDRESS, NOT DATA, IS IMPORTANT
F831	5A	5040	POP DX ; RECOVER VALUE
F832		5041	V7: ; RETURN_NO_RESET
F832	5F	5042	POP DI
F833	5E	5043	POP SI
F834	1F	5044	POP DS ; DISCARD SAVED BX,CX,DX
F835	1F	5045	POP DS
F836	1F	5046	POP DS
F837	1F	5047	POP DS
F838	07	5048	POP ES
F839	CF	5049	IRET
		5050	READ_LPEN ENDP

```

5051 :-----
5052 :--- INT 12 -----
5053 : MEMORY_SIZE_DET
5054 : THIS ROUTINE DETERMINES THE AMOUNT OF MEMORY IN THE SYSTEM
5055 : AS REPRESENTED BY THE SWITCHES ON THE PLANAR. NOTE THAT THE
5056 : SYSTEM MAY NOT BE ABLE TO USE I/O MEMORY UNLESS THERE IS A FULL
5057 : COMPLEMENT OF 64K BYTES ON THE PLANAR.
5058 : INPUT
5059 : NO REGISTERS
5060 : THE MEMORY_SIZE_VARIABLE IS SET DURING POWER ON DIAGNOSTICS
5061 : ACCORDING TO THE FOLLOWING HARDWARE ASSUMPTIONS:
5062 : PORT 60 BITS 3,2 = 00 - 16K BASE RAM
5063 : 01 - 32K BASE RAM
5064 : 10 - 48K BASE RAM
5065 : 11 - 64K BASE RAM
5066 : PORT 62 BITS 3-0 INDICATE AMOUNT OF I/O RAM IN 32K INCREMENTS
5067 : E.G., 0000 - NO RAM IN I/O CHANNEL
5068 : 0010 - 64K RAM IN I/O CHANNEL, ETC.
5069 : OUTPUT
5070 : (AX) = NUMBER OF CONTIGUOUS 1K BLOCKS OF MEMORY
5071 :-----
5072 : ASSUME CS:CODE,DS:DATA
5073 : ORG 0F841H
5074 : MEMORY_SIZE_DET PROC FAR
5075 :
5076 : PUSH DS ; INTERRUPTS BACK ON
5077 : CALL DDS ; SAVE SEGMENT
5078 : MOV AX,MEMORY_SIZE ; GET VALUE
5079 : POP DS ; RECOVER SEGMENT
5080 : IRET ; RETURN TO CALLER
5081 : MEMORY_SIZE_DET ENDP
5082 :
5083 :--- INT 11 -----
5084 : EQUIPMENT_DETERMINATION
5085 : THIS ROUTINE ATTEMPTS TO DETERMINE WHAT OPTIONAL
5086 : DEVICES ARE ATTACHED TO THE SYSTEM.
5087 : INPUT
5088 : NO REGISTERS
5089 : THE EQUIP_FLAG_VARIABLE IS SET DURING THE POWER ON
5090 : DIAGNOSTICS USING THE FOLLOWING HARDWARE ASSUMPTIONS:
5091 : PORT 60 = LOW ORDER BYTE OF EQUIPMENT
5092 : PORT 3FA = INTERRUPT ID REGISTER OF 8250
5093 : BITS 7-3 ARE ALWAYS 0
5094 : PORT 378 = OUTPUT PORT OF PRINTER -- 8255 PORT THAT
5095 : CAN BE READ AS WELL AS WRITTEN
5096 : OUTPUT
5097 : (AX) IS SET, BIT SIGNIFICANT, TO INDICATE ATTACHED I/O
5098 : BIT 15,14 = NUMBER OF PRINTERS ATTACHED
5099 : BIT 13 NOT USED
5100 : BIT 12 = GAME I/O ATTACHED
5101 : BIT 11,10,9 = NUMBER OF RS232 CARDS ATTACHED
5102 : BIT 8 UNUSED
5103 : BIT 7,6 = NUMBER OF DISKETTE DRIVES
5104 : 00=1, 01=2, 10=3, 11=4 ONLY IF BIT 0 = 1
5105 : BIT 5,4 = INITIAL VIDEO MODE
5106 : 00 - UNUSED
5107 : 01 - 40X25 BW USING COLOR CARD
5108 : 10 - 80X25 BW USING COLOR CARD
5109 : 11 - 80X25 BW USING BW CARD
5110 : BIT 3,2 = PLANAR RAM SIZE (00=16K,01=32K,10=48K,11=64K)
5111 : BIT 1 NOT USED
5112 : BIT 0 = IPL FROM DISKETTE -- THIS BIT INDICATES THAT
5113 : THERE ARE DISKETTE DRIVES ON THE SYSTEM
5114 :
5115 : NO OTHER REGISTERS AFFECTED
5116 :-----
5117 : ASSUME CS:CODE,DS:DATA
5118 : ORG 0F84DH
5119 : EQUIPMENT_DET PROC FAR
5120 :
5121 : PUSH DS ; INTERRUPTS BACK ON
5122 : CALL DDS ; SAVE SEGMENT REGISTER
5123 : MOV AX,EQUIP_FLAG ; GET THE CURRENT SETTINGS
5124 : POP DS ; RECOVER SEGMENT
5125 : IRET ; RETURN TO CALLER
5126 : EQUIPMENT_DET ENDP
5127 :
5128 :--- INT 15 -----
5129 : DUMMY CASSETTE I/O ROUTINE-RETURNS 'INVALID_CMD' IF THE ROUTINE IS
5130 : EVER CALLED BY ACCIDENT (AH=86H, CARRY_FLAG=1)
5131 :-----
5132 : ORG 0F859H
5133 : CASSETTE_I/O PROC FAR
5134 :
5135 : STC ; CARRY INDICATOR=1
5136 : MOV AH,86H
5137 : RET 2
5138 : CASSETTE_I/O ENDP

```

F841
F841
F841 FB
F842 IE
F843 E81302
F846 A11300
F849 IF
F84A CF

F84D
F84D
F84D FB
F84E IE
F84F E80702
F852 A11000
F855 IF
F856 CF

F859
F859
F859 F9
F85A B486
F85C CA0200

```

LOC OBJECT          LINE  SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82

5138
5139
5140 :-----
5141 : THIS ROUTINE WILL PRINT A PARITY CHECK 1 OR 2 MESSAGE :
5142 : AND ATTEMPT TO FIND THE STORAGE LOCATION CONTAINING THE :
5143 : BAD PARITY. IF FOUND, THE SEGMENT ADDRESS WILL BE :
5144 : PRINTED. IF NO PARITY ERROR CAN BE FOUND (INTERMITTANT :
5145 : READ PROBLEM) ??????-WILL BE PRINTED WHERE THE ADDRESS :
5146 : WOULD NORMALLY GO. :
5147 : IF ADDRESS IN ERROR IS IN THE I/O EXPANSION BOX, THE :
5148 : ADDRESS WILL BE FOLLOWED BY A '(E)', IF IN SYSTEM UNIT, :
5149 : A '(S)' WILL FOLLOW THE ADDRESS :
5150 :-----
F85F 5151 NMI_INT PROC NEAR
5152         ASSUME DS,DATA
5153         PUSH AX ; SAVE ORIG CONTENTS OF AX
5154         IN AL,PORT_C
5155         TEST AL,0C0H ; PARITY CHECK?
5156         JNZ NMI_1
5157         JMP D14 ; NO, EXIT FROM ROUTINE
F869 5158 NMI_1:
F86A 5159         MOV DX,DATA
F86C 5160         MOV DS,DX
F86E 5161         MOV SI,OFFSET D1 ; ADDR OF ERROR MSG
F872 5162         TEST AL,40H ; I/O PARITY CHECK
F874 5163         D13 D13 ; DISPLAY ERROR MSG
F876 5164         MOV SI,OFFSET D2 ; MUST BE PLANAR
F87A 5165
F87A 5166         MOV AH,0 ; INIT AND SET MODE FOR VIDEO
F87C 5167         MOV AL,CRT_MODE
F87F 5168         INT 10H ; CALL VIDEO I/O PROCEDURE
F881 5169         CALL P_MSG ; PRINT ERROR MSG
5170
5171 :----- SEE IF LOCATION THAT CAUSED PARITY CHECK CAN BE FOUND
5172
F884 5173         MOV AL,00H ; DISABLE TRAP
F886 5174         OUT 0A0H,AL
F888 5175         IN AL,PORT_B
F88A 5176         OR AL,00110000B ; TOGGLE PARITY CHECK ENABLES
F88C 5177         OUT PORT_B,AL
F88E 5178         AND AL,1T001111B
F890 5179         OUT PORT_B,AL
F892 5180         MOV BX,MEMORY_SIZE ; GET MEMORY SIZE WORD
F896 5181         CLD ; SET DIR FLAG TO INCREMENT
F897 5182         SUB DX,DX ; POINT DX AT START OF MEM
F899 5183 NMI_LOOP:
F899 5184         MOV DS,DX
F89B 5185         MOV ES,DX
F89D 5186         MOV CX,4000H ; SET FOR 16KB SCAN
F8A0 5187         SUB SI,SI ; SET SI TO BE RELATIVE TO
; START OF ES
; READ 16KB OF MEMORY
F8A2 5188
F8A3 5189         REP LODSB
F8A4 5190
F8A6 5191         IN AL,PORT_C ; SEE IF PARITY CHECK HAPPENED
F8A8 5192         AND AL,11000000B
F8AA 5193         JNZ PRT_NMI ; GO PRINT ADDRESS IF IT DID
F8AE 5194         ADD DX,0A0H ; POINT TO NEXT 16K BLOCK
F8B1 5195         JNZ NMI_LOOP
F8B3 5196         MOV SI,OFFSET D2A1 ; PRINT ROW OF ????? IF PARITY
F8B7 5197         CALL P_MSG ; CHECK COULD NOT BE RE-CREATED
F8BA 5198         CLI
F8BB 5199         HLT ; HALT SYSTEM
F8BC 5200
PRT_NMI:
F8BC 5201         MOV DX,DS
F8BE 5202         CALL PRT_SEG ; PRINT SEGMENT VALUE
F8C1 5203         MOV DX,0213H
F8C4 5204         MOV AL,00H ; DISABLE EXPANSION BOX
F8C6 5205         OUT DX,AL ; (CAN'T WRITE TO MEM)
F8C7 5206         MOV AL,'('
F8C9 5207         CALL PRT_HEX
F8CC 5208         MOV AX,0A55AH
F8CF 5209         MOV CX,AX
F8D1 5210         SUB BX,BX
F8D3 5211         MOV [BX],AX ; WRITE A WORD TO SEGMENT THAT
F8D5 5212         NOP
F8D6 5213         NOP
F8D7 5214         MOV AX,[BX] ; HAD THE ERROR
F8D9 5215         CMP AX,CX ; IS IT THERE?
F8DB 5216         JE SYS_BOX_ERR ; YES- MUST BE SYS UNIT
F8DD 5217         MOV AL,'E' ; NO-MUST BE IN EXP. BOX
F8DF 5218         CALL PRT_HEX
F8E2 5219         JMP SHORT HLT_NMI
F8E4 5220 SYS_BOX_ERR:
F8E4 5221         MOV AL,'S'
F8E6 5222         CALL PRT_HEX
F8E9 5223
HLT_NMI:
F8E9 5224         MOV AL,')'
F8EB 5225         CALL PRT_HEX
F8EE 5226         CLI ; HALT SYSTEM
F8EF 5227         HLT
F8F0 5228 D14:
F8F0 5229         POP AX ; RESTORE ORIG CONTENTS OF AX
F8F1 5230         IRET
5231 NMI_INT ENDP
5232
5233 :-----
5234 : ROS CHECKSUM SUBROUTINE :
5235 :-----
F8F2 5236 ROS_CHECKSUM PROC NEAR ; NEXT ROS_MODULE
F8F2 5237         MOV CX,8192 ; NUMBER OF BYTES TO ADD
F8F5 5238 ROS_CHECKSUM_CNT: ; ENTRY FOR OPTIONAL ROS TEST
F8F5 5239         XOR AL,AL
F8F7 5240 C26:
F8F7 5241         ADD AL,DS:[BX]
F8F9 5242         INC BX ; POINT TO NEXT BYTE
F8FA 5243         LOOP C26 ; ADD ALL BYTES IN ROS MODULE
F8FC 5244         OR AL,AL ; SUM = 0?
F8FE 5245         RET
5246 ROS_CHECKSUM ENDP

```


LOC OBJECT

LINE SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82

```

5247 :-----:
5248 : MESSAGE AREA FOR POST :
5249 :-----:
5250 E0 DB '101',13,10 ; SYSTEM BOARD ERROR

F8FF 313031
F902 0D
F903 0A
F904 20320301 5251 E1 DB '201',13,10 ; MEMORY ERROR
F908 0D
F909 0A
F90A 524F4D 5252 F3A DB 'ROM',13,10 ; ROM CHECKSUM ERROR
F90D 0D
F90E 0A
F90F 31383031 5253 F3C DB '1801',13,10 ; EXPANSION IO BOX ERROR
F913 0D
F914 0A
F915 50415249545920 5254 D1 DB 'PARITY CHECK 2',13,10
434845434B2032
F923 0D
F924 0A
F925 50415249545920 5255 D2 DB 'PARITY CHECK 1',13,10
434845434B2031
F933 0D
F934 0A
F935 3F3F3F3F3F 5256 D2A DB '?????',13,10
F93A 0D
F93B 0A

5257 :-----:
5258 : BLINK LED PROCEDURE FOR MFG RUN-IN TESTS :
5259 : IF LED IS ON, TURN IT OFF. IF OFF, TURN ON. :
5260 :-----:
5261 :
5262 : ASSUME DS:DATA
5263 BLINK_INT PROC NEAR
5264 STI AX
5265 PUSH AX ; SAVE AX REG. CONTENTS
5266 IN AL,PORT_B ; READ CURRENT VAL OF PORT B
5267 MOV AH,AL
5268 NOT AL ; FLIP ALL BITS
5269 AND AL,01000000B ; ISOLATE CONTROL BIT
5270 AND AH,01111111B ; MASK OUT OF ORIGINAL VAL
5271 OR AL,AH ; OR NEW CONTROL BIT IN
5272 OUT PORT_B,AL
5273 MOV AL,ED1
5274 OUT INTA00,AL ; RESTORE AX REG
5275 POP AX
5276 IRET
5277 BLINK_INT ENDP
5278
5279 :-----:
5280 : THIS ROUTINE CHECKS SUMS OPTIONAL ROM MODULES AND :
5281 : IF CHECKSUM IS OK, CALLS INIT/TEST CODE IN MODULE :
5282 :-----:
5283 ROM_CHECK PROC NEAR
5284 MOV ES,AX ; POINT ES TO DATA AREA
5285 MOV AX,AX ; ZERO OUT AH
5286 SUB AH,AH
5287 MOV AL,[BX+2] ; GET LENGTH INDICATOR
5288 CL_09H ; MULTIPLY BY 512
5289 SHL AX,CL
5290 MOV CX,AX ; SET COUNT
5291 PUSH CX ; SAVE COUNT
5292 MOV CX,4 ; ADJUST
5293 SHR AX,CL
5294 ADD DX,AX ; SET POINTER TO NEXT MODULE
5295 POP CX ; RETRIEVE COUNT
5296 CALL ROM_CHECKSUM_CNT ; DO CHECKSUM
5297 JZ ROM_CHECK_1
5298 CALL ROM_ERR ; POST CHECKSUM ERROR
5299 JMP ROM_CHECK_END
5300 ROM_CHECK_1:
5301 PUSH DX ; SAVE POINTER
5302 MOV ES:10_ROM_INIT,0003H ; LOAD OFFSET
5303 MOV ES:10_ROM_SEG,05 ; LOAD SEGMENT
5304 CALL DWORD PTR ES:10_ROM_INIT ; CALL INIT./TEST ROUTINE
5305 POP DX
5306 ROM_CHECK_END:
5307 RET ; RETURN TO CALLER
5308 ROM_CHECK ENDP
5309
5310 :-----:
5311 : CONVERT AND PRINT ASCII CODE :
5312 : AL MUST CONTAIN NUMBER TO BE CONVERTED. :
5313 : AX AND BX DESTROYED. :
5314 :-----:
5315 XPC_BYTE PROC NEAR
5316 PUSH AX ; SAVE FOR LOW NIBBLE DISPLAY
5317 MOV CL,4 ; SHIFT COUNT
5318 SHR AL,CL ; NYBBLE SWAP
5319 CALL XLAT_PR ; DO THE HIGH NIBBLE DISPLAY
5320 POP AX ; RECOVER THE NIBBLE
5321 AND AL,0FH ; ISOLATE TO LOW NIBBLE
5322 ; FALL INTO LOW NIBBLE CONVERSION
5323 XLAT_PR PROC NEAR
5324 ADD AL,090H ; CONVERT 00-OF TO ASCII CHARACTER
5325 DAA ; ADJUST FOR NUMERIC AND ALPHA RANGE
5326 ADD AL,040H ; ADD CONVERSION AND ADJUST LOW NIBBLE
5327 DAA ; ADJUST HIGH NIBBLE TO ASCII RANGE
5328 PRT_HEX PROC NEAR
5329 MOV AH,14 ; DISPLAY CHARACTER IN AL
5330 MOV BH,0
5331 INT 10H ; CALL VIDEO_10
5332 RET
5333 PRT_HEX ENDP
5334 XLAT_PR ENDP
5335 XPC_BYTE ENDP
5336
5337 F4 LABEL WORD ; PRINTER SOURCE TABLE
5338 DW 3BCH
5339 DW 378H
5340 DW 278H
5341 F4E LABEL WORD
5342
F98B
F98B 50
F98C B104
F98E 02E8
F990 E80300
F993 58
F994 240F
5322
F996
F996 0490
F998 27
F999 1440
F99B 27
F99C
F99C B40E
F99E B700
F9A0 CD10
F9A2 C3
5336
F9A3
F9A3 BC03
F9A5 7803
F9A7 7802
F9A9
5337
5338
5339
5340
5341
5342

```

```

LOC OBJECT      LINE  SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82
5343  ;-----
5344  ; THIS SUBROUTINE WILL PRINT A MESSAGE ON THE DISPLAY  ;
5345  ;-----
5346  ENTRY REQUIREMENTS:
5347  ; SI = OFFSET (ADDRESS) OF MESSAGE BUFFER
5348  ; CX = MESSAGE BYTE COUNT
5349  ; MAXIMUM MESSAGE LENGTH IS 36 CHARACTERS
5350  ;-----
F9A9      E_MSG  PROC    NEAR
F9AB      B8EE    MOV     BP,SI
F9AD      E81C00  CALL    P_MSG
F9AE      1E      PUSH    DS
F9AF      E8A700  CALL    DOS
F9B2      A01000  MOV     AL,BYTE PTR EQUIP_FLAG
F9B5      2401    AND     AL,01H
F9B7      750F    JNZ     G12
F9B9      MFG_HALT;
F9BB      FA      CLD
F9BA      B089    MOV     AL,B9H
F9BC      E663    OUT     CMD_PORT,AL
F9BE      B085    MOV     AL,10000101B
F9C0      E661    OUT     PORT_B,AL
F9C2      A01500  MOV     AL,MFG_ERR_FLAG
F9C5      E660    OUT     PORT_A,AL
F9C7      F4      HLT
F9C8      1F      G12:   POP     DS
F9C9      C3      E_MSG  ENDP
; WRITE_MSG:

F9CA      5373    P_MSG  PROC    NEAR
F9CB      5374    G12A:   MOV     AL,CS:[SI]
F9CC      2EBA04  INC     SI
F9CD      46      INC     AX
F9CE      50      CALL    PRT_HEX
F9CF      E8CAFF  AX      AX
F9D2      58      CMP     AL,10
F9D3      3C0A    JNE     G12A
F9D5      75F3    RET
F9D7      C3      P_MSG  ENDP
; NO,KEEP PRINTING STRING

5385  ;-----
5386  ; INITIAL RELIABILITY TEST -- SUBROUTINES
5387  ;-----
5388  ASSUME  CS:CODE,DS:DATA
5389  ;-----
5390  SUBROUTINES FOR POWER ON DIAGNOSTICS
5391  ;-----
5392  ; THIS PROCEDURE WILL ISSUE ONE LONG TONE (3 SECS) AND ONE OR
5393  ; MORE SHORT TONES (1 SEC) TO INDICATE A FAILURE ON THE PLANAR
5394  ; BOARD, A BAD RAM MODULE, OR A PROBLEM WITH THE CRT.
5395  ; ENTRY PARAMETERS:
5396  ; DH = NUMBER OF LONG TONES TO BEEP
5397  ; DL = NUMBER OF SHORT TONES TO BEEP.
5398  ;-----
F9D8      5399    ERR_BEEP PROC    NEAR
F9D9      5400    PUSHF
F9DA      1E      CLD
F9DB      E87B00  CALL    DS
F9DE      0AF6    OR      DH,DH
F9E0      7414    JZ      G3
F9E2      5406    MOV     BL,6
F9E4      E82100  CALL    BEEP
F9E7      G2:     LOOP    G2
F9E7      E2FE    DEC     DH
F9E9      FECE    JNZ     G1
F9EB      75F5    JNZ     G1
F9ED      803E120001  CMP     MFG_TST,1
F9F2      7502    JNE     G3
F9F4      EBC3    JMP     MFG_HALT
F9F6      G3:     MOV     BL,1
F9F6      B301    CALL    BEEP
F9F8      E8D000  MOV     BL,1
F9FB      5419    CALL    BEEP
F9FB      E2FE    LOOP    G4
F9FD      FECA    DEC     DL
F9FF      75F5    JNZ     G3
FA01      5423    G5:     LOOP    G5
FA01      E2FE    LOOP    G5
FA03      5425    G6:     LOOP    G6
FA03      E2FE    LOOP    G6
FA05      1F      POP     DS
FA06      9D      POPF
FA07      C3      RET
ERR_BEEP  ENDP

5430  ;-----
5431  ;-----
5432  ;-----
5433  ;-----
5434  BEEP  PROC    NEAR
5435      MOV     AL,10110110B
5436      OUT     TIMER+3,AL
5437      MOV     AX,533H
5438      OUT     TIMER+2,AL
5439      MOV     AL,AH
5440      OUT     TIMER+2,AL
5441      IN      AL,PORT_B
5442      MOV     AH,AL
5443      OR      AL,03
5444      OUT     PORT_B,AL
5445      SUB     CX,CX
5446      G7:     LOOP    G7
5447      DEC     BL
5448      JNZ     G7
5449      MOV     AL,AH
5450      OUT     PORT_B,AL
5451      RET
5452      BEEP  ENDP
; RETURN TO CALLER

```

LOC OBJECT

LINE SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82

```

5454
5455 ;-----
5456 ; THIS PROCEDURE WILL SEND A SOFTWARE RESET TO THE KEYBOARD. ;
5457 ; SCAN CODE AA SHOULD BE RETURNED TO THE CPU. ;
5458 ;-----
FA2A KBD_RESET PROC NEAR
5459 DS:AB50
5460 MOV AL,0B08 ; SET KBD CLK LINE LOW
5461 OUT PORT_B,AL ; WRITE 8255 PORT B
5462 MOV CX,10582H ; HOLD KBD CLK LOW FOR 20 MS
5463
G8:
5464 LOOP G8 ; LOOP FOR 20 MS
5465 MOV AL,0C8H ; SET CLK, ENABLE LINES HIGH
5466 OUT PORT_B,AL
5467
SP_TEST
5468 MOV AL,48H ; ENTRY FOR MANUFACTURING TEST 2
5469 OUT PORT_B,AL ; SET KBD CLK HIGH, ENABLE LOW
5470 MOV AL,0F0H ;
5471 INTA01,AL ; ENABLE KEYBOARD INTERRUPTS
5472 MOV DATA_AREA[OFFSET INTR_FLAG] ; WRITE 8259 IMR
5473 STI ; RESET INTERRUPT INDICATOR
5474 SUB CX,CX ; ENABLE INTERRUPTS
5475
G9:
5476 TEST DATA_AREA[OFFSET INTR_FLAG],02H ; DID A KEYBOARD INTR OCCUR?
5477 JNZ G10 ; YES - READ SCAN CODE RETURNED
5478 LOOP G9 ; NO - LOOP TILL TIMEOUT
5479
G10:
5480 IN AL,PORT_A ; READ KEYBOARD SCAN CODE
5481 MOV BL,AL ; SAVE SCAN CODE JUST READ
5482 MOV AL,0C8H ; CLEAR KEYBOARD
5483 OUT PORT_B,AL
5484 RET
5485 KBD_RESET ENDP
5486
FA59 DDS PROC NEAR
5487 PUSH AX ; SAVE AX
5488 MOV AX,DATA ;
5489 MOV DS,AX ; SET SEGMENT
5490 POP AX ; RESTORE AX
5491 RET
5492 DDS ENDP
5493
5494 ;-----
5495 ; CHARACTER GENERATOR GRAPHICS FOR 320X200 AND 640X200 GRAPHICS ;
5496 ;-----
FA6E ORG 0FA6EH
FA6E CRT_CHAR_GEN LABEL BYTE
5497 DB 000H,000H,000H,000H,000H,000H,000H,000H ; D_00
5498 DB 01EH,081H,0A5H,081H,0BDH,099H,081H,07EH ; D_01
5499 DB 07EH,0FFH,00BH,0FFH,0C3H,0E7H,0FFH,07EH ; D_02
5500 DB 06CH,0FEH,0FEH,0FEH,07CH,03BH,010H,000H ; D_03
5501 DB 010H,03BH,07CH,0FEH,07CH,03BH,010H,000H ; D_04
5502 DB 03BH,07CH,03BH,0FEH,0FEH,07CH,03BH,07CH ; D_05
5503 DB 010H,010H,03BH,07CH,0FEH,07CH,03BH,07CH ; D_06
5504 DB 000H,000H,01BH,03CH,03CH,01BH,000H,000H ; D_07
5505 DB 0FFH,0FFH,0E7H,0C3H,0C3H,0E7H,0FFH,0FFH ; D_08
5506 DB 000H,000H,03EH,042H,042H,066H,03EH,000H ; D_09
5507 DB 0FFH,0C3H,099H,0BDH,0BDH,099H,0C3H,0FFH ; D_0A
5508 DB 0FFH,077H,00FH,07DH,0CCH,0CCH,0CCH,07BH ; D_0B
5509 DB 03CH,06BH,06BH,06BH,03CH,01BH,07EH,01BH ; D_0C
5510 DB 03FH,03BH,03FH,03DH,03DH,070H,0F0H,0E0H ; D_0D
5511 DB 07FH,063H,07FH,063H,063H,067H,0E6H,0CCH ; D_0E
5512 DB 099H,05AH,03CH,0E7H,0E7H,03CH,05AH,099H ; D_0F
5513 DB 0BDH,0E0H,0F8H,0FEH,0F8H,0E0H,0BDH,000H ; D_10
5514 DB 002H,00EH,03EH,0FEH,03EH,00EH,002H,000H ; D_11
5515 DB 01BH,03CH,07EH,01BH,01BH,07EH,03CH,01BH ; D_12
5516 DB 066H,066H,066H,066H,066H,000H,066H,000H ; D_13
5517 DB 07FH,0BDH,0BDH,07BH,01BH,01BH,01BH,000H ; D_14
5518 DB 03EH,063H,03BH,06CH,06CH,03BH,0CCH,07BH ; D_15
5519 DB 000H,000H,000H,000H,07EH,07EH,07EH,000H ; D_16
5520 DB 01BH,03CH,07EH,01BH,07EH,03CH,01BH,0FFH ; D_17
5521 DB 01BH,03CH,07EH,01BH,01BH,01BH,01BH,000H ; D_18
5522 DB 01BH,01BH,01BH,01BH,07EH,03CH,01BH,000H ; D_19
5523 DB 000H,01BH,0CCH,0FEH,0CCH,01BH,000H,000H ; D_1A
5524 DB 000H,03BH,06CH,0FEH,06CH,03BH,000H,000H ; D_1B
5525 DB 000H,000H,0CCH,0CCH,0CCH,0FEH,000H,000H ; D_1C
5526 DB 000H,024H,066H,0FFH,066H,024H,000H,000H ; D_1D
5527 DB 000H,01BH,03CH,07EH,0FFH,0FFH,000H,000H ; D_1E
5528 DB 000H,0FFH,0FFH,07EH,03CH,01BH,000H,000H ; D_1F
5529 DB 030H,07BH,07BH,030H,030H,000H,030H,000H ; D_20
5530 DB 06CH,06CH,06CH,000H,000H,000H,000H,000H ; D_21
5531 DB 06CH,06CH,0FEH,06CH,0FEH,06CH,06CH,000H ; D_22
5532 DB 030H,07CH,0CCH,07BH,0CCH,07BH,030H,000H ; D_23
5533 DB 000H,0C6H,0CCH,01BH,030H,066H,0C6H,000H ; D_24
5534 DB 03BH,0C6H,0CCH,076H,0CCH,0CCH,076H,000H ; D_25
5535 DB 060H,060H,0CCH,000H,000H,000H,000H,000H ; D_26
5536 DB 01BH,030H,060H,060H,060H,030H,01BH,000H ; D_27
5537 DB 060H,030H,01BH,01BH,01BH,030H,060H,000H ; D_28
5538 DB 000H,066H,03CH,0FFH,03CH,066H,000H,000H ; D_29
5539 DB 000H,030H,030H,0CCH,030H,030H,000H,000H ; D_2A
5540 DB 000H,000H,000H,0CCH,030H,030H,060H,000H ; D_2B
5541 DB 000H,000H,000H,0CCH,000H,000H,000H,000H ; D_2C
5542 DB 000H,000H,000H,0CCH,000H,000H,000H,000H ; D_2D
5543 DB 006H,0CCH,01BH,030H,060H,0CCH,080H,000H ; D_2E
5544 DB 07CH,0CCH,0E6H,0E6H,0F6H,0E6H,07CH,000H ; D_2F
5545 DB 030H,070H,030H,030H,030H,030H,07CH,000H ; D_30
5546 DB 07BH,0CCH,0CCH,03BH,060H,0CCH,0FCH,000H ; D_31
5547 DB 07BH,0CCH,0CCH,03BH,060H,0CCH,0FCH,000H ; D_32
5548 DB 01CH,03CH,06CH,0CCH,0FEH,0CCH,01EH,000H ; D_33
5549 DB 0FCH,0CCH,0F8H,0CCH,0CCH,0CCH,0F8H,000H ; D_34
5550 DB 03BH,0CCH,0CCH,0F8H,0CCH,0F8H,0CCH,000H ; D_35
5551 DB 0FCH,0CCH,0CCH,01BH,030H,030H,030H,000H ; D_36
5552 DB 07BH,0CCH,0CCH,07BH,0CCH,0CCH,07BH,000H ; D_37
5553 DB 000H,030H,030H,000H,000H,030H,070H,000H ; D_38
5554 DB 000H,030H,030H,000H,000H,030H,030H,060H ; D_39
5555 DB 01BH,030H,060H,0CCH,060H,030H,01BH,000H ; D_3A
5556 DB 000H,000H,0FCH,000H,000H,0FCH,000H,000H ; D_3B
5557 DB 060H,030H,01BH,0CCH,01BH,0CCH,060H,000H ; D_3C
5558 DB 07BH,0CCH,0CCH,01BH,030H,000H,030H,000H ; D_3D
5559 DB 07BH,0CCH,0CCH,01BH,030H,000H,030H,000H ; D_3E
5560 DB 07BH,0CCH,0CCH,01BH,030H,000H,030H,000H ; D_3F

```

```

LOC OBJECT                               LINE  SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82

FC6E 7C6CDEDEDEC07800 5565     DB      01CH,0C6H,0DEH,0DEH,0DEH,0C0H,078H,000H ; P D 40
FC76 3078C6CFCGCC0000 5566     DB      030H,078H,0CCH,0CCH,0CCH,0FCH,0CCH,0CCH,000H ; A D 41
FC7E FC66667C6666FC00 5567     DB      0FCH,066H,066H,07CH,066H,066H,0FCH,000H ; B D 42
FC86 3C66C0C0C0663C00 5568     DB      03CH,066H,0C0H,0C0H,0C0H,066H,03CH,000H ; C D 43
FC8E F8C6C6666666CF80 5569     DB      0F8H,0C6H,066H,066H,066H,06CH,0F8H,000H ; D D 44
FC96 FE6268786862FE00 5570     DB      0FEH,062H,068H,078H,068H,062H,0FEH,000H ; E D 45
FC9E FE6268786862FE00 5571     DB      0FEH,062H,068H,078H,068H,060H,0F0H,000H ; F D 46
FCA6 3C66C0C0C0663C00 5572     DB      03CH,066H,0C0H,0C0H,0CEH,066H,03EH,000H ; G D 47
FCAE CCCCCCFC6CCCC000 5573     DB      0CCH,0CCH,0CCH,0FCH,0CCH,0CCH,0CCH,000H ; H D 48
FCB6 7830303030307800 5574     DB      078H,030H,030H,030H,030H,030H,078H,000H ; I D 49
FCBE 1E0C0C0C0C0C0C7800 5575     DB      01EH,0CCH,0CCH,0CCH,0CCH,0CCH,078H,000H ; J D 4A
FCCE E66666786C666600 5576     DB      0E6H,066H,06CH,078H,066H,066H,0E6H,000H ; K D 4B
FCEE F06060606266FE00 5577     DB      0F0H,060H,060H,066H,062H,066H,0F0H,000H ; L D 4C
FCD6 C6EEFEF0D6C6C600 5578     DB      0C6H,0EEH,0FEH,0FEH,0D6H,0C6H,0C6H,000H ; M D 4D
FCE6 C6E6F6DECEC6C600 5579     DB      0C6H,0E6H,0F6H,0DEH,0CEH,0C6H,0C6H,000H ; N D 4E
FCEB 38C6C6C6C6C63800 5580     DB      038H,0C6H,0C6H,0C6H,0C6H,06CH,038H,000H ; O D 4F
FCEE FC66667C6060F000 5581     DB      0FCH,066H,066H,07CH,066H,0CCH,0CCH,000H ; P D 50
FCF6 78CCCCC0DCT781C00 5582     DB      078H,0CCH,0CCH,0CCH,0DCH,078H,01CH,000H ; Q D 51
FCFE FC66667C6066E600 5583     DB      0FCH,066H,066H,07CH,066H,066H,0E6H,000H ; R D 52
FDD6 78C0C07010C0C780 5584     DB      078H,0CCH,0E0H,078H,070H,01CH,0CCH,078H,000H ; T D 53
FDDF FCB4303030307800 5585     DB      0FCH,0B4H,030H,030H,030H,030H,078H,000H ; U D 54
FD16 CCCCCC0C0C0CFC00 5586     DB      0CCH,0CCH,0CCH,0CCH,0CCH,0CCH,0FCH,000H ; U D 55
FD1E CCCCCC0C0C0C783000 5587     DB      0CCH,0CCH,0CCH,0CCH,0CCH,0CCH,078H,000H ; V D 56
FD26 C6C6C6D6FEEEC600 5588     DB      0C6H,0C6H,0C6H,0D6H,0FEH,0E6H,0C6H,000H ; W D 57
FD2E C6C6C63838C6C600 5589     DB      0C6H,0C6H,0C6H,038H,038H,06CH,0C6H,000H ; X D 58
FD36 CCCCCC7830307800 5590     DB      0CCH,0CCH,0CCH,078H,030H,030H,078H,000H ; Y D 59
FD3E FE6C9C183264FE00 5591     DB      0FEH,0C6H,08CH,018H,032H,066H,0FEH,000H ; Z D 5A
FD46 7860606060607800 5592     DB      078H,06H,060H,060H,060H,060H,078H,000H ; [ D 5B
FD4E C06030180C060200 5593     DB      0C0H,060H,030H,018H,0C0H,066H,0D2H,000H ; BACKSLASH D_5C
FD56 7818181818187800 5594     DB      078H,018H,018H,018H,018H,018H,078H,000H ; [ D 5D
FD66 10386C6A00000000 5595     DB      010H,038H,06CH,066H,000H,000H,000H,000H ; CIRCUMFLEX D_5E
FD6E 00000000000000FF 5596     DB      000H,000H,000H,000H,000H,000H,000H,000H ; D 5F
FD76 3030180000000000 5597     DB      030H,030H,018H,000H,000H,000H,000H,000H ; T D 60
FD7E 000078C0C7C7C71600 5598     DB      000H,000H,078H,00C7,0C7C,0CCH,078H,000H ; LOWER CASE A_D_61
FD86 E060607C8664D0C0 5599     DB      0E0H,060H,06CH,078H,066H,066H,0DCH,000H ; L.C. B D 62
FD8E 000078CC0C0C7800 5600     DB      000H,000H,078H,0CCH,0CCH,0CCH,078H,000H ; L.C. C D 63
FD96 1C0C0C7C0C0C7C1600 5601     DB      01CH,0CCH,0CCH,078H,0CCH,0CCH,076H,000H ; L.C. D D 64
FD9E 386C6F0606060F00 5603     DB      000H,000H,078H,0CCH,0FCH,0CCH,078H,000H ; L.C. E D 65
FDA6 00007C6C7C0C0CF8 5604     DB      000H,000H,076H,0CCH,0CCH,0CCH,0F8H,000H ; L.C. F D 66
FDAE E060C7666666E600 5605     DB      0E0H,060H,06CH,076H,066H,066H,0E6H,000H ; L.C. H D 68
FDB6 3000703030307800 5606     DB      030H,000H,070H,030H,030H,030H,078H,000H ; L.C. J D 6A
FDC6 E060666C786C6800 5608     DB      0E0H,060H,066H,06CH,078H,06CH,0E6H,000H ; L.C. K D 69
FDE6 7030303030307800 5609     DB      070H,030H,030H,030H,030H,030H,078H,000H ; L.C. L D 6C
FDEB 0000CFE7ED6C6600 5610     DB      000H,000H,0CCH,0FEH,0FEH,0D6H,0C6H,000H ; L.C. N D 6E
FDEE 000078CCCC0C0C16 5611     DB      000H,000H,0F8H,0CCH,0CCH,0CCH,0CCH,000H ; L.C. O D 67
FDEA 000078CCCC0C0C16 5612     DB      000H,000H,078H,0CCH,0CCH,0CCH,078H,000H ; L.C. M D 6F
FDEE 0000DC66667C60F0 5613     DB      000H,000H,0DCH,066H,066H,07CH,060H,0F0H ; L.C. P D 70
FDFA 00007C6C7C0C0C16 5614     DB      000H,000H,076H,0CCH,0CCH,0CCH,0CCH,01EH ; L.C. Q D 71
FDFE 0000DCT76666F000 5615     DB      000H,000H,0DCH,076H,066H,066H,0F0H,000H ; L.C. R D 72
FE06 00007C0780CF8000 5616     DB      000H,000H,07CH,0C0H,078H,0CCH,0F8H,000H ; L.C. S D 73
FE0E 10307C3030341800 5617     DB      010H,030H,07CH,030H,030H,034H,018H,000H ; L.C. T D 74
FE16 0000C0C0C0C0C160 5618     DB      000H,000H,0CCH,0CCH,0CCH,0CCH,076H,000H ; L.C. U D 75
FE1E 0000CCCCC7830000 5619     DB      000H,000H,0CCH,0CCH,0CCH,0CCH,078H,000H ; L.C. V D 76
FE26 0000C6D6FEFE6C00 5620     DB      000H,000H,0C6H,0D6H,0FEH,0FEH,0C6H,000H ; L.C. W D 77
FE2E 0000C6386C38C6C600 5621     DB      000H,000H,0C6H,06CH,038H,06CH,0C6H,000H ; L.C. X D 78
FE36 0000C0C0C0C0C0CF8 5622     DB      000H,000H,0CCH,0CCH,0CCH,0CCH,0CCH,0F8H ; L.C. Y D 79
FE3E 0000F9C83064FC00 5623     DB      000H,000H,0FCH,098H,030H,064H,0FCH,000H ; L.C. Z D 7A
FE46 1C3030E030301C00 5624     DB      01CH,030H,030H,0E0H,030H,030H,01CH,000H ; [ D 7B
FE4E 1818180018181800 5625     DB      018H,018H,018H,000H,018H,018H,018H,000H ; [ D 7C
FE56 E030301C3030E000 5626     DB      0E0H,030H,030H,01CH,030H,030H,0E0H,000H ; [ D 7D
FE5E 76C0D00000000000 5627     DB      076H,0DCH,000H,000H,000H,000H,000H ; TILDE D_7E
FE66 0010386C6C66FE00 5628     DB      000H,010H,038H,06CH,0C6H,0C6H,0FEH,000H ; DELTA D_7F

5630 :--- INT IA ---
5631 : TIME_OF_DAY
5632 : THIS ROUTINE ALLOWS THE CLOCK TO BE SET/READ
5633 :
5634 : INPUT
5635 : (AH) = 0 READ THE CURRENT CLOCK SETTING
5636 : RETURNS CX = HIGH PORTION OF COUNT
5637 : DX = LOW PORTION OF COUNT
5638 : AL = 0 IF TIMER HAS NOT PASSED
5639 : 24 HOURS SINCE LAST READ
5640 : <0 IF ON ANOTHER DAY
5641 : (AH) = 1 SET THE CURRENT CLOCK
5642 : CX = HIGH PORTION OF COUNT
5643 : DX = LOW PORTION OF COUNT
5644 : NOTE: COUNTS OCCUR AT THE RATE OF
5645 : 1193180/65536 COUNTS/SEC
5646 : (OR ABOUT 18.2 PER SECOND -- SEE EQUATES BELOW)
5647 :
5648 ASSUME CS:CODE,DS:DATA
5649 ORG 0FE6H
5650 TIME_OF_DAY PROC FAR
5651 STI ; INTERRUPTS BACK ON
5652 PUSH DS ; SAVE SEGMENT
5653 CALL DDS
5654 OR AH,AH ; AH=0
5655 JZ T2 ; READ_TIME
5656 DEC AH ; AH=1
5657 JZ T3 ; SET_TIME
5658 T1: ; TOD_RETURN
5659 STI ; INTERRUPTS BACK ON
5660 POP DS ; RECOVER SEGMENT
5661 IRET ; RETURN TO CALLER
5662 T2: ; READ_TIME
5663 CLI ; NO TIMER INTERRUPTS WHILE READING
5664 MOV AL,TIMER_OFL
5665 MOV CX,TIMER_HIGH
5666 MOV DX,TIMER_LOW
5667 JMP T3 ; TOD_RETURN
5668 T3: ; SET_TIME
5669 CLI ; NO INTERRUPTS WHILE WRITING
5670 MOV TIMER_LOW,DX
5671 MOV TIMER_HIGH,CX
5672 MOV TIMER_OFL,0
5673 JMP T1 ; RESET_OVERFLOW
5674 T1: ; TOD_RETURN
5675 TIME_OF_DAY ENDP

```

```

5676
5677 :-----
5678 : THIS ROUTINE HANDLES THE TIMER INTERRUPT FROM :
5679 : CHANNEL 0 OF THE 8253 TIMER. INPUT FREQUENCY :
5680 : IS 1.19318 MHZ AND THE DIVISOR IS 65536, RESULTING :
5681 : IN APPROX. 18.2 INTERRUPTS EVERY SECOND. :
5682 : :
5683 : THE INTERRUPT HANDLER MAINTAINS A COUNT OF INTERRUPTS :
5684 : SINCE POWER ON TIME, WHICH MAY BE USED TO ESTABLISH :
5685 : TIME OF DAY. :
5686 : THE INTERRUPT HANDLER ALSO DECREMENTS THE MOTOR :
5687 : CONTROL COUNT OF THE DISKETTE, AND WHEN IT EXPIRES, :
5688 : WILL TURN OFF THE DISKETTE MOTOR, AND RESET THE :
5689 : MOTOR RUNNING FLAGS. :
5690 : THE INTERRUPT HANDLER WILL ALSO INVOKE A USER ROUTINE :
5691 : THROUGH INTERRUPT ICH AT EVERY TIME TICK. THE USER :
5692 : MUST CODE A ROUTINE AND PLACE THE CORRECT ADDRESS IN :
5693 : THE VECTOR TABLE. :
5694 :-----
5695 ORG OFEA5H
5696 TIMER_INT PROC FAR
5697
5698 STI DS ; INTERRUPTS BACK ON
5699 PUSH AX
5700 PUSH DX ; SAVE MACHINE STATE
5701 CALL DDS
5702 INC TIMER_LOW ; INCREMENT TIME
5703 JNZ T4 ; TEST DAY
5704 INC TIMER_HIGH ; INCREMENT HIGH WORD OF TIME
5705
5706 T4: CMP TIMER_HIGH,018H ; TEST FOR COUNT EQUALING 24 HOURS
5707 JNZ T5 ; TEST DAY
5708 CMP TIMER_LOW,0B0H ; DISKETTE_CTL
5709 JNZ T5
5710
5711 :----- TIMER HAS GONE 24 HOURS
5712
5713 SUB AX,AX
5714 MOV TIMER_HIGH,AX
5715 MOV TIMER_LOW,AX
5716 MOV TIMER_OFL,I
5717
5718 :----- TEST FOR DISKETTE TIME OUT
5719
5720 T5: DEC MOTOR_COUNT ; DISKETTE_CTL
5721 JNZ T6 ; RETURN IF COUNT NOT OUT
5722 AND MOTOR_STATUS,0F0H ; TURN OFF MOTOR RUNNING BITS
5723 MOV AL,0CH
5724 MOV DX,03F2H ; FDC CTL PORT
5725 OUT DX,AL ; TURN OFF THE MOTOR
5726
5727 T6: INT ICH ; TIMER RET1
5728 MOV AL,E01 ; TRANSFER CONTROL TO A USER ROUTINE
5729 OUT 020H,AL ; END OF INTERRUPT TO 8259
5730 POP DX
5731 POP AX
5732 POP DS
5733 IRET ; RESET MACHINE STATE
5734
5735 TIMER_INT ENDP
5736

```

FEA5
FEA5
FEA5 FB
FEA6 IE
FEA7 50
FEA8 52
FEA9 EBADEFB
FEAC FF066C00
FEB0 7504
FEB2 FF066E00
FEB6
FEB6 833E6E0018
FEBB 7515
FEBD 813E6C00B000
FEC3 750D
FEC5 2BC0
FEC7 A36E00
FECA A36C00
FECF C606700001
FED2
FED2 FE0E4000
FED6 750B
FED8 80263F00F0
FEDD 800C
FEDF BAF203
FEE2 EE
FEE3
FEE3 CD1C
FEE5 B020
FEE7 E620
FEE9 5A
FEEA 58
FEEB 1F
FEED CF

```

5737 :-----
5738 : THESE ARE THE VECTORS WHICH ARE MOVED INTO :
5739 : THE 8086 INTERRUPT AREA DURING POWER ON. :
5740 : ONLY THE OFFSETS ARE DISPLAYED HERE, CODE :
5741 : SEGMENT WILL BE ADDED FOR ALL OF THEM, EXCEPT :
5742 : WHERE NOTED. :
5743 :-----
5744 : ASSUME CS:CODE
5745 : ORG 0FF53H
FEF3 VECTOR_TABLE LABEL WORD ; VECTOR TABLE FOR MOVE TO INTERRUPTS
FEF3 :
FEF3 A5FE :
FEF5 81E9 :
FEF7 23FF :
FEF9 23FF :
FEFB 23FF :
FEFD 23FF :
FEFF 57EF :
FF01 23FF :
FF03 65F0 :
FF05 4DF8 :
FF07 41F8 :
FF09 59EC :
FF0B 39E7 :
FF0D 59F8 :
FF0F 2EE8 :
FF11 D2EF :
FF13 0000 :
FF15 F2E6 :
FF17 6EFE :
FF19 4BFF :
FF1B 4BFF :
FF1D A4F0 :
FF1F C7EF :
FF21 0000 :
5746 :
5747 :
5748 :
5749 :
5750 :
5751 :
5752 :
5753 :
5754 :
5755 :
5756 :
5757 :
5758 :
5759 :
5760 :
5761 :
5762 :
5763 :
5764 :
5765 :
5766 :
5767 :
5768 :
5769 :
5770 :
5771 :
5772 :
5773 :
5774 :
5775 :
5776 :
5777 :
5778 :
5779 :
5780 :
5781 :
5782 :
5783 :
5784 :
5785 :
5786 :
5787 :
5788 :
5789 :
5790 :
5791 :
5792 :
5793 :
5794 :
5795 :
5796 :
5797 :
5798 :
5799 :
5800 :
5801 :
5802 :
5803 :
5804 :
5805 :
5806 :
5807 :
5808 :
5809 :
5810 :
5811 :
5812 :
5813 :
5814 :
5815 :
5816 :
5817 :
5818 :
5819 :
5820 :
:-----
: 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 INTERRUPTS THAT WAS :
: EXECUTED ACCIDENTLY. :
:-----
FF23 DII PROC NEAR
FF23 IE ASSUME DS:DATA
FF24 52 PUSH DS
FF25 50 PUSH AX ; SAVE REG AX CONTENTS
FF26 E830FB CALL DD5
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 ; 007 (NO HARDWARE ISR ACTIVE)
FF34 7504 JNZ HW_INT
FF36 B4FF MOV AH,0FFH
FF38 E80A JMP SHORT SET_INTR_FLAG ; SET FLAG TO FF IF NON-HWWARE
FF3A :
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,E01
FF42 E620 OUT INTA00,AL
FF44 :
FF44 88266B00 SET_INTR_FLAG: MOV INTR_FLAG,AH ; SET FLAG
FF48 58 POP AX ; RESTORE REG AX CONTENTS
FF49 5A POP DX
FF4A 1F POP DS
FF4B :
FF4B CF DUMMY_RETURN: ; NEED IRET FOR VECTOR TABLE
5812 IRET
5813 DII ENDP
5814 :
5815 :-----
5816 : DUMMY RETURN FOR ADDRESS COMPATIBILITY :
5817 :-----
5818 : ORG 0FF53H
5819 : IRET
5820 :

```

```

5821 :-- INT 5
5822 : THIS LOGIC WILL BE INVOKED BY INTERRUPT 05H TO PRINT THE
5823 : SCREEN. THE CURSOR POSITION AT THE TIME THIS ROUTINE IS INVOKED
5824 : WILL BE SAVED AND RESTORED UPON COMPLETION. THE ROUTINE IS
5825 : INTENDED TO RUN WITH INTERRUPTS ENABLED. IF A SUBSEQUENT
5826 : 'PRINT SCREEN' KEY IS DEPRESSED DURING THE TIME THIS ROUTINE
5827 : IS PRINTING IT WILL BE IGNORED.
5828 : ADDRESS 50:0 CONTAINS THE STATUS OF THE PRINT SCREEN:
5829 :
5830 : 50:0 =0 EITHER PRINT SCREEN HAS NOT BEEN CALLED
5831 : OR UPON RETURN FROM A CALL THIS INDICATES
5832 : A SUCCESSFUL OPERATION.
5833 : =1 PRINT SCREEN IS IN PROGRESS
5834 : =255 ERROR ENCOUNTERED DURING PRINTING
5835 :-----
5836 : ASSUME CS:CODE,DS:XXDATA
5837 : ORG OFF54H
5838 : PRINT_SCREEN PROC FAR
5839 : STI DS ; MUST RUN WITH INTERRUPTS ENABLED
5840 : PUSH AX ; MUST USE 50:0 FOR DATA AREA STORAGE
5841 : PUSH AX
5842 : PUSH BX
5843 : PUSH CX
5844 : PUSH DX
5845 : MOV AX,XXDATA
5846 : MOV DS,AX
5847 : CMP STATUS_BYTE,1 ; SEE IF PRINT ALREADY IN PROGRESS
5848 : JZ EXIT ; JUMP IF PRINT ALREADY IN PROGRESS
5849 : MOV STATUS_BYTE,1 ; INDICATE PRINT NOW IN PROGRESS
5850 : MOV AH,15 ; WILL REQUEST THE CURRENT SCREEN MODE
5851 : INT 10H ; [AH]=NUMBER COLUMNS/LINE
5852 : ; [BH]=VISUAL PAGE
5853 :
5854 :-----
5855 : AT THIS POINT WE KNOW THE COLUMNS/LINE ARE IN
5856 : [AX] AND THE PAGE IF APPLICABLE IS IN[BH]. THE STACK
5857 : HAS DS,AX,BX,CX,DX PUSHED. [A] HAS VIDEO MODE
5858 :-----
5859 : MOV CL,AH ; WILL MAKE USE OF [CX] REGISTER TO
5860 : MOV CH,25 ; CONTROL ROW & COLUMNS
5861 : CALL CRLF ; CARRIAGE RETURN LINE FEED ROUTINE
5862 : PUSH CX ; SAVE SCREEN BOUNDS
5863 : MOV AH,3 ; WILL NOW READ THE CURSOR.
5864 : INT 10H ; AND PRESERVE THE POSITION
5865 : POP CX ; RECALL SCREEN BOUNDS
5866 : PUSH DX ; RECALL [BH]=VISUAL PAGE
5867 : XOR DX,DX ; WILL SET CURSOR POSITION TO [0,0]
5868 :
5869 :-----
5870 : THE LOOP FROM PRI10 TO THE INSTRUCTION PRI0 TO PRI20
5871 : IS THE LOOP TO READ EACH CURSOR POSITION FROM THE
5872 : SCREEN AND PRINT.
5873 :-----
5874 : PRI10: MOV AH,2 ; TO INDICATE CURSOR SET REQUEST
5875 : INT 10H ; NEW CURSOR POSITION ESTABLISHED
5876 : MOV AH,8 ; TO INDICATE READ CHARACTER
5877 : INT 10H ; CHARACTER NOW IN [AL]
5878 : OR AL,AL ; SEE IF VALID CHAR
5879 : JNZ PRI15 ; JUMP IF VALID CHAR
5880 : MOV AL,' ' ; MAKE A BLANK
5881 :
5882 : PRI15: PUSH DX ; SAVE CURSOR POSITION
5883 : XOR DX,DX ; INDICATE PRINTER 1
5884 : XOR AH,AH ; TO INDICATE PRINT CHAR IN [AL]
5885 : INT 17H ; PRINT THE CHARACTER
5886 : POP DX ; RECALL CURSOR POSITION
5887 : TEST AH,25H ; TEST FOR PRINTER ERROR
5888 : JNZ ERR10 ; JUMP IF ERROR DETECTED
5889 : INCFD ; ADVANCE TO NEXT COLUMN
5890 : CMOVL ; IF AT END OF LINE
5891 : JNZ PRI10 ; IF NOT PROCEED
5892 : XOR DX,DL ; BACK TO COLUMN 0
5893 : MOV AH,DL ; [AH]=0
5894 : PUSH DX ; SAVE NEW CURSOR POSITION
5895 : CALL CRLF ; LINE FEED CARRIAGE RETURN
5896 : POP DX ; RECALL CURSOR POSITION
5897 : INC DH ; ADVANCE TO NEXT LINE
5898 : CMP CH,DH ; FINISHED?
5899 : JNZ PRI10 ; IF NOT CONTINUE
5900 :
5901 : PRI20: POP DX ; RECALL CURSOR POSITION
5902 : MOV AH,2 ; TO INDICATE CURSOR SET REQUEST
5903 : INT 10H ; CURSOR POSITION RESTORED
5904 : MOV STATUS_BYTE,0 ; INDICATE FINISHED
5905 : JMP SHORT EXIT ; EXIT THE ROUTINE
5906 :
5907 : ERR10: POP DX ; GET CURSOR POSITION
5908 : MOV AH,2 ; TO REQUEST CURSOR SET
5909 : INT 10H ; CURSOR POSITION RESTORED
5910 :
5911 : ERR20: MOV STATUS_BYTE,OFFH ; INDICATE ERROR
5912 :
5913 : EXIT: POP DX ; RESTORE ALL THE REGISTERS USED
5914 : POP CX
5915 : POP BX
5916 : POP AX
5917 : POP DS
5918 : IRET
5919 : PRINT_SCREEN ENDP
5920 :
5921 :----- CARRIAGE RETURN, LINE FEED SUBROUTINE
5922 :
5923 : CRLF PROC NEAR
5924 : XOR DX,DX ; PRINTER 0
5925 : XOR AH,AH ; WILL NOW SEND INITIAL LF,CR
5926 : ; TO PRINTER
5927 : MOV AL,12Q
5928 : INT 17H ; SEND THE LINE FEED
5929 : XOR AH,AH ; NOW FOR THE CR
5930 : MOV AL,15Q
5931 : INT 17H ; SEND THE CARRIAGE RETURN
5932 : RET
5933 : CRLF ENDP

```

LOC	OBJECT	LINE	SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82
		5934	
		5935	-----
		5936	PRINT A SEGMENT VALUE TO LOOK LIKE A 20 BIT ADDRESS ;
		5937	DX MUST CONTAIN SEGMENT VALUE TO BE PRINTED ;
		5938	-----
FFDA		5939	PRT_SEG PROC NEAR
FFDA 8AC6		5940	MOV AL,0H ;GET MSB
FFDC E8ACF9		5941	CALL XPC_BYTE
FFDF 8AC2		5942	MOV AL,0L ;LSB
FFE1 E8A7F9		5943	CALL XPC_BYTE
FFE4 B030		5944	MOV AL,'0' ; PRINT A '0 '
FFE6 E8B3F9		5945	CALL PRT_HEX
FFE9 B020		5946	MOV AL,' '
FFEB E8AEF9		5947	CALL PRT_HEX ;SPACE
FFEE C3		5948	RET
		5949	PRT_SEG ENDP
		5950	
----		5951	CODE ENDS
		5952	
		5953	-----
		5954	POWER ON RESET VECTOR ;
		5955	-----
----		5956	VECTOR SEGMENT AT 0FFFFH
		5957	
		5958	----- POWER ON RESET
		5959	
0000 EA5BE00F0		5960	JMP RESET
		5961	
0005 31312F30382F38		5962	DB '11/08/82' ; RELEASE MARKER
32			
----		5963	VECTOR ENDS
		5964	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

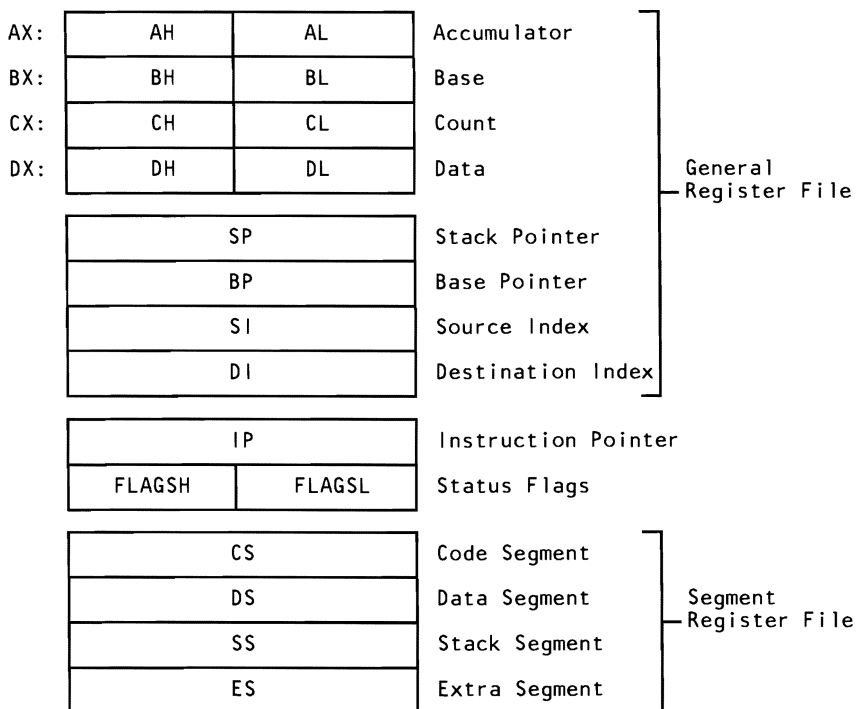
Notes:



8088 Register Model

Notes:

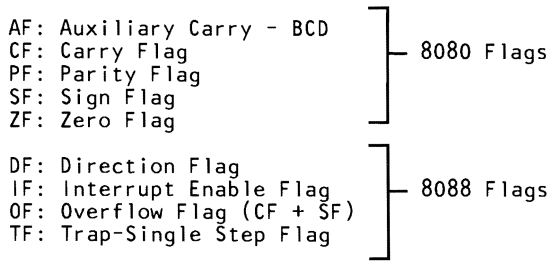
if d = 1 then "to"; if d = 0 then "from"
 if w = 1 then word instruction; if w = 0 then byte instruction
 if s:w = 01 then 16 bits of immediate data from the operand
 if s:w = 11 then an immediate data byte is signed extended to form the 16-bit operand
 if v = 0 the "count" = 1; if v = 1 the "count" 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



Instructions which reference the flag register file as a 16-bit object, use the symbol FLAGS to represent the file:



X = Don't Care



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	

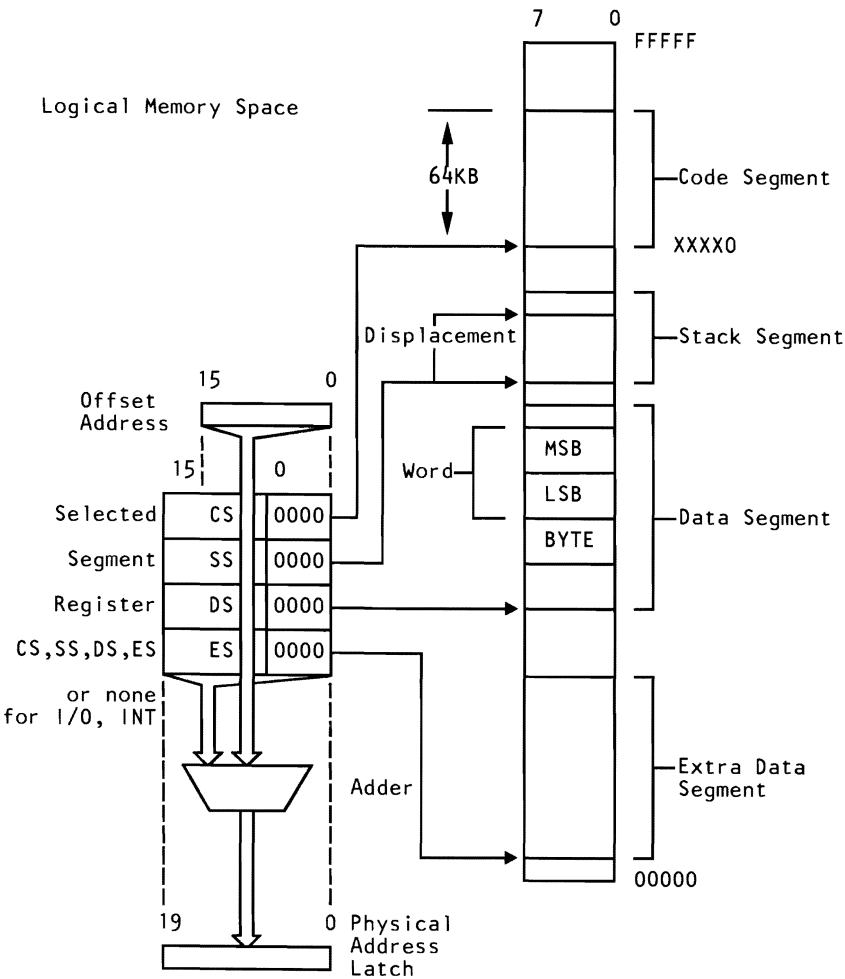
Second Instruction Byte Summary

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



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
----------	-------------

Immediate 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

LDS = 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

L0	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

L0	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,(l+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	r/m	Operand Address
00 - 32-bit Real	000	(BX) + (SI) + DISP
01 - 32-bit Integer	001	(BX) + (DI) + DISP
10 - 64-bit Real	010	(BP) + (SI) + DISP
11 - 64-bit Integer	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
------------	----------

FCBS = Change Sign of ST(0)

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

Transcendental

FPTAN = Partial Tangent of ST(0)

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

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

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

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

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

FYL2X = ST(1) x Log₂ [ST(0)]

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

FYL2XP1 = ST(1) x Log₂ [ST(0) + 1]

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

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**: $-0 \leq \text{ST}(0) \leq +\infty$

For **FSCALE**: $-2^{15} \leq \text{ST}(1) < +2^{15}$ and ST(1) interger

For **F2XM1**: $0 \leq \text{ST}(0) \leq 2^{-1}$

For **FYL2X**: $0 < \text{St}(0) < \infty - \infty < \text{ST}(1) < +\infty$

For **FYL2XP1**: $0 < |\text{ST}(0)| < (2-\sqrt{2})/2 - \infty < \text{ST}(1) < \infty$

For **FPTAN**: $0 \leq \text{ST}(0) < \pi/4$

For **FPATAN**: $0 \leq \text{ST}(0) < \text{ST}(1) < +\infty$

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, Ctrl Esc		Blue	Light Cyan	High Intensity
1C	28	└─	Ctrl \		Blue	Light Red	High Intensity
1D	29	↔	Ctrl]		Blue	Light Magenta	High Intensity
1E	30	▲	Ctrl 6		Blue	Yellow	High Intensity
1F	31	▼	Ctrl —		Blue	White	High Intensity
20	32	Blank Space	Space Bar, Shift, Space, Ctrl Space, Alt Space		Green	Black	Normal
21	33	!	!	Shift	Green	Blue	Underline
22	34	”	”	Shift	Green	Green	Normal
23	35	#	#	Shift	Green	Cyan	Normal
24	36	\$	\$	Shift	Green	Red	Normal
25	37	%	%	Shift	Green	Magenta	Normal
26	38	&	&	Shift	Green	Brown	Normal
27	39	'	'		Green	Light Grey	Normal
28	40	(	(	Shift	Green	Dark Grey	High Intensity
29	41	)	)	Shift	Green	Light Blue	High Intensity Underline
2A	42	*	*	Note 1	Green	Light Green	High Intensity
2B	43	+	+	Shift	Green	Light Cyan	High Intensity
2C	44	,	,		Green	Light Red	High Intensity
2D	45	-	-		Green	Light Magenta	High Intensity
2E	46	.	.	Note 2	Green	Yellow	High Intensity

Value		As Characters			As Text Attributes		
					Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground	
2F	47	/	/		Green	White	High Intensity
30	48	0	0	Note 3	Cyan	Black	Normal
31	49	1	1	Note 3	Cyan	Blue	Underline
32	50	2	2	Note 3	Cyan	Green	Normal
33	51	3	3	Note 3	Cyan	Cyan	Normal
34	52	4	4	Note 3	Cyan	Red	Normal
35	53	5	5	Note 3	Cyan	Magenta	Normal
36	54	6	6	Note 3	Cyan	Brown	Normal
37	55	7	7	Note 3	Cyan	Light Grey	Normal
38	56	8	8	Note 3	Cyan	Dark Grey	High Intensity
39	57	9	9	Note 3	Cyan	Light Blue	High Intensity Underline
3A	58	:	:	Shift	Cyan	Light Green	High Intensity
3B	59	;	;		Cyan	Light Cyan	High Intensity
3C	60	<	<	Shift	Cyan	Light Red	High Intensity
3D	61	=	=		Cyan	Light Magenta	High Intensity
3E	62	>	>	Shift	Cyan	Yellow	High Intensity
3F	63	?	?	Shift	Cyan	White	High Intensity
40	64	@	@	Shift	Red	Black	Normal
41	65	A	A	Note 4	Red	Blue	Underline
42	66	B	B	Note 4	Red	Green	Normal
43	67	C	C	Note 4	Red	Cyan	Normal
44	68	D	D	Note 4	Red	Red	Normal
45	69	E	E	Note 4	Red	Magenta	Normal
46	70	F	F	Note 4	Red	Brown	Normal
47	71	G	G	Note 4	Red	Light Grey	Normal
48	72	H	H	Note 4	Red	Dark Grey	High Intensity
49	73	I	I	Note 4	Red	Light Blue	High Intensity Underline
4A	74	J	J	Note 4	Red	Light Green	High Intensity

Value		As Characters			As Text Attributes		
					Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground	
4B	75	K	K	Note 4	Red	Light Cyan	High Intensity
4C	76	L	L	Note 4	Red	Light Red	High Intensity
4D	77	M	M	Note 4	Red	Light Magenta	High Intensity
4E	78	N	N	Note 4	Red	Yellow	High Intensity
4F	79	O	O	Note 4	Red	White	High Intensity
50	80	P	P	Note 4	Magenta	Black	Normal
51	81	Q	Q	Note 4	Magenta	Blue	Underline
52	82	R	R	Note 4	Magenta	Green	Normal
53	83	S	S	Note 4	Magenta	Cyan	Normal
54	84	T	T	Note 4	Magenta	Red	Normal
55	85	U	U	Note 4	Magenta	Magenta	Normal
56	86	V	V	Note 4	Magenta	Brown	Normal
57	87	W	W	Note 4	Magenta	Light Grey	Normal
58	88	X	X	Note 4	Magenta	Dark Grey	High Intensity
59	89	Y	Y	Note 4	Magenta	Light Blue	High Intensity Underline
5A	90	Z	Z	Note 4	Magenta	Light Green	High Intensity
5B	91	[	[		Magenta	Light Cyan	High Intensity
5C	92	\	\		Magenta	Light Red	High Intensity
5D	93	]	]		Magenta	Light Magenta	High Intensity
5E	94	^	^	Shift	Magenta	Yellow	High Intensity
5F	95	_	_	Shift	Magenta	White	High Intensity
60	96	'	'		Brown	Black	Normal
61	97	a	a	Note 5	Brown	Blue	Underline
62	98	b	b	Note 5	Brown	Green	Normal
63	99	c	c	Note 5	Brown	Cyan	Normal
64	100	d	d	Note 5	Brown	Red	Normal
65	101	e	e	Note 5	Brown	Magenta	Normal
66	102	f	f	Note 5	Brown	Brown	Normal

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	Underliné
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	<i>f</i>	Alt 159	Note 6	Blue	White	High Intensity
A0	160	á	Alt 160	Note 6	Green	Black	Normal
A1	161	í	Alt 161	Note 6	Green	Blue	Underline
A2	162	ó	Alt 162	Note 6	Green	Green	Normal
A3	163	ú	Alt 163	Note 6	Green	Cyan	Normal
A4	164	ñ	Alt 164	Note 6	Green	Red	Normal
A5	165	Ñ	Alt 165	Note 6	Green	Magenta	Normal
A6	166	<u>a</u>	Alt 166	Note 6	Green	Brown	Normal
A7	167	<u>o</u>	Alt 167	Note 6	Green	Light Grey	Normal
A8	168	¿	Alt 168	Note 6	Green	Dark Grey	High Intensity
A9	169	┐	Alt 169	Note 6	Green	Light Blue	High Intensity Underline
AA	170	└	Alt 170	Note 6	Green	Light Green	High Intensity
AB	171	½	Alt 171	Note 6	Green	Light Cyan	High Intensity
AC	172	¼	Alt 172	Note 6	Green	Light Red	High Intensity
AD	173	i	Alt 173	Note 6	Green	Light Magenta	High Intensity
AE	174	<<	Alt 174	Note 6	Green	Yellow	High Intensity
AF	175	>>	Alt 175	Note 6	Green	White	High Intensity
B0	176	⋮	Alt 176	Note 6	Cyan	Black	Normal
B1	177	⋮	Alt 177	Note 6	Cyan	Blue	Underline
B2	178	⋮	Alt 178	Note 6	Cyan	Green	Normal
B3	179	▬	Alt 179	Note 6	Cyan	Cyan	Normal
B4	180	▬	Alt 180	Note 6	Cyan	Red	Normal
B5	181	▬	Alt 181	Note 6	Cyan	Magenta	Normal
B6	182	▬	Alt 182	Note 6	Cyan	Brown	Normal

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

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

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

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	â	û	ä				μ	÷
7	7	ç	ù	ó				γ	≈
8	8	ê	ÿ	ı				Φ	°
9	9	ë	Ö	┐				Θ	•
10	A	è	Ü	┐				Ω	•
11	B	ï	¢	½				δ	√
12	C	î	£	¼				∞	n
13	D	ì	¥	ı				φ	²
14	E	Ä	℔	«				€	■
15	F	Å	f	»				∩	BLANK 'FF'

Notes:



Glossary

This glossary includes terms and definitions from the *IBM Vocabulary for Data Processing, Telecommunications, and Office Systems*, GC20-1699.

μ. Prefix micro; 0.000 001.

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, hexadecimal can be noted as X' '.

hexadecimal. (1) Pertaining to a selection, choice, or condition that has 16 possible different values or states. These values or states are usually symbolized by the ten digits 0 through 9 and the six letters A through F. (2) Pertaining to a fixed radix numeration system having a radix of 16.

high impedance state. A state in which the output of a device is effectively isolated from the circuit.

highlighting. In computer graphics, emphasizing a given display group by changing its attributes relative to other display groups in the same display field.

high-order position. The leftmost position in a string of characters. See also most-significant digit.

hither plane. In computer graphics, a plane that is perpendicular to the line joining the viewing reference point and the view point and that lies between these two points. Any part of an object between the hither plane and the view point is not seen. See also yon plane.

housekeeping. Operations or routines that do not contribute directly to the solution of the problem but do contribute directly to the operation of the computer.

Hz. Hertz

image. A fully processed unit of operational data that is ready to be transmitted to a remote unit; when loaded into control storage in the remote unit, the image determines the operations of the unit.

immediate instruction. An instruction that contains within itself an operand for the operation specified, rather than an address of the operand.

index register. A register whose contents may be used to modify an operand address during the execution of computer instructions.

indicator. (1) A device that may be set into a prescribed state, usually according to the result of a previous process or on the occurrence of a specified condition in the equipment, and that usually gives a visual or other indication of the existence of the prescribed state, and that may in some cases be used to determine the selection among alternative processes; for example, an overflow indicator. (2) An item of data that may be interrogated to determine whether a particular condition has been satisfied in the execution of a computer program; for example, a switch indicator, an overflow indicator.

inhibited. (1) Pertaining to a state of a processing unit in which certain types of interruptions are not allowed to occur. (2) Pertaining to the state in which a transmission control unit or an audio response unit cannot accept incoming calls on a line.

initialize. To set counters, switches, addresses, or contents of storage to 0 or other starting values at the beginning of, or at prescribed points in, the operation of a computer routine.

input/output (I/O). (1) Pertaining to a device or to a channel that may be involved in an input process, and, at a different time, in an output process. In the English language, "input/output" may be used in place of such terms as "input/output data," "input/output signal," and "input/output terminals," when such usage is clear in a given context. (2) Pertaining to a device

whose parts can be performing an input process and an output process at the same time. (3) Pertaining to either input or output, or both.

instruction. In a programming language, a meaningful expression that specifies one operation and identifies its operands, if any.

instruction set. The set of instructions of a computer, of a programming language, or of the programming languages in a programming system.

intensity. In computer graphics, the amount of light emitted at a display point

interface. A device that alters or converts actual electrical signals between distinct devices, programs, or systems.

interleave. To arrange parts of one sequence of things or events so that they alternate with parts of one or more other sequences of the same nature and so that each sequence retains its identity.

interrupt. (1) A suspension of a process, such as the execution of a computer program, caused by an event external to that process, and performed in such a way that the process can be resumed. (2) In a data transmission, to take an action at a receiving station that causes the transmitting station to terminate a transmission. (3) Synonymous with interruption.

I/O. Input/output.

I/O area. Synonym for buffer.

irrecoverable error. An error that makes recovery impossible without the use of recovery techniques external to the computer program or run.

joystick. In computer graphics, a lever that can pivot in all directions and that is used as a locator device.

k. Prefix kilo; 1000.

K. When referring to storage capacity, 1024. ($1024 = 2$ to the 10th power.)

KB. 1024 bytes.

key lock. A device that deactivates the keyboard and locks the cover on for security.

kg. Kilogram; 1000 grams.

kHz. Kilohertz; 1000 hertz.

kilo (k). Prefix 1000

kilogram (kg). 1000 grams.

kilohertz (kHz). 1000 hertz

latch. (1) A simple logic-circuit storage element. (2) A feedback loop in sequential digital circuits used to maintain a state.

least-significant digit. The rightmost digit. See also low-order position.

LED. Light-emitting diode.

light-emitting diode (LED). A semiconductor device that gives off visible or infrared light when activated.

load. In programming, to enter data into storage or working registers.

look-up table (LUT). (1) A technique for mapping one set of values into a larger set of values. (2) In computer graphics, a table that assigns a color value (red, green, blue intensities) to a color index.

low power Schottky TTL. A version (LS series) of TTL giving a good compromise between low power and high speed. See also transistor-transistor logic and Schottky TTL.

low-order position. The rightmost position in a string of characters. See also least-significant digit.

luminance. The luminous intensity per unit projected area of a given surface viewed from a given direction.

LUT. Look-up table.

m. (1) Prefix milli; 0.001. (2) Meter.

M. (1) Prefix mega; 1,000,000. (2) When referring to computer storage capacity, 1,048,576. (1,048,576 = 2 to the 20th power.)

mA. Milliampere; 0.001 ampere.

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

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

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

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

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

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

masked. Synonym for disabled.

matrix. (1) A rectangular array of elements, arranged in rows and columns, that may be manipulated according to the rules of matrix algebra. (2) In computers, a logic network in the form of an array of input leads and output leads with logic elements connected at some of their intersections.

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

MB. 1,048,576 bytes.

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

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

memory. Term for main storage.

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

MFM. Modified frequency modulation.

MHz. Megahertz; 1,000,000 hertz.

micro (μ). Prefix 0.000,001.

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

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

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

microsecond (μ s). 0.000,001 second.

milli (m). Prefix 0.001.

milliampere (mA). 0.001 ampere.

millisecond (ms). 0.001 second.

mnemonic. A symbol chosen to assist the human memory; for example, an abbreviation such as "mpy" for "multiply."

mode. (1) A method of operation; for example, the binary mode, the interpretive mode, the alphanumeric mode. (2) The most frequent value in the statistical sense.

modeling transformation. Operations on the coordinates of an object (usually matrix multiplications) that cause the object to be rotated about any axis, translated (moved without rotating), and/or scaled (changed in size along any or all dimensions). See also viewing transformation.

modem (modulator-demodulator). A device that converts serial (bit by bit) digital signals from a business machine (or data communication equipment) to analog signals that are suitable for transmission in a telephone network. The inverse function is also performed by the modem on reception of analog signals.

modified frequency modulation (MFM). The process of varying the amplitude and frequency of the 'write' signal. MFM pertains to the number of bytes of storage that can be stored on the recording media. The number of bytes is twice the number contained in the same unit area of recording media at single density.

modulation. The process by which some characteristic of one wave (usually high frequency) is varied in accordance with another wave or signal (usually low frequency). This technique is used in modems to make business-machine signals compatible with communication facilities.

modulation rate. The reciprocal of the measure of the shortest nominal time interval between successive significant instants of the modulated signal. If this measure is expressed in seconds, the modulation rate is expressed in baud.

module. (1) A program unit that is discrete and identifiable with respect to compiling, combining with other units, and loading. (2) A packaged functional hardware unit designed for use with other components.

modulo check. A calculation performed on values entered into a system. This calculation is designed to detect errors.

modulo-N check. A check in which an operand is divided by a number N (the modulus) to generate a remainder (check digit)

that is retained with the operand. For example, in a modulo-7 check, the remainder will be 0, 1, 2, 3, 4, 5, or 6. The operand is later checked by again dividing it by the modulus; if the remainder is not equal to the check digit, an error is indicated.

modulus. In a modulo-N check, the number by which the operand is divided.

monitor. Synonym for cathode ray tube display (CRT display).

most-significant digit. The leftmost (non-zero) digit. See also high-order position.

ms. Millisecond; 0.001 second.

multiplexer. A device capable of interleaving the events of two or more activities, or capable of distributing the events of an interleaved sequence to the respective activities.

multiprogramming. (1) Pertaining to the concurrent execution of two or more computer programs by a computer. (2) A mode of operation that provides for the interleaved execution of two or more computer programs by a single processor.

n. Prefix nano; 0.000,000,001.

NAND. A logic operator having the property that if P is a statement, Q is a statement, R is a statement,..., then the NAND of P, Q, R,... is true if at least one statement is false, false if all statements are true.

NAND gate. A gate in which the output is 0 only if all inputs are 1.

nano (n). Prefix 0.000,000,001.

nanosecond (ns). 0.000,000,001 second.

negative true. Synonym for active low.

negative-going edge. The edge of a pulse or signal changing in a negative direction. Synonymous with falling edge.

non-return-to-zero change-on-ones recording (NRZI). A transmission encoding method in which the data terminal equipment changes the signal to the opposite state to send a binary 1 and leaves it in the same state to send a binary 0.

non-return-to-zero (inverted) recording (NRZI). Deprecated term for non-return-to-zero change-on-ones recording.

NOR. A logic operator having the property that if P is a statement, Q is a statement, R is a statement,..., then the NOR of P, Q, R,... is true if all statements are false, false if at least one statement is true.

NOR gate. A gate in which the output is 0 only if at least one input is 1.

NOT. A logical operator having the property that if P is a statement, then the NOT of P is true if P is false, false if P is true.

NRZI. Non-return-to-zero change-on-ones recording.

ns. Nanosecond; 0.000,000,001 second.

NUL. The null character.

null character (NUL). A control character that is used to accomplish media-fill or time-fill, and that may be inserted into or removed from, a sequence of characters without affecting the meaning of the sequence; however, the control of the equipment or the format may be affected by this character.

odd-even check. Synonym for parity check.

offline. Pertaining to the operation of a functional unit without the continual control of a computer.

one-shot. A circuit that delivers one output pulse of desired duration for each input (trigger) pulse.

open circuit. (1) A discontinuous circuit; that is, one that is broken at one or more points and, consequently, cannot conduct current. Contrast with closed circuit. (2) Pertaining to a no-load condition; for example, the open-circuit voltage of a power supply.

open collector. A switching transistor without an internal connection between its collector and the voltage supply. A connection from the collector to the voltage supply is made through an external (pull-up) resistor.

operand. (1) An entity to which an operation is applied. (2) That which is operated upon. An operand is usually identified by an address part of an instruction.

operating system. Software that controls the execution of programs; an operating system may provide services such as resource allocation, scheduling, input/output control, and data management.

OR. A logic operator having the property that if P is a statement, Q is a statement, R is a statement,..., then the OR of P, Q, R,...is true if at least one statement is true, false if all statements are false.

OR gate. A gate in which the output is 1 only if at least one input is 1.

output. Pertaining to a device, process, or channel involved in an output process, or to the data or states involved in an output process.

output process. (1) The process that consists of the delivery of data from a data processing system, or from any part of it. (2) The return of information from a data processing system to an end user, including the translation of data from a machine language to a language that the end user can understand.

overcurrent. A current of higher than specified strength.

overflow indicator. (1) An indicator that signifies when the last line on a page has been printed or passed. (2) An indicator that is set on if the result of an arithmetic operation exceeds the capacity of the accumulator.

overrun. Loss of data because a receiving device is unable to accept data at the rate it is transmitted.

overvoltage. A voltage of higher than specified value.

parallel. (1) Pertaining to the concurrent or simultaneous operation of two or more devices, or to the concurrent performance of two or more activities. (2) Pertaining to the concurrent or simultaneous occurrence of two or more related activities in multiple devices or channels. (3) Pertaining to the simultaneity of two or more processes. (4) Pertaining to the simultaneous processing of the individual parts of a whole, such as the bits of a character and the characters of a word, using separate facilities for the various parts. (5) Contrast with serial.

parameter. (1) A variable that is given a constant value for a specified application and that may denote the application. (2) A name in a procedure that is used to refer to an argument passed to that procedure.

parity bit. A binary digit appended to a group of binary digits to make the sum of all the digits either always odd (odd parity) or always even (even parity).

parity check. (1) A redundancy check that uses a parity bit. (2) Synonymous with odd-even check.

PEL. Picture element.

personal computer. A small home or business computer that has a processor and keyboard and that can be connected to a television or some other monitor. An optional printer is usually available.

phototransistor. A transistor whose switching action is controlled by light shining on it.

picture element (PEL). The smallest displayable unit on a display.

polling. (1) Interrogation of devices for purposes such as to avoid contention, to determine operational status, or to determine readiness to send or receive data. (2) The process whereby stations are invited, one at a time, to transmit.

port. An access point for data entry or exit.

positive true. Synonym for active high.

positive-going edge. The edge of a pulse or signal changing in a positive direction. Synonymous with rising edge.

potentiometer. A variable resistor with three terminals, one at each end and one on a slider (wiper).

power supply. A device that produces the power needed to operate electronic equipment.

printed circuit. A pattern of conductors (corresponding to the wiring of an electronic circuit) formed on a board of insulating material.

printed-circuit board. A usually copper-clad plastic board used to make a printed circuit.

priority. A rank assigned to a task that determines its precedence in receiving system resources.

processing program. A program that performs such functions as compiling, assembling, or translating for a particular programming language.

processing unit. A functional unit that consists of one or more processors and all or part of internal storage.

processor. (1) In a computer, a functional unit that interprets and executes instructions. (2) A functional unit, a part of another unit such as a terminal or a processing unit, that interprets and executes instructions. (3) Deprecated term for processing program. (4) See microprocessor.

program. (1) A series of actions designed to achieve a certain result. (2) A series of instructions telling the computer how to handle a problem or task. (3) To design, write, and test computer programs.

programmable read-only memory (PROM). A read-only memory that can be programmed by the user.

programming language. (1) An artificial language established for expressing computer programs. (2) A set of characters and rules with meanings assigned prior to their use, for writing computer programs.

programming system. One or more programming languages and the necessary software for using these languages with particular automatic data-processing equipment.

PROM. Programmable read-only memory.

propagation delay. (1) The time necessary for a signal to travel from one point on a circuit to another. (2) The time delay between a signal change at an input and the corresponding change at an output.

protocol. (1) A specification for the format and relative timing of information exchanged between communicating parties. (2) The set of rules governing the operation of functional units of a communication system that must be followed if communication is to be achieved.

pulse. A variation in the value of a quantity, short in relation to the time schedule of interest, the final value being the same as the initial value.

radio frequency (RF). An ac frequency that is higher than the highest audio frequency. So called because of the application to radio communication.

radix. (1) In a radix numeration system, the positive integer by which the weight of the digit place is multiplied to obtain the weight of the digit place with the next higher weight; for example, in the decimal numeration system the radix of each digit place is 10. (2) Another term for base.

radix numeration system. A positional representation system in which the ratio of the weight of any one digit place to the weight of the digit place with the next lower weight is a positive integer (the radix). The permissible values of the character in any digit place range from 0 to one less than the radix.

RAM. Random access memory. Read/write memory.

random access memory (RAM). Read/write memory.

RAS. In the IBM Personal Computer, row address strobe.

raster. In computer graphics, a predetermined pattern of lines that provides uniform coverage of a display space.

read. To acquire or interpret data from a storage device, from a data medium, or from another source.

read-only memory (ROM). A storage device whose contents cannot be modified. The memory is retained when power is removed.

read/write memory. A storage device whose contents can be modified. Also called RAM.

recoverable error. An error condition that allows continued execution of a program.

red-green-blue-intensity (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.

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 Library*. 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

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

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
- instruction 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

- 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
- Overflow 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

- 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





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