



*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 4. Keyboards

(

Section 5. System BIOS

Section 6. Instruction Set

Notes:

Section 7. Characters, Keystrokes, and Colors

Glossary

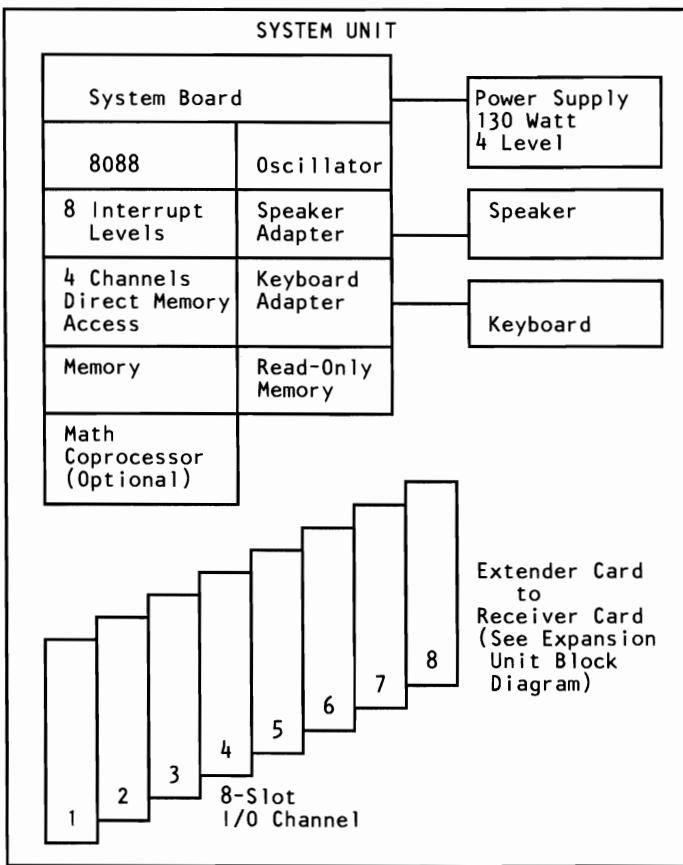
Bibliography

Index

Notes:

System Block Diagram (XT)

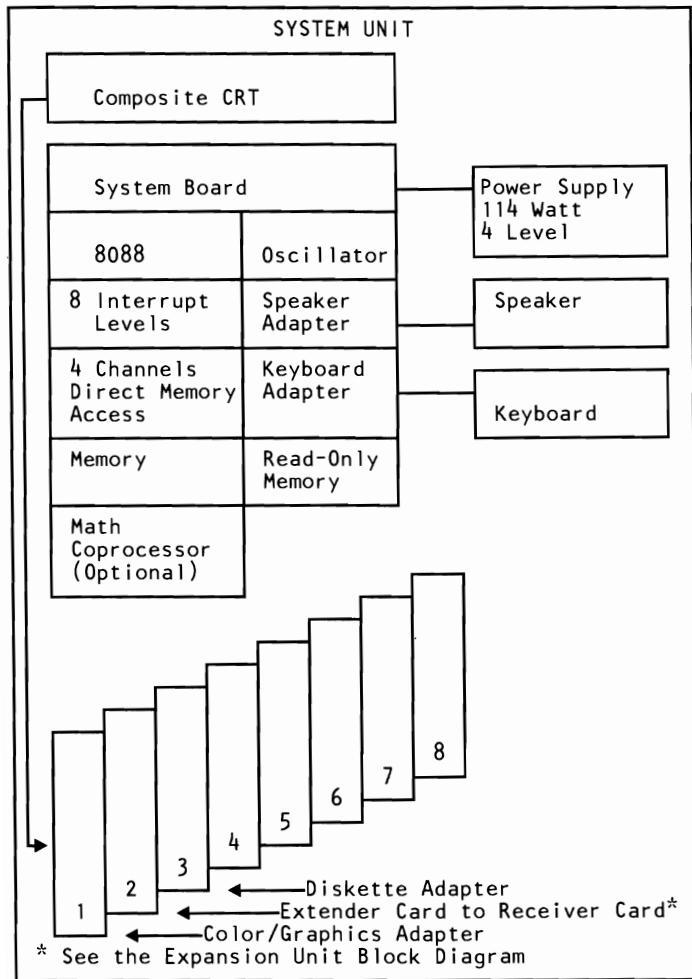
The following is a system block diagram of the IBM Personal Computer XT.



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

System Block Diagram (Portable)

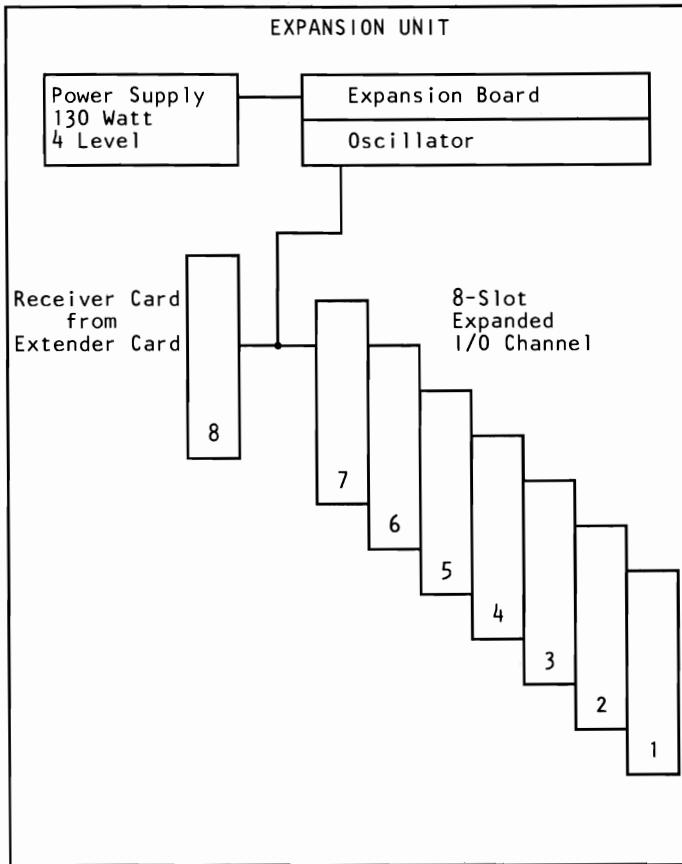
The following is a system block diagram of the IBM Portable Personal Computer.



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

Expansion Unit Block Diagram

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



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

Notes:

SECTION 1. SYSTEM BOARD

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

Notes:

Description

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

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

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

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

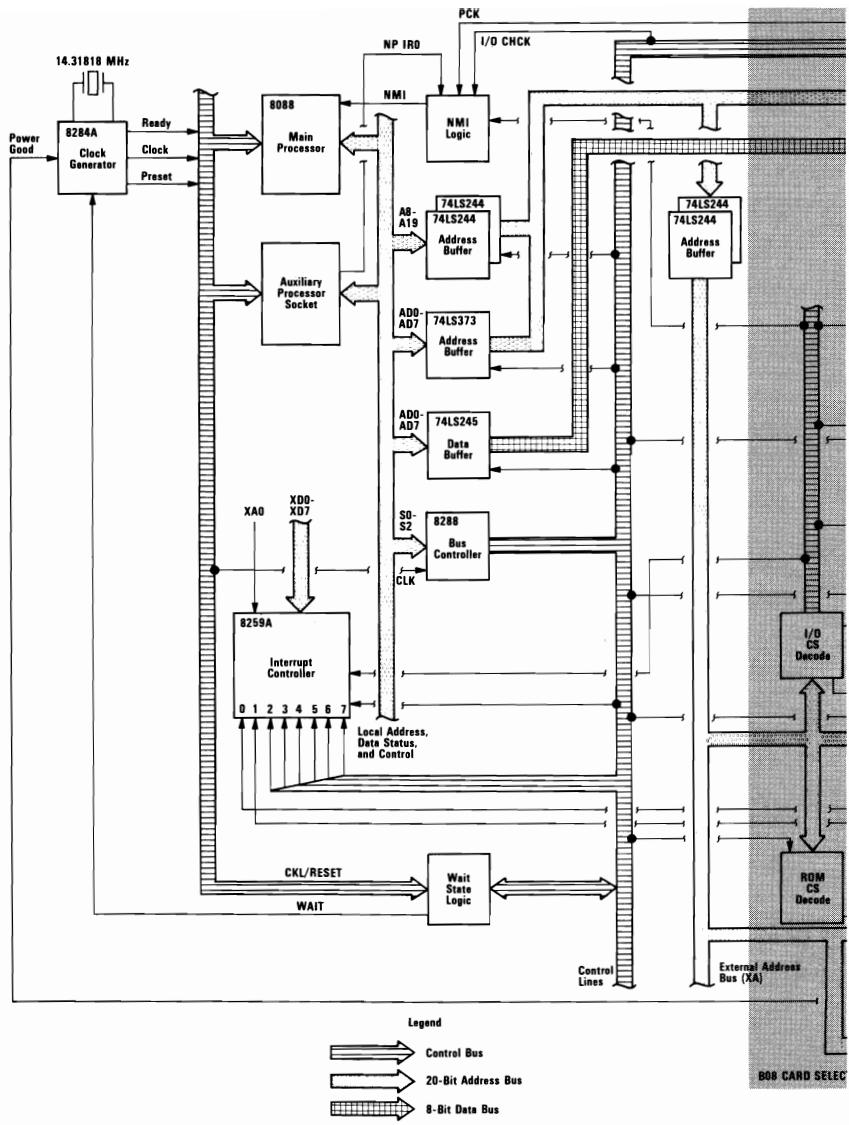
Micropocessor

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).

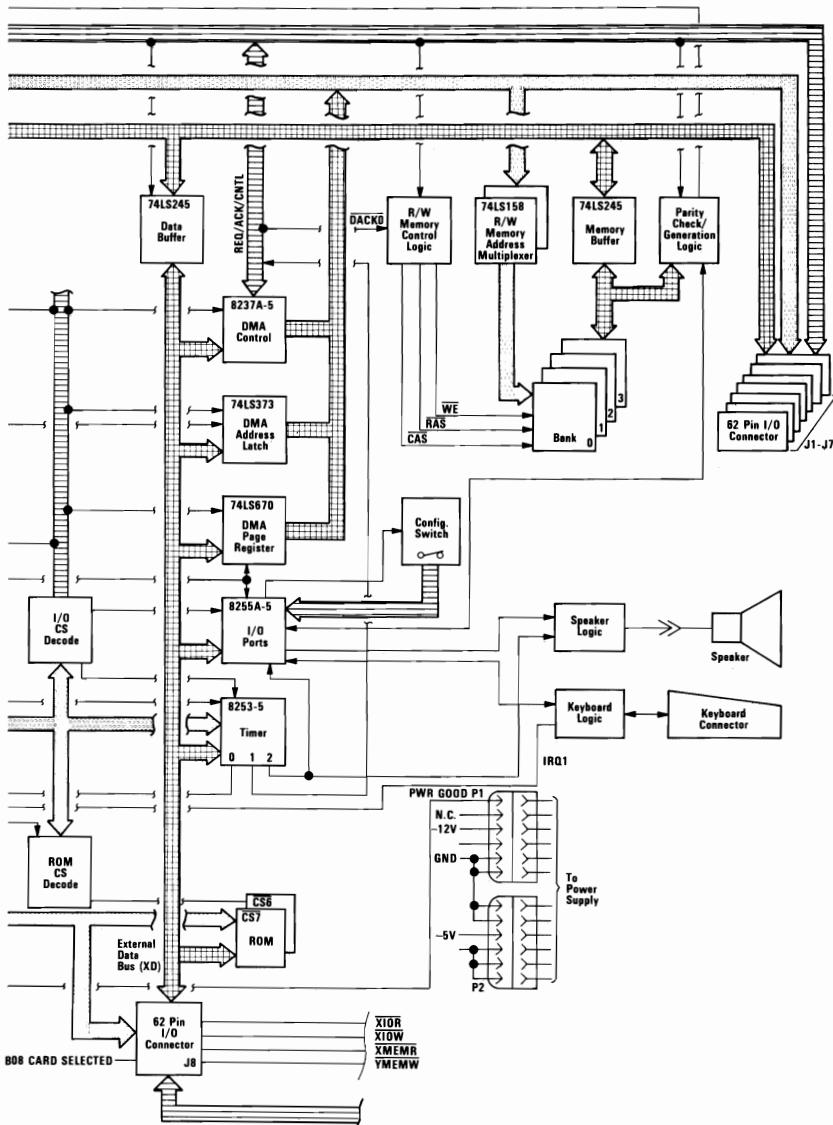
Data Flow Diagrams

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



Legend

- Control Bus
- 20-Bit Address Bus
- 8-Bit Data Bus



System Memory Map

Start Address		Function	
Decimal	Hex	64/256K	256/640K
0K	00000		
16K	04000		
32K	08000		
48K	0C000		
64K	10000		
80K	14000		
86K	18000		
112K	1C000		
128K	20000	128-256K Read/Write Memory on the System Board	256-640K Read/Write Memory on the System Board
144K	24000		
160K	28000		
176K	2C000		
192K	30000		
208K	34000		
224K	38000		
240K	3C000		
256K	40000		
272K	44000		
288K	48000		
304K	4C000		
320K	50000		
336K	54000		
352K	58000		
368K	5C000		
384K	60000	384K R/W Memory Expansion in the I/O Channel	
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	A0000	
656K	A4000	
672K	A8000	128K Reserved
688K	AC000	
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	D4000	
864K	D8000	
880K	DC000	192K Read Only Memory
896K	E0000	Expansion and Control
912K	E4000	
928K	E8000	
944K	EC000	
960K	F0000	
976K	F4000	
992K	F8000	64K Base system
1008K	FC000	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

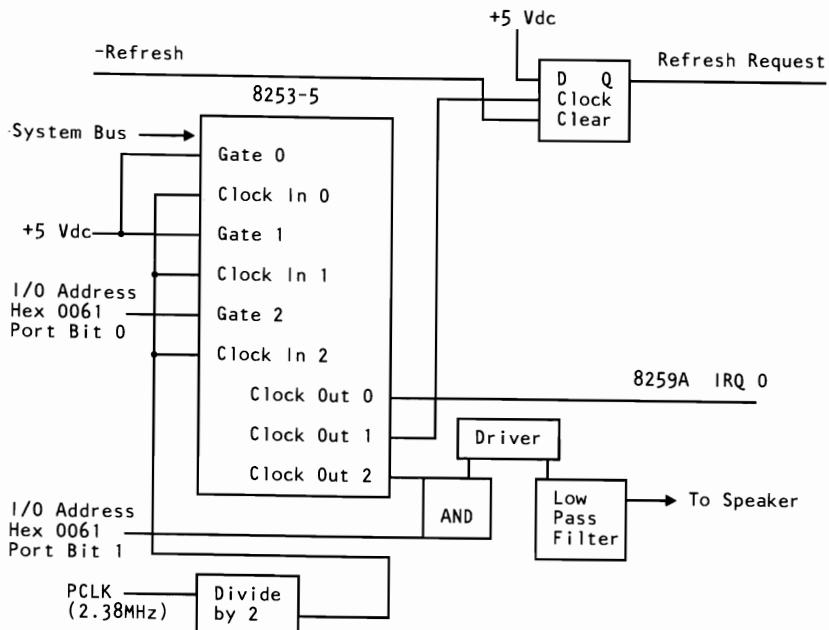
Note: Channel 1 is programmed as a rate generator to produce a 15-microsecond period signal.

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

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.



System-Timer Block Diagram

System Interrupts

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

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

The following diagram contains the System Interrupt Listing.

Number	Usage
NMI	Parity 8087
0	Timer
1	Keyboard
2	EGA Display, PC Net, 3278/79
3	Asynchronous Communications (Alternate) PC Net(Alternate) 3278/79(Alternate)
	SDLC Communications
	BSC Communications
4	Cluster (Primary) 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 1.05 μ s. (See I/O CH RDY on page 1-22.) Refresh cycles occur once every 72 clocks (approximately 15 μ s) and require four clocks or approximately 5.6% of the bus bandwidth.

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

$$64K \times 1$$

$$\% \text{ Bandwidth used} = \frac{4 \text{ cycles} \times 128}{1.93\text{ms}/210\text{ns}} = \frac{512}{9190} = 5.6\%$$

$$256K \times 1$$

$$\% \text{ Bandwidth used} = \frac{4 \text{ cycles} \times 256}{3.86\text{ms}/210\text{ns}} = \frac{1024}{19048} = 5.6\%$$

I/O Channel

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

The I/O channel contains an 8-bit, bidirectional data bus, 20 address lines, 6 levels of interrupt, control lines for memory and I/O read or write, clock and timing lines, 3 channels of DMA control lines, memory refresh-timing control lines, a 'channel check' line, and power and ground for the adapters. Four voltage levels are provided for I/O cards: +5 Vdc \pm 5%, -5 Vdc \pm 10%, +12 Vdc \pm 5%, and -12 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 μ s/byte.
- DMA transfers require five clocks for a cycle time of 1.05 μ s/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.

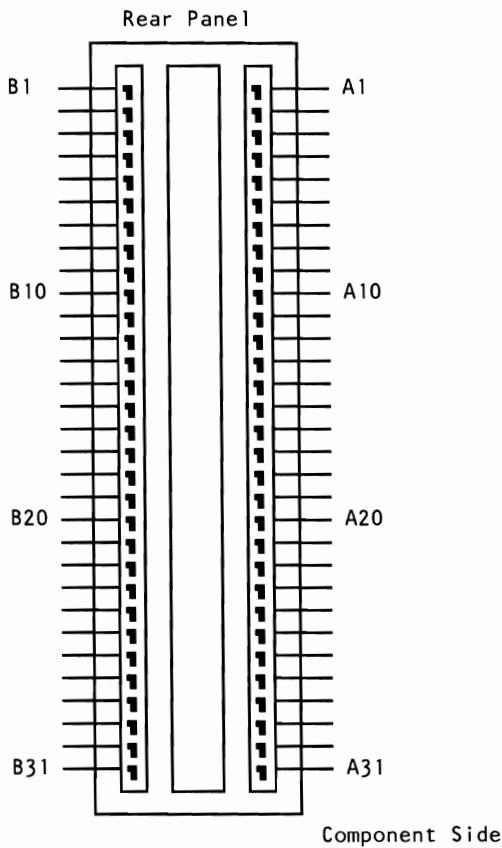
A 'channel check' line exists for reporting error conditions to the microprocessor. Activating this line results in a non-maskable interrupt (NMI) to the 8088 microprocessor. Memory expansion options use this line to report parity errors.

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

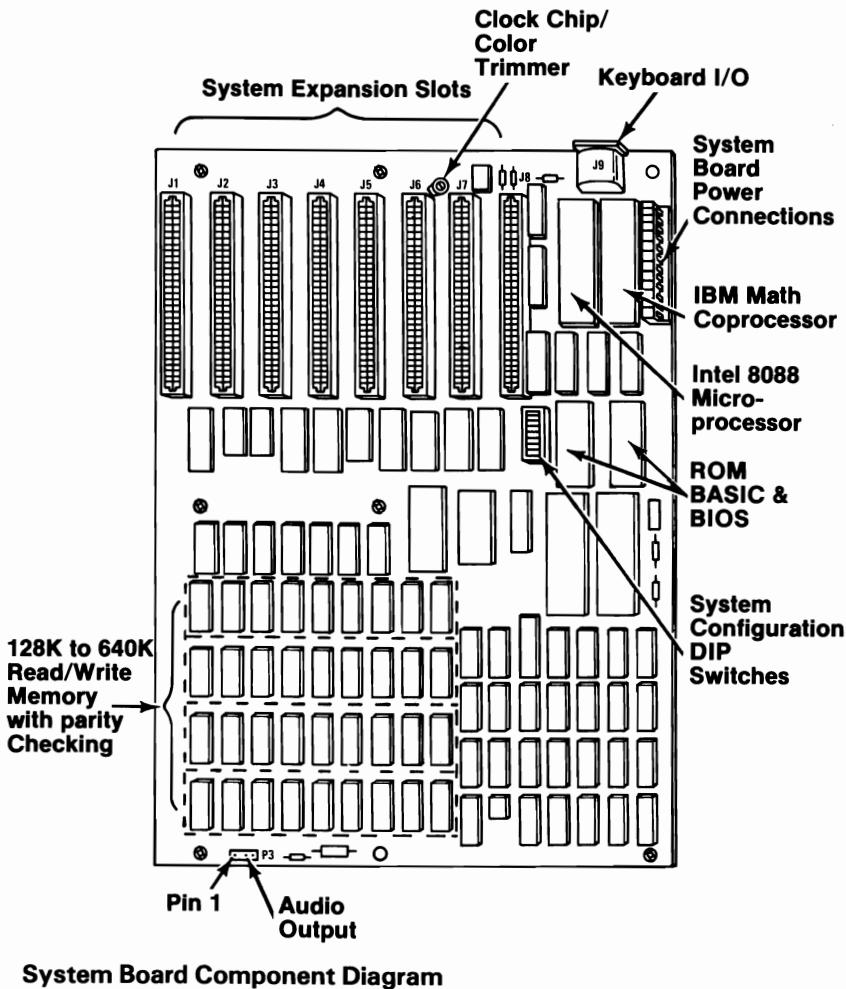
I/O Channel (A-Side, J1 through J8)

I/O Pin	Signal Name	I/O
B1	GND	Ground
B2	RESET DRV	0
B3	+5 Vdc	Power
B4	IRQ_2	1
B5	-5 Vdc	Power
B6	DRQ2	1
B7	-12 Vdc	Power
B8	-CARD SLCTD	1
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	1
B17	-DACK1	0
B18	DRQ1	1
B19	-DACK0	I/O
B20	CLK	0
B21	IRQ7	1
B22	IRQ6	1
B23	IRQ5	1
B24	IRQ4	1
B25	IRQ3	1
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.



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 OFF, are reserved for the system board I/O. Hex 100 to 3FF are available on the I/O channel.

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

** At power-on-time, the Non Mask Interrupt into the 8088 is masked off. This mask bit can be set and reset through system software as follows:
Set mask: Write hex 80 to I/O Address
hex A0(enable NMI)
Clear mask: Write hex 00 to I/O Address
hex A0(disable NMI)

Planar I/O Address Map

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

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

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

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

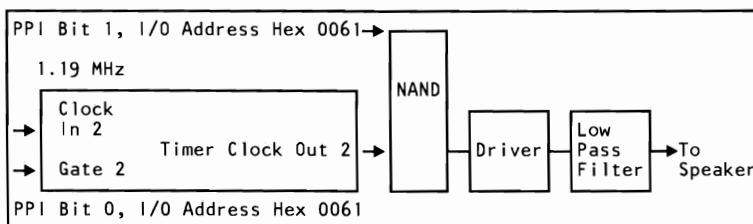
Channel I/O Address Map

Other Circuits

Speaker Circuit

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

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



Speaker Drive System Block Diagram

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

Speaker Tone Generation

The speaker connection is a 4-pin Berg connector.

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	+ Keyboard Scan Code	0	Or	Diagnostic Outputs	0					
			1			1					
			2			2					
			3			3					
			4			4					
			5			5					
			6			6					
			7			7					
0060	PB0	+ 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	Loop on POST	Sw-1		Display 0	**Sw-5					
		+ CoProcessor Installed	Sw-2		Display 1	**Sw-6					
		+ Planar RAM Size 0	* Sw-3		# Drives	***Sw-7					
		+ Planar RAM Size 1	* Sw-4		# Drives 1	***Sw-8					
		Spare									
		+ Timer Channel 2 Out									
		+ I/O Channel Check									
		+ RAM Parity Check									
0063	Command/Mode Register		Hex 99								
	Mode Register Value		7	6	5	4					
			3	2	1	0					
			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

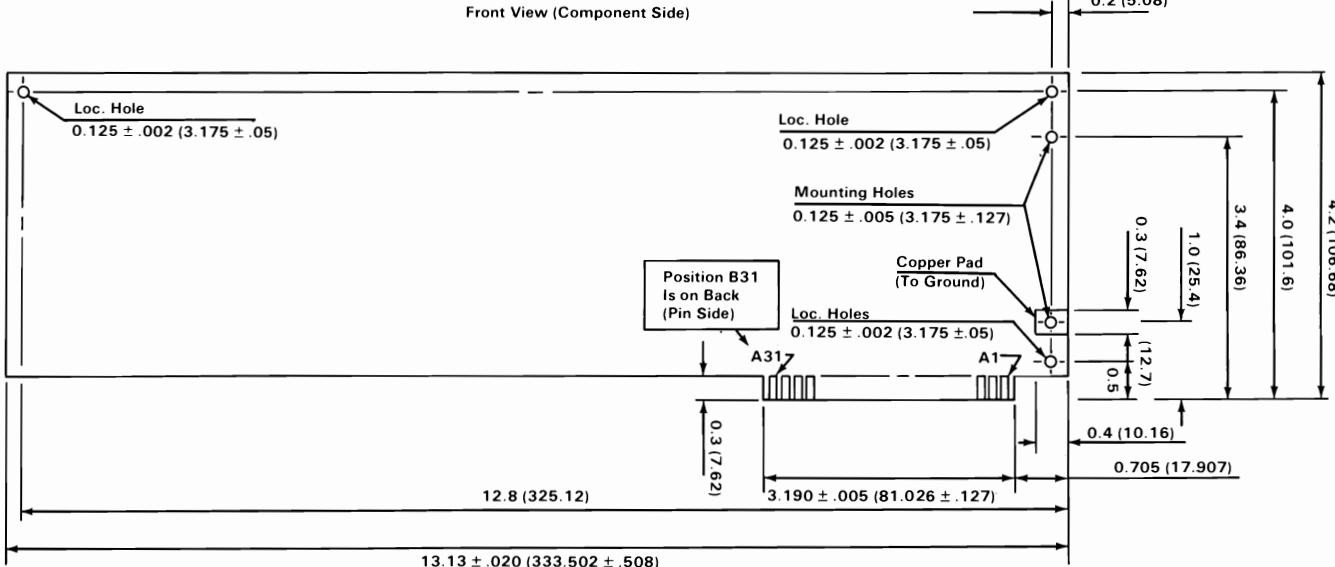
- 43 decibels average-noise rating (without printer)

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 $\pm .010$ (.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,
 $0.100 \pm .0005$ ($2.54 \pm .0127$) Center
 to Center, $0.06 \pm .0005$ ($1.524 \pm .0127$)
 Width.
5. Numbers in Parentheses
 are in Millimeters. All Others
 are in Inches.

Connectors

The system board has the following additional connectors:

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

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

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

Power Supply Connectors (P1, P2)

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

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

Speaker Connector (J19)

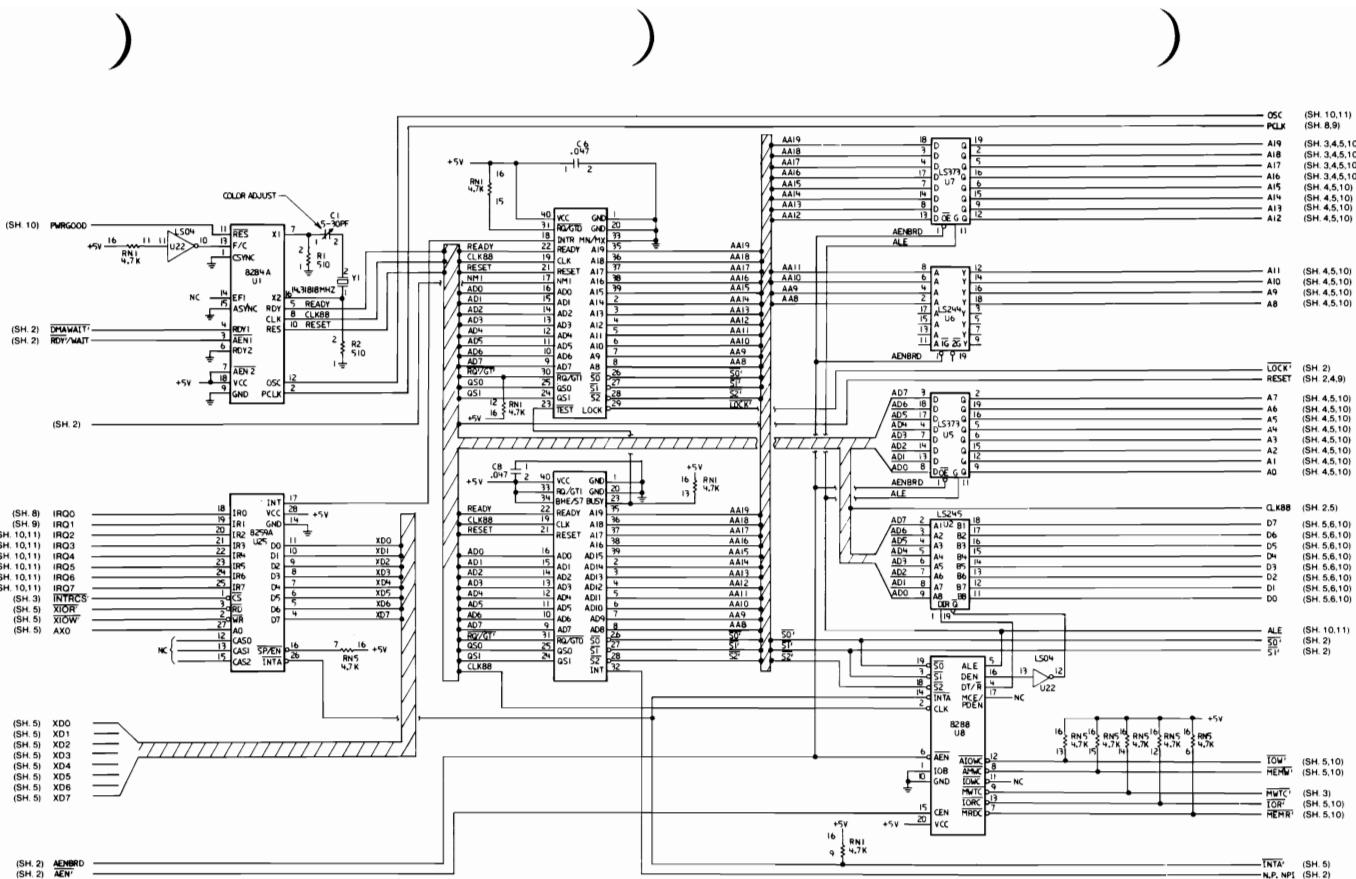
The keyboard connector, J22, is a 5-pin, 90-degree printed circuit board (PCB) mounting, DIN connector. For pin numbering, see the “Keyboard” section. The pin assignments are as follows:

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

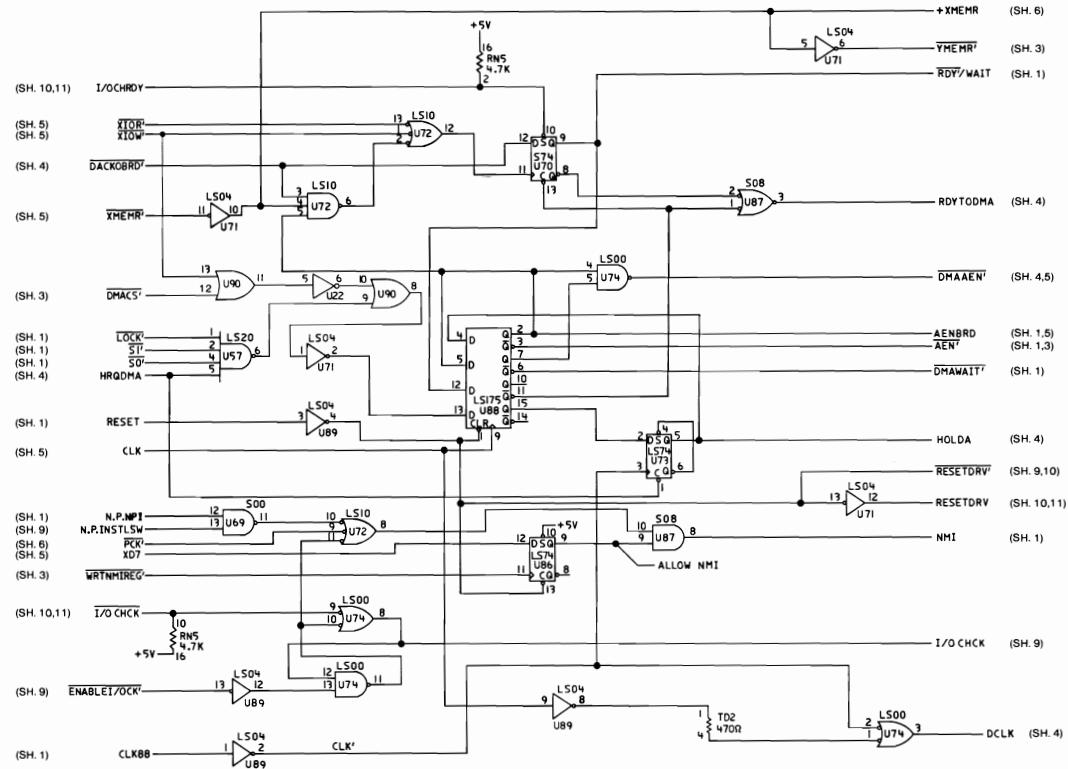
Keyboard Connector (J22)

Logic Diagrams - 64/256K

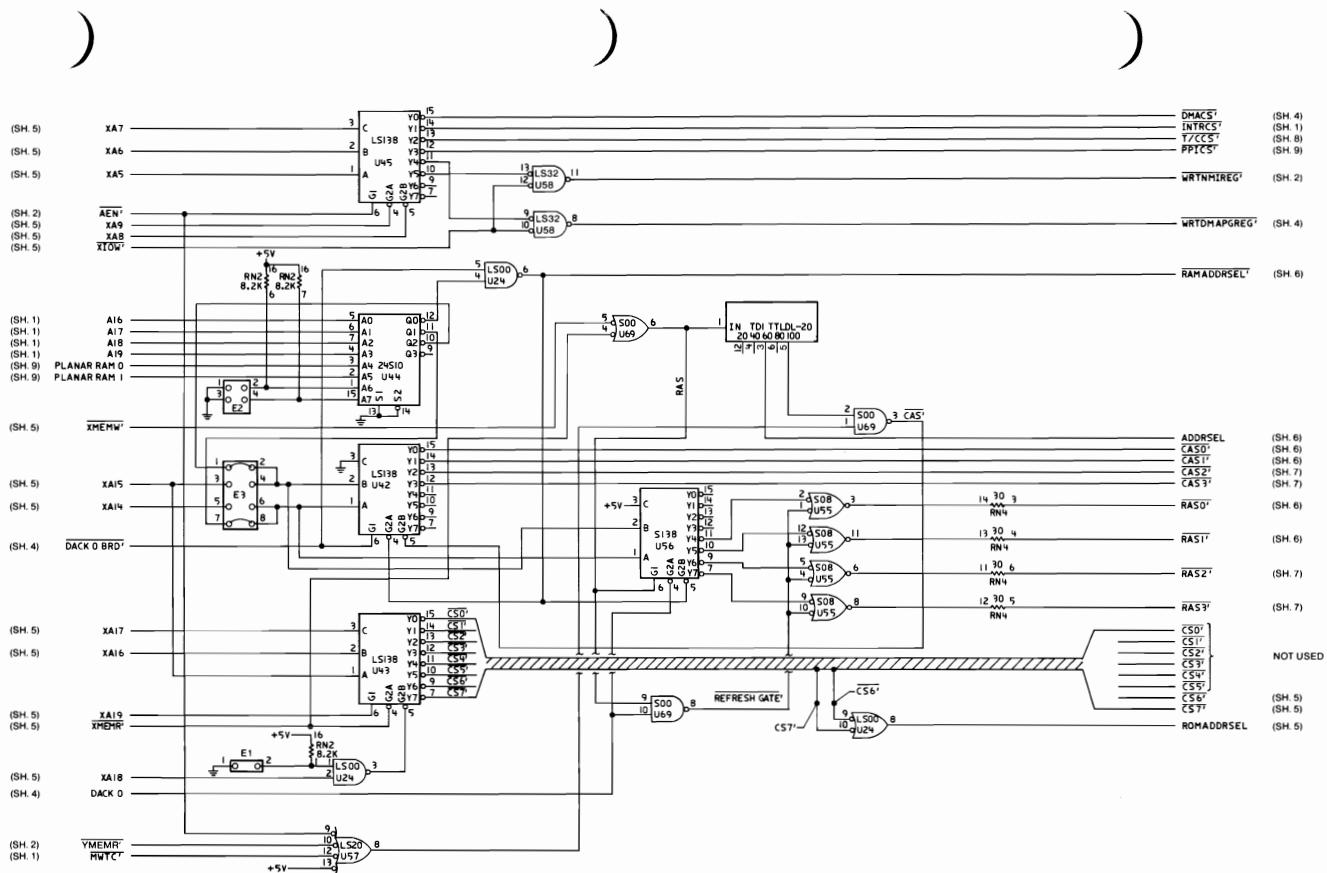
The following pages contain the logic diagrams for the 64/256K system board.



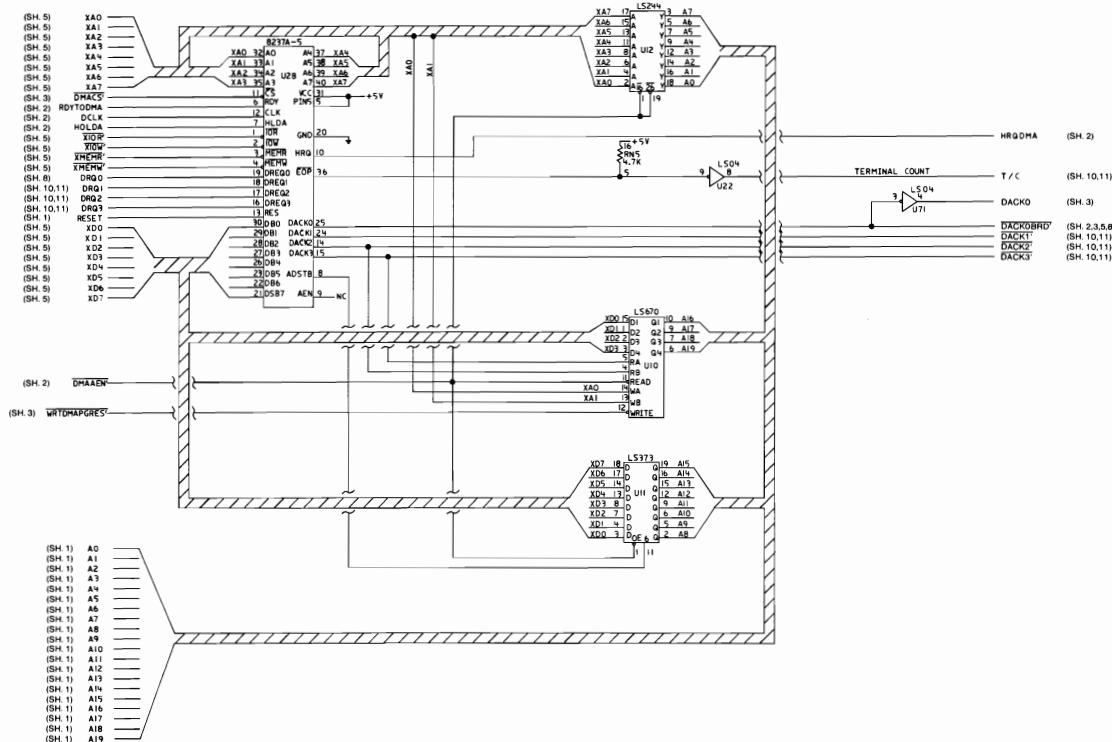
64/256K System Board (Sheet 1 of 11)



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

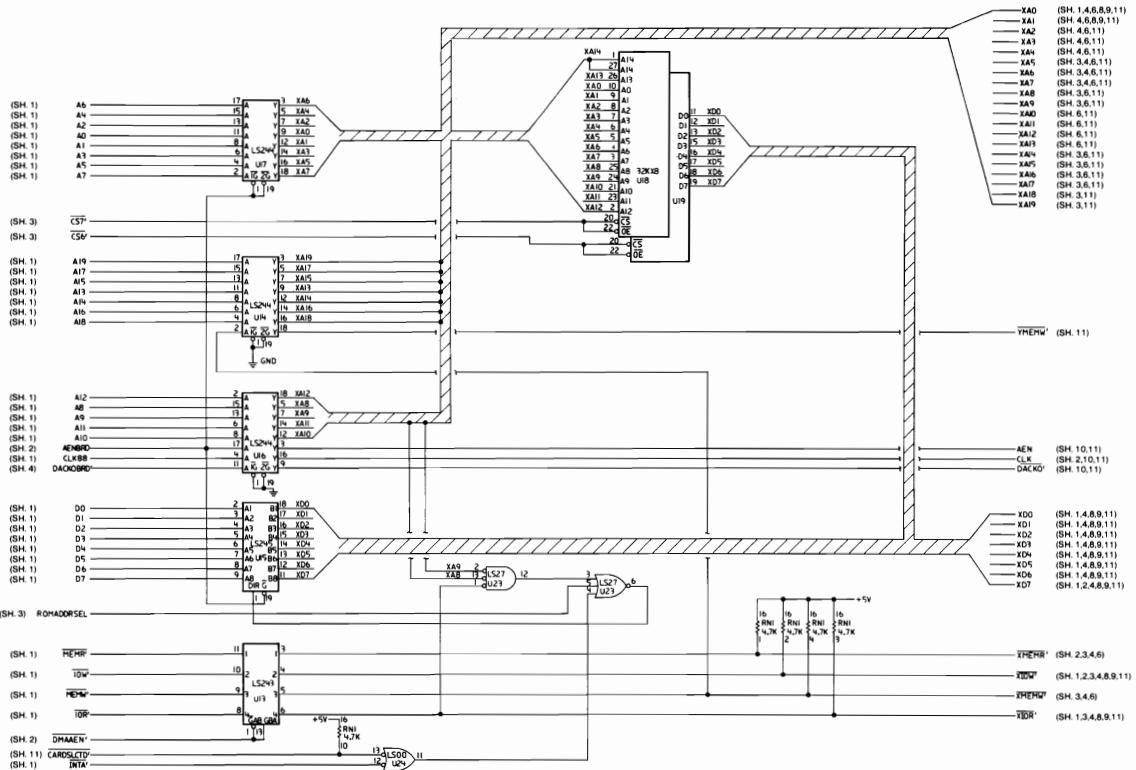


64/256K System Board (Sheet 3 of 11)

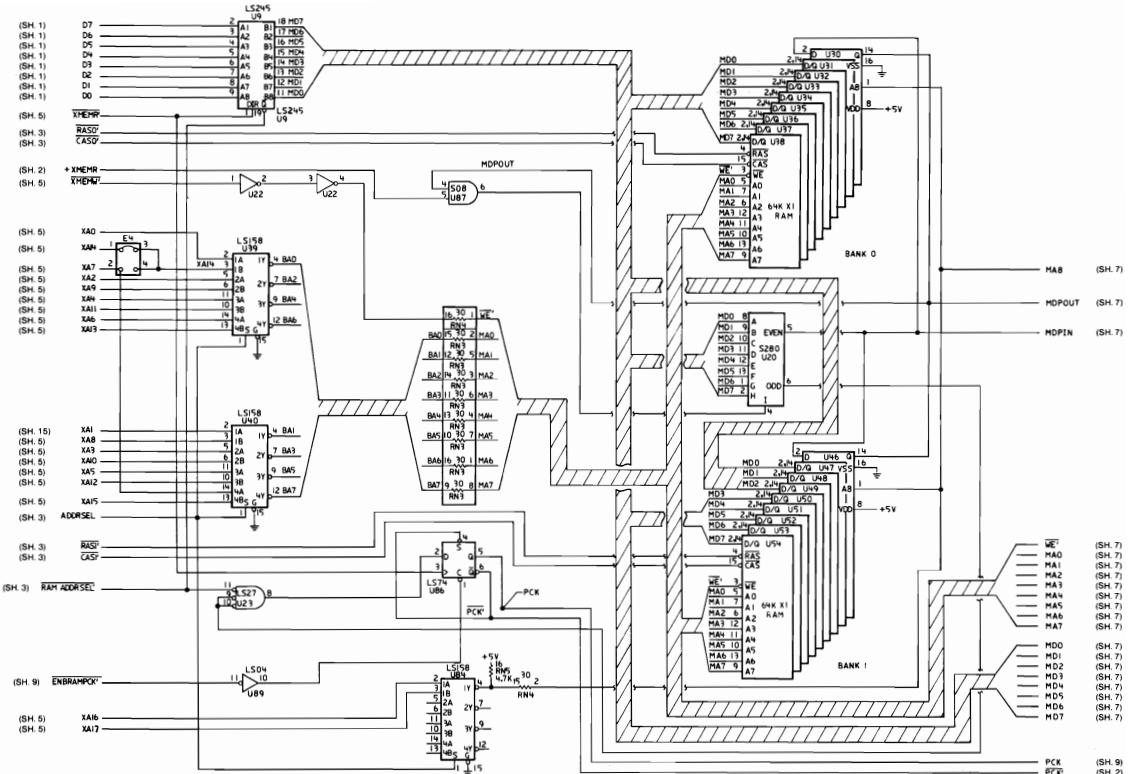


64/256K System Board (Sheet 4 of 11)

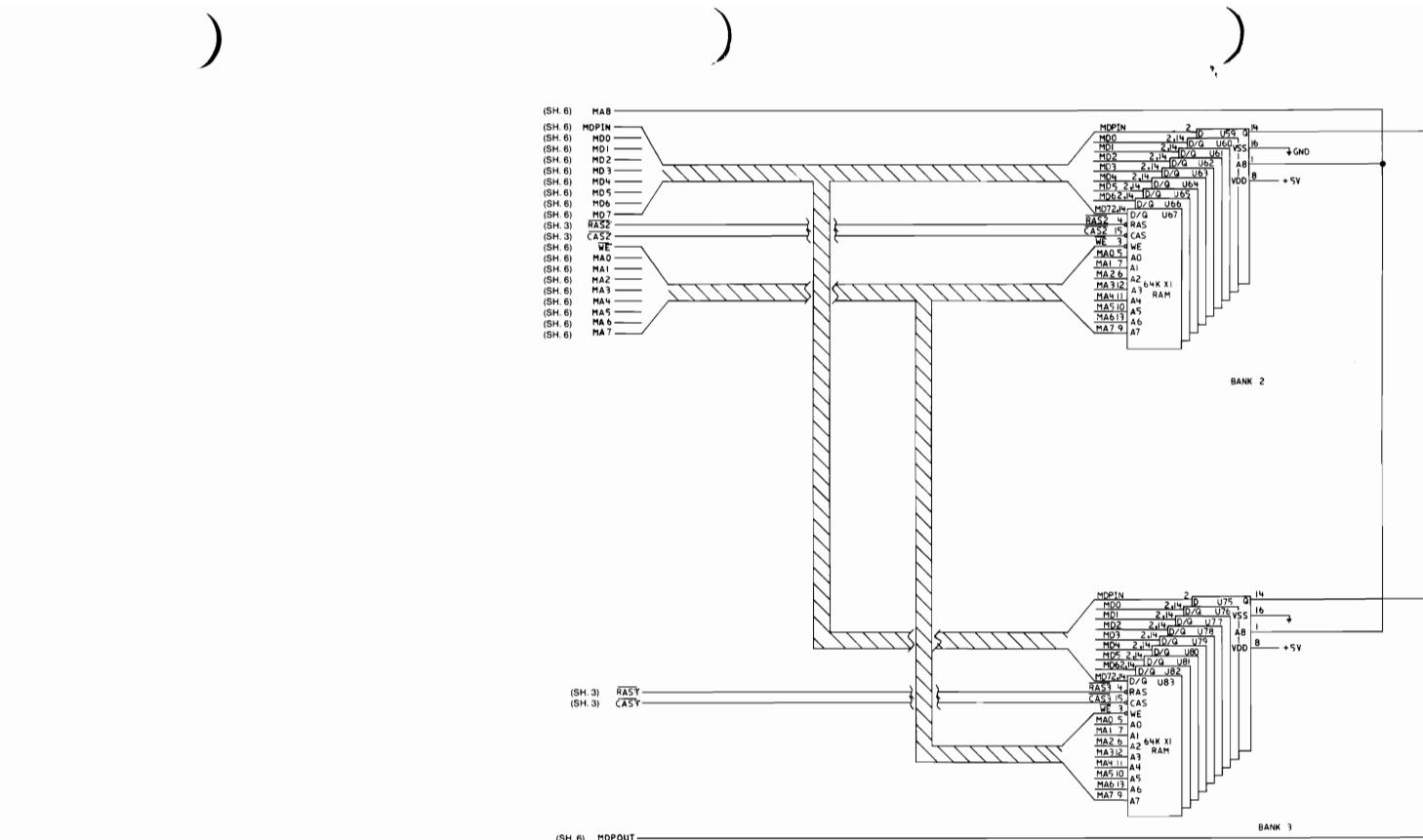
)))



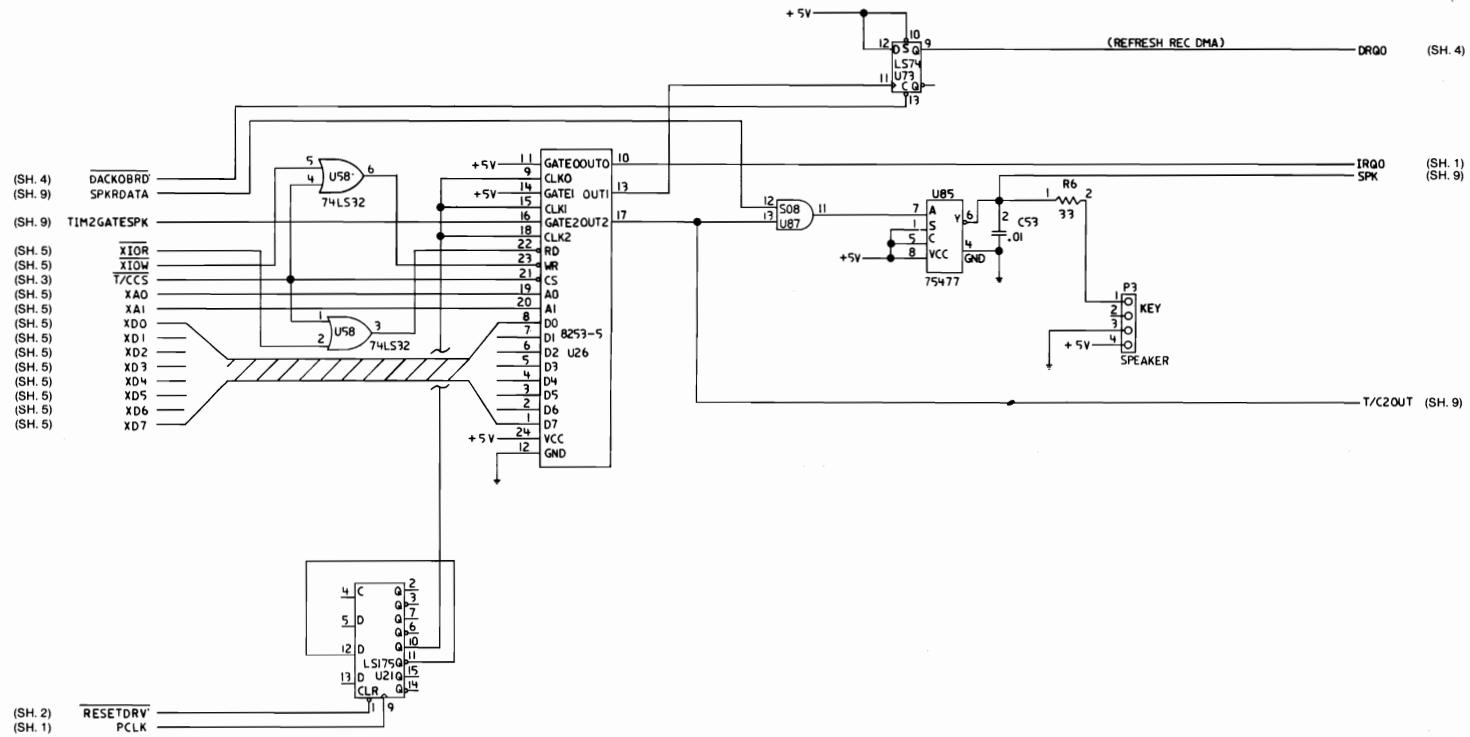
64/256K System Board (Sheet 5 of 11)



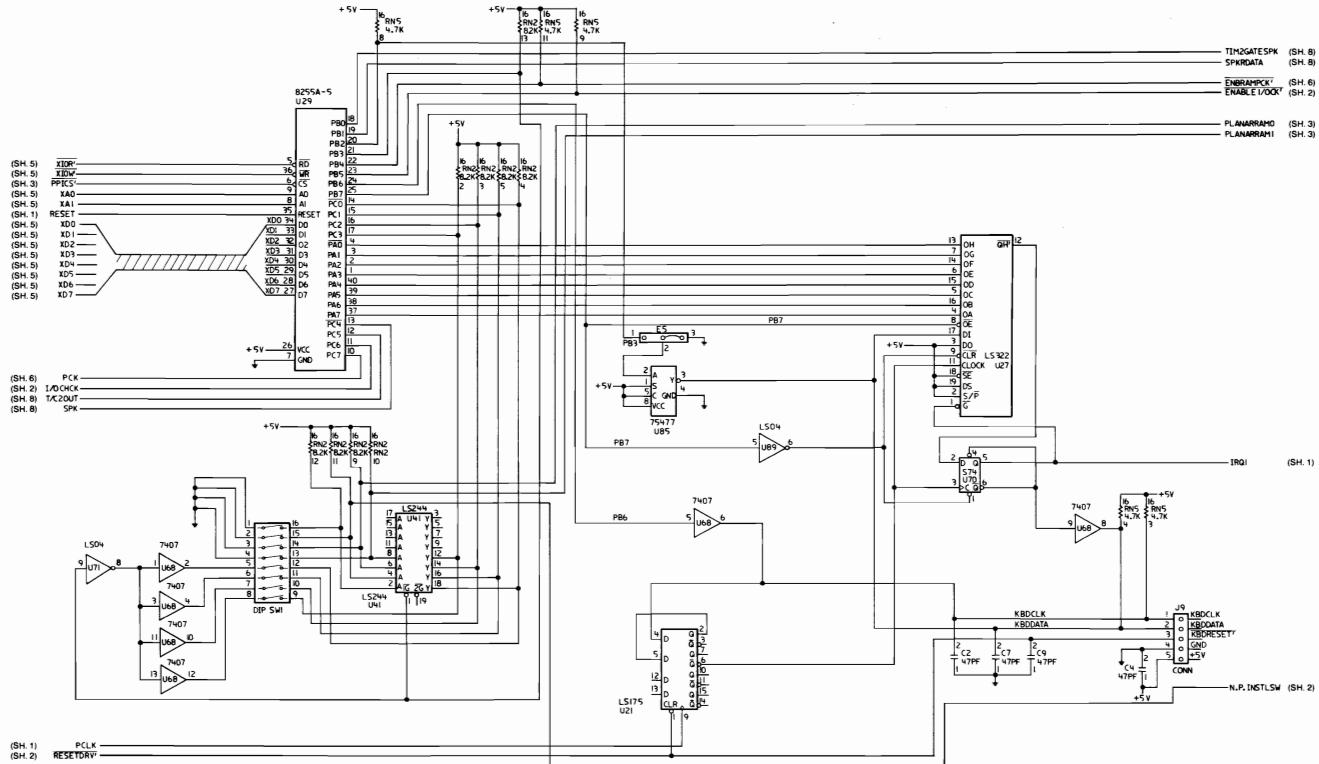
64/256K System Board (Sheet 6 of 11)



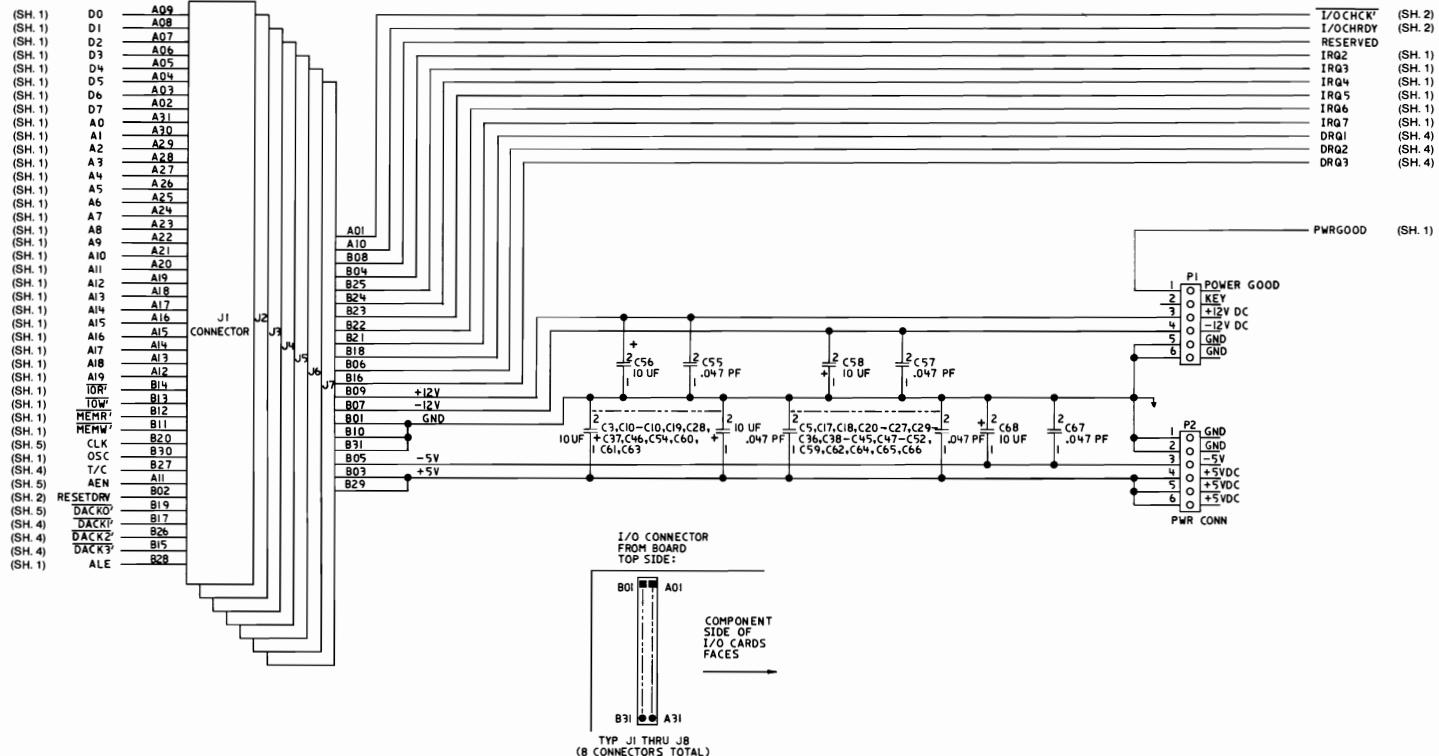
64/256K System Board (Sheet 7 of 11)



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



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



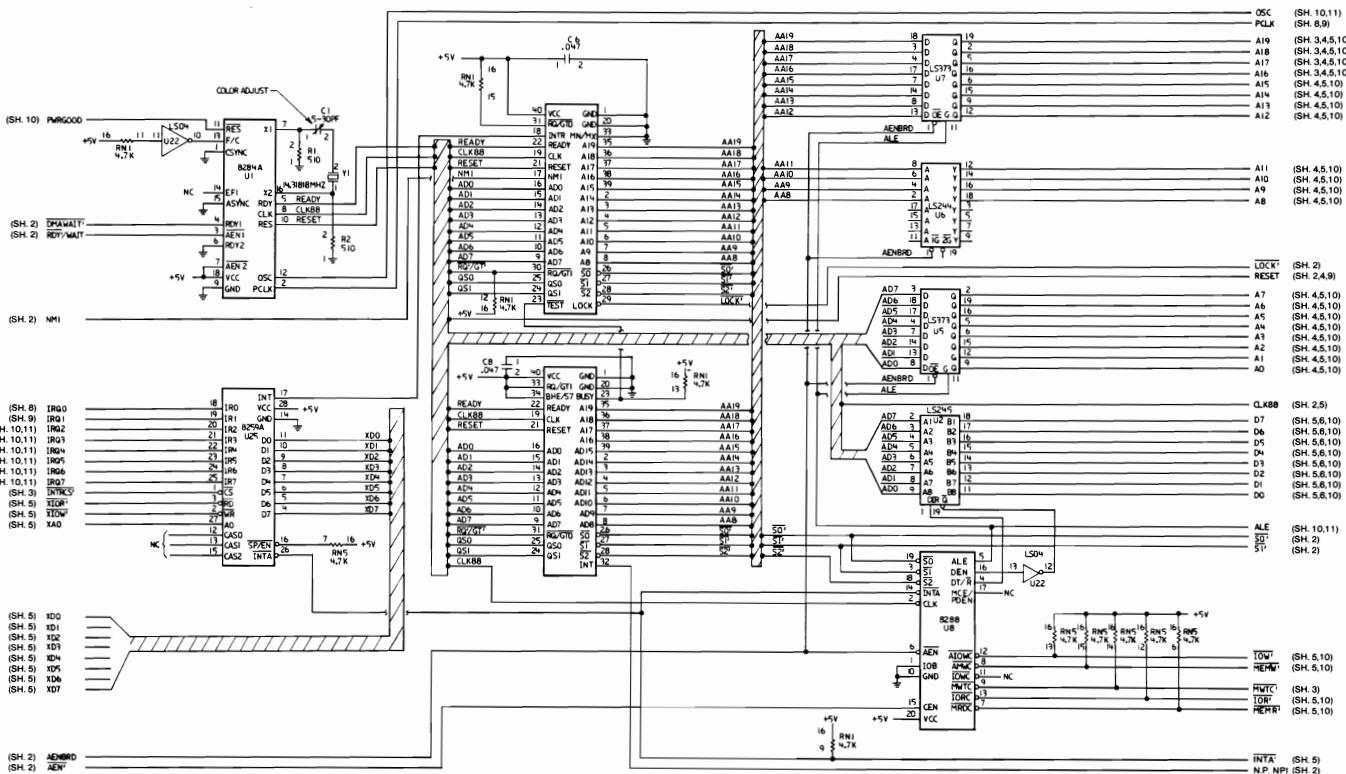
64/256K System Board (Sheet 10 of 11)

(SH. 5)	XD0	A09	
(SH. 5)	XD1	A08	
(SH. 5)	XD2	A07	
(SH. 5)	XD3	A06	
(SH. 5)	XD4	A05	
(SH. 5)	XD5	A04	
(SH. 5)	XD6	A03	
(SH. 5)	XD7	A02	
(SH. 5)	XA0	A31	
(SH. 5)	XA1	A30	
(SH. 5)	XA2	A29	
(SH. 5)	XA3	A28	
(SH. 5)	XA4	A27	
(SH. 5)	XA5	A26	
(SH. 5)	XA6	A25	
(SH. 5)	XA7	A24	
(SH. 5)	XA8	A23	
(SH. 5)	XA9	A22	
(SH. 5)	XA10	A21	
(SH. 5)	XAI1	A20	
(SH. 5)	XAI2	A19	
(SH. 5)	XAI3	A18	
(SH. 5)	XAI4	A17	
(SH. 5)	XAI5	A16	
(SH. 5)	XAI6	A15	
(SH. 5)	XAI7	A14	
(SH. 5)	XAI8	A13	
(SH. 5)	XAI9	A12	
(SH. 5)	XIOR'	B14	
(SH. 5)	XIOW'	B13	
(SH. 2)	YMEMR'	B12	
(SH. 5)	YMEMW'	B11	
(SH. 5)	CLK	B20	
(SH. 1)	OSC	B30	
(SH. 4)	T/C	B27	
(SH. 5)	AEN	A11	
(SH. 2)	RESETDRV	B02	
(SH. 5)	DACK0'	B19	
(SH. 4)	DACK1'	B17	
(SH. 4)	DACK2'	B26	
(SH. 4)	DACK3'	B15	
(SH. 1)	ALE	B28	
			J8 CONNECTOR
		A01	I/OCHCK' (SH. 2)
		A10	I/OCHRDY (SH. 2)
		B08	CARDSLCTD (SH. 5)
		B04	IRQ2 (SH. 1)
		B25	IRQ3 (SH. 1)
		B24	IRQ4 (SH. 1)
		B23	IRQ5 (SH. 1)
		B22	IRQ6 (SH. 1)
		B21	IRQ7 (SH. 1)
		B18	DRQ1 (SH. 4)
		B06	DRQ2 (SH. 4)
		B16	DRQ3 (SH. 4)
		B09	+12V
		B07	-12V
		B01	
		B10	
		B31	
		B05	GND
		B03	-5V
		B29	+5V

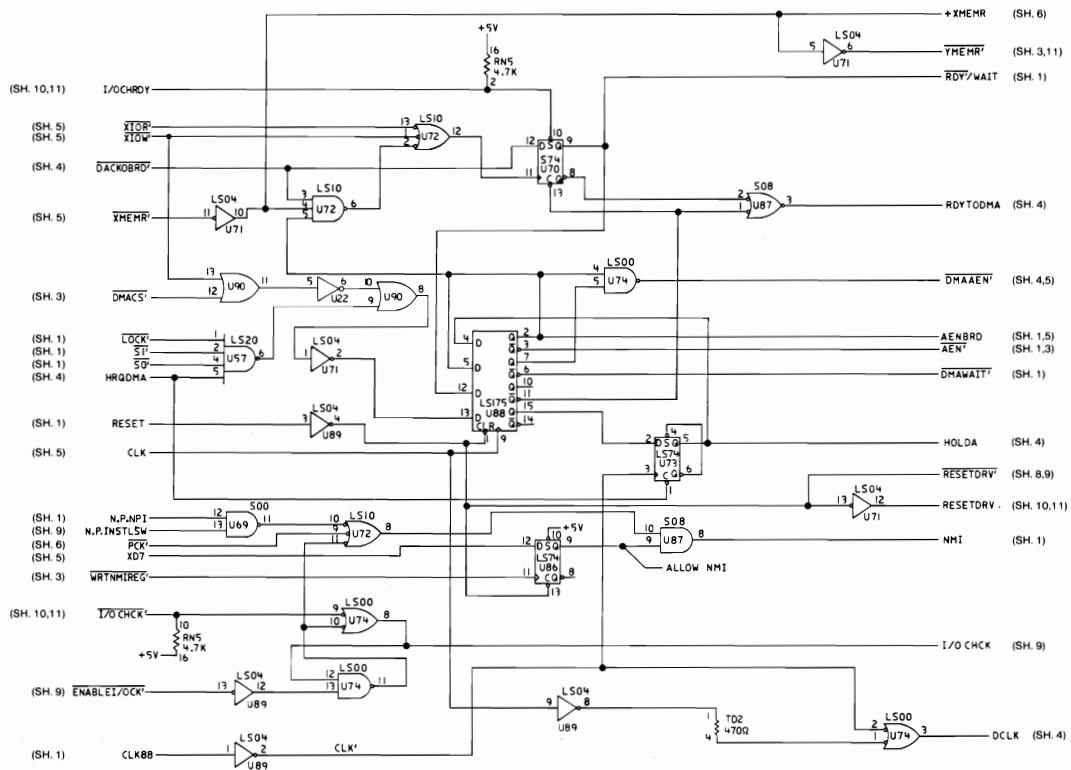
64/256K System Board (Sheet 11 of 11)

Logic Diagrams - 256/640K

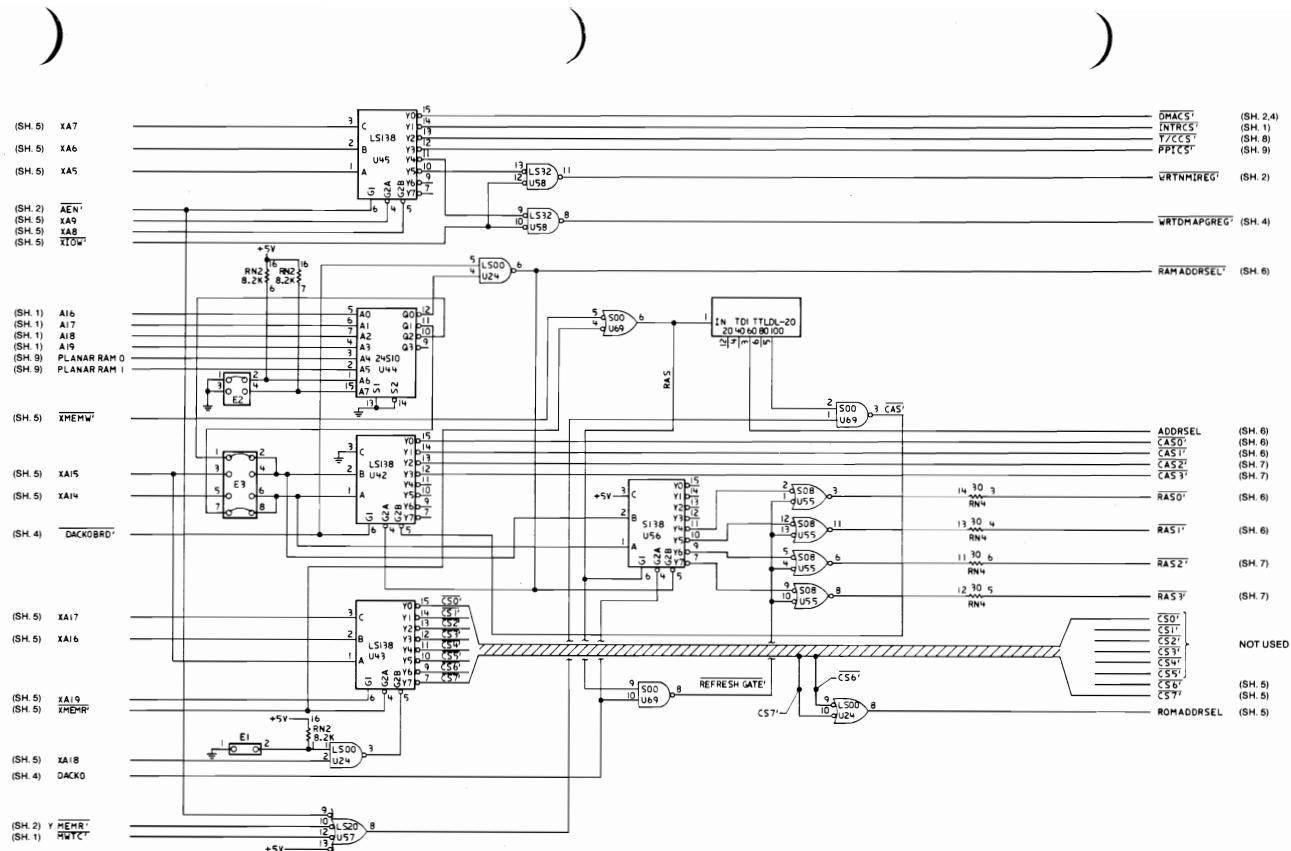
The following pages contain the logic diagrams for the 256/640K system board.



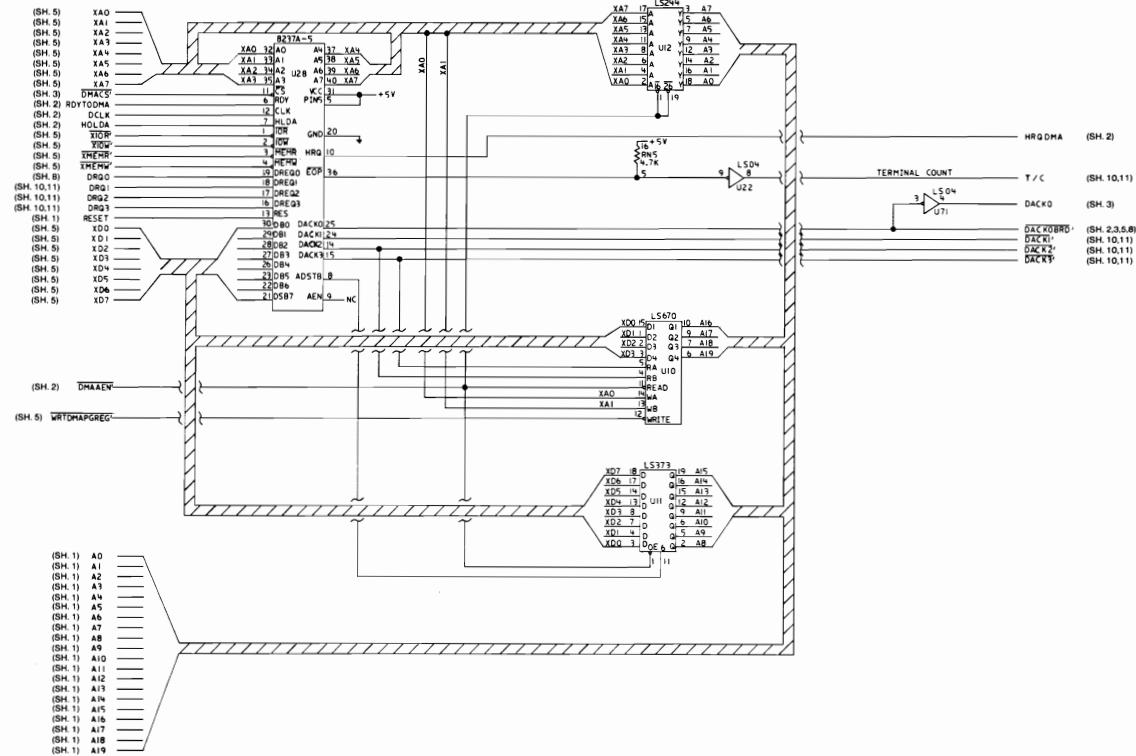
256/640K System Board (Sheet 1 of 11)



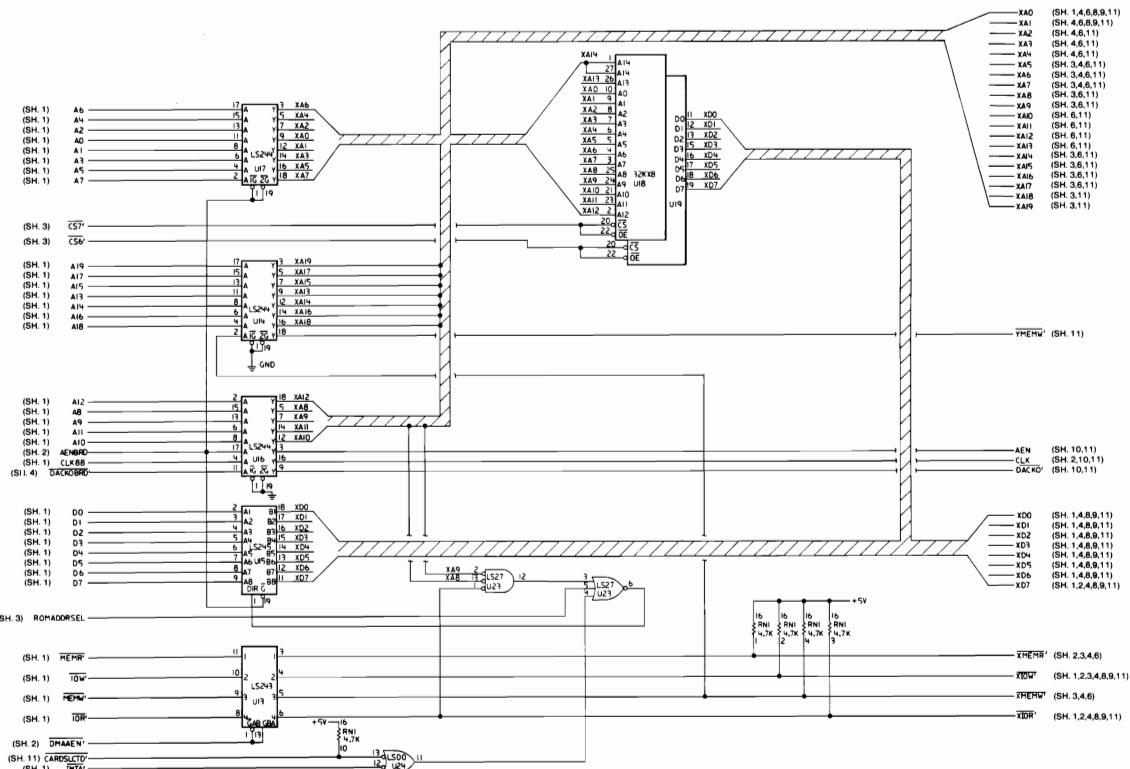
256/640K System Board (Sheet 2 of 11)



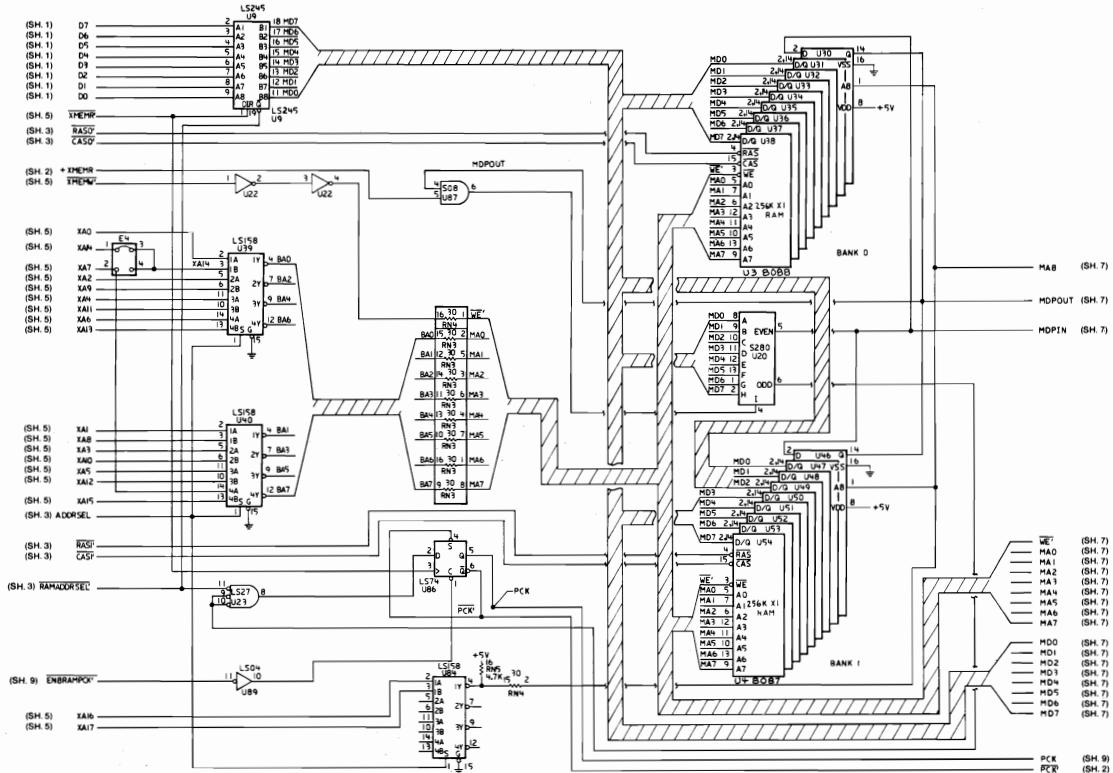
256/640K System Board (Sheet 3 of 11)



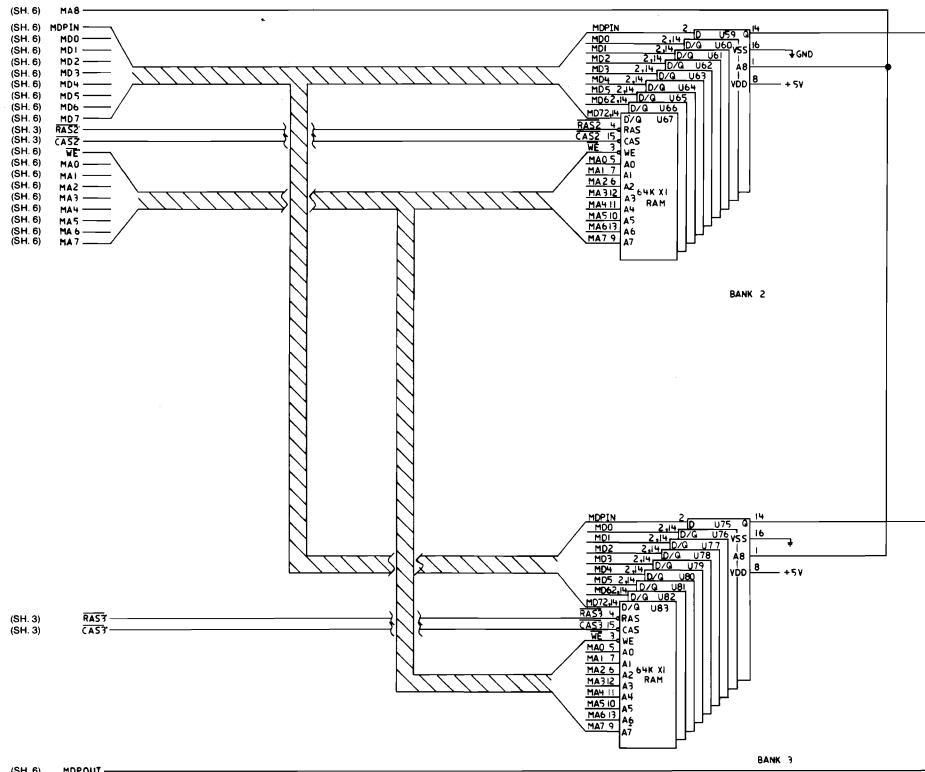
256/640K System Board (Sheet 4 of 11)



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

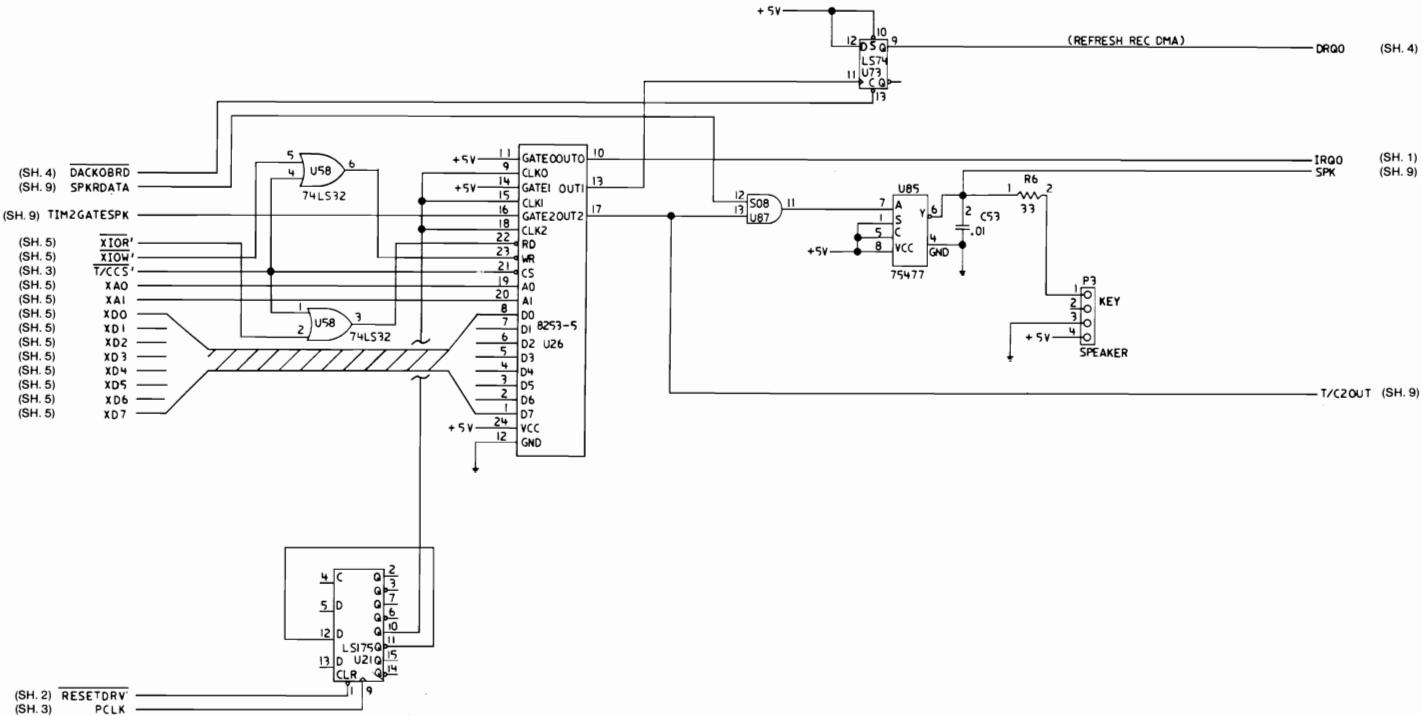


256/640K System Board (Sheet 6 of 11)

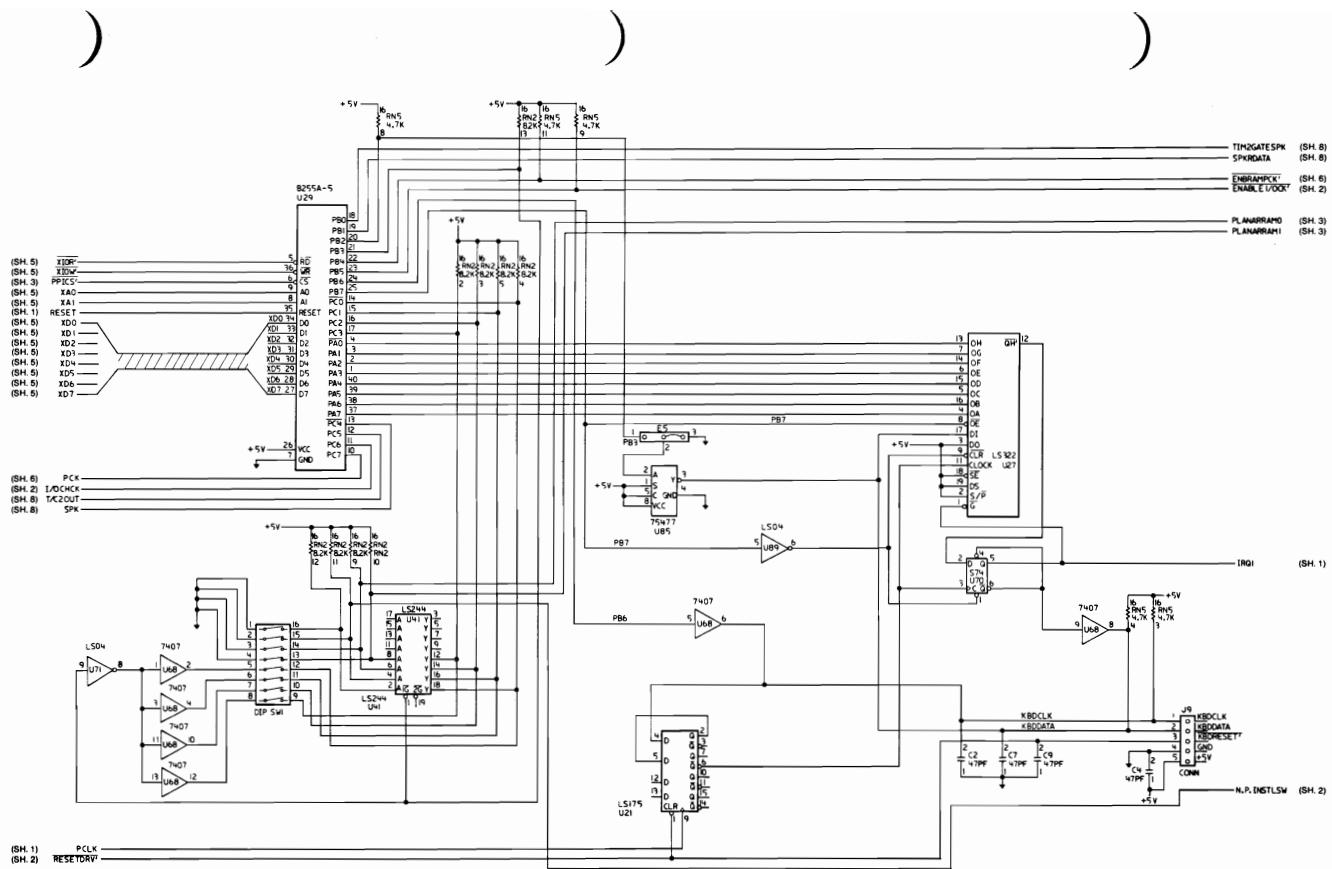


256/640K System Board (Sheet 7 of 11)

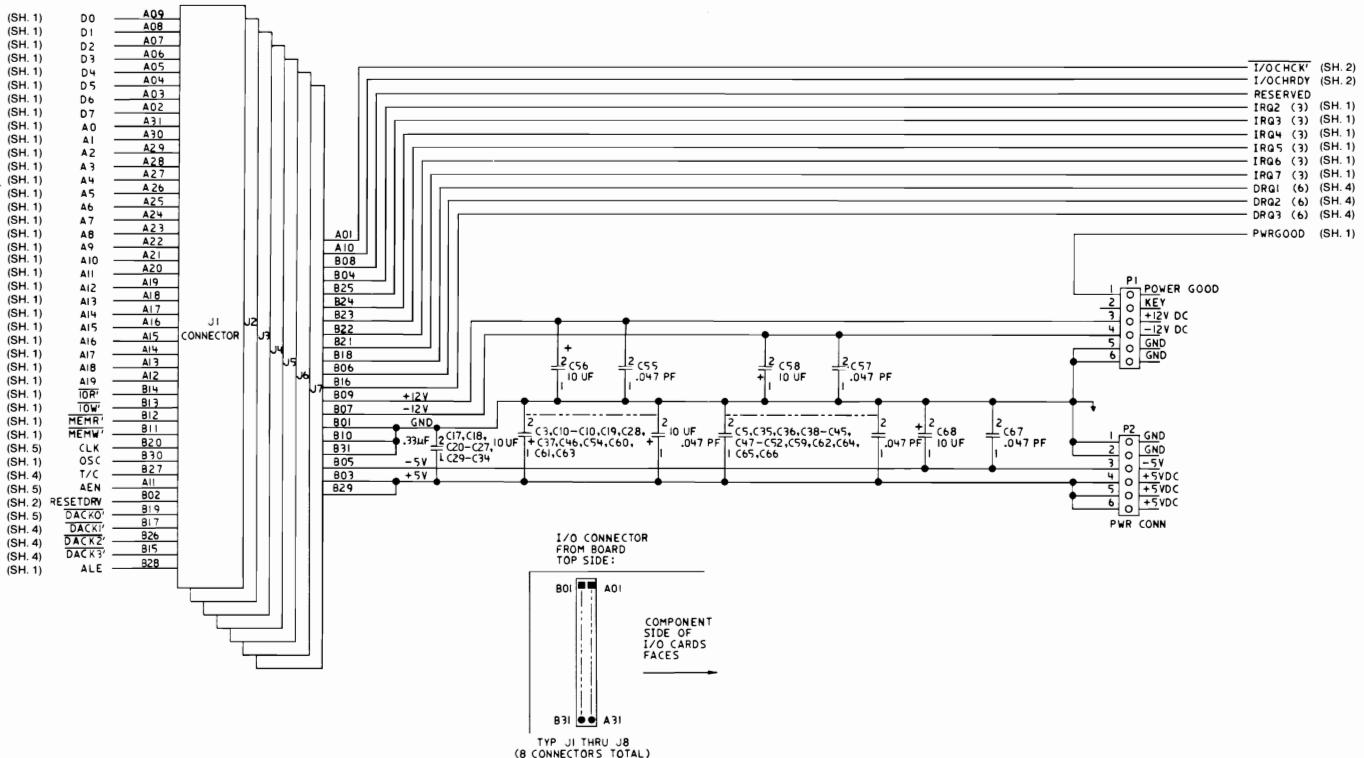
SECTION 1



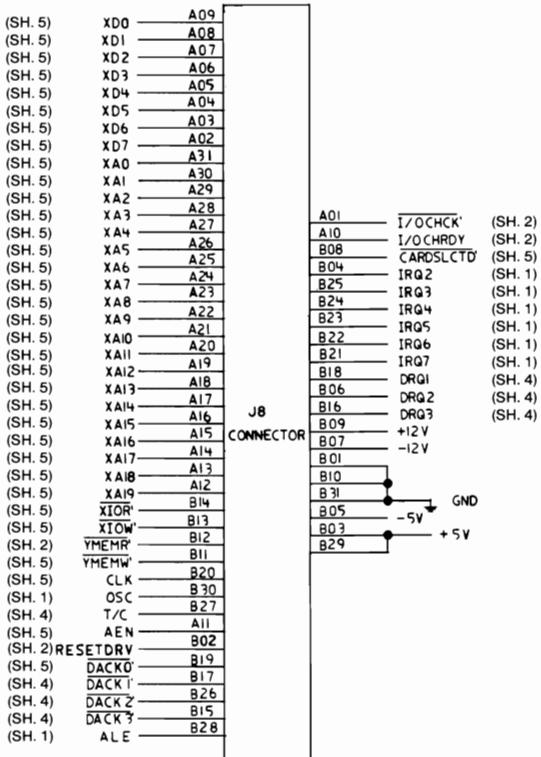
256/640K System Board (Sheet 8 of 11)



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



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



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

Notes:

SECTION 2. COPROCESSOR

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

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).

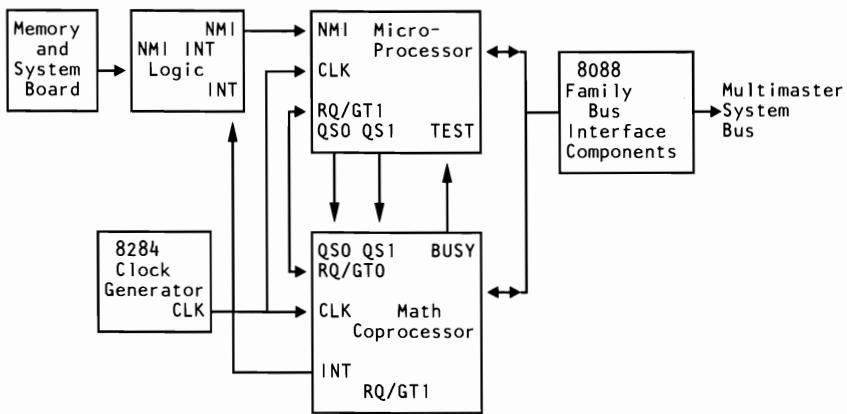
When an incorrect instruction is sent to the coprocessor (for example, divide by 0 or load a full register), the coprocessor can signal the microprocessor with an interrupt. There are three conditions that will disable the coprocessor interrupt to the microprocessor:

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 ± 3 Hz		
Nominal Vac	Minimum Vac	Maximum Vac
110	90	137
220/240	180	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	0.0	1.0	90	137
220/240	0.0	0.5	180	259

Vac Output

The sense levels of the dc outputs are:

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

Vdc Sense Levels

Overvoltage/Overscurrent Protection

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

Voltage and Current Protection

Power Good Signal

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

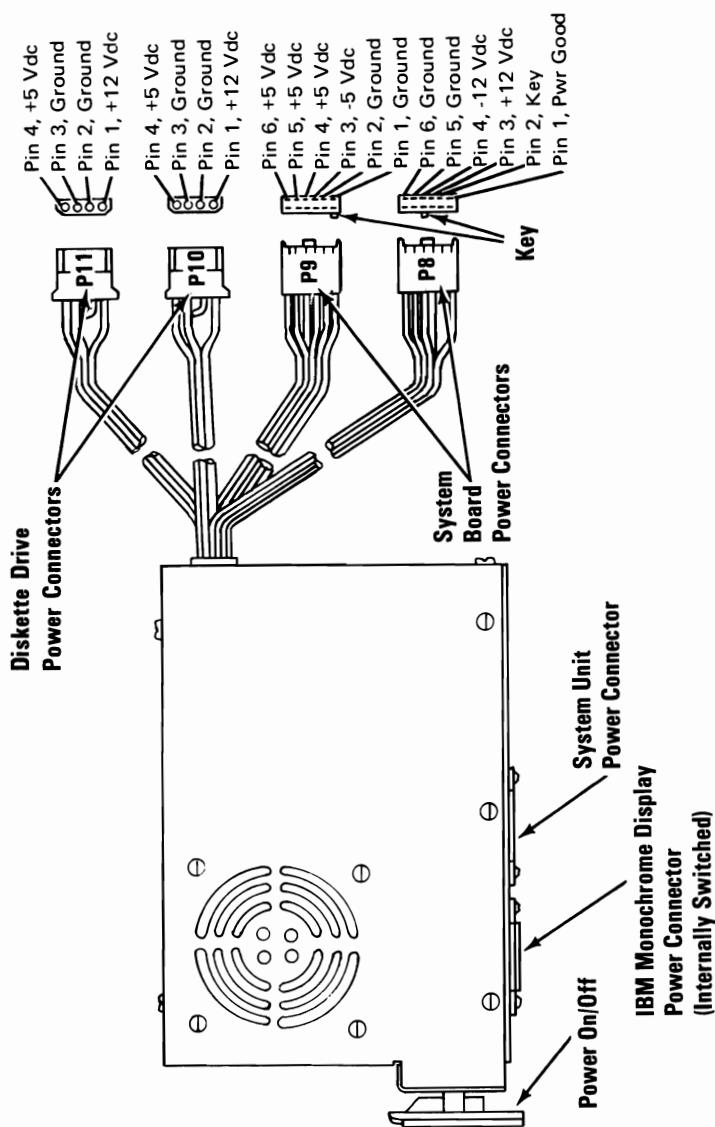
The 'power good' signal indicates that there is adequate power to continue processing. If the power goes below the specified levels, the 'power good' signal triggers a system shutdown.

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

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

Connector Specifications and Pin Assignments

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



Power Supply and Connectors

IBM Portable Personal Computer Power Supply

Description

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

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

Voltage and Current Requirements

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

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

Nominal Output(Vdc)	Load Current (A)		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.

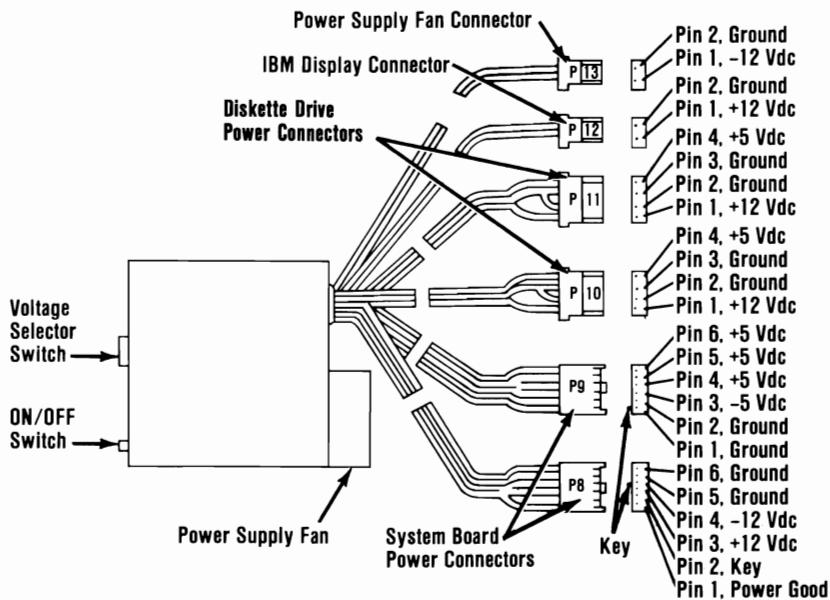
This signal is the logical **AND** of the dc output-voltage sense signal and the ac input-voltage fail signal. This signal is **TTL**-compatible up-level for normal operation or down-level for fault conditions. The ac 'fail' signal causes 'power good' to go to a down-level when any output voltage falls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

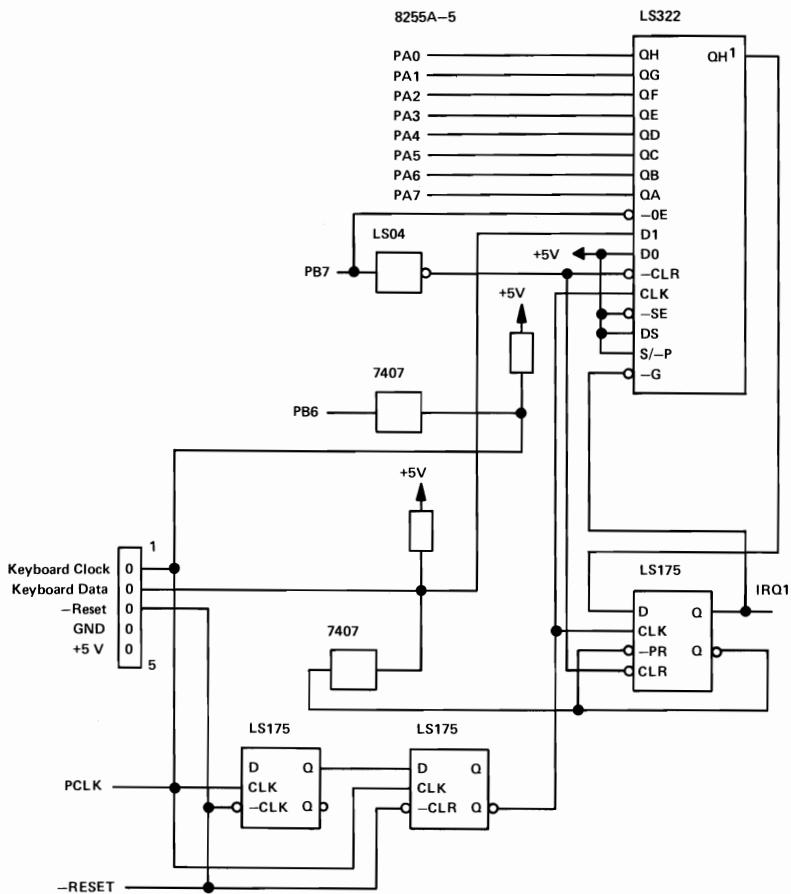
The keyboard has 83 keys arranged in three major groupings. The central portion of the keyboard is a standard typewriter keyboard layout. On the left side are 10 function keys. These keys are user-defined by the software. On the right is a 15-key keypad. These keys are also defined by the software, but 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	Ctr1	Alt
1	Esc	Esc	-1	-1
2	1	!	-1	(*)
3	2	@	Nul(000)	(*)
4	3	#	(*)	(*)
5	4	\$	(*)	(*)
6	5	%	(*)	(*)
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 alphabetic or graphics mode, with unrecognizable characters printing as blanks.

Extended Functions

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

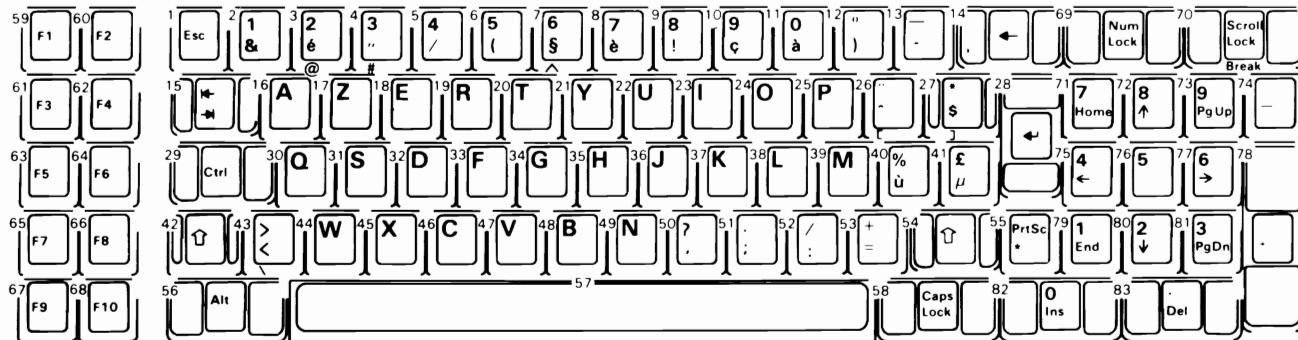
Also, the keyboard routine suppresses the typematic action of the following keys: Ctrl, Shift, Alt, Num Lock, Scroll Lock, Caps Lock, and Ins.

Keyboard Layouts

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

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

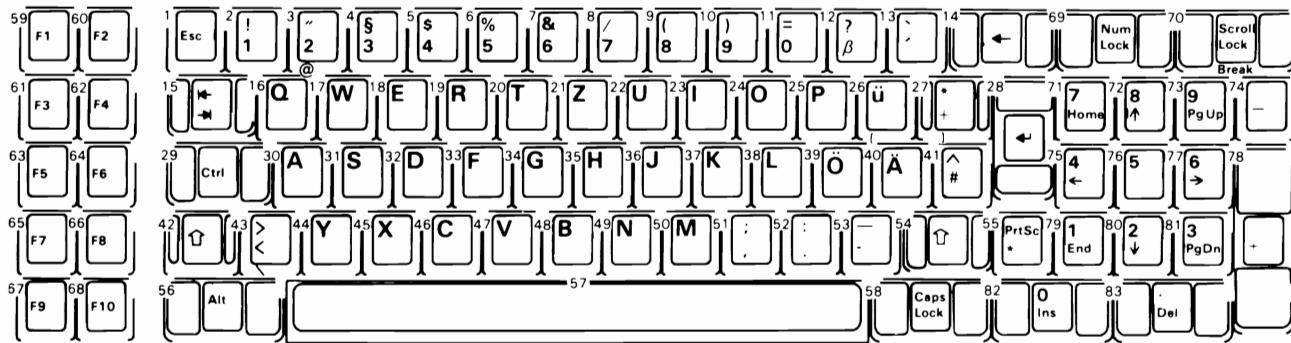
French Keyboard



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

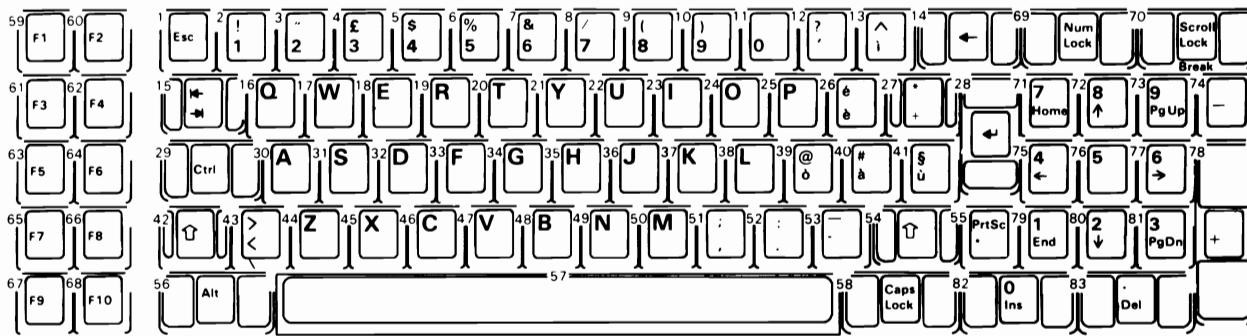
German Keyboard

4-14 83-Key Keyboard



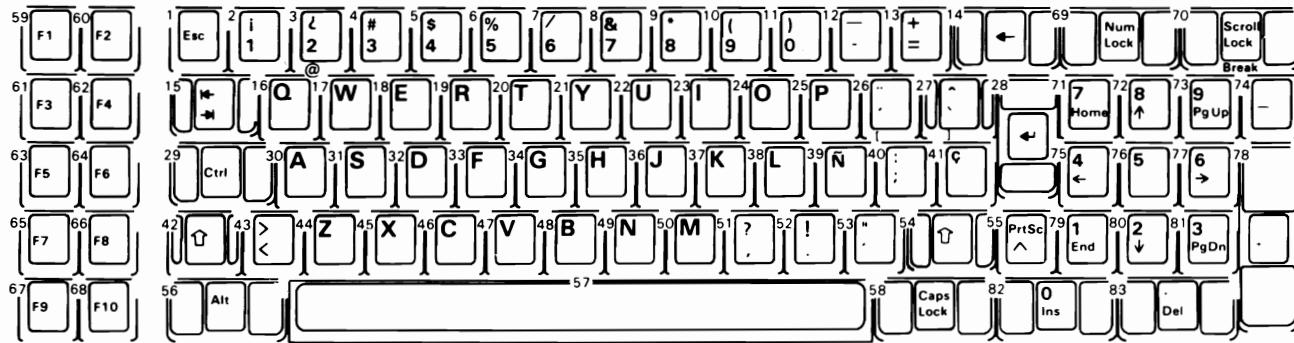
Note: Nomenclature is on both the top and front face of keybuttons as shown. The number to the upper left designates the button position.

Italian Keyboard



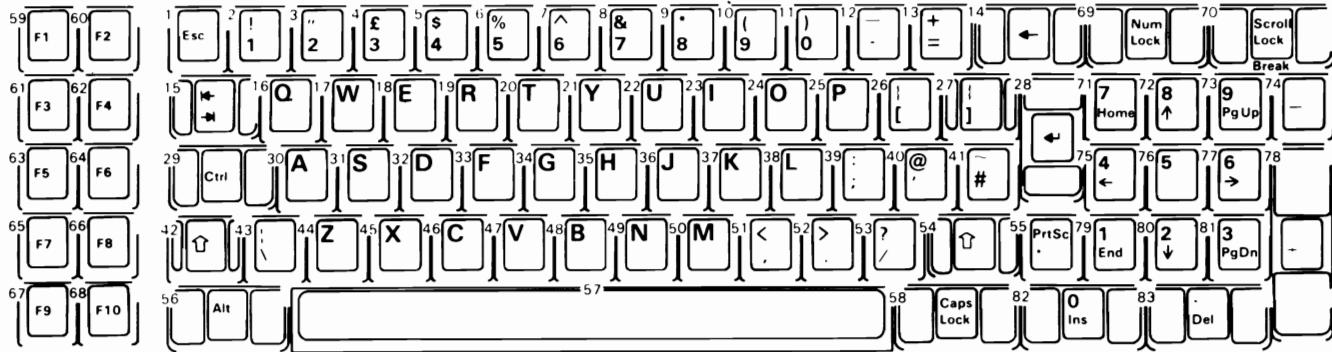
Note: Nomenclature is on both the top and front face of keybuttons as shown. The number to the upper left designates the button position.

Spanish Keyboard



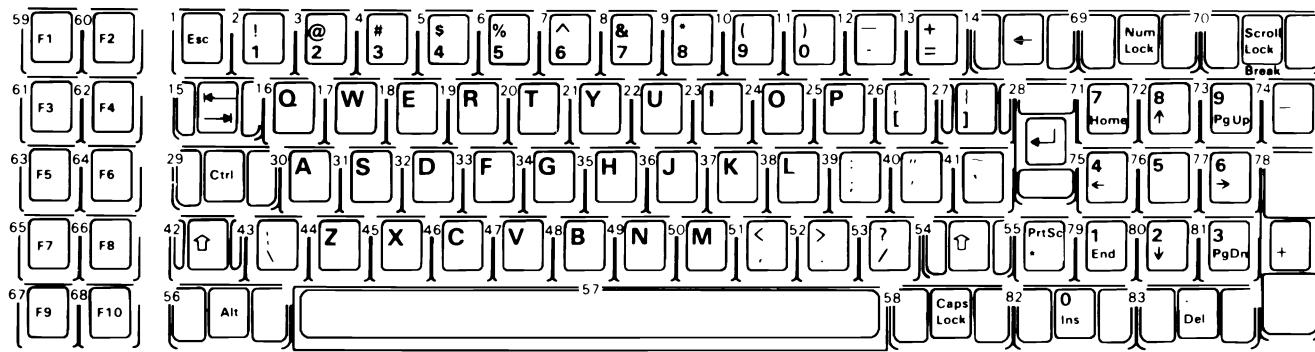
Note: Nomenclature is on both the top and front face of keybuttons as shown. The number to the upper left designates the button position.

UK Keyboard



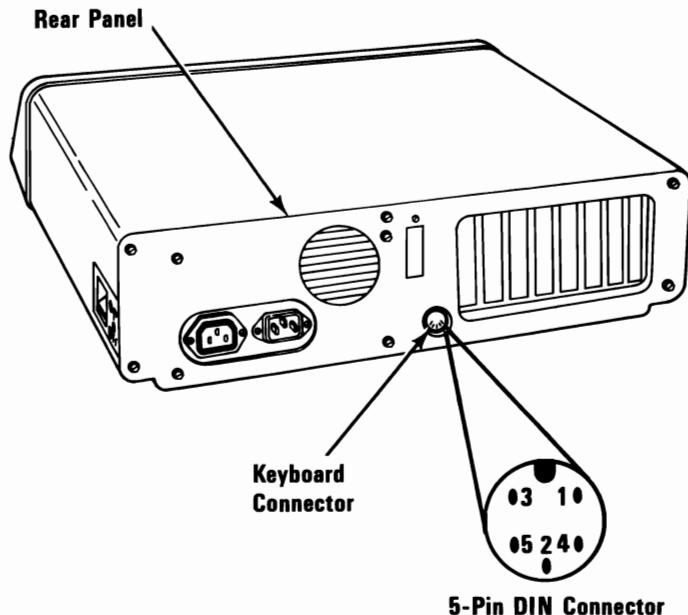
Note: Nomenclature is on both the top and front face of keybuttons as shown. The number to the upper left designates the button position.

US Keyboard



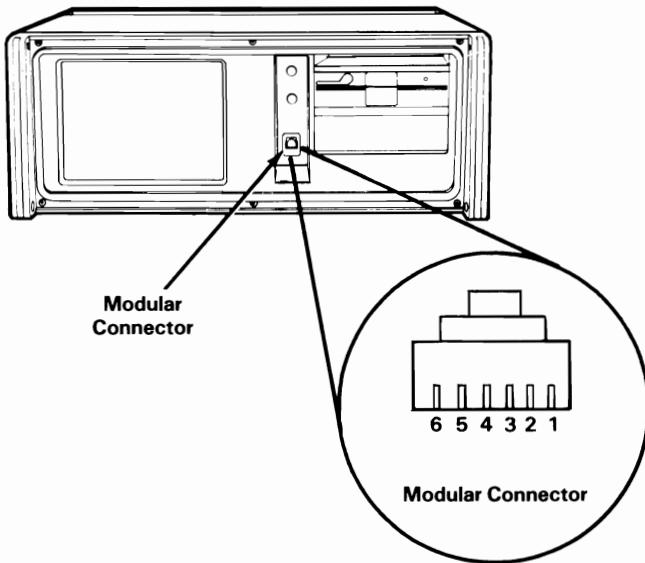
Note: Nomenclature is on both the top and front face of keybuttons as shown. The number to the upper left designates the button position.

Connector Specifications



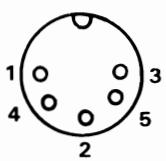
Pin	TTL Signal	Signal Level
1	+ Keyboard Clock	+ 5 Vdc
2	+ Keyboard Data	+ 5 Vdc
3	- Keyboard Reset (Not used by keyboard)	
	Power Supply Voltages	Voltage
4	Ground	0
5	+ 5 Volts	+ 5 Vdc

Keyboard Interface Connector Specifications

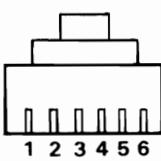


Keyboard Cable Connections

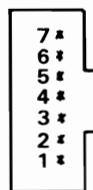
DIN Connector



Modular Connector



Keyboard Connector



Pin Side

Clock

1

Pin Side

4

Wire Side

6

Data

2

5

5

Ground

4

3

4

+5 Volts

5

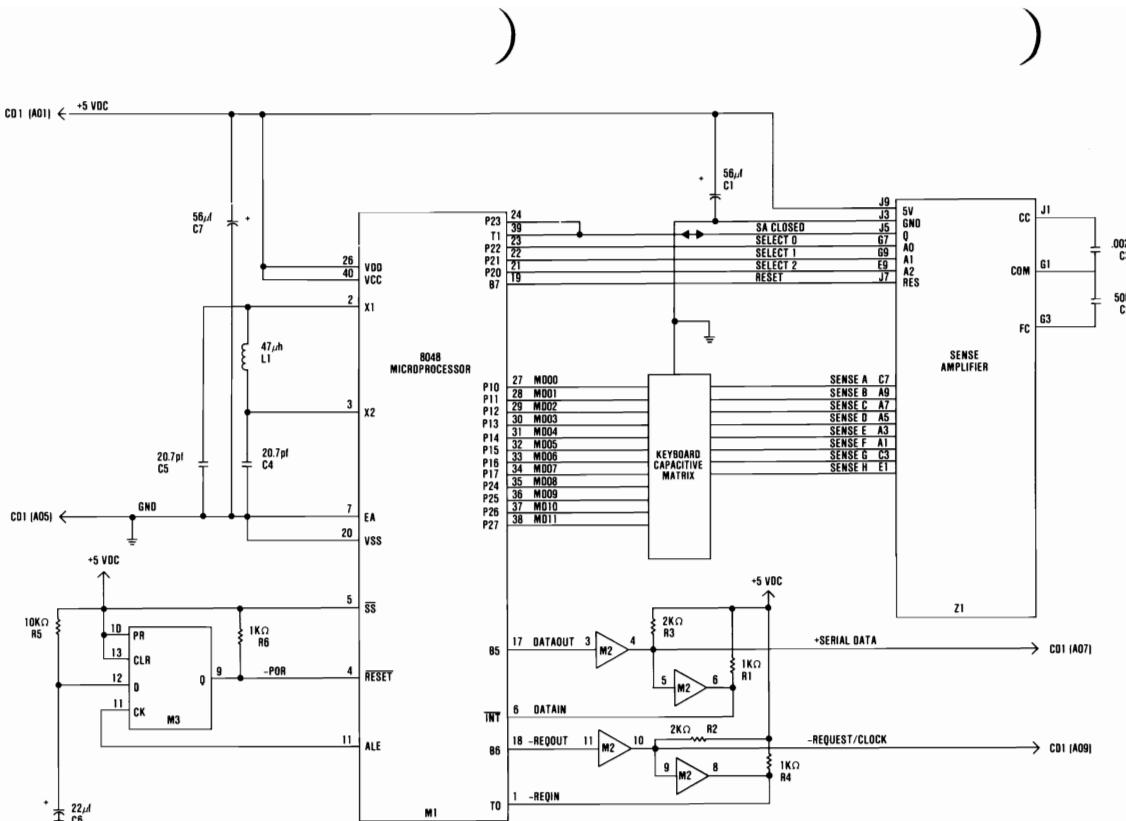
2

2

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

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

Keyboard Logic Diagram



83-Key Keyboard

SECTION 4

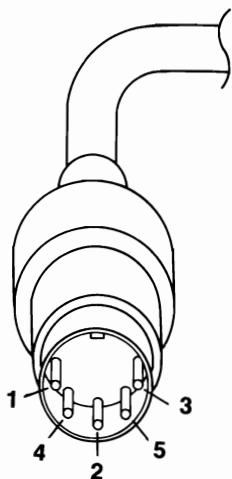
101/102-Key Keyboard

Description

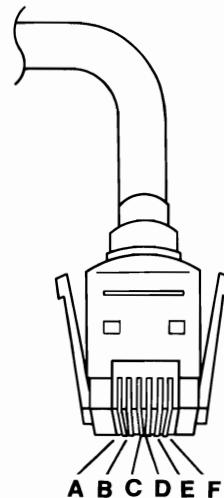
The keyboard has 101 keys (102 in countries outside the U. S.). At system power-on, the keyboard monitors the signals on the 'clock' and 'data' lines and establishes its line protocol.

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	
5	B	+5.0 Vdc	Power
Shield	F	Not used	Power
	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-overrun condition occurs when more than 16 bytes are placed in the keyboard buffer. An overrun code replaces the 17th byte. If more keys are pressed before the system allows keyboard output, the additional data is lost.

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

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

Keys

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

Additionally, except for the Pause key, all keys are *typematic*. When a key is pressed and held down, the keyboard sends the make code for that key, delays 500 milliseconds \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.

Overrun (Hex FF)

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

Keyboard Scan Codes

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

Scan Code Tables

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

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

* 101-key keyboard only.

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

** 102-key keyboard only.

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,

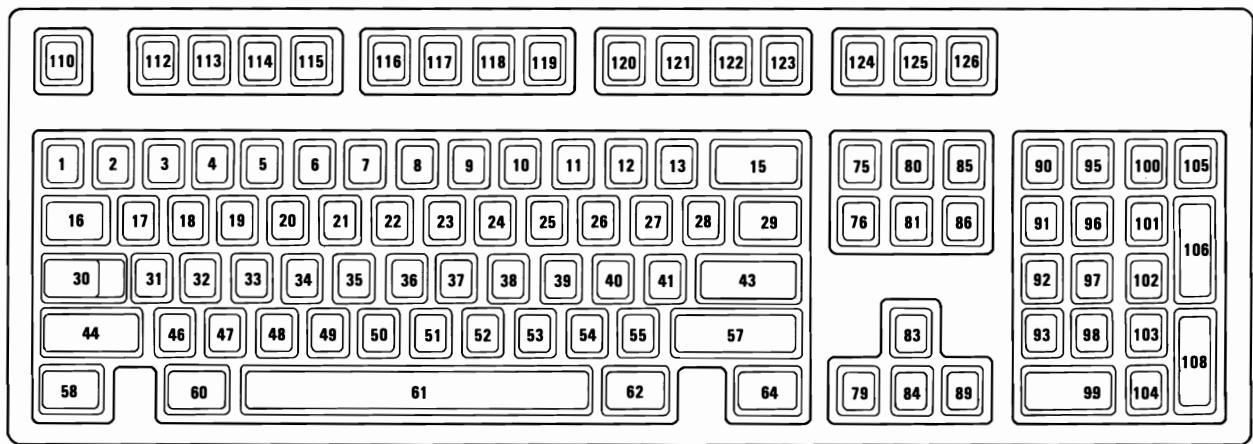
which resides in the read/write memory. This section discusses only the ROM routine.

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	@	Nul(000) (*)	(*)
4	3	#	-1	(*)
5	4	\$	-1	(*)
6	5	%	-1	(*)
7	6	^	RS(030)	(*)
8	7	&	-1	(*)
9	8	(-1	(*)
10	9)	-1	(*)
11	0		-1	(*)
12	-		US(031)	(*)
13	=		-1	(*)
15	Backspace (008)	Backspace (008)	Del(127)	(*)
16	→ (009)	← (*)	(*)	(*)
17	q	Q	DC1(017)	(*)
18	w	W	ETB(023)	(*)
19	e	E	ENQ(005)	(*)
20	r	R	DC2(018)	(*)
21	t	T	DC4(020)	(*)
22	y	Y	EM(025)	(*)
23	u	U	NAK(021)	(*)
24	i	I	HT(009)	(*)
25	o	O	SI(015)	(*)
26	p	P	DLE(016)	(*)
27	{	{	Esc(027)	(*)
28	}	}	GS(029)	(*)
29	\		FS(028)	(*)
30	Caps Lock	-1	-1	-1
31	a	A	SOH(001)	(*)
32	s	S	DC3(019)	(*)
33	d	D	EOT(004)	(*)
34	f	F	ACK(006)	(*)
35	g	G	BEL(007)	(*)
36	h	H	BS(008)	(*)
37	j	J	LF(010)	(*)
38	k	K	VT(011)	(*)
39	l	L	FF(012)	(*)
40	;	:	-1	(*)
41	,	,	-1	(*)
43	CR	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	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	Enter	LF(010)	(*)
110 Esc	Esc	Esc	Esc	(*)
112 Null (*)	Null (*)	Null (*)	Null (*)	Null(*)
113 Null (*)	Null (*)	Null (*)	Null (*)	Null(*)
114 Null (*)	Null (*)	Null (*)	Null (*)	Null(*)
115 Null (*)	Null (*)	Null (*)	Null (*)	Null(*)
116 Null (*)	Null (*)	Null (*)	Null (*)	Null(*)
117 Null (*)	Null (*)	Null (*)	Null (*)	Null(*)
118 Null (*)	Null (*)	Null (*)	Null (*)	Null(*)
119 Null (*)	Null (*)	Null (*)	Null (*)	Null(*)
120 Null (*)	Null (*)	Null (*)	Null (*)	Null(*)
121 Null (*)	Null (*)	Null (*)	Null (*)	Null(*)
122 Null (*)	Null (*)	Null (*)	Null (*)	Null(*)
123 Null (*)	Null (*)	Null (*)	Null (*)	Null(*)
125 Scroll Lock	-1	-1	-1	-1
126 Pause(**)	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.

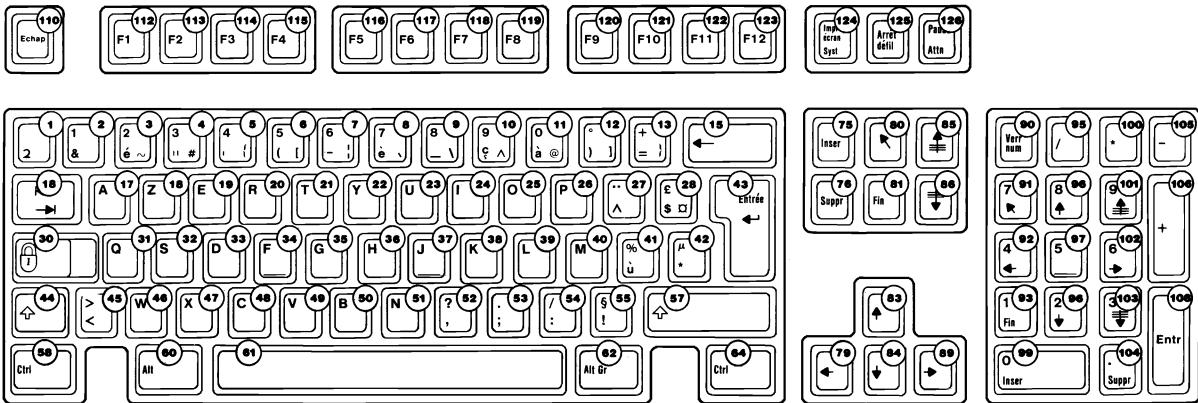
Keyboard Layouts

The keyboard is available in six layouts:

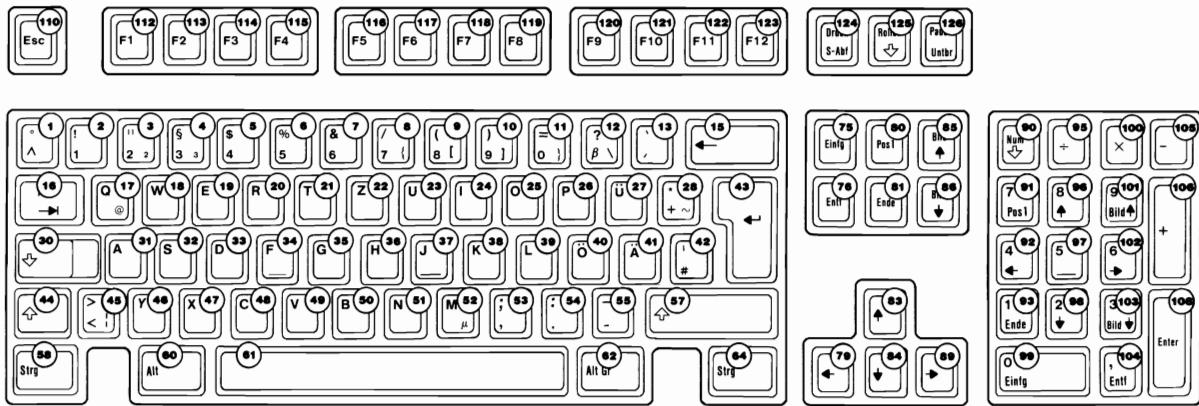
- French
- German
- Italian
- Spanish
- UK English
- US English

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.

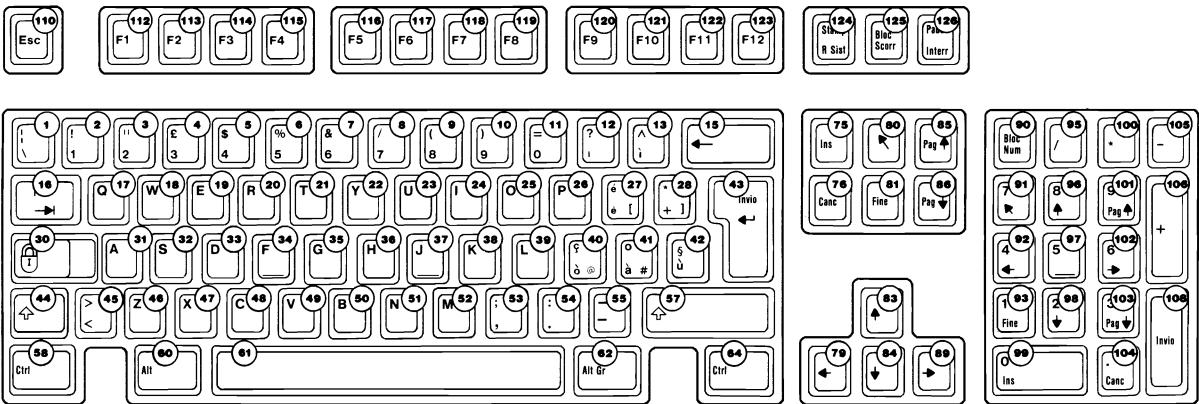
French Keyboard



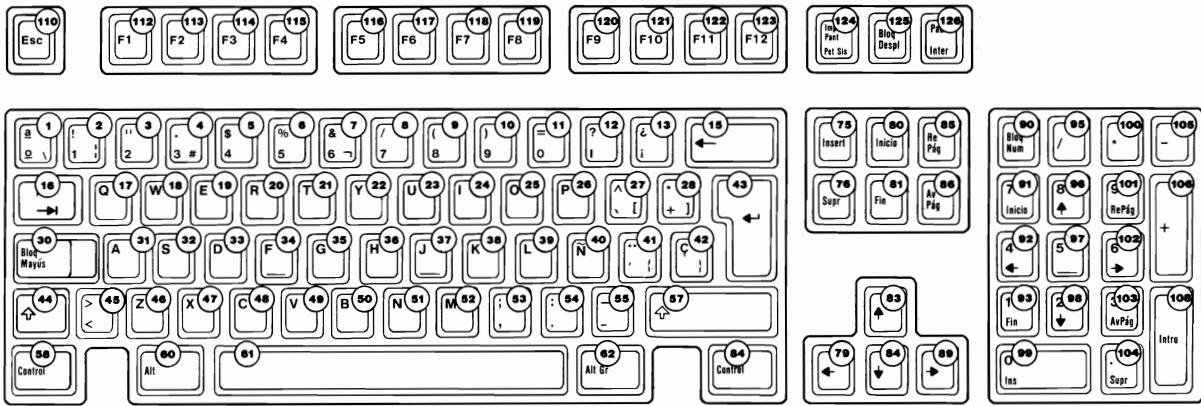
German Keyboard



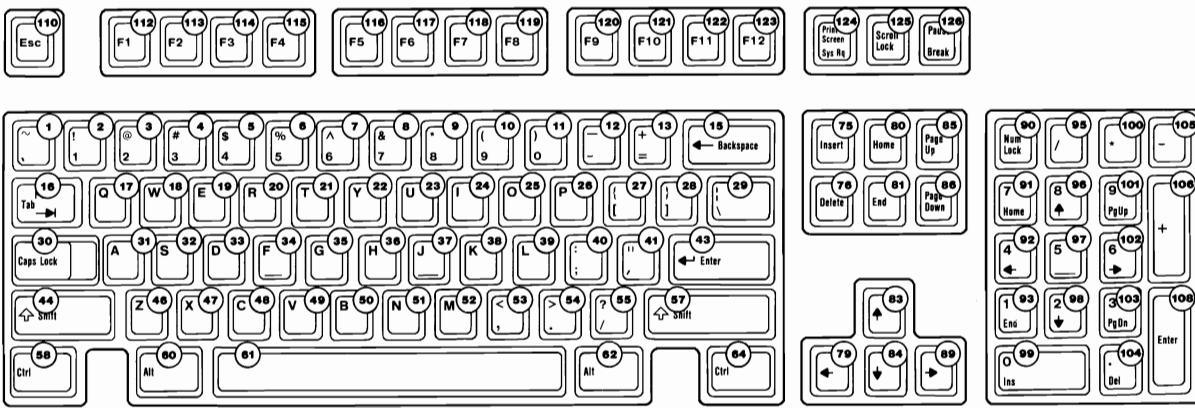
Italian Keyboard



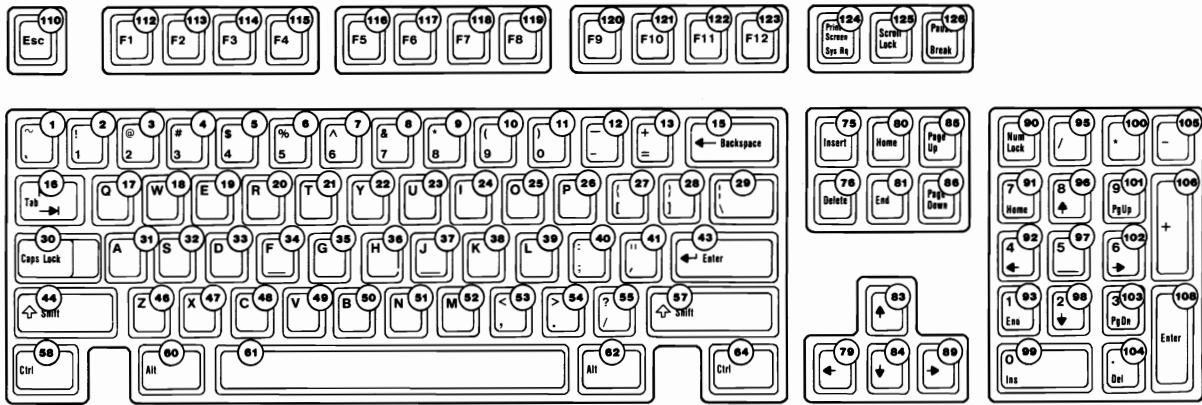
Spanish Keyboard



UK English Keyboard



US English Keyboard



Specifications

The specifications for the keyboard follow.

Power Requirements

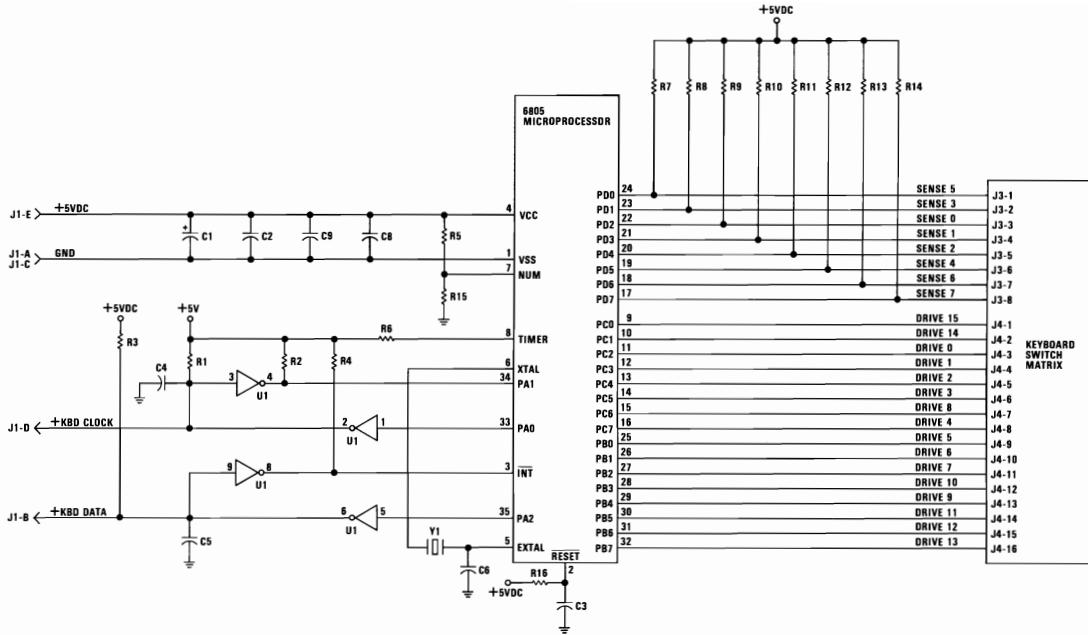
- +5 Vdc \pm 10%
- Current cannot exceed 275 mA.

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)



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_10
11	44-47	Equipment Check	EQUIPMENT
12	48-4B	Memory	MEMORY_SIZE DETERMINE
13	4C-4F	Diskette	DISKETTE_10
14	50-53	Communications	RS232_10
15	54-57	Cassette	CASSETTE_10
16	58-5B	Keyboard	KEYBOARD_10
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		Reserved for DOS and BASIC
500	DOS	Print screen status flag store 0=Print screen not active or successful print screen operation 1=Print screen in progress 255=Error encountered during print screen operation
504	DOS	Single drive mode status byte
510-511	BASIC	BASIC's segment address store
512-515	BASIC	Clock interrupt vector segment:offset store
516-519	BASIC	Break key interrupt vector segment:offset store
51A-51D	BASIC	Disk error interrupt vector segment:offset store

Reserved Memory Locations

If you do DEF SEG (Default workspace segment):

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

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

Basic Workspace Variables

Example

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

L	H
Hex 64	Hex 00

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

BIOS Memory Map

BIOS Programming Hints

The BIOS code is invoked through software interrupts. The programmer should not “hard code” BIOS addresses into application programs. The internal workings and absolute addresses within BIOS are subject to change without notice.

If an error is reported by the disk or diskette code, you should reset the drive adapter and retry the operation. A specified number of retries should be required on diskette reads to ensure the problem is not due to motor startup.

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

Adapter Cards with System-Accessible ROM Modules

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

The absolute addresses hex C8000 through hex F4000 are scanned in 2K blocks in search of a valid adapter card ROM. A valid ROM is defined as follows:

- Byte 0:** Hex 55
- Byte 1:** Hex AA
- Byte 2:** A length indicator representing the number of 512-byte blocks in the ROM (length/512). A checksum is also done to test the integrity of the ROM module. Each byte in the defined ROM is summed modulo hex 100. This sum must be 0 for the module to be deemed valid.

When the POST identifies a valid ROM, it does a far call to byte 3 of the ROM (which should be executable code). The adapter card may now perform its power-on initialization tasks. The feature ROM should return control to the BIOS routines by executing a far return.

System BIOS Listing - 01/10/86

Quick Reference - 256/640K Board

Map	5-13
Header	5-14
EQUATES	5-15
ABS0	5-19
DATA Segment	5-20
Diskette	5-23
INT 13H	5-23
Drive Type	5-25
Diskette_IO_1	5-25
DMA_Setup	5-36
Motor_On	5-40
Disk_Int	5-44
Diskette_Setup	5-45
Keyboard BIOS	5-46
Scan Tables	5-56
Printer BIOS	5-57
RS232 BIOS	5-59
Video BIOS	5-62
BIOS1	5-80
INT 15H	5-80
Joystick Support	5-82
POST	5-84
Determine Configuration	5-87
8259 Test	5-89
Keyboard Test	5-90
Expansion Test	5-91
Boot_Strip (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	Address	Publics by Value
F0001:EF29	A1	F0001:0000	HEADER
F0001:15CC	ACT_DISP_PAGE	F0001:0062	DISKETTE_IO_1
F0001:6000	BASIC	F0001:0A40	NEC_OUTPUT
F0001:EC5C	BEEP	F0001:0A46	SEEK
F0001:1E4F	CASSETTE_IO_1	F0001:0B25	RESULTS
F0001:E73C	CONF_TBL	F0001:0BC4	DISPINT_1
F0001:FA6E	CRT_CHAR_GEN	F0001:0B0B	DISKETTE_SETUP
F0001:FA12	DDS	F0001:0C57	KEYBOARD_IO_1
F0001:0062	DISKETTE_IO_1	F0001:0D78	KB_INT_1
F0001:EF77	DISK_BASE	F0001:12B2	PRINTER_IO_1
F0001:1E54	DISK_INT_1	F0001:1344	R5232_IO_1
F0001:0B0B	DISKETTE_SETUP	F0001:140E	VIDEOPARM_1
F0001:1D37	FILL	F0001:1485	SET_MODE
F0001:0000	HEADER	F0001:1563	VIDEO_RETURN
F0001:0D78	KB_INT_1	F0001:156C	SET_CTYPE
F0001:0C57	KEYBOARD_IO_1	F0001:158D	SET_CPOS
F0001:1E54	M5	F0001:15B5	READ_CURSOR
F0001:F0EC	M6	F0001:16C4	ACT_DISP_PAGE
F0001:F0F4	M7	F0001:16EE	SET_CURSOR
F0001:EF79	MD_TBL_1	F0001:1614	VIDEO_STATE
F0001:EF86	MD_TBL_2	F0001:1635	SCROLL_UP
F0001:EF93	MD_TBL_3	F0001:16D3	SCROLL_DOWN
F0001:1E40	MD_TBL_4	F0001:1725	READ_AC_CURRENT
F0001:EFAD	MD_TBL_5	F0001:1782	WRITE_AC_CURRENT
F0001:EF8A	MD_TBL_6	F0001:1784	WRITING_C_CURRENT
F0001:0A40	NEC_OUTPUT	F0001:17A4	WRITE_STRING
F0001:12B8	PRINTER_IO_1	F0001:17E1	WRITE_STRING
F0001:FFF0	P_O_R	F0001:1865	READ_DOT
F0001:1E5C	READ_AC_CURRENT	F0001:1876	WRITE_DOT
F0001:15B5	READ_CURSOR	F0001:1B24	WRITE_TTY
F0001:15B5	READ_DOT	F0001:1B4B	READ_CURSOR
F0001:1865	READ_DOT	F0001:1B4F	CASSETTE_IO_1
F0001:1BAB	READ_LPEN	F0001:1D37	FILL
F0001:E05B	RESET	F0001:1600	BASIC
F0001:0B32	RESULTS	F0001:E05B	RESET
F0001:1E44	RS232C_IO_1	F0001:EF29	A1
F0001:1603	SCROLL_DOWN	F0001:EF3C	CONF_TBL
F0001:1635	SCROLL_UP	F0001:EC5C	BEEP
F0001:0A64	SEEK	F0001:EC80	WAITF
F0001:15EE	SET_COLOR	F0001:EF79	MD_TBL_1
F0001:158D	SET_CPOS	F0001:EF84	MD_TBL_2
F0001:1E5C	SET_MODE	F0001:EF93	MD_TBL_3
F0001:1485	SET_MODE	F0001:EFAD	MD_TBL_4
F0001:144E	VIDEO_IO_1	F0001:EF04	MD_TBL_5
F0001:F0A4	VIDEO_PARMS	F0001:EFBA	MD_TBL_6
F0001:1563	VIDEO_RETURN	F0001:EF77	DISK_BASE
F0001:1614	VIDEO_STATE	F0001:F0A4	VIDEO_PARMS
F0001:1700	WAITF	F0001:FOE4	M5
F0001:1782	WRITE_AC_CURRENT	F0001:FOEC	M6
F0001:1784	WRITE_C_CURRENT	F0001:FOF4	M7
F0001:1876	WRITE_DOT	F0001:FA12	DDS
F0001:17E1	WRITE_TTY	F0001:FA6E	CRT_CHAR_GEN
F0001:1B24	WRITE_TTY	F0001:FFF0	P_O_R

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

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65

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_I_O_1 - INT 13H BIOS ENTRY (40H) -INT 13H
DISKETTE_I_O_1 - HARDWARE INTERRUPT HANDLER -INT OEM
DSKETTE_SETUP - POST SETUP DRIVE TYPES

KEYBRD.ASM --> KEYBOARD BIOS
KEYBOARD_I_O_1 - INT 16H BIOS ENTRY -INT 16H
KB_INT_1 - HARDWARE INTERRUPT -INT 09H
SND_DATA - KEYBOARD TRANSMISSION

PRT.ASM --> PRINTER ADAPTER BIOS -INT 17H

RS232.ASM --> COMMUNICATIONS BIOS FOR RS232 -INT 14H

VIDEO.ASM --> VIDEO BIOS -INT 10H

BIOS1.ASM --> INTERRUPT 15H BIOS ROUTINES -INT 15H
DEV_OPEN - NULL DEVICE OPEN HANDLER
DEV_CLOSE - NULL DEVICE CLOSE HANDLER
PORT_TERM - NULL PORT TERMINATION
JOY_STICK - JOYSTICK PORT HANDLER
SYS_REQ - NULL SYSTEM REQUEST KEY
EXT_MEMORY - EXTENDED MEMORY SIZE DETERMINE
DEVTC_E_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_SCREEN - PRINT SCREEN ROUTINE -INT 05H
TIMER_INT_1 - TIMER INTERRUPT HANDLER -> INT 1CH
DDS - LOAD DDS INTO 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 SYSTEM      EQU    0          ; 0 PC-XT, 1 PC-AT
72          C = 0000      EQU    0FBH     ; SYSTEM MODEL BYTE
73          C = 00FB      EQU    000H     ; SYSTEM SUB-MODEL TYPE
74          C = 0000      EQU    001H     ; BIOS LEVEL
75          C = 0001      EQU    ENDIF
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 = 0060      EQU    061H     ; PORT B READ/WRITE DIAGNOSTIC REGISTER
81          C = 0061      EQU    061H     ; PORT B READ/WRITE DIAGNOSTIC REGISTER
82          C = 0062      EQU    062H     ; 8255 PORT C ADDR
83          C = 0063      EQU    063H     ;
84          C |
85          C KB_DATA     EQU    60H      ; KEYBOARD SCAN CODE PORT
86          C KB_CTL      EQU    61H      ; CONTROL BITS FOR KEYBOARD SENSE DATA
87          C = 0054      EQU    054H     ; ALTERNATE 2ND ID CHAR FOR KBX
88          C = 00E0      EQU    224      ; GENERAL MARKER CODE
89          C = 00E1      EQU    225      ; PAUSE KEY MARKER CODE
90          C |
91          C -----
92          C RAM_PAR_ON   EQU    11110011B ; AND MASK FOR PARITY CHECKING ENABLE ON
93          C RAM_PAR_OFF  EQU    00001100B ; OR MASK FOR PARITY CHECKING ENABLE OFF
94          C = 00C0      EQU    11000000B ; R/W MEMORY - I/O CHANNEL PARITY ERROR
95          C = 0011      EQU    00000000B ; TIMER 1 INPUT GATE CLOCK BIT
96          C = 0002      EQU    00000000B ; SPEAKER OUTPUT DATA ENABLE BIT
97          C = 0010      EQU    00010000B ; REFRESH TEST BIT
98          C = 0020      EQU    00100000B ; SPEAKER TIMER OUT2 INPUT BIT
99          C = 0040      EQU    01000000B ; I/O (MEMORY) CHECK OCCURRED BIT MASK
100         C = 0080     EQU    10000000B ; MEMORY PARITY CHECK OCCURRED BIT MASK
101         C ENDIF
```

```
102      C PAGE
103      C ;----- KEYBOARD RESPONSE -----
104      C KB_OK      EQU  OAAH          ; RESPONSE FROM SELF DIAGNOSTIC
105      C KB_NACK    EQU  OFAH          ; ACKNOWLEDGE FROM TRANSMISSION
106      C KB resend   EQU  OFEH          ; RESEND REQUEST
107      C KB_over_run EQU  OFFH          ; OVER RUN SCAN CODE
108
109      C ;----- FLAG EQUATES WITHIN *KB_FLAG -----
110      C RIGHT_SHIFT EQU  00000001B ; RIGHT SHIFT KEY DEPRESSED
111      C LFT_SHIFT   EQU  00000002B ; LEFT SHIFT KEY DEPRESSED
112      C CTL_SHIFT   EQU  00000004B ; CONTROL SHIFT KEY DEPRESSED
113      C ALT_SHIFT   EQU  00001000B ; ALTERNATE SHIFT KEY DEPRESSED
114      C SCROLL_STATE EQU  00010000B ; SCROLL LOCK STATE HAS BEEN TOGGLED
115      C NUM_STATE    EQU  00100000B ; NUM LOCK STATE HAS BEEN TOGGLED
116      C CAPS_STATE   EQU  01000000B ; CAPS LOCK STATE HAS BEEN TOGGLED
117      C INS_STATE    EQU  10000000B ; INSERT STATE IS ACTIVE
118
119      C ;----- FLAG EQUATES WITHIN *KB_FLAG_1 -----
120      C SYS_SHIFT   EQU  00000100B ; SYSTEM KEY DEPRESSED AND HELD
121      C HOLD_STATE  EQU  00001000B ; SUSPEND KEY HAS BEEN TOGGLED
122      C SCROLL_SHIFT EQU  00001000B ; SCROLL LOCK KEY IS DEPRESSED
123      C NC_SHIFT    EQU  00100000B ; NUM LOCK KEY IS DEPRESSED
124      C CAPS_SHIFT  EQU  01000000B ; CAPS LOCK KEY IS DEPRESSED
125      C INS_SHIFT   EQU  10000000B ; INSERT KEY IS DEPRESSED
126
127      C ;----- FLAGS EQUATES WITHIN *KB_FLAG_2 -----
128      C KB.LEDS     EQU  00000111B ; KEYBOARD LED STATE BITS
129      C ;           1: RESERVED (MUST BE ZERO)
130      C ;           2: ACROSS EDGE KEY RECEIVED
131      C KB_FA       EQU  00000008B ; KB FA
132      C KB_FE       EQU  00100000B ; KB FE
133      C KB_PR_LED   EQU  01000000B ; KB PR LED
134      C KB_ERR      EQU  10000000B ; KB ERR
135
136      C ;----- FLAGS EQUATES WITHIN *KB_FLAG_3 -----
137      C LC_E1       EQU  00000001B ; LAST CODE WAS THE E1 HIDDEN CODE
138      C LC_E0       EQU  000000010B ; LAST CODE WAS THE E0 HIDDEN CODE
139      C R_CTL_SHIFT EQU  00000100B ; RIGHT CTL KEY DOWN
140      C ;           1: RESERVED (MUST BE ZERO)
141      C ;           2: KB INSTALLED
142      C SET_NUM_LK  EQU  00010000B ; FORCED NUMBER LOCK IF READ ID AND KBX
143      C LC_AB       EQU  01000000B ; LAST CHARACTER WAS FIRST ID CHARACTER
144      C RD_ID       EQU  10000000B ; DOING A READ ID (MUST BE BIT0)
145
146      C ;----- KEYBOARD SCAN CODES -----
147      C ID_1        EQU  0ABH          ; 1ST ID CHARACTER FOR KBX
148      C ID_2        EQU  0A1H          ; 2ND ID CHARACTER FOR KBX
149      C ;           1: SCAN CODE FOR ALTERNATE SHIFT KEY
150      C ALT_KEY     EQU  56
151      C CTL_KEY     EQU  29          ; SCAN CODE FOR CONTROL KEY
152      C CAPS_KEY   EQU  58          ; SCAN CODE FOR SHIFT LOCK KEY
153      C DEL_KEY     EQU  83          ; SCAN CODE FOR DELETE KEY
154      C BSKEY       EQU  62          ; SCAN CODE FOR INSERT KEY
155      C INS_KEY     EQU  62          ; SCAN CODE FOR INSERT KEY
156      C LFT_SHIFT   EQU  42          ; SCAN CODE FOR LEFT SHIFT
157      C NUM_KEY     EQU  69          ; SCAN CODE FOR NUMBER LOCK KEY
158      C RIGHT_KEY   EQU  54          ; SCAN CODE FOR RIGHT SHIFT
159      C SCROLL_KEY  EQU  70          ; SCAN CODE FOR SCROLL LOCK KEY
160      C SYS_KEY     EQU  84          ; SCAN CODE FOR SYSTEM KEY
161      C F11_M       EQU  87          ; F11 KEY MAKE
162      C F12_M       EQU  88          ; F12 KEY MAKE
```

```

161          C PAGE
162          C      ENDIF
163
164          C----- DISKETTE EQUATES -----
165      = 0050  C CARD_ID     EQU 0100000B ; CONTROLLER CARD I.D. BIT
166      = 0001  C DUAL        EQU 00000001B ; MASK FOR FDC ADAPTER I.D.
167      = 0080  C INT_FLAG    EQU 10000000B ; INTERRUPT OCCURRENCE FLAG
168      = 0080  C DSK_CHG     EQU 10000000B ; DISKETTE CHANGE FLAG MASK BIT
169      = 0010  C DETERMINED  EQU 00000000B ; SET STATE DETERMINED IN STATE BITS
170      = 0010  C NONE        EQU 00000000B ; TRACER
171      = 0004  C SENSE_DRY_ST EQU 00000000B ; SENSE DRIVE STATUS COMMAND
172      = 0030  C TRK_SLAP    EQU 030H   ; CRASH STOP (48 TPI DRIVES)
173      = 000A  C QUIET_SEEK EQU 00AH   ; SEEK TO TRACK 10
174      = 0002  C MAX_DRY    EQU 2      ; MAX NUMBER OF DRIVES
175      = 000F  C H12_SETTLE EQU 15     ; 1.2 M HEAD SETTLE TIME
176      = 0004  C H32_SETTLE  EQU 20     ; 320 K HEAD SETTLE TIME
177      = 0025  C MOTOR_WAIT EQU 3T    ; 1/2 SECONDS OF COUNTS FOR MOTOR TURN OFF
178
179          C----- DISKETTE ERRORS -----
180      = 0080  C TIME_OUT    EQU 080H ; ATTACHMENT FAILED TO RESPOND
181      = 0040  C BAD_SEEK   EQU 040H ; SEEK OPERATION FAILED
182      = 0000  C BAD_DUL    EQU 020H ; DUAL ATTACH CONTROLLER HAS FAILED
183      = 0010  C BAD_CRC    EQU 010H ; BAD CRC ON DISKETTE READ
184      = 000C  C MED_NOT_FND EQU 00CH ; MEDIA TYPE FOUND
185      = 0009  C DMA_BOUNDARY EQU 009H ; ATTEMPT TO DMA ACROSS 64K BOUNDARY
186      = 0008  C BAD_DMA    EQU 008H ; DMA OVERRUN ON OPERATION
187      = 0016  C MED_CHANGE  EQU 005H ; MEDIA REMOVED ON DUAL ATTACH CARD
188      = 0004  C RECORD_NOT_FND EQU 004H ; RECORD NOT FOUND
189      = 0003  C WRITE_PROTECT EQU 003H ; WRITE ATTEMPTED ON WRITE PROTECT DISK
190      = 0002  C BAD_ADDR_MARK EQU 002H ; ADDRESS MARK NOT FOUND
191      = 0001  C BAD_CMD    EQU 001H ; BAD COMMAND PASSED TO DISKETTE 1/0
192
193          C----- DISK CHANGE LINE EQUATES -----
194      = 0001  C NOCHGLN   EQU 001H ; NO DISK CHANGE LINE AVAILABLE
195      = 0002  C CHGLN     EQU 002H ; DISK CHANGE LINE AVAILABLE
196
197          C----- MEDIA/DRIVE STATE INDICATORS -----
198      = 0001  C TRC_CAPA   EQU 00000001B ; 80 TRACK CAPABILITY
199      = 0002  C FMT_CAPA   EQU 000000010B ; MULTIPLE FORMAT CAPABILITY (1.2M)
200      = 0014  C DMV_DET    EQU 000000000B ; DMV DETERMINED
201      = 0010  C MED_DET    EQU 000000000B ; MEDIA DETERMINED
202      = 0020  C DBL_STEP   EQU 001000000B ; DOUBLE STEP BIT
203      = 00C0  C RATE_MSK   EQU 110000000B ; MASK FOR CLEARING ALL BUT RATE
204      = 0000  C RATE_500   EQU 000000000B ; 500 KBS DATA RATE
205      = 0040  C RATE_300   EQU 010000000B ; 300 KBS DATA RATE
206      = 0000  C RATE_250   EQU 100000000B ; 250 KBS DATA RATE
207      = 000C  C STRT_MSK   EQU 00001100B ; INITIATION START RATE MASK
208      = 00C0  C SEND_MSK   EQU 110000000B ; MASK FOR SEND RATE BITS
209
210          C----- MEDIA/DRIVE STATE INDICATORS COMPATIBILITY -----
211      = 0000  C MID360    EQU 00000000B ; 360 MEDIA/DRIVE NOT ESTABLISHED
212      = 0001  C M360U    EQU 00000001B ; 360 MEDIA/DRIVE NOT ESTABLISHED
213      = 0002  C MID120    EQU 000000010B ; 1.2 MEDIA/DRIVE NOT ESTABLISHED
214      = 0007  C MED_UNK   EQU 0000011B ; NONE OF THE ABOVE

```

215
216
217 = 0020 C PAGE
218 = 0020 C ----- INTERRUPT EQUATES -----
219 = 0011 C EO1 EQU 020H ; END OF INTERRUPT COMMAND TO 8259
220 = 0040 C INTA00 EQU 020H ; 8259 PORT
221 = 0041 C INTB01 EQU 021H ; 8259 PORT
222 = 0041 C INTB01 EQU 00AH ; 2ND 8259
223 = 0010 C INT_TYPE EQU 070H ; START OF 8259 INTERRUPT TABLE LOCATION
224 = 0010 C INT_VIDEO EQU 010H ; VIDEO VECTOR
225 = 0008 C :-----
226 = 0000 C DMA08 EQU 008H ; DMA STATUS REGISTER PORT ADDRESS
227 = 0000 C DMA10 EQU 000H ; DMA CH.0 ADDRESS REGISTER PORT ADDRESS
228 = 00C0 C DMA18 EQU 000H ; 2ND DMA STATUS PORT ADDRESS
229 = 00C0 C DMA1 EQU 0C0H ; 2ND DMA CH.0 ADDRESS REGISTER ADDRESS
230 = 0040 C :-----
231 = 0043 C TIMER EQU 040H ; 8253 TIMER - BASE ADDRESS
232 = 0040 C TIM_CTL EQU 043H ; 8253 TIMER CONTROL PORT ADDR
233 = 0040 C TIMERD EQU 040H ; 8253 TIMER/CNTER 0 PORT ADDR
234 = 0000 C :----- MANUFACTURING PORT -----
235 = 0080 C MFG_PORT EQU 80H ; MANUFACTURING AND POST CHECKPOINT PORT
236 = 0000 C :----- DMA CHANNEL 0 PAGE REGISTER ADDRESS -----
237 = 0000 C :----- MANUFACTURING BIT DEFINITION FOR @MFG_ERR_FLAG+1 -----
238 = 0001 C MEM_FAIL EQU 00000001B ; STORAGE TEST FAILED (ERROR 20X)
239 = 0002 C PRO_FAIL EQU 000000010B ; VIRTUAL MODE TEST FAILED (ERROR 104)
240 = 0004 C LMCS_FAIL EQU 00000100B ; LOW MEG CHIP SELECT FAILED (ERROR 109)
241 = 0008 C KYCLK_FAIL EQU 00000000B ; KEYBOARD CLOCK TEST FAILED (ERROR 304)
242 = 0010 C KY_SYS_FAIL EQU 00010000B ; KEYBOARD OR SYSTEM FAILED (ERROR 303)
243 = 0020 C KY_BD_FAIL EQU 00000000B ; KEYBOARD FAILED (ERROR 301)
244 = 0040 C DSK_FAIL EQU 01000000B ; DISKETTE TEST FAILED (ERROR 601)
245 = 0080 C KEY_FAIL EQU 10000000B ; KEYBOARD LOCKED (ERROR 302)
246 = 0000 C :-----
247 = 0081 C DMA_PAGE EQU 081H ; START OF DMA PAGE REGISTERS
248 = 008F C LAST_DMA_PAGE EQU 08FH ; LAST DMA PAGE REGISTER
249 = 0000 C :-----
250 = 0000 C :----- i287 -----
251 = 0000 C i287 EQU 0F0H ; MATH COPROCESSOR CONTROL PORT
252 = 0000 C :-----
253 = 0000 C POST_SS EQU 00000H ; POST STACK SEGMENT
254 = 0000 C POST_SP EQU 00000H ; POST STACK PTR
255 = 0000 C STACK_SS EQU 30H ; STACK SEGMENT USED DURING POST
256 = 0030 C TOS EQU 100H ; STACK -- USED DURING POST ONLY
257 = 0100 C :----- USE WILL OVERLAY INTERRUPTS VECTORS -----
258 = 0000 C :-----
259 = 0000 C CR EQU 000DH ; CARRIAGE RETURN CHARACTER
260 = 000A C LF EQU 000AH ; LINE FEED CHARACTER
261 = 0008 C RVRT EQU 00001000B ; VIDEO VERTICAL RETRACE BIT
262 = 0001 C RHRZ EQU 00000001B ; VIDEO HORIZONTAL RETRACE BIT
263 = 0000 C H EQU 256 ; HIGH BYTE FACTOR (X 00H)
264 = 0101 C X EQU H+1 ; HIGH AND LOW BYTE FACTOR (X 10H)
265 = 0000 C :-----
266 = 0000 C :-----
267 = 0000 C :-----
268 = 0000 C :-----
269 = 0000 C :-----
270 = 0000 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      C ABS0      SEGMENT AT 0      ; ADDRESS= 0000:0000
282 0000 ??    C *STG_LOCO   DB   ?      ; START OF INTERRUPT VECTOR TABLE
283
284
285 0008      C ORG 4*002H
286 0008 ??????? C *NMI_PTR   DD   ?      ; NON-MASKABLE INTERRUPT VECTOR
287
288 0014      C *INT5_PTR  ORG 4*005H
289 0014 ??????? C *INT_PTR   DD   ?      ; PRINT SCREEN INTERRUPT VECTOR
290
291 0020      C ORG 4*008H
292 0020 ??????? C *INT_PTR   DD   ?      ; HARDWARE INTERRUPT POINTER (8-F)
293
294 0040      C *VIDEO_INT ORG 4*010H
295 0040 ??????? C DD   ?
296
297 004C      C *ORG_VECTOR ORG 4*013H
298 004C ??????? C DD   ?      ; DISKETTE/DISK INTERRUPT VECTOR
299
300 0060      C *BASIC_PTR ORG 4*015H
301 0060 ??????? C DD   ?      ; POINTER TO CASSETTE BASIC
302
303 0074      C *PARM_PTR ORG 4*01DH
304 0074 ??????? C DD   ?      ; POINTER TO VIDEO PARAMETERS
305
306 0078      C *DISK_POINTER ORG 4*01EH
307 0078 ??????? C DD   ?      ; POINTER TO DISKETTE PARAMETER TABLE
308
309 007C      C *EXT_PTR   ORG 4*01FH
310 007C ??????? C DD   ?      ; POINTER TO GRAPHIC CHARACTERS 128-255
311
312 0100      C *DISK_VECTOR ORG 4*040H
313 0100 ??????? C DD   ?      ; POINTER TO DISKETTE INTERRUPT CODE
314
315 0104      C *HF_TBL_VEC ORG 4*041H
316 0104 ??????? C DD   ?      ; POINTER TO FIRST DISK PARAMETER TABLE
317
318 0118      C *HF1_TBL_VEC ORG 4*046H
319 0118 ??????? C DD   ?      ; POINTER TO SECOND DISK PARAMETER TABLE
320
321 01C0      C *SLAVE_INT_PTR ORG 4*070H
322 01C0 ??????? C DD   ?      ; POINTER TO SLAVE INTERRUPT HANDLER
323
324 01D8      C *HDISK_INT  ORG 4*076H
325 01D8 ??????? C DD   ?      ; POINTER TO FIXED DISK INTERRUPT CODE
326
327 0400      C ORG 400H
328 0400      C DATA_AREA  LABEL BYTE
329 0400      C DATA_WORD   LABEL WORD
330
331 0500      C *MFG_TEST_RTN ORG 0500H
332 0500      C LABEL FAR
333
334 TC00      C *BOOT_LOCN  ORG TCOOH
335 TC00      C LABEL FAR
336
337 7C00      C ABS0      ENDS

```

```

338 PAGE
339
340 C:----- ROM BIOS DATA AREAS -----
341
342 C:----- DATA SEGMENT AT 40H -----
343 0000 DATA SEGMENT AT 40H : ADDRESS= 0040:0000
344 0000 ???? : ADDRESS= 0040:0000
345 0000 ???? : BASE ADDRESSES OF RS232 ADAPTERS
346 0002 ???? : #RS232_BASE DW ? : SECOND LOGICAL RS232 ADAPTER
347 0004 ???? : DW ? : RESERVED
348 0006 ???? : DW ? : RESERVED
349 0008 ???? : @PRINTER_BASE DW ? : BASE ADDRESSES OF PRINTER ADAPTERS
350 000A ???? : DW ? : SECOND LOGICAL PRINTER ADAPTER
351 000C ???? : DW ? : THIRD LOGICAL PRINTER ADAPTER
352 000E ???? : DW ? : RESERVED
353 0010 ???? : @EQUIP_FLAG DW ? : INSTALLED HARDWARE FLAGS
354 0012 ???? : @MFG_TST DB ? : INITIALIZATION FLAGS
355 0013 ???? : @MEMORY_SIZE DW ? : BASE MEMORY SIZE IN K BYTES (X 1024)
356 0015 ???? : @MFG_ERR_FLAG DB ? : SCRATCHPAD FOR MANUFACTURING
357 0016 ???? : DB ? : ERROR CODES
358
359 C:----- KEYBOARD DATA AREAS -----
360
361
362
363 0017 ?? : @KB_FLAG DB ? : KEYBOARD SHIFT STATE AND STATUS FLAGS
364 0018 ?? : @KB_FLAG_1 DB ? : SECOND BYTE OF KEYBOARD STATUS
365 0019 ?? : @ALT_INPUT DB ? : STORAGE FOR ALTERNATE KEY PAD ENTRY
366 001A ???? : @BUFFER_HEAD DW ? : POINTER TO HEAD OF KEYBOARD BUFFER
367 001C ???? : @BUFFER_TAIL DW ? : POINTER TO TAIL OF KEYBOARD BUFFER
368
369 ;----- HEAD = TAIL INDICATES THAT THE BUFFER IS EMPTY
370 001E 10 [ ???? ] : @KB_BUFFER DW 16 DUP(?) : ROOM FOR 15 SCAN CODE ENTRIES
371
372
373
374
375
376 C:----- DISKETTE DATA AREAS -----
377
378
379 003E ?? : @SEEK_STATUS DB ? : DRIVE RECALIBRATION STATUS
380 ; BIT 0 = 0 DRIVE 3-0 RECALIBRATION
381 ; BEFORE NEXT SEEK IF BIT 15 = 0
382 003F ?? : @MOTOR_STATUS DB ? : MOTOR STATUS
383
384
385 0040 ?? : @MOTOR_COUNT DB ? : BIT 3-0 = DRIVE 3-0 CURRENTLY RUNNING
386 0041 ?? : @DISKETTE_STATUS DB ? : BIT 7 = CURRENT OPERATION IS A WRITE
387
388 0042 07 [ ???? ] : @NEC_STATUS DB 7 DUP(?) : TIME OUT COUNTER FOR MOTOR(S) TURN OFF
389
390
391
392
393
394 C:----- VIDEO DISPLAY DATA AREA -----
395
396
397 0049 ?? : @CRT_MODE DB ? : CURRENT DISPLAY MODE (TYPE)
398 004A ?? : @CRT_PAGES DB ? : NUMBER OF PAGES ON SCREEN
399 004C ?? : @CRT_LEN DW ? : LENGTH OF REGEN BUFFER IN BYTES
400 004E ?? : @CRT_START DW ? : STARTING ADDRESS IN REGEN BUFFER
401 0050 08 [ ???? ] : @CURSOR_POSN DW 8 DUP(?) : CURSOR FOR EACH OF UP TO 8 PAGES
402
403
404
405 0060 ?? : @CURSOR_MODE DW ? : CURRENT CURSOR MODE SETTING
406 0062 ?? : @ACTIVE_PAGE DB ? : CURRENT PAGE BEING DISPLAYED
407 0063 ?? : @ADDR 6845 DW ? : BASE ADDRESS FOR ACTIVE DISPLAY CARD
408 0065 ?? : @CRT_MODE_SET DB ? : CURRENT SETTING OF THE 3x8 REGISTER
409 0066 ?? : @CRT_PALETTE DB ? : CURRENT PALETTE SETTING - COLOR CARD
410
411
412 C:----- POST AND BIOS WORK DATA AREA -----
413
414
415
416 0067 ?? : @IO_ROM_INIT DW ? : STACK SAVE, ETC.
417 0069 ?? : @IO_ROM_SEG DW ? : POINTER TO ROM INITIALIZATION ROUTINE
418 006B ?? : @INTR_FLAG DB ? : POINTER TO I/O ROM SEGMENT
419
420
421 C:----- TIMER DATA AREA -----
422
423
424 006C ?? : @TIMER_LOW DW ? : LOW WORD OF TIMER COUNT
425 006E ?? : @TIMER_HIGH DW ? : HIGH WORD OF TIMER COUNT
426 0070 ?? : @TIMER_OFLOW DB ? : TIMER HAS ROLLED OVER SINCE LAST READ
427
428
429 C:----- SYSTEM DATA AREA -----
430
431
432 0071 ?? : @BIOS_BREAK DB ? : BIT 7=1 IF BREAK KEY HAS BEEN PRESSED
433 0072 ?? : @RESET_FLAG DW ? : WORD=1234H IF KEYBOARD RESET UNDERRUN
434
435
436 C:----- FIXED DISK DATA AREAS -----
437
438
439 0074 ?? : @DISK_STATUS1 DB ? : FIXED DISK STATUS
440 0075 ?? : @MF_NDM DB ? : COUNT OF FIXED DISK DRIVES
441 0076 ?? : @CONTROL_BYTE DB ? : HEAD CONTROL BYTE
442 0077 ?? : @PORT_OFF DB ? : RESERVED (PORT OFFSET)

```

IBM Personal Computer MACRO Assembler Version 2.00 I-8
 HEADER --- 01/08/86 POWER ON SELF TEST (POST) 01-10-86
 DSEG.INC - DATA SEGMENTS

```

443      PAGE
444
445      :-----+
446      : TIME-OUT VARIABLES
447      :-----+
448      PRINT_TIM_OUT DB ?    ; TIME OUT COUNTERS FOR PRINTER RESPONSE
449      DB ?    ; SECOND LOGICAL PRINTER ADAPTER
450      DB ?    ; THIRD LOGICAL PRINTER ADAPTER
451      DB ?    ; RESERVED
452      RS232_TIM_OUT DB ?   ; TIME OUT COUNTERS FOR RS232 RESPONSE
453      DB ?   ; SECOND LOGICAL RS232 ADAPTER
454      DB ?   ; RESERVED
455      DB ?   ; RESERVED
456
457      :-----+
458      : ADDITIONAL KEYBOARD DATA AREA
459      :-----+
460
461      BUFFER_START DW ?    ; BUFFER LOCATION WITHIN SEGMENT 40H
462      0080 ????? ; OFFSET OF KEYBOARD BUFFER START
463      0082 ????? ; OFFSET OF END OF BUFFER
464
465      :-----+
466      : EGA/PGA DISPLAY WORK AREA
467      :-----+
468      ROWS        DB ?    ; ROWS ON THE ACTIVE SCREEN (LESS 1)
469      0084 ??    ; BYTES PER CHARACTER
470      0085 ???   ; MODE OPTIONS
471      0087 ??   ; FEATURE BIT SWITCHES
472      0088 ??   ; RESERVED FOR DISPLAY ADAPTERS
473      0089 ??   ; RESERVED FOR DISPLAY ADAPTERS
474      008A ??   ; RESERVED
475
476      :-----+
477      : ADDITIONAL MEDIA DATA
478      :-----+
479
480      008B ??    ; LAST DISKETTE DATA RATE SELECTED
481      008C ??    ; STATUS REGISTER
482      008D ??    ; ERROR REGISTER
483      008E ??    ; FIXED DISK INTERRUPT FLAG
484      008F ??    ; BIT 0 = PC-1/IDEAL FDC ADAPTER CARD
485      0090 ??    ; BIT 1 = FDC STATE
486      0091 ??    ; DRIVE 0 MEDIA STATE
487      0092 ??    ; DRIVE 0 OPERATION START STATE
488      0093 ??    ; DRIVE 0 PRESENT CYLINDER
489      0094 ??    ; DRIVE 1 MEDIA STATE
490      0095 ??    ; DRIVE 1 OPERATION START STATE
491
492      :-----+
493      : ADDITIONAL KEYBOARD FLAGS
494      :-----+
495
496      0096 ??    ; KEYBOARD MODE STATE AND TYPE FLAGS
497      0097 ??    ; KEYBOARD LED FLAGS
498
499      IFE SYSTEM
500
501      :-----+
502      : REAL TIME CLOCK DATA AREA
503      :-----+
504
505      0098 ????? ; OFFSET ADDRESS OF USERS WAIT FLAG
506      0099 ????? ; SEGMENT ADDRESS OF USER WAIT FLAG
507      009C ????? ; RTC LOW WORD OF USER WAIT FLAG
508      009E ????? ; RTC HIGH WORD OF USER WAIT FLAG
509      00A0 ??     ; RTC_WAIT_FLAG DB ?    ; WAIT ACTIVE FLAG (01=BUSY, 80=POSTED)
510
511      ENDIF          ; (00=POST ACKNOWLEDGED)
512
513      :-----+
514      : AREA FOR NETWORK ADAPTER
515
516      00A1 07 [ ?? ] ; RESERVED FOR NETWORK ADAPTERS
517
518
519
520
521
522
523      00A8 ???????? ; EGA/PGA PALETTE POINTER
524
525
526
527
528
529      00CE
530      00CE ???     ; *SAVE_PTR DD ?    ; POINTER TO EGA PARAMETER CONTROL BLOCK
531
532
533
534
535
536
537      0100          ; *DAY_COUNT ORG 0CEH ; COUNT OF DAYS FROM 1-1-80
538
539      0100 ??       ; 1 RESERVED
540
541
542      0101          ; *DATA AREA - PRINT SCREEN
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 '
556          34 20 43 4F 50 52      ;COPYRIGHT NOTICE
557          2E 20 49 42 4D 20
558          45 20 52 50 2E 20
559          31 39 34 31 2C 31
560          39 38 36 20
561
562          EVEN
563          0022 20 20 20 20 20  DB   '
564          20 20 20 20 20 20
565          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
570
571  0050      HEADER ENDP
572  0050      CODE  ENDS
573      END
```

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


```

215          PAGE
216
217          MD_STRUC    STRUC
218          MD_SPEC1   DB ?      ; SRTD=0, HD UNLOAD=0F - 1ST SPECIFY BYTE
219          MD_SPEC2   DB ?      ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
220          MD_OFF_TIM DB ?      ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
221          MD_BYT_SEC DB ?      ; 512 BYTES/SECTOR
222          MD_SEC_TRK DB ?      ; EOT ( LAST SECTOR ON TRACK)
223          MD_GAP     DB ?      ; GAP LENGTH
224          MD_DTL     DB ?      ; DTL
225          MD_SP3     DB ?      ; GAP LENGTH FOR FORMAT
226          MD_FIL_BYT DB ?      ; FILL BYTE FOR FORMAT
227          MD_HD_TIM  DB ?      ; HEAD SETTLE TIME (MILLISECONDS)
228          MD_STR_TIM DB ?      ; MOTOR START TIME (1/8 SECONDS)
229          MD_MAX_TRK DB ?      ; MAX. TRACK NUMBER
230          MD_RATE    DB ?      ; DATA TRANSFER RATE
231          MD_STRUC    ENDS
232
233          = 007F        BITOFF  EQU 7FH
234          = 0080        BITON  EQU 80H
235
236          PUBLIC DISK_INT_I
237          PUBLIC DSKETTE_SETUP
238          PUBLIC DSKETTE_IO_I
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 PUBLIC
254
255          ASSUME CS:CODE,DS:DATA,ES:DATA
256
257
258
259          ;-----DRIVE TYPE TABLE-----;
260          0000        DR_TYPE    LABEL  BYTE
261          0000 01    DR_TYPE_E  DB 0      ; DRIVE TYPE, MEDIA TABLE
262          0001 0000 E DW OFFSET MD_TBL1
263          0003 82    DB 02+BITTON
264          0004 0000 E DW OFFSET MD_TBL2
265          0006 02    DB 02
266          0008 0000 E DW OFFSET MD_TBL3
267          0009 03    DB 03
268          000A 0000 E DW OFFSET MD_TBL4
269          000C 84    DB 04+BITTON
270          000E 0000 E DW OFFSET MD_TBL5
271          000F 04    DB 04
272          0010 0000 E DW OFFSET MD_TBL6
273          = 0012        DR_CNT    EQU (DR_TYPE_E-DR_TYPE)/3 ; NUMBER OF DRIVE TYPES
274          = 0006
275
276          0012        DSKETTE_IO_I PROC FAR
277          ST1
278          PUSH BP
279          PUSH DI
280          PUSH CX
281          PUSH BX
282          PUSH DX
283          MOV  BP,SP
284
285          ;>>> ENTRY POINT FOR ORG 00C59H
286          ;INTERRUPTS BACK ON
287          ;USER REGISTER
288          ;USER REGISTER
289          ;HEAD #, DRIVE # OR USER REGISTER
290          ;BUFFER OFFSET PARAMETER OR REGISTER
291          ;TRACK #, SECTOR # OR USER REGISTER
292          ;BP => PARAMETER LIST DEP. ON AH
293          ;[BP] = SECTOR #
294          ;[BP+1] = HEAD #
295          ;[BP+2] = BUFFER OFFSET
296          ;FOR RETURN OF DRIVE PARAMETERS:
297          ;CL/[BP] = BITS 7&6 HI BITS OF MAX CYL
298          ;BITS 0-5 MAX SECTORS/TRACK
299          ;CH/[BP+1] = LOW 8 BITS OF MAX CYL
300          ;BL/[BP+2] = BITS 7-4 = 0
301          ;DL/[BP+3] = BITS 3-0 = VALID CMOS TYPE
302          ;DH/[BP+4] = # DRIVES INSTALLED
303          ;DI/[BP+5] = MAX HEAD #
304          ;ES/[BP+6] = OFFSET TO DISK BASE
305          ;FS/[BP+7] = SEGMENT PARM OR USER REGISTER
306          ;GS/[BP+8] = USER REGISTERS
307          ;DS/[BP+9] = SEGMENT OF BIOS DATA AREA TO DS
308          ;SI/[BP+10] = CHECK FOR > LARGEST FUNCTION
309          ;DI/[BP+11] = FUNCTION OR
310          ;REPLACE WITH KNOWN INVALID FUNCTION
311
312          001A 1E    PUSH  DS
313          001B 54    PUSH  SI
314          001C E8    CALL   DDS
315          001F 80 FC 19  CMP   AH,[FNC-TAE-FNC_TAB]/2
316          0022 72 02  JB    OK_FUNC
317          0024 84 14  MOV   AH,14H
318
319          0026 80 FC 01  CMP   AH,I
320          0026 76 0C  JBE   OK_DRV
321          0028 80 FC 08  CMP   AH,B
322          002E 74 07  JZ    OK_DRV
323          0030 80 FA 03  CMP   DL,3
324          0032 80 02  JBE   OK_DRV
325          0034 80 02  MOV   AH,14H
326
327          0037 8A CC  MOV   CL,AH
328          0039 32 ED  XOR   CH,CH
329          0041 00 11  SHL   CL,1
330          003D BB 0060 R MOV   BX,[OFFSET FNC_TAB]
331          0040 03 D9  ADD   BX,CX
332          0042 8A E6  MOV   AH,DH
333          0044 32 F6  XOR   DH,DH
334          0046 BB F0  MOV   SI,AX
335          0048 BB FA  MOV   DI,DX
336          004A C6 26 0041 R MOV   AH,[DSKETTE_STATUS]
337          004E C6 06 0041 R MOV   AH,[DSKETTE_STATUS,0]
338
339          ; CL = FUNCTION
340          ; CX = FUNCTION
341          ; FUNCTION TIMES 2
342          ; ADD/START INTO FUNCTION TABLE
343          ; ADD OFFSET INTO TABLE => ROUTINE
344          ; AX = HEAD #, # OF SECTORS OR DASD TYPE
345          ; SI = HEAD #, # OF SECTORS OR DASD TYPE
346          ; DI = DRIVE #
347          ; LOAD STATUS TO AH FOR STATUS FUNCTION
348          ; INITIALIZE FOR ALL OTHERS
349
350          ; THROUGHTOUT THE DISKETTE BIOS, THE FOLLOWING INFORMATION IS CONTAINED IN
351          ; THE FOLLOWING MEMORY LOCATIONS AND REGISTERS. NOT ALL DISKETTE BIOS
352          ; FUNCTIONS REQUIRE ALL OF THESE PARAMETERS.
353
354          ;-----DISKETTE BIOS-----;
355          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          ; ACROSS CALLS TO SUBROUTINES THE CARRY FLAG (CY=1), WHERE INDICATED IN
337          ; SUBROUTINE PROLOGUES, REPRESENTS AN EXCEPTION RETURN (NORMALLY AN ERROR
338          ; CONDITION). IN MOST CASES, WHEN CY = 1, #DISKETTE_STATUS CONTAINS THE
339          ; SPECIFIC ERROR CODE.
340
341 0053 2E: FF 17          CALL WORD PTR CS:[BX]      ; (AH) = #DISKETTE_STATUS
342
343 0056 5E          POP SI           ; RESTORE ALL REGISTERS
344 0057 1F          POP DS           ;
345 .0058 59          POP CX           ;
346 0059 5B          POP BX           ;
347 005A 5A          POP DX           ;
348 005B FF          POP DI           ;
349 005C 5D          POP BP           ;
350 005D CA 0002        RET 2            ; THROW AWAY SAVED FLAGS
351
352
353 0060 0092 R          FNC_TAB DW 0000H             ; DISK_RESET      ; AH = 00; RESET
354 0062 00EA R          DW 0000H             ; DISK_STATUS     ; AH = 01; STATUS
355 0064 00FA R          DW 0000H             ; DISK_READ       ; AH = 02; READ
356 0066 0102 R          DW 0000H             ; DISK_WRITE      ; AH = 03; WRITE
357 0068 010E R          DW 0000H             ; DISK_VRF        ; AH = 04; VERIFY
358 006A 011A R          DW 0000H             ; DISK_FORMAT     ; AH = 05; FORMAT
359 006C 011C R          DW 0000H             ; FNC_ERR         ; AH = 06; INVALID
360 006E 017D R          DW 0000H             ; FNC_ERR         ; AH = 07; INVALID
361 0070 0167 R          DW 0000H             ; DISK_PARMS     ; AH = 08; READ DRIVE PARAMETERS
362 0072 017D R          DW 0000H             ; FNC_ERR         ; AH = 09; INVALID
363 0074 017D R          DW 0000H             ; FNC_ERR         ; AH = 0A; INVALID
364 0076 017D R          DW 0000H             ; FNC_ERR         ; AH = 0B; INVALID
365 0078 017D R          DW 0000H             ; FNC_ERR         ; AH = 0C; INVALID
366 007A 017D R          DW 0000H             ; FNC_ERR         ; AH = 0D; INVALID
367 007C 017D R          DW 0000H             ; FNC_ERR         ; AH = 0E; INVALID
368 007E 017D R          DW 0000H             ; FNC_ERR         ; AH = 0F; INVALID
369 0080 017D R          DW 0000H             ; FNC_ERR         ; AH = 10; INVALID
370 0082 017D R          DW 0000H             ; FNC_ERR         ; AH = 11; INVALID
371 0084 017D R          DW 0000H             ; FNC_ERR         ; AH = 12; INVALID
372 0086 017D R          DW 0000H             ; FNC_ERR         ; AH = 13; INVALID
373 0088 017D R          DW 0000H             ; FNC_ERR         ; AH = 14; INVALID
374 008A 027C R          DW 0000H             ; DISK_TYPE      ; AH = 15; READ DASD TYPE
375 008C 0280 R          DW 0000H             ; DISK_CHANGE    ; AH = 16; CHANGE STATUS
376 008E 02E5 R          DW 0000H             ; FORMAT_SET    ; AH = 17; SET DASD TYPE
377 0090 034D R          DW 0000H             ; SET_MEDIA     ; AH = 18; SET MEDIA TYPE
378 .0092               SET_DOU $             ; END
379 0092               DISKETTE_IO_1 ENDP
380
381
382          ; DISK_RESET      ; RESET THE DISKETTE SYSTEM.
383
384
385          ; ON EXIT: #DISKETTE_STATUS, CY REFLECT STATUS OF OPERATION
386
387 0092               DISK_RESET PROC NEAR
388 0092 BA 03F2          MOV DX,03F2H          ; ADAPTER CONTROL PORT
389 0095 FA          CLI
390 0097 0003F R          MOV AL, #MOTOR_STATUS
391 0099 24 3F          AND AL, 00011111B
392 009B 00 D0          ROL AL, 1
393 009D 00 D0          ROL AL, 1
394 009F 00 D0          ROL AL, 1
395 00A1 00 D0          ROL AL, 1
396 00A3 00 D8          OR AL, 00000100B
397 00A6 C6 06 003E R 00  OUT DX,AL
398 00AB EB 00          MOV AX, SEEK_STATUS,0
399 00AD OC 04          JMP $+2
400 00AF EE          OR AL, 00000100B
401 00B0 F9          OUT DX,AL
402 00B1 EB 00 ABA R    STI
403 00B2 72 2D          JC DR_ERR
404 00B6 B9 00C0        MOV CX, 11000000B
405
406          ;---- Nxt_Drv:
407 00B9 51          PUSH CX           ; SAVE FOR CALL
408 00BA B8 00E2 R    MOV AX, OFFSET DR_POP_ERR
409 00BD 50          PUSH AX           ; LOAD NEC_OUTPUT ERROR ADDRESS
410 00BE B4 08          MOV AH, 08H
411 00C0 E8 09F0 R    CALL NEC_OUTPUT
412 00C3 58          POP AX           ; SENSE INTERRUPT STATUS COMMAND
413 00C6 00 E4E2 R    CALL RESULTS
414 00C7 59          POP CX           ; REAM IN THE RESULTS
415 00C8 72 19          JC DR_ERR
416 00CA 3A 0E 0042 R  CMP CL, *NEC_STATUS
417 00CE 75 13          JNZ DR_ERR
418 00D0 FE C1          INC CL
419 00D2 80 F9 C3    CMP CL, 11000011B
420 00D5 76 E2          JBE NXT_DRV
421
422          ;---- SEND_SPECIFY_COMMAND_TO_NEC
423
424 00D7
425 00D7 E8 03D1 R    JT: CALL SEND_SPEC
426 00DA RESBAC1
427 00DA E8 0832 R    CALL SETUP_END
428 00DE B8 DE          MOV BX, SI-
429 00DF B8 C3          MOV AL, BL
430 00E1 C3          RET
431
432 00E2 DR_POP_ERR:   DR_POP_ERR: POP CX           ; CLEAR STACK
433 00E2 59          DR_ERR: OR #DISKETTE_STATUS,BAD_NEC
434 00E3
435 00E3 80 0E 0041 R 20  JMP SHORT RESBAC
436 00E8 EB F0          DISK_RESET ENDP
437 00EA
438
439
440          ;---- DISK_STATUS
441 00E3 _DISKETTE_STATUS
442 00E3 ; ON ENTRY: AH = STATUS OF PREVIOUS OPERATION

```

```

443 ; ON EXIT:    #DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
444
445 ;-----DISK_STATUS PROC NEAR
446 00EA          MOV    #DSKETTE_STATUS,AH      ; PUT BACK FOR SETUP_END
447 00F0 88 26 0041 R    ORL   SI,_END           ; VARIOUS CLEANUPS
448 00F5 E8 0032 R    CALL  SETL _END           ; GET SAVED AL TO BL
449 00F1 BB DE      MOV    BX,SI              ; STORE STATUS IN AL
450 00F3 8A C4      MOV    AL,AH
451 00F5 C3      RET
452 00F6
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
464 ; ON EXIT:    #DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
465
466 ;-----DISK_READ    PROC NEAR
467 00F6 80 26 003F R TF    AND   #MOTOR_STATUS,0111111B ; INDICATE A READ OPERATION
468 00FB E8 E646    MOV    AX,0E646H           ; AX = NEC COMMAND, DMA COMMAND
469 00FE E8 04B3 R    CALL  RD_WR_VF           ; COMMON READ/WRITE/VERIFY
470 0101 C3      RET
471 0102
472
473 ;-----DISK_WRITE   PROC NEAR
474 ; DISKETTE WRITE.
475 ; ON ENTRY:    DI    = DRIVE #
476 ;             SI-HI = HEAD #
477 ;             SI-LOW = # OF SECTORS
478 ;             ES    = BUFFER SEGMENT
479 ;             [BP]  = SECTOR #
480 ;             [BP+1] = TRACK #
481 ;             [BP+2] = BUFFER OFFSET
482
483 ; ON EXIT:    #DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
484
485 ;-----DISK_WRITE   PROC NEAR
486 0102 BB C54A    AND   #MOTOR_STATUS,0111111B ; INDICATE WRITE OPERATION
487 0105 80 0E 003F R 80    OR    AX,0E646H           ; AX = NEC COMMAND, DMA COMMAND
488 010A E8 04B3 R    CALL  RD_WR_VF           ; COMMON READ/WRITE/VERIFY
489 010D C3      RET
490 010E
491
492 ;-----DISK_VERIFY  PROC NEAR
493 ; DISKETTE VERIFY.
494 ; ON ENTRY:    DI    = DRIVE #
495 ;             SI-HI = HEAD #
496 ;             SI-LOW = # OF SECTORS
497 ;             ES    = BUFFER SEGMENT
498 ;             [BP]  = SECTOR #
499 ;             [BP+1] = TRACK #
500 ;             [BP+2] = BUFFER OFFSET
501
502 ; ON EXIT:    #DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
503
504 ;-----DISK_VERIFY  PROC NEAR
505 010E 80 26 003F R 7F    AND   #MOTOR_STATUS,0111111B ; INDICATE A READ OPERATION
506 0113 BB E642    MOV    AX,0E642H           ; AX = NEC COMMAND, DMA COMMAND
507 0116 E8 008F R 01    CALL  RD_WR_VF           ; COMMON READ/WRITE/VERIFY
508 0119 C3      RET
509 011A
510
511 ;-----DISK_FORMAT  PROC NEAR
512 ; DISKETTE FORMAT.
513 ; ON ENTRY:    DI    = DRIVE #
514 ;             SI-HI = HEAD #
515 ;             SI-LOW = # OF SECTORS
516 ;             ES    = BUFFER SEGMENT
517 ;             [BP]  = SECTOR #
518 ;             [BP+1] = TRACK #
519 ;             [BP+2] = BUFFER OFFSET
520 ;             #DISK_POINTER POINTS TO THE PARAMETER TABLE OF
521 ;             THIS DRIVE
522
523 ; ON EXIT:    #DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
524
525 ;-----DISK_FORMAT  PROC NEAR
526 011A E8 0404 R    CALL  X8000_NEW           ; TRANSLATE STATE TO PRESENT ARCH.
527 011D E8 05A0 R    CALL  PFM_INIT           ; ESTABLISH STATE IF UNESTABLISHED
528 0120 80 0E 003F R 80    OR    #MOTOR_STATUS,10000000B ; INDICATE WRITE OPERATION
529 0125 F6 06 008F R 01    TEST  #HF_CTRNL_DUAL ; TEST CONTROLLER I.D.
530 012A 14 05      JZ    NO_CHG_CHECK        ; NO CHG CHECK
531 012C E8 05F5 R    CALL  MED_CHANGE          ; CHECK MEDIA CHANGE AND RESET IF SO
532 012D 72 41      JC    FM_DON             ; MEDIA CHANGED, SKIP
533 0131
534 0131 E8 0658 R    NO_CHG_CHECK:     CALL  CHK_LASTRATE        ; ZF=1 ATTEMPT RATE IS SAME AS LAST RATE
535 0134 T4 06      CALL  FM_WR             ; YES, SKIP SPECIFY COMMAND
536 0136 E8 03D1 R    CALL  SEND_SPEC          ; SEND SPECIFY COMMAND TO NEC
537 0139 E8 0637 R    CALL  SEND_RATE          ; SEND DATA RATE TO CONTROLLER
538 0140
539 013C 80 4A      FM_WR:            MOV    AL,04AH           ; WILL WRITE TO THE DISKETTE
540 013E E8 0668 R    CALL  DMA_SETUP          ; SET UP THE DMA
541 0141 72 2F      JC    FM_DON             ; RETURN WITH ERROR
542 0143 B4 4D      MOV    AH,04DH           ; ESTABLISH THE FORMAT COMMAND
543 0144 80 05CB R    CALL  NEC_INIT           ; INITIALIZE THE NEC
544 0148 72 20      JC    FM_DON             ; LOAD ERROR ADDRESS
545 014A E8 0172 R    MOV    AX,OFFSET_FM_DON ; PUSH AX, ERROR RETURN
546 014D 50      PUSH  AX             ; PUSH NEC OUT ERROR RETURN
547 014E B2 03      MOV    DL,3              ; BYTES/SECTOR VALUE TO NEC
548 0150 E8 08FE R    CALL  GET_PARM          ; SECTORS/TRACK VALUE TO NEC
549 0151 E8 09F0 R    MOV    DL,4              ; GAP LENGTH VALUE TO NEC
550 0156 B2 04      CALL  GET_PARM          ; NEC_OUTPUT
551 0158 E8 08FE R    CALL  GET_PARM          ; NEC_OUTPUT
552 0158 E8 09F0 R    MOV    DL,T              ; NEC_OUTPUT
553 015E B2 07      CALL  GET_PARM          ; NEC_OUTPUT
554 0160 E8 08FE R    CALL  GET_PARM          ; NEC_OUTPUT
555 0161 E8 09F0 R    CALL  GET_PARM          ; NEC_OUTPUT
556 0164 B2 00      MOV    DL,B              ; 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 E8 0832 R FM_DON: CALL XLAT_OLD ; TRANSLATE STATE TO COMPATIBLE MODE
563 0175 E8 0832 R CALL SETUP_END ; VARIOUS CLEANUPS
564 0178 8B DE MOV BX,S1- ; GET SAVED AL TO BL
565 017A 8A C3 MOV AL,BL ; PUT BACK FOR RETURN
566 017C C3 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 ;-----+
576 017D BB C6 FNC_ERR PROC NEAR ; INVALID FUNCTION REQUEST
577 017F B4 01 MOV AH,51 ; RESTORE AL
578 0181 88 26 0041 R MOV AH,BAD_CMD ; SET BAD COMMAND ERROR
579 0185 F9 MOV #DSKETTE_STATUS,AH ; STORE IN DATA AREA
580 0186 C3 STC ; SET CARRY INDICATING ERROR
581 0187 RET
582 FNC_ERR ENDP

583 ;-----+
584 ; DSK_PARAMS:
585 ;   READ DRIVE PARAMETERS.
586 ; ON ENTRY: 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 ; CHECK FOR FIXED MEDIA TYPE REQUEST
606 0187 B1 FF 0080 CMP DI,80H ; CONTINUE IF NOT REQUEST FALL THROUGH
607 0188 72 06 JB DISK_P2
608 ;
609 ;-----+ FIXED DISK REQUEST FALL THROUGH ERROR
610 ;
611 018D BB C6 MOV AX,51 ; RESTORE AL WITH CALLERS VALUE
612 018D B4 01 MOV AH,BAD_CMD ; SET BAD COMMAND ERROR IN (AH)
613 018E 58 SET_PARM_Flags STC ; SET ERROR RETURN CODE
614 0192 C3 RET
615 ;
616 0193 DISK_P2: CALL XLAT_NEW ; TRANSLATE STATE TO PRESENT ARCH.
617 0193 E8 0404 R MOV WORD PTR [BP+2],0 ; DRIVE TYPE 0
618 0193 C7 46 02 0000 MOV AX,_EQUIP_FLAG ; LOAD EQUIPMENT FLAG FOR # DISKETTES
619 0193 80 00 00 00 AND AL,1000001B ; LOAD EQUIPMENT FLAG FOR # DISKETTES
620 019E 24 C1 CALL SHR ; KEEPS DISKETTE DRIVE BITS
621 01A0 DO E8 SHR AL,1 ; ARE THERE ANY DRIVES INSTALLED?
622 01A2 73 7C JNC NON_DRV ; NC-->NO DRIVES, ZERO PARAMETERS
623 01A4 DO C0 ROL AL,T ; ROTATE TO ORIGINAL POSITION
624 01A6 DO C0 ROL AL,I ; ROTATE BITS 6 AND 7 TO 0 AND 1
625 01A8 00 00 ROL AL,I
626 01AA FE C0 INC AL ; CONVERT TO RELATIVE !
627 01AC 88 46 04 MOV [BP+4],AL ; STORE NUMBER OF DRIVES
628 01AF F6 00 00FB R 01 TEST 0FH,CNTRL,DUAL ; CHECK CONTROLLER I.D.
629 01B0 75 03 JNZ DP1_CONT ; CONTINUE WITH USUAL PARMS CHECK
630 01B0 E9 0256 R JMP DET_PARMS ; RETURN THIS CONTROLLERS PARMS
631 ;
632 01B9 83 FF 01 DP1_CONT: CMP D1,I ; CHECK FOR VALID DRIVE
633 01C0 77 66 JA NON_DRV1 ; DRIVE INVALID
634 01C6 C6 46 05 01 MOV BYTE PTR [BP+5],1 ; MAXIMUM HEAD NUMBER = 1
635 01C2 8E 0BCF R CALL CMOS_TYPE ; RETURN DRIVE TYPE IN AL
636 01C5 72 16 JC CMOS_EST ; ON CMOS BAD CHECK ESTABLISHED
637 01C8 04 C0 OR AL,AL ; TEST FOR NO DRIVE TYPE
638 01C9 00 00 JZ CMOS_EST ; RETURN 0
639 01CB E8 03B1 R CALL DR_TYPE_CHECK ; RTN CS1BX = MEDIA/DRIVE PARAM TBL
640 01CE 72 03B1 R JC CMOS_EST ; TYPE NOT IN TABLE (IMPOSSIBLE BAD CMOS
641 01D3 2E: 8A 4F 04 MOV [BP+2],AL ; STORE VALID CMOS DRIVE TYPE
642 01D3 2E: 8A 4F 04 MOV CL,CS1[BX].MD_SEC_TRK ; GET SECTOR/TRACK
643 01D7 2E: 8A 6F 0B MOV CH,CS1[BX].MD_MAX_TRK ; GET MAX. TRACK NUMBER
644 01D8 EB 32 JMP SHORT STO_CX ; CMOS GOOD, USE CMOS
645 ;
646 01DD CHK_EST: MOV AH,#DSK_STATE[D1] ; LOAD STATE FOR THIS DRIVE
647 01DD 8A A5 0090 R TEST AH,MD_DET ; CHECK FOR ESTABLISHED STATE
648 01E1 F6 C4 10 JZ NON_DRV1 ; CMOS BAD/INVALID ID AND UNESTABLISHED
649 01E4 74 3E
650 ;
651 USE_EST: AND AH,RATE_MSK ; ISOLATE STATE
652 01E6 80 E4 C0 CMP AH, RATE_250 ; RATE 250 ?
653 01E9 80 FC 80 JNE USE_EST2 ; NO, GO CHECK OTHER RATE
654 01EC 75 54
655 ;
656 01E6 DATA_RATE IS 250 KBS, TRY 360 KB TABLE FIRST
657 ;
658 01E6 B0 01 MOV AL,01 ; DRIVE TYPE 1 (360KB)
659 01F0 E8 03B1 R CALL DR_TYPE_CHECK ; RTN CS1BX = MEDIA/DRIVE PARAM TBL
660 01F3 2E: 8A 4F 04 MOV CL,CS1[BX].MD_SEC_TRK ; GET SECTOR/TRACK
661 01F7 2E: 8A 6F 0B MOV CH,CS1[BX].MD_MAX_TRK ; GET MAX. TRACK NUMBER
662 01FB F6 85 0090 R 01 TEST #DSK_STATE[D1],TRK_CAPA ; 80 TRACK ?
663 0200 74 0D JZ STO_CX ; MUST BE 360KB DRIVE
664 ;
665 ;
666 0202 PARM_HDR_80T: RET ; DRIVE TYPE 4
667 0202 B0 04 CALL DR_TYPE_CHECK ; RTN CS1BX = MEDIA/DRIVE PARAM TBL
668 0204 E8 03B1 R MOV CL,CS1[BX].MD_SEC_TRK ; GET SECTOR/TRACK
669 0207 2E: 8A 4F 04

```

```

671 020B 2E: 8A 6F 0B      MOV    CH,CS:[BX].MD_MAX_TRK   ; GET MAX. TRACK NUMBER
672
673 020F                   STO_CX: MOV    [BP],CX           ; SAVE IN STACK FOR RETURN
674 020F 89 4E 00          ES_DI:  MOV    AX,[BP+6],BX     ; ADDRESS OF MEDIA/DRIVE PARAM TABLE
675 0212                   MOV    AX,CS           ; SEGMENT MEDIA/DRIVE PARAMETER TABLE
676 0212 89 5E 06          MOV    ES,AX           ; ES IS SEGMENT OF TABLE
677 0217 80 C0
678 0217 80 C0
679
680 0219                   DP_OUT: CALL   XLAT_OLD        ; TRANSLATE STATE TO COMPATIBLE MODE
681 0219 E8 0432 R         XOR    AX,AX           ; CLEAR
682 021C 33 C0
683 021E F8
684 021F C3
685
686
687
688 0220                   NON_DRV: MOV    BYTE PTR [BP+4],0   ; CLEAR NUMBER OF DRIVES
689
690
691 0224                   NON_DRV1: CMP   DI,80H        ; CHECK FOR FIXED MEDIA TYPE REQUEST
692 0224 81 FF 0080        JB    NON_DRV2       ; CONTINUE IF NOT REQUEST FALL THROUGH
693 0224 72 09
694
695
696
697 022A                   FD_REQ_ERR: CALL   XLAT_OLD        ; ELSE TRANSLATE TO COMPATIBLE MODE
698 022A E8 0432 R         MOV    AX,ST           ; RESTORE AL
699 022B 8B C6             MOV    AH,BAD_CMD     ; SET BAD COMMAND ERROR
700 022B 84 01             STC
701 0231 F9
702 0232 C3
703
704 0233                   NON_DRV2: XOR   AX,AX           ; CLEAR PARMs IF NO DRIVES OR CMOS BAD
705 0233 33 C0
706 0235 85 46 00          MOV    [BP],AX
707 0235 85 66 05          MOV    [BP+5],AH
708 0238 85 46 06          MOV    [BP+6],AX
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                   USE_EST2: MOV   AL,02          ; DRIVE TYPE 2 (1.2MB)
715 0242 80 02             CALL   DR_TYPE_CHECK   ; RTN CS:[BX] = MEDIA/DRIVE PARAM TBL
716 0244 E8 03B1 R         MOV   CL,CS:[BX].MD_SEC_TRK ; GET SECTOR/TRACK
717 0247 2E: 8A 4F 04        MOV   CL,CS:[BX].MD_MAX_TRK ; GET MAX. TRACK NUMBER
718 0248 2E: 8A 6F 0B        CMP   AH, RATE_300    ; RATE 300 ?
719 0248 80 40             JE    STD_CX          ; MUST BE 1.2MB DRIVE
720 0252 74 00             JMP   SHORT PARM_HDR_80T ; ELSE, HIGH DATA RATE/80 TRACK DRIVE
721 0254 EB AC
722 0256
723 0256 83 FF 03          DET_PARMS: CMP   DI,3           ; REQUEST FOR FIXED DISK?
724 0259 77 D8             JA    NON_DRV2       ; YES-->DRIVE NUMBER INVALID
725 0260 80 09
726 0260 F4 85 0090 R 01
727 0262 80 01             TEST  AL,[DI]        ; IS DRIVE 80 TRACKS? (RELATIVE ZERO)
728 0264 85 27             MOV   CL,5            ; SET CMOS TYPE 1
729 0266 74 04             MOV   CH,39           ; NUMBER OF TRACKS (RELATIVE ZERO)
730 0268 80 03             JZ    SET_TYP1       ; IF ZERO TYPE = 1
731 0268 85 4F             MOV   AL,3            ; SET CMOS TYPE 3
732 026C
733 026C 88 46 02          SET_TYP1: MOV   CH,79           ; NUMBER OF TRACKS (RELATIVE ZERO)
734 026F C6 46 03 00
735 0273 C6 46 05 01
736 0277 E8 03B1 R
737 027A EB 93
738
739 027C                   DISK_PARMS: ENDP
740
741
742
743
744
745
746
747
748 027C                   DISK_TYPE: PROC  NEAR
749 027C F6 06 008F R 01
750 0281 74 22
751 0283 E8 0404 R
752 0286 8A 85 0090 R
753 028A 80 C0
754 028C 74 13
755 0290 A0 01
756 0290 80 01
757 0292 74 02
758 0294 B4 02
759
760 0296
761 0296 80 50
762 0297 E8 0432 R
763 029A 58
764 029B
765 029F F8
766 029C BB DE
767 029D 8A C3
768 02A0 C3
769 02A1
770 02A1 32 E4
771 02A3 EB F1
772 02A5
773 02A5 A1 0010 R
774 02A8 D0 E8
775 02AA 73 F5
776 02AC B4 01
777 02AE EB EB
778 02B0
779
780
781
782
783
784
;
```

```

T85 : ON EXIT: AH = #DSKETTE_STATUS
T86 :          00 - DISK CHANGE LINE INACTIVE, CY = 0
T87 :          01 - DISK CHANGE LINE ACTIVE, CY = 1
T88 :-----+
T89 02B0  DISK_CHANGE PROC NEAR
T90 02B0 F6 06 008F R 01 TEST #HF_CNTL,DUAL ; TEST CONTROLLER I.D.
T91 02B5 75 03 JNZ DC1 ; ERROR FOR THIS KIND OF CONTROLLER
T92 02B7 E9 01D7 R JMP FNC_ERR
T93 02C1 00 0000 R DC1: ; GET MEDIA STATE INFORMATION
T94 02B4 E8 0404 R CALL XLAT_NEW ; TRANSLATE STATE TO PRESENT ARCH.
T95 02B0 8A 05 0090 R MOV #DSK_STATE[DI] ; GET MEDIA STATE INFORMATION
T96 OR AL,AL ; DRIVE PRESENT ?
T97 02C3 T4 19 JZ DC_NON ; JUMP IF NO DRIVE
T98 02C5 A8 01 TEST AL,TRK_CAPA ; IS 80 TRACK DRIVE ?
T99 02C7 T4 05 JZ SETIT ; IF SO , CHECK CHANGE LINE
T100 02C9 E8 0B21 R DCO: CALL READ_DSKCHNG ; GO CHECK STATE OF DISK CHANGE LINE
T101 02CC T4 05 JZ FINIS ; CHANGE LINE NOT ACTIVE
T102 02CE C6 06 0041 R 06 SETIT: MOV #DSKETTE_STATUS,MEDIA_CHANGE ; INDICATE MEDIA REMOVED
T103 02D3 E8 0432 R FINIS: CALL XLAT_OLD ; TRANSLATE STATE TO COMPATIBLE MODE
T104 02D6 E8 0832 R CALL SETUP_END ; VARIOUS CLEANUPS
T105 02D9 8B DE MOV BX,S1 ; GET SAVED AL TO BL
T106 02DB 8A C3 MOV AL,BL ; PUT BACK FOR RETURN
T107 02DD C3 RET
T108 02DE DC_NON: ;-----+
T109 02DE 80 0E 0041 R 80 OR #DSKETTE_STATUS,TIME_OUT ; SET TIMEOUT, NO DRIVE
T110 02E3 EB EE JMP SHORT FINIS
T111 02E5 DISK_CHANGE ENDP
T112 02E6 ;-----+
T113 02E6 E8 0404 R FORMAT_SET PROC NEAR
T114 02E8 56 CALL XLAT_NEW ; TRANSLATE STATE TO PRESENT ARCH.
T115 02E9 8B C6 PUSH SI ; SAVE DASD TYPE
T116 02EB 32 E4 MOV AX,SI ; AH = ? , AL = DASD TYPE
T117 02ED 8B F0 XOR AH,AH ; AH 0 , AL = DASD TYPE
T118 02F0 80 A5 0090 R 0F MOV SI,AX ; SI = DASD TYPE
T119 02F4 80 0000 R 0F AND #DSK_STATE[DI],NOT_MED_DET+DBL_STEP+RATE_MSK ; CLEAR STATE
T120 02F5 75 07 DEC S1 ; CHECK FOR 320/360K MEDIA & DRIVE
T121 02F7 80 8D 0090 R 90 NOT_320 JNZ NOT_320 ; BYPASS IF NOT
T122 02FC EB 3E OR #DSK_STATE[DI],MED_DET+RATE_250 ; SET TO 320/360
T123 02E5 SHORT S0
T124 02FE NOT_320: ;-----+
T125 02FE F6 06 008F R 01 TEST #HF_CNTL,DUAL ; TEST CONTROLLER I.D.
T126 0303 74 0A JZ S3 ; CHECK FOR TIME_OUT
T127 0305 E8 05F5 R CALL MED_CHANGE ; CHECK FOR TIME_OUT
T128 0303 80 3E 0041 R 80 CMP #DSKETTE_STATUS,TIME_OUT
T129 0300 74 2D JZ S0 ; IF TIME OUT TELL CALLER
T130 030F 4E S3: DEC S1 ; CHECK FOR 320/360K IN 1.2M DRIVE
T131 0310 75 07 JNZ NOT_320_12 ; BYPASS IF NOT
T132 0312 80 8D 0090 R 70 OR #DSK_STATE[DI],MED_DET+DBL_STEP+RATE_300 ; SET STATE
T133 0317 EB 23 JMP SHORT S0
T134 0319 NOT_320_12: ;-----+
T135 0319 4E DEC S1 ; CHECK FOR 1.2M MEDIA IN 1.2M DRIVE
T136 031A 75 07 JNZ NOT_12 ; BYPASS IF NOT
T137 031C 80 8D 0090 R 10 OR #DSK_STATE[DI],MED_DET+RATE_500 ; SET STATE VARIABLE
T138 0321 EB 19 JMP SHORT S0 ; RETURN TO CALLER
T139 0323 NOT_12: ;-----+
T140 0323 4E DEC S1 ; CHECK FOR SET DASD TYPE 04
T141 0324 75 20 JNZ FS_ERR ; BAD COMMAND EXIT IF NOT VALID TYPE
T142 0326 80 50 TEST #DSK_STATE[DI].DRY_DET ; DRIVE DETERMINED ?
T143 0328 T4 09 JZ ASSUME ; IF STILL NOT DETERMINED ASSUME
T144 032D B0 50 MOV AL,MED_DET+RATE_300 ; MULTIPLE FORMAT CAPABILITY ?
T145 032E F6 85 0090 R 02 TEST #DSK_STATE[DI].FMT_CAPA ; IF 1.2 M THEN DATA RATE 300
T146 0334 75 02 JNZ OR_IT_IN
T147 0336 ASSUME: MOV AL,MED_DET+RATE_250 ; SET UP
T148 0336 B0 90
T149 0338 OR_IT_IN: OR #DSK_STATE[DI].AL ; OR IN THE CORRECT STATE
T150 0338 SHORT S0 ;-----+
T151 033C E8 0432 R CALL XLAT_OLD ; TRANSLATE STATE TO COMPATIBLE MODE
T152 033F 80 0832 R CALL SETUP_END ; VARIOUS CLEANUPS
T153 0342 5D 00 MOV BX ; GET SAVED AL TO BL
T154 0343 8A C3 MOV AL,BL ; PUT BACK FOR RETURN
T155 0345 C3 RET
T156 0346 FS_ERR: ;-----+
T157 0346 C6 06 0041 R 01 MOV #DSKETTE_STATUS,BAD_CMD ; UNKNOWN STATE,BAD COMMAND
T158 0348 EB EF JMP SHORT S0
T159 0340 FORMAT_SET ENDP
T160 0340 ;-----+
T161 0340 SET_MEDIA ;-----+
T162 0340 D0 00 D0 00 D0 00 D0 00 ; THIS ROUTINE SETS THE TYPE OF MEDIA AND DATA RATE
T163 0340 00 00 00 00 ; TO BE USED FOR THE FOLLOWING FORMAT OPERATION.
T164 0340 ;-----+
T165 0340 ON_ENTRY: [BP] = SECTOR PER TRACK
T166 0340 [BP+1] = TRACK #
T167 0340 D = DRIVE #
T168 0340 ;-----+
T169 0340 ON_EXIT: #DSKETTE_STATUS REFLECTS STATUS
T170 0340 IF NO ERROR: ;-----+

```



```

1012      ; ON ENTRY: CS:BX = MEDIA/DRIVE PARAMETER TABLE           I
1013      ; ON EXIT:  NONE                                         I
1014      ; REGISTERS ALTERED: AX,CX,DX                           I
1015
1016 03EC SEND_SPEC_M0 PROC NEAR
1017 03EC B8 0403 R MOV AX,[BX].MD_SPEC1 ; LOAD ERROR ADDRESS
1018 03EC 50          PUSH AX ; PUSH NOT-OUT ERROR RETURN
1019 03D0 00 03        MOV AL,03H ; SPECIFY COMMAND
1020 03F2 E8 09F0 R CALL NEC_OUTPUT ; GET 1ST SPECIFY BYTE
1021 03F5 2E 8A 27    MOV AH,[BX].MD_SPEC1 ; GET 1ST SPECIFY BYTE
1022 03FB E8 09F0 R CALL NEC_OUTPUT ; OUTPUT THE COMMAND
1023 03FB 2E 8A 67 01 MOV AH,[BX].MD_SPEC2 ; GET SECOND SPECIFY BYTE
1024 03FB E8 09F0 R CALL NEC_OUTPUT ; GET SECOND SPECIFY BYTE
1025 03F5 2E 58       CALL NEC_OUTPUT ; OUTPUT THE COMMAND
1026 0403 POP AX ; POP ERROR RETURN
1027 0403 C3         SCPE_ESBAC1
1028 0404 RET
1029 SEND_SPEC_M0 ENDP
1030
1031 ;----- XLAT_NEW: TRANSLATES DISKETTE STATE LOCATIONS FROM COMPATIBLE MODE TO NEW ARCHITECTURE.
1032
1033
1034
1035
1036 0404 ; ON ENTRY: DI : DRIVE
1037 0404 F6 06 008F R 01 XLAT_NEW PROC NEAR
1038 0409 74 22        TEST BH,CNTRL,DUAL ; TEST CONTROLLER I.D.
1039 0408 83 FF 01     JZ XN_OUT ; IF INVALID BACK
1040 040E 77 1D        CMP DI,-1 ; NO DRIVE ?
1041 0410 80 BD 0090 R 00 JZ DO_DET ; IF NO DRIVE ATTEMPT DETERMINE
1042 0415 74 17        CMP #DSK_STATE[DI],0 ; NO DRIVE ?
1043 0416 80 CF        MOV CX,DI ; CX = DRIVE NUMBER
1044 0419 00 E1        SHL CL,1 ; CL = SHIFT COUNT, A=0, B=4
1045 0418 00 E1        SHL CL,1
1046 0410 A0 008F R   MOV AL,OHF_CNTRL ; DRIVE INFORMATION
1047 0422 D2 C8        ROR AL,CL ; TO LOW NIBBLE
1048 0422 24 07        AND AL,DRV_DET+FMT_CAPA+TRK_CAPA ; KEEP DRIVE BITS
1049 0429 80 A5 0090 R F8 AND #DSK_STATE[DI].NOT DRV_DET+FMT_CAPA+TRK_CAPA
1050 0429 08 85 0090 R OR #DSK_STATE[DI].AL ; UPDATE DRIVE STATE
1051 042D C3          XN_OUT: RET
1052 042D
1053
1054 042E DO_DET: CALL DRIVE_DET ; TRY TO DETERMINE
1055 042E E8 0B2B R   RET
1056 0431 C3
1057
1058 0432 XLAT_NEW ENDP
1059
1060 ;----- XLAT_OLD: TRANSLATES DISKETTE STATE LOCATIONS FROM NEW ARCHITECTURE TO COMPATIBLE MODE.
1061
1062
1063
1064 ; ON ENTRY: DI : DRIVE
1065
1066 0432 XLAT_OLD PROC NEAR
1067 0432 F6 06 008F R 01 TEST BH,CNTRL,DUAL ; TEST CONTROLLER I.D.
1068 0437 74 79        JZ XN_OUT ; IF INVALID BACK
1069 0439 83 FF 01     CMP DI,-1 ; NO DRIVE ?
1070 043C 77 74        JZ DO_DET ; IF NO DRIVE TRANSLATE DONE
1071 043E 80 BD 0090 R 00 JZ XN_OUT ; IF NO DRIVE TRANSLATE DONE
1072 0443 74 6D
1073
1074 ;----- TEST FOR SAVED DRIVE INFORMATION ALREADY SET
1075
1076 0445 8B CF        MOV CX,DI ; CX = DRIVE NUMBER
1077 0447 00 E1        SHL CL,1 ; CL = SHIFT COUNT, A=0, B=4
1078 0448 00 D2        SHL CL,1
1079 044D D2 CC        MOV AH,FMT_CAPA ; LOAD MULTI DATA RATE BIT MASK
1080 044D D2 CC        ROR AH,CL ; ROTATE BY MASK
1081 044F B4 26 008F R TEST OHF_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 OHF_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 D4        NOT AH ; TRANSLATE MASK
1089 045B 20 26 008F R AND OHF_CNTRL,AH ; KEEP BITS FROM OTHER DRIVE INTACT
1090
1091 ;----- ACCESS CURRENT DRIVE BITS AND STORE IN OHF_CNTRL
1092
1093 045F 8A 85 0090 R MOV AL,#DSK_STATE[DI] ; ACCESS STATE
1094 0463 24 07        AND AL,DRV_DET+FMT_CAPA+TRK_CAPA ; KEEP DRIVE BITS
1095 0465 D2 C8        ROR AL,CL ; FIX FOR THIS DRIVE
1096 0467 08 06 008F R OR OHF_CNTRL,AL ; UPDATE SAVED DRIVE STATE
1097
1098 ;----- TRANSLATE TO COMPATIBILITY MODE
1099
1100 046B SAVE_SET: MOV AH,#DSK_STATE[DI] ; ACCESS STATE
1101 046B 8A A5 0090 R MOV BH,AH ; TO BH FOR LATER
1102 046F 8A FC        AND AH,RATE_MSK ; KEEP ON RATE
1103 0470 80 00 C0      CMP AH,RATE_500 ; RATE 500 ?
1104 0474 80 FC 00      JZ CHK_HDR_80T ; YES 1.2 OR HIGH DATA RATE 80 TRK
1105 0477 74 10        MOV AL,M3D1U ; AL = 360 IN 1.2 UNESTABLISHED
1106 0479 B0 01        MOV BH,M3D1U ; AL = 360 IN 1.2 UNESTABLISHED
1107 047B 80 FC 40      CMP AH,RATE_300 ; RATE 300 ?
1108 047E 75 16        JNZ CHK_250 ; NO, 360/360, 720/720
1109 0480 F6 C7 20      TEST BH,DBL_STEP ; YES, DOUBLE STEP ?
1110 0483 75 1D        JNZ TST_DET ; YES, MUST BE 360 IN 1.2
1111
1112 0485 UNKNOD: MOV AL,MED_UNK ; 'NONE OF THE ABOVE'
1113 0485 B0 07        JMP SHORT AL_SET ; PROCESS COMPLETE
1114 0487 EB 20
1115
1116 0489 CHK_HDR_80T: CALL CMOS_TYPE ; RETURN DRIVE TYPE IN (AL)
1117 0489 E8 08CF R JC UNKNOD ; ERROR, SET 'NONE OF THE ABOVE'
1118 048C 72 F7        CMP AL,02 ; 1.2MB DRIVE ?
1119 048E 3C 02        JNE UNKNOD ; NO, GO SET 'NONE OF THE ABOVE'
1120 0490 75 F3        MOV AL,M1D1U ; AL = 1.2 IN 1.2 UNESTABLISHED
1121 0492 B0 02        JMP SHORT TST_DET
1122 0494 EB 0C
1123
1124 0496 CHK_250: MOV AL,M3D3U ; AL = 360 IN 360 UNESTABLISHED
1125 0496 B0 00

```

```

1126 0498 80 FC 80      CMP     AH,RATE_250      I RATE 250 ?
1127 0499 75 E9      JNZ     UNKNO                I IF SO FALL THRU
1128 049D F6 C7 01      TEST    BH,TRK_CAPA      I DO TRACK CAPABILITY ?
1129 04A0 75 E3      JNZ     UNKNO                I IF SO JUMP, FALL THRU TEST DET
1130
1131 04A2      TST_DET: TEST    BH, MED_DET      I DETERMINED ?
1132 04A3 F6 C7 10      JZ      AL_SET              I IF LOWEN SET
1133 04A4 04 03      ADD    AL,S                I MAKE DETERMINED/ESTABLISHED
1135
1136 04A9      AL_SET: AND    #DSK_STATE[DI],NOT DRV_DET+FMT_CAPA+TRK_CAPA I CLEAR DRIVE
1137 04A9 80 A5 0090 R F8 OR     #DSK_STATE[DI],AL      I REPLACE WITH COMPATIBLE MODE
1138 04B0 88 85 0090 R
1139 04B2
1140 04B2 C3      RET
1141 04B3      XLAT_OLD ENDP

1142
1143
1144      ;----- RD_WR_VF -----
1145      ; COMMON READ, WRITE AND VERIFY;
1146      ; MAIN LOOP FOR STATE RETRIES.
1147
1148      ; ON ENTRY:   AH : READ/WRITE/VERIFY NEC PARAMETER
1149      ; AL : READ/WRITE/VERIFY DMA PARAMETER
1150
1151      ; ON EXIT:    #DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION !
1152
1153 04B3      RD_WR_VF PROC NEAR
1154 04B3 50      PUSH   AX      I SAVE DMA, NEC PARAMETERS
1155 04B4 E8 0404 R CALL    XLAT_NEW          I TRANSLATE STATE TO PRESENT ARCH.
1156 04B7 E8 0561 R CALL    SETUP_STATE      I INITIALIZE START AND END RATE
1157 04B8 58      POP    AX      I RESTORE READ/WRITE/VERIFY
1158
1159 04B8      DO AGAIN: TEST   #HF_CNTRL,DUAL      I TEST CONTROLLER I.O.
1160 04B8 F6 00BF R 01 JZ      RWV_
1161 04C0 74 0A      PUSH   AX      I SAVE READ/WRITE/VERIFY PARAMETER
1162 04C2 50      PUSH   AX      I MEDIA CHANGE AND RESET IF CHANGED
1163 04C3 00 05F5 R CALL    WORD_CHANGE      I RESTORE READ/WRITE/VERIFY
1164 04C6 58      POP    AX
1165 04C7 73 03      JNC    RWV_
1166 04C9 E9 0552 R JMP    RWV_END
1167 04CC
1168 04C4 50      PUSH   AX      I SAVE READ/WRITE/VERIFY PARAMETER
1169
1170 04CD 8A B5 0090 R MOV    DH,#DSK_STATE[DI]      I GET RATE STATE OF THIS DRIVE
1171 04D1 80 E6 C0      AND    DH,RATE_MSK      I KEEP ONLY RATE
1172
1173 04D4 E8 08CF R CALL   CMOS_TYPE          I RETURN DRIVE TYPE IN (AL)
1174 04D5 08 C0      OR     AL,AL              I TEST FOR NO DRIVE
1175 04D6 04 02      JZ      RWV_ASSUME        I ASSUME TYPE, USE MAX TRACK
1176 04D8 E8 03B1 R CALL   DR_TYPE_CHECK      I RTYC CS1BX = MEDIA/DRIVE PARAM TABL
1177 04DE 72 2A      JC      RWV_ASSUME        I TYPE NOT IN TABLE ASSUME DEFAULT
1178
1179      ;--- SEARCH FOR MEDIA/DRIVE PARAMETER TABLE
1180
1181 04E0 57      PUSH   DI      I SAVE DRIVE #
1182 04E1 33 DB      XOR    BX,BX          I BX = INDEX TO DR_TYPE TABLE
1183 04E3 B9 0006      MOV    CX,DR_CNT        I CX = LOOP COUNT
1184 04E6
1185 04E6 2E; SA A7 0000 R      RWV_DR_SEARCH: MOV    AH,CS;DR_TYPE[BX]      I GET DRIVE TYPE
1186 04E8 80 E4 7F      AND    AH,BIT7OFF      I MASK OUT MSB
1187 04E9 3A C4      CMP    AH,AH              I DRIVE TYPE MATCH ?
1188 04F0 75 0B      JNE    RWV_NXT_MD      I NO, CHECK NEXT DRIVE TYPE
1189 04F2
1190 04F2 2E; BB BF 0001 R      RWV_DR_FND: MOV    DI,WORD PTR CS:DR_TYPE[BX+1] I DI = MEDIA/DRIVE PARAMETER TABLE
1191 04F7      RWV_MD_SEARCH: CMP    DH,CS:[DI].MD_RATE      I MATCH ?
1192 04F7 2E; 3A 75 0C      JZ      RWV_MD_FND      I YES, GO GET 1ST SPECIFY BYTE
1193 04F8 74 1D      JNE    RWV_NXT_MD
1194 04FD
1195 04FD 83 C3 03      RWV_NXT_MD: ADD    BX,3          I CHECK NEXT DRIVE TYPE
1196 0500 E2 E4      LOOP   RWV_DR_SEARCH      I FORCE IT TO RETRY
1197 0502 C6 06 0041 R FF      MOV    #DSKETTE_STATUS,0FFH I RESTORE DRIVE #
1198 0507 5F
1199 0508 EB 3F      POP    DI
1200      SHORT CHK_RET
1201
1202      ;--- ASSUME PRIMARY DRIVE IS INSTALLED AS SHIPPED
1203 050A
1204 0505 BB 0000 E      MOV    BX,OFFSET MD_TBL1      I POINT TO 40 TK 250 KBS
1205 0506 B5 0090 R 01      TEST   #DSKETTE[DT].TRK_CAPA      I TEST FOR 80 TRACK
1206 0512 74 09      JZ      RWV_MD_FND      I JUST F 40 TRACK
1207 0514 BB 0000 E      MOV    BX,OFFSET MD_TBL3      I POINT TO 80 TK 500 KBS
1208 0517 EB 04 90      JMP    RWV_MD_FND1      I GO SET SPECIFY PARAMETERS
1209
1210
1211
1212 051A      RWV_ASSUM: MOV    BX,OFFSET MD_TBL1      I POINT TO 40 TK 250 KBS
1213 051A BB DF      TEST   #DSKETTE[DT].TRK_CAPA      I TEST FOR 80 TRACK
1214 051C 5F
1215 051D      RWV_MD_FND1:
1216
1217
1218      ;--- SEND THE SPECIFY COMMAND TO THE CONTROLLER
1219 051D E8 03EC R      CALL   SEND_SPEC_M0      I ZF=1 ATTEMPT RATE IS SAME AS LAST RATE
1220 0520 EB 0658 R      CALL   CHK_LASTRATE      I YES, SKIP SEND RATE COMMAND
1221 0523 74 03      JZ      RWV_DBL          I SEND DATA RATE TO NEC
1222 0525 EB 0637 R      CALL   SEND_RATE
1223
1224 0528 53      RWV_DBL: PUSH   BX      I SAVE MEDIA/DRIVE PARAM ADDRESS
1225 0529 EB 084C R      CALL   SETUP_DBL          I CHECK FOR DOUBLE STEP
1226 052C 5B      POP    BX      I RESTORE ADDRESS
1227 052D 72 1A      JC      CHK_RET            I ERROR FROM READ ID, POSSIBLE RETRY
1228 052E 58      POP    AX      I RESTORE NEC/DMA COMMAND
1229 0530 00      PUSH   AX      I MOVE NEW COMMAND
1230 0531 53      PUSH   BX      I SAVE MEDIA/DRIVE PARAM ADDRESS
1231 0532 EB 0668 R      CALL   DMA_SETUP          I SET UP THE DMA
1232 0535 5B      POP    BX      I RESTORE ADDRESS
1233 0536 58      POP    AX      I RESTORE NEC COMMAND
1234 0537 5B 1F      JC      RWV_BAC            I CHECK FOR DMA BOUNDARY ERROR
1235 0538 59 00      PUSH   AX      I SAVE MEDIA/DRIVE PARAM ADDRESS
1236 053A 53      PUSH   BX      I RESTORE ADDRESS
1237 053B EB 06CB R      CALL   NEC_INIT          I INITIALIZE NEC
1238 053E 5B      POP    BX      I RESTORE ADDRESS

```

```

1239 053F 72 08 JC CHK_RET ; IF ERROR DO NOT SEND MORE COMMANDS
1240 0541 E8 06F1 R CALL RWV_COM ; OP 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: CALL RETRY ; CHECK FOR, SETUP RETRY
1245 0549 E8 07BE R POP AX ; RESTORE READ/WRITE/VERIFY PARAMETER
1246 054C 58 JNC RWV_END ; CY = 0 NO RETRY
1247 054D 73 03 JMP DO AGAIN ; CY = 1 MEANS RETRY
1248 054F E9 04BB R
1249
1250 0552 RWV_END: CALL DSTATE ; ESTABLISH STATE IF SUCCESSFUL
1251 0552 E8 077B R CALL NUM_TRANS ; AL = NUMBER TRANSFERRED
1252 0555 E8 0805 R
1253
1254 0558 RWV_BAC: PUSH AX ; BAD DMA ERROR ENTRY
1255 0558 50 CALL XLAT_OLD ; SAVE NUMBER TRANSFERRED
1256 0559 E8 0432 R POP AX ; TRANSLATE STATE TO COMPATIBLE MODE
1257 055C 58 CALL SETUP_END ; RESTORE NUMBER TRANSFERRED
1258 055D E8 0832 R
1259 0560 C3 RET ; VARIOUS CLEANUPS
1260 0561 RD_WF_VF ENDP
1261
1262 ;-----; : SETUP_STATE: INITIALIZES START AND END RATES.
1263 ;-----;
1264 0561 SETUP_STATE PROC NEAR ; TEST CONTROLLER I.D.
1265 0561 F6 06 008F R 01 TEST .0HF_CNTL,DUAL ; MEDIA DETERMINED ?
1266 0564 74 37 JZ .0DSK_STATE[D1],MED_DET ; NO STATES IF DETERMINED
1267 0568 F6 85 0090 R 10 TEST JZC ; AH = START RATE, AL = END RATE
1268 056D 75 30 JNC .0DSK_STATE[D1],DRV_TYPE ; DRIVE ?
1269 056F B5 4080 MOV AX,RATE_300*H RATE_250 ; DO NOT KNOW DRIVE
1270 0572 F6 85 0090 R 04 TEST JZC ; STEP UP FOR 1.2 M END RATE
1271 0574 75 04 MOV AX,RATE_500 ; JUMP WITH FIXED END RATE
1272 0579 B0 00 TEST JZC ; START & END RATE = 250 FOR 360 DRIVE
1273 057B F6 85 0090 R 02 JNZ AX,SET ; JUMP IF AL 0
1274 0580 75 03 MOV AX,RATE_250*X ; START & END RATE = 250 FOR 360 DRIVE
1275 0582 B8 0800
1276
1277 0585 AX_SET: AND .0DSK_STATE[D1],NOT RATE_MSK+DBL_STEP ; TURN OFF THE RATE
1278 0585 80 A5 0090 R IF OR .0DSK_STATE[D1],AH ; RATE FIRST TO TRY
1279 058A 80 A5 0090 R OR .0LASTRATE,NOT STRT_MSK ; ERASE LAST TO TRY RATE BITS
1280 058E 80 26 008B R F3 AND ROR AL,1 ; TO OPERATION LAST RATE LOCATION
1281 0593 D0 C8 ROR AL,1
1282 0595 00 C8 ROR AL,1
1283 0597 D0 C8 ROR AL,1
1284 0599 00 C8 ROR AL,1
1285 059B 06 008B R OR .0LASTRATE,AL ; LAST RATE
1286 059F C3 JIC1: RET
1287 059F C3 SETUP_STATE ENDP
1288 05A0 ;-----; : FMT_INIT: ESTABLISH STATE IF UNESTABLISHED AT FORMAT TIME.
1289 ;-----;
1290 ;-----;
1291 ;-----;
1292 05A0 FMT_INIT PROC NEAR ; TEST CONTROLLER I.D.
1293 05A0 F6 06 008F R 01 TEST .0HF_CNTL,DUAL ; IS MEDIA ESTABLISHED
1294 05A5 74 49 JZ FI_OUT ; RETURN DRIVE TYPE IN AL
1295 05A6 76 85 0090 R 10 TEST .0DSK_STATE[D1],MED_DET ; IS DRIVE DETERMINED
1296 05A6 7E 42 JZ .0DSK_STATE[D1],DRV_TYPE ; TREAT AS NON 1.2 DRIVE
1297 05A6 E8 04CF R CALL CMOS_TYPE ; ERROR IN CMOS ASSUME NO DRIVE
1298 05B1 72 3E JC CL_DRV ; MAKE ZERO ORIGIN
1299 05B3 FE C8 DEC AL ; NO DRIVE IF AL 0
1300 05B5 78 3A JS CL_DRV ; AH = CURRENT STATE
1301 05B6 80 A5 0090 R MOV AH,.0DSK_STATE[D1] ; AH = NOT MED_DET+DBL_STEP+NOT
1302 05B8 80 04 OF AND AH,.0DSK_STATE[D1],NOT ; CLEAR
1303 05B9 04 C0 OR AL,AL ; IF 360 WILL BE 0
1304 05C0 75 05 JNZ N_360 ; ESTABLISH MEDIA
1305 05C2 80 CC 90 OR AH,MED_DET+RATE_250 ; SKIP OTHER STATE PROCESSING
1306 05C5 EB 25 JMP SHORT SKP_STATE
1307
1308 05C7 N_360: DEC AL ; 1.2 M DRIVE
1309 05C7 FE C8 JNZ N_12 ; JUMP IF NOT
1310 05C9 75 05 OR AH,MED_DET+RATE_500 ; SET FORMAT RATE
1311 05CB 80 CC 10 JMP SHORT SKP_STATE ; SKIP OTHER STATE PROCESSING
1312 05CE EB 1C
1313
1314 05D0 N_12: DEC AL ; CHECK FOR TYPE 3
1315 05D0 FE C8 JNZ N_720 ; JUMP IF NOT
1316 05D2 75 0F TEST AH,DRV_DET ; IS DRIVE DETERMINED
1317 05D4 F6 C4 04 JZ ISNT_12 ; TREAT AS NON 1.2 DRIVE
1318 05D7 74 10 TEST AH,FMT_CAPA ; IS 1.2M
1319 05D9 F6 C4 02 JZ ISNT_12 ; JUMP IF NOT
1320 05D9 75 0A OR AH,MED_DET+RATE_300 ; RATE 300
1321 05DE 80 CC 50 JMP SHORT SKP_STATE ; CONTINUE
1322 05E1 EB 09
1323 05E3 N_720: DEC AL ; CHECK FOR TYPE 4
1324 05E3 FE C8 JNZ CL_DRV ; NO DRIVE, CMOS BAD
1325 05E5 75 0A CALL SHORT FI_RATE
1326 05E5 EB E2
1327 05E9 ISNT_12: OR AH,MED_DET+RATE_250 ; MUST BE RATE 250
1328 05E9 80 CC 90 SKP_STATE: MOV .0DSK_STATE[D1],AH ; STORE AWAY
1329 05EC
1330 05EC 88 A5 0090 R FI_OUT: RET
1331 05F0 C3 CL_DRV: XOR AH,AH ; CLEAR STATE
1332 05F1 CL_DRV: JMP SHORT SKP_STATE ; SAVE IT
1333 05F1 FMT_INIT ENDP
1334 ;-----; : MED_CHANGE
1335 ;-----; : CHECKS FOR MEDIA CHANGE, RESETS MEDIA CHANGE,
1336 ;-----; : CHECKS MEDIA CHANGE AGAIN.
1337 ;-----; : ON EXIT: CY = 1 MEANS MEDIA CHANGE OR TIMEOUT
1338 ;-----; : DSKETTE_STATUS = ERROR CODE
1339 ;-----;
1340 ;-----;
1341 ;-----;
1342 ;-----;
1343 ;-----;
1344 ;-----;
1345 05F5 MED_CHANGE PROC NEAR ; TEST CONTROLLER I.D.
1346 05F5 F6 06 008F R 01 TEST .0HF_CNTL,DUAL ; READ DISK CHANGE LINE STATE
1347 05FA 74 37 JZ OK2 ; BYPASS HANDLING DISK CHANGE LINE
1348 05FC E8 0B21 R CALL READ_DSKCHNG ; AND .0DSK_STATE[D1],NOT MED_DET ; CLEAR STATE FOR THIS DRIVE
1349 05FF 74 34 JZ MC_OUT
1350 0601 80 A5 0090 R EF AND MC_OUT
1351
1352 ;-----; 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 BB 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 0610 06 003F R AND #MOTOR_STATUS,AL ; INTERRUPT MASK INDICATOR
1362 0613 FB STI ; INTERRUPTS ENABLED
1363 0614 EB 0913 R CALL MOTOR_ON ; TURN MOTOR ON
1364
1365 ;----- THIS SEQUENCE OF SEEKS IS USED TO RESET DISKETTE CHANGE SIGNAL
1366
1367 0617 E8 0092 R CALL DISK_RESET ; RESET NEC
1368 061A B5 01 MOV CH,0TH ; MOVE TO CYLINDER 1
1369 061C EB 0A14 R CALL SEEK ; ISSUE SEEK
1370 061F 32 ED XOR CH,CH ; MOVE TO CYLINDER 0
1371 0621 EB 0A14 R CALL SEEK ; ISSUE SEEK
1372 0624 C6 06 0041 R 06 MOV #DSKETTE_STATUS,MEDIA_CHANGE ; STORE IN STATUS
1373
1374 0629 EB 0B21 R OK1: CALL READ_DSKCHNG ; CHECK MEDIA CHANGED AGAIN
1375 062C T4 05 JZ OK2 ; IF ACTIVE, NO DISKETTE, TIMEOUT
1376
1377 062E C6 06 0041 R B0 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 F8 MC_OUT: CLC ; NO MEDIA CHANGED, CLEAR CY
1382 0636 C3 RET
1383 0637 MED_CHANGE ENDP ;-----+
1384
1385 ;-----+
1386 ;-----+
1387 ;-----+
1388 ;-----+
1389 ;-----+
1390 ;-----+
1391 ;-----+
1392 0637 SEND_RATE PROC NEAR ; TEST CONTROLLER I.D.
1393 0637 F6 06 008F R 01 TEST #HF_CNTL,DUAL ;-----+
1394 063C T4 19 C_S_OUT ;-----+
1395 063E 50 PUSH AX ;-----+
1396 0640 26 008B R 3F AND #LASTRATE_NOT_SEND_MSK ;-----+
1397 0644 84 85 0090 R MOV AL,#DSK_STATE[DI] ;-----+
1398 0648 24 C0 AND AL,SEND_MSK ;-----+
1399 064A 08 06 008B R OR #LASTRATE_AL ;-----+
1400 064E D0 C0 ROL AL,1 ;-----+
1401 0650 D0 C0 ROL AL,1 ;-----+
1402 0652 BA 03F7 MOV DX,03FTH ;-----+
1403 0654 EE OUT DX,AL ;-----+
1404 0656 58 POP AX ;-----+
1405 0657 RET ;-----+
1406 0657 C3 SEND_RATE ENDP ;-----+
1407 0658
1408
1409 ;-----+
1410 ;-----+
1411 ;-----+
1412 ;-----+
1413 ;-----+
1414 ;-----+
1415 ;-----+
1416 ;-----+
1417 ;-----+
1418 ;-----+
1419 0658 CHK_LASTRATE PROC NEAR ;-----+
1420 0658 50 PUSH AX ;-----+
1421 0658 84 26 008B R MOV AH,#LASTRATE ;-----+
1422 065D 84 85 0090 R MOV AL,#DSK_STATE[DI] ;-----+
1423 0661 25 C0C0 AND AX,SEND_MSK*X ;-----+
1424 0664 3A C4 CMP AL,AH ;-----+
1425
1426 0666 58 POP AX ;-----+
1427 0667 C3 RET ;-----+
1428 0668 CHK_LASTRATE ENDP ;-----+
1429

```

```

1430
1431
1432
1433 ;----- THIS ROUTINE SETS UP THE DMA FOR READ/WRITE/VERIFY
1434 ; OPERATIONS.
1435
1436 ; ON ENTRY: AL = DMA COMMAND
1437
1438 ; ON EXIT: 0DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
1439 ;----- DMA_SETUP PROC NEAR
1440 0668
1441 0668 FA
1442 0669 E6 0C
1443 066B EB 00
1444 066D EB 0B
1445 066E EC 02
1446 0671 T5 04
1447 0673 33 C0
1448 0675 EB 15
1449 0677
1450 0679 8C C0
1451 0679 D1 C0
1452 067B D1 C0
1453 067D D1 C0
1454 067F D1 C0
1455 0681 8A E0
1456 0683 24 F0
1457 0685 00 02
1458 0688 T3 02
1459 068A FE C5
1460 068C
1461 068C 50
1462 068D E0 04
1463 068E EB 00
1464 0691 8A C4
1465 0693 E6 04
1466 0695 8A C5
1467 0697 EB 00
1468 0699 24 0F
1469 069B E6 81
1470
1471 ;----- DETERMINE COUNT
1472
1473 069D 8B C6
1474 06A1 84 C4
1475 06A1 00 00
1476 06A3 D1 E8
1477 06A5 50
1478 06A6 B2 03
1479 06A8 E8 08FE R
1480 06AB 80 CC
1481 06B0 55
1482 06AE D3 E0
1483 06B0 48
1484 06B1 50
1485 06B2 E6 05
1486 06B4 EB 00
1487 06B5 8A C4
1488 06B8 EB 05
1489 06BA FB
1490 06BB 59
1491 06BC 58
1492 06BD 03 C1
1493 06B8 BB 00
1494 06C1 E6 0A
1495
1496 06C3 T3 05
1497 06C5 C6 06 0041 R 09
1498
1499 06CA
1500 06CA C3
1501 06CB
1502
1503 ;----- NEC_INIT
1504 ; THIS ROUTINE SEEKS TO THE REQUESTED TRACK AND
1505 ; INITIALIZES THE NEC FOR THE READ/WRITE/VERIFY/FORMAT
1506 ; OPERATION.
1507
1508 ; ON ENTRY: AH = NEC COMMAND TO BE PERFORMED
1509
1510 ; ON EXIT: 0DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
1511
1512 06CB
1513 06CB 50
1514 06CC E8 0913 R
1515
1516 ;----- DO THE SEEK OPERATION
1517
1518 06CF 8A 6E 01
1519 06D2 E8 0014 R
1520 06D5 58
1521 06D6 T2 18
1522 06DB BB 06F0 R
1523 06DB 53
1524
1525 ;----- SEND OUT THE PARAMETERS TO THE CONTROLLER
1526
1527 06DC E8 09F0 R
1528 06DF BB C6
1529 06E0 00 DF
1530 06E3 D0 E4
1531 06E5 D0 E4
1532 06E7 80 E4 04
1533 06E8 00 E3
1534 06EC E8 09F0 R
1535 06F0 5B
1536 06F0
1537 06F0 C3
1538 06F1
1539
1540 ;----- RWY_COM
1541 ; THIS ROUTINE SENDS PARAMETERS TO THE NEC SPECIFIC
1542 ; TO THE READ/WRITE/VERIFY OPERATIONS.
1543

```



```

1658
1659 0798 E8 0BCF R CALL CMOS_TYPE ; RETURN DRIVE TYPE IN (AL)
1660 07A2 T2 14 JC M_12_ ; CMOS BAD ASSUME DEFAULT
1661 07A6 TAC 3C 02 CMP AL_2 ; TYPE 2 DRIVE ?
1662 07A6 TAC 00 00 JE M_12 ; YES->ASSUME MULTI FORMAT CAPABILITY
1663 07A8 TAC 3C 04 CMP AL_4 ; TYPE 4 DRIVE
1664 07AA TAC 74 0C JE M_12 ; YES->ASSUME MULTI FORMAT CAPABILITY
1665 07AC M_720: AND #DSK_STATE[DI],NOT FMT_CAPA ; TURN OFF FORMAT CAPABILITY
1666 07AC TAC 80 A5 0090 R FD OR #DSK_STATE[DI],DRV_DET ; MARK DRIVE DETERMINED
1667 07B1 TAC 80 BD 0090 R 04 JMP SHORT SETBAC ; BACK
1668 07B2 TAC EB 05
1669
1670 07B8 M_12: OR #DSK_STATE[DI],DRV_DET+FMT_CAPA ; TURN ON DETERMINED & FMT CAPA
1671 07B8 80 BD 0090 R 06
1672
1673 07BD SETBAC:
1674 07BD C3 RET
1675 07BE DSTATE ENDP
1676
1677 : RETRY
1678 : DETERMINES WHETHER A RETRY IS NECESSARY. IF RETRY IS
1679 : REQUIRED THEN STATE INFORMATION IS UPDATED FOR RETRY.
1680
1681 : ON EXIT: CY = 1 FOR RETRY, CY = 0 FOR NO RETRY
1682
1683 07BE RETRY PROC NEAR
1684 07BE 80 3E 0041 R 00 CMP #DSKETTE_STATUS,0 ; GET STATUS OF OPERATION
1685 07C3 74 3E JZ NO_RETRY ; SUCCESSFUL OPERATION
1686 07C5 80 3E 0041 R 80 CMP #DSKETTE_STATUS,TIME_OUT ; IF TIME OUT BUT NOT RETRY
1687 07C6 74 37 JZ NO_RETRY
1688 07D0 F6 C4 00 TEST AH,MEDIA STATE[DI] ; GET MEDIA STATE OF DRIVE
1689 07D0 F6 C4 00 AH,MDN_BET ; ESTABLISHED/DETERMINED ?
1690 07D3 75 2E JNZ NO_RETRY ; IF ESTABLISHED STATE THEN TRUE ERROR
1691 07D5 80 E4 C0 AND AH,RATE_MSK ; ISOLATE RATE
1692 07D6 82 E4 00 MOV CH,DL@RATE ; GET START OPERATION STATE
1693 07D8 D0 C5 ROL CH,1 ; TO CORRESPONDING BITS
1694 07D9 D0 C5 ROL CH,1
1695 07E0 D0 C5 ROL CH,1
1696 07E2 D0 C5 AND CH,RATE_MSK ; ISOLATE RATE BITS
1697 07E4 80 E5 C0 CMP CH,AH ; ALL MEDIUMS TRIED
1698 07E7 3A EC NO_RETRY ; IF YES, THEN TRUE ERROR
1699 07E9 T4 18
1700
1701 : SETUP STATE INDICATOR FOR RETRY ATTEMPT TO NEXT RATE
1702 : 0000000B (500) -> 1000000B (250)
1703 : 1000000B (250) -> 0100000B (300)
1704 : 0100000B (300) -> 0010000B (500)
1705
1706 07EB 80 FC 01 CMP AH,RATE_500+1 ; SET CY FOR RATE 500
1707 07EE D0 DC RCR AH,1 ; TO NEXT STATE
1708 07F0 80 E4 C0 AND AH,RATE_MSK ; KEEP ONLY RATE BITS
1709 07F3 80 A5 0090 R IF AND #DSK_STATE[DI],NOT RATE_MSK+DBL_STEP ; 1 RATE, DBL STEP OFF
1710 07F8 08 A5 0090 R OR #DSK_STATE[DI],AH ; TURN ON NEW RATE
1711 07F7 C6 06 0041 R 00 MOV #DSKETTE_STATUS,0 ; RESET STATUS FOR RETRY
1712 0801 F9 STC ; SET CARRY FOR RETRY
1708 0802 C3 RET ; RETRY RETURN
1714
1715 0803 NO_RETRY: CLC ; CLEAR CARRY NO RETRY
1716 0803 F8 RET ; NO RETRY RETURN
1717 0804 C3
1718 0805
1719
1720 : NUM_TRANS THIS ROUTINE CALCULATES THE NUMBER OF SECTORS THAT
1721 : WERE ACTUALLY TRANSFERRED TO/FROM THE DISKETTE.
1722
1723 : ON ENTRY: [BP+I] = TRACK
1724 : SI-HI = HEAD
1725 : [BP] = START SECTOR
1726
1727 : ON EXIT: AL = NUMBER ACTUALLY TRANSFERRED
1728
1729 0805 NUM_TRANS PROC NEAR
1730 0805 XOR AL,1 ; CLEAR FOR ERROR
1731 0805 32 C0 CMP #DSKETTE_STATUS,0 ; CHECK FOR ERROR
1732 0805 10 00 JE 0041 R 00 JNZ NT_OUT ; IF ERROR 0 TRANSFERRED
1733 080C 75 23 MOV DL-4 ; SECTORS/TRACK OFFSET TO DL
1734 080E B2 04 MOV AH,1 ; SECTORS/TRACK
1735 0810 E0 08FE R CALL GET_PARM ; GET PARM
1736 0813 8A 0047 R MOV BL,_NEC_STATUS+5 ; GET ENDING SECTOR
1737 0817 8B CE MOV CX,SI ; CH = HEAD @ STARTED
1738 0819 8A 0046 R CMP CH,_NEC_STATUS+4 ; GET HEAD ENDED UP ON
1739 081D 75 0B JNZ DIF_HD ; IF ON SAME HEAD, THEN NO ADJUST
1740
1741 081F 8A 0045 R MOV CH,_NEC_STATUS+3 ; GET TRACK ENDED UP ON
1742 0823 8A 06 01 CMP CH,[BP+T] ; IS IT ASKED FOR TRACK
1743 0826 74 04 JZ SAME_TRK ; IF SAME TRACK NO INCREASE
1744
1745 0828 02 DC ADD BL,AH ; ADD SECTORS/TRACK
1746 082A ADD BL,AH ; ADD SECTORS/TRACK
1747 082A 02 DC SAME_TRK: SUB BL,[BP] ; SUBTRACT START FROM END
1748 082C SUB AL,BL ; TO AL
1749 082C 2A 5E 00
1750 082F 8A C3
1751
1752 0831 NT_OUT: RET
1753 0831 C3 NUM_TRANS ENDP
1754 0832
1755
1756 : SETUP_END AX ; RESTORES #MOTOR COUNT TO PARAMETER PROVIDED IN TABLE
1757 : AND LOADS #DSKETTE_STATUS TO AH, AND SETS CY.
1758
1759 : ON EXIT: AH, #DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
1760
1761 SETUP_END PROC NEAR
1762 0835 XOR DL,2 ; GET THE MOTOR WAIT PARAMETER
1763 0834 50 50 PUSH AX ; SAVE NUMBER TRANSFERRED
1764 0835 E5 08FE R CALL GET_PARM ; STORE UPON RETURN
1765 0838 8A 26 0040 R MOV #MOTOR_COUNT,AH ; RESTORE NUMBER TRANSFERRED
1766 0838 5C 58 POP AX ; GET STATUS OF OPERATION
1767 0839 8A 26 0041 R MOV AH,#DSKETTE_STATUS ; GET STATUS OF OPERATION
1768 0839 8A 26 0041 R OR AH,1 ; GET NUMBER TRANSFERRED
1769 0843 44 02 JZ NUN_ERR ; NO ERROR
1770 0843 44 02 XOR AL,AL ; CLEAR NUMBER RETURNED
1771 0845 32 C0

```

```

1772
1773 0847 NUN_ERR:
1774 0847 80 FC 01 CMP AH,1 ; SET THE CARRY FLAG TO INDICATE
1775 084A F5 CMC ; SUCCESS OR FAILURE
1776 084B C3 RET
1777 084C SETUP_END ENDP
-----+
1778
1779 ; SETUP_DBL
1780 ; -CHECK DOUBLE STEP,
1781 ; ON ENTRY: DI = DRIVE
1782 ; ON EXIT: CY = 1 MEANS ERROR
1783
1784 ;-----+
1785 084C SETUP_DBL PROC NEAR
1786 084C F6 06 008F R 01 TEST OFH_CNTL,DUAL ; TEST CONTROLLER I.D.
1787 0851 74 65 JZ NO_DBL ; NO DOUBLE STEPPING REQUIRED
1788 0852 80 A5 0090 R MOV AH,0DSK_STATE[DI] ; ACCESS STATE
1789 0857 65 10 TEST AH,RED_DET ; ESTABLISHED STATE ?
1790 085A 75 5C UNZ 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 MOOR_ON ; EXECUTE MOVE TO STAY ON
1796 0864 B5 00 MOV CH,1 ; LOAD TRACK 0
1797 0866 E8 0414 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 0DSK_STATE[DI],TRK_CAPA ; TEST FOR 80 TRACK CAPABILITY
1805 0876 74 02 JZ CNT_0K ; IF NOT COUNT IS SETUP
1806 0878 B1 A0 MOV CL,0AH ; 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 READY; IF NOT
1810 ; THEN SET DOUBLE STEP ON.
1811
1812 087A CNT_0K:
1813 087A 51 PUSH CX ; SAVE TRACK, COUNT
1814 087B C6 06 0041 R 00 MOV 0DSKETTE_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 0414 R CALL SEEK ; SEEK TO TRACK
1822 088E 58 POP AX ; RESTORE HEAD
1823 088F 0B F8 OR DI,AX ; DI = HEAD OR'D ED DRIVE
1824 0890 E8 08BA R CALL READ_ID ; READ ID/HEAD 0
1825 0894 50 PUSHF ; SAME AS FROM READ_ID
1826 0895 81 E7 00FB AND DI,1111011B ; 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 D1 JNZ CNT_0K ; 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 08A7 80 D0 0094 R MOV 0DSK_TRK[DI],CL ; STORE TRACK NUMBER
1843 08A9 D0 ED SHR CH,1 ; HALVE TRACK
1844 08AF 3A E9 CMP CH,CL ; 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 0DSK_STATE[DI],DBL_STEP ; TURN ON DOUBLE STEP REQUIRED
1847
1848 0856 NO_DBL: CLC ; CLEAR ERROR FLAG
1849 0859 F8 RET
1850 0859 C3
1851 08BA
1852
1853 ;-----+ READ_ID
1854 ; READ ID FUNCTION.
1855 ; ON ENTRY: DI = BIT 2 = HEAD; BITS 1,0 = DRIVE
1856 ; ON EXIT: DI = BIT 2 IS RESET, BITS 1,0 = DRIVE
1857 ; 0DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
1858
1859 08BA READ_ID PROC NEAR
1860 08BA B8 08CE R PUSH AX ; MOVE NEC OUTPUT ERROR ADDRESS
1862 08BD 50 MOV AH,4AH ; READ ID COMMAND
1863 08BE B4 4A CALL NEC_OUTPUT ; TO CONTROLLER
1864 08C0 E8 09F0 R MOV AX,DI ; DRIVE # TO AH, HEAD 0
1865 08C3 8B C7 MOV AH,AL
1866 08C5 8A E0 CALL NEC_OUTPUT ; TO CONTROLLER
1867 08C7 8B 00 CALL NEC_TERM ; WAIT FOR OPERATION, GET STATUS
1868 08CA E8 0727 R POP AX ; THROW AWAY ERROR ADDRESS
1869 08CD 58
1870 08CE
1871 08CE C3
1872 08CF RET
1873
1874 ;-----+ CMOS_TYPE
1875 ; RETURNS DISKETTE TYPE FROM CMOS
1876
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,1100001B ; KEEP DISKETTE DRIVE BITS
1884 08D4 D0 E8 SHR AL,1 ; ARE THERE ANY DRIVES INSTALLED?
1885 08D6 73 20 JNC TYP_ZERO ; NC--> NO DRIVES TYPE ZERO

```

```

1886 08D8 D0 C0      ROL  AL,1           ; ROTATE TO ORIGINAL POSITION
1887 08D9 D0 C0      ROL  AL,1           ; ROTATE BITS 6 AND 7 TO 0 AND 1
1888 08E0 D0 C0      XOR  AH,AH          ; AX=NUMBER OF DRIVES
1889 08E0 32 E4      CMP  AL,DI          ; IS DRIVE REQUESTED PRESENT
1890 08E0 3B C7      JC   TYP_ZERO        ; C-->REQUESTED DRIVE NOT PRESENT
1891 08E2 72 14      TEST 0HF-CNTRL,DUAL ; TEST CONTROLLER I.D.
1892 08E4 F6 06 00BF R 01    TEST 0DSK_STATE[DI],TRK_CAPA ; TEST FOR 80 TRACKS
1893 08E9 75 10      MOV  AL,1           ; DRIVE TYPE HAS 40 TRACKS
1894 08E9 75 10      JNZ  CR2           ; DRIVE TYPE HAS 80 TRACKS
1895 08F0 80 01      TEST 0DSK_STATE[DI],TRK_CAPA ; TEST FOR 80 TRACKS
1896 08F2 74 06      JZ   CR1           ; DRIVE TYPE HAS 40 TRACKS
1897 08F4 B0 03      MOV  AL,3           ; DRIVE TYPE HAS 80 TRACKS
1898 08F6 EB 02      JMP  SHORT CR1
1899 08F8             TYP_ZERO:
1900 08F9 32 C0      XOR  AL,AL          ; DRIVE TYPE 0
1901 08FA             CR1:  RET
1902 08FB C3         CR2:  RET
1903 08FB             STC
1904 08FB F9         JMP  CR1           ; EXIT WITH CARRY IF DUAL CARD
1905 08FC EB FC
1906 08FE             CMOS_TYPE: ENDP
1907             -----
1908             GET_PARM:
1909             THIS ROUTINE FETCHES THE INDEXED POINTER FROM THE
1910             DISK BASE BLOCK POINTED TO BY THE DATA VARIABLE
1911             0DSK_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 1E         PUSH DS
1922 08FF 56         PUSH SI
1923 0900 2B C0         SUB  AX,AX          ; DS = 0 , BIOS DATA AREA
1924 0902 8E D8         MOV  DS,AX
1925 0904 87 D3         XCHG DX,AX
1926 0906 2A FF         SUB  BX,BH          ; BL = INDEX
1927             ASSUME DS:1AB50
1928 0908 C5 36 0078 R    LDS  SI,0DSK_POINTER ; POINT TO BLOCK
1929 090C 8A 20         MOV  AH,[SI+BX] ; GET THE WORD
1930 090E 8A D3         XCHG DX,BX
1931 0910 9E         POP  SI
1932 0911 1F         POP  DS
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 (0FFH) 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             THAT IS BUILT IN IS CALLED TO WAIT FOR MOTOR START UP. IF THE
1943             FUNCTION RETURNS WITH CY=1 IT MEANS THAT THE MINIMUM WAIT
1944             HAS BEEN COMPLETED. AT THIS POINT A CHECK IS MADE TO ENSURE
1945             THAT THE MOTOR WASN'T TURNED OFF BY THE TIMER. IF THE HOOK DID
1946             NOT WAIT, THE WAIT FUNCTION (AH=086H) IS CALLED TO WAIT THE
1947             PRESCRIBED AMOUNT OF TIME. IF THE CARRY FLAG IS SET ON RETURN,
1948             IT MEANS THAT THE FUNCTION IS IN USE AND DID NOT PERFORM THE
1949             WAIT. A TIMER1 WAIT LOOP WILL THEN DO THE WAIT.
1950
1951             ON ENTRY:  DI = DRIVE #
1952
1953             ON EXIT:   AX,CX,DX DESTROYED
1954             -----
1955             -----
1956 0913             MOTOR_ON PROC NEAR
1957 0913 53         PUSH BX          ; SAVE REG.
1958 0914 E8 095E R    CALL TURN_ON       ; TURN ON MOTOR
1959 0917 72 43         JC   MOT_TS_ON     ; IF CY=1 NO WAIT
1960 0919 E8 0432 R    CALL XLAT_OLD     ; TRANSLATE STATE TO COMPATIBLE MODE
1961 091B 80 00         MOV  MOT_TS,00FDH ; LOAD SYSTEM TYPE
1962 091F CD 15         INT  15H          ; TELL OPERATING SYSTEM ABOUT TO DO WAIT
1963 0921 9C         PUSHF
1964 0922 E8 0404 R    CALL XLAT_NEW     ; SAVE CY FOR TEST
1965 0925 9D         POPF
1966 0926 73 05         JNC  M_WAIT       ; TRANSLATE STATE TO PRESENT ARCH.
1967 0928 E8 095E R    CALL TURN_ON     ; RESTORE CY FOR TEST
1968 0928 72 2F         JC   MOT_TS_ON     ; BYPASS LOOP IF OS SYSTEM HANDLED WAIT
1969             -----
1970 092D             M_WAIT:          ; CHECK AGAIN IF MOTOR ON
1971 092D B2 0A         MOV  DL,10          ; IF NO WAIT MEANS IT IS ON
1972 092E E8 08FE R    CALL GET_PARM
1973 0932 8C         MOV  AL,TH          ; AL = MOTOR WAIT PARAMETER
1974 0934 32 E4         XOR  AH,AH          ; AX = MOTOR WAIT PARAMETER
1975 0936 3C 08         CMP  AL,0           ; SEE IF AT LEAST A SECOND IS SPECIFIED
1976 0938 73 02         JAE  GP2          ; IF YES, CONTINUE
1977 093A B0 08         MOV  AL,B           ; ONE SECOND WAIT FOR MOTOR START UP
1978
1979             -----
1980             AX CONTAINS NUMBER OF 1/8 SECONDS (125000 MICROSECONDS) TO WAIT
1981 093C 50         GP2:  PUSH AX          ; SAVE WAIT PARAMETER
1982 093D BA F424     MOV  DX,62500        ; LOAD LARGEST POSSIBLE MULTIPLIER
1983 0940 F7 E2         MUL  DX
1984 0941 BB CA         MOV  CX,DX          ; MULTIPLY BY HALF OF WHAT'S NECESSARY
1985 0944 8B D0         MOV  DX,AX          ; CX = HIGH WORD
1986 0946 F8         CLC
1987 0947 D1 D2         RCL  DX,1           ; CLEAR CARRY FOR ROTATE
1988 0949 D1 D1         RCL  CX,1           ; DOUBLE LOW WORD, CY CONTAINS OVERFLOW
1989 094B B4 84         MOV  AH,86H          ; DOUBLE HI, INCLUDING LOW WORD OVERFLOW
1990 094D CD 15         INT  15H          ; LOAD WAIT CODE
1991 094F 58         POP  AX
1992 0950 T3 0A         JNC  MOT_IS_ON    ; RESTORE WAIT PARAMETER
1993             -----
1994             -----
1995             FOLLOWING LOOPS REQUIRED WHEN RTC WAIT FUNCTION IS ALREADY IN USE
1996             -----
1997 0952 B9 205E     J13:  MOV  CX,8286        ; WAIT FOR 1/8 SECOND PER (AL)
1998 0955 E8 0000 E    CALL WAITF          ; COUNT FOR 1/8 SECOND AT 15.085737 US
1999 0958 FE C8         DEC  AL           ; GO TO FIXED WAIT ROUTINE
                                         ; DECREMENT TIME VALUE

```

```

2000 095A 7F F6 JNZ J13 ; ARE WE DONE YET
2001
2002 095C MOT_IS_ON; ; RESTORE REG.
2003 095C 5B POP BX
2004 095D C3 RET
2005 095E MOTOR_ON ENDP
2006
2007 ; TURN_ON ;-----+
2008 ;-----+ TURN MOTOR ON AND RETURN WAIT STATE.
2009
2010 ; ON ENTRY: DI = DRIVE #
2011 ; ON EXIT: CY = 0 MEANS WAIT REQUIRED
2012 ; CY = 1 MEANS NO WAIT REQUIRED
2013 ; AX,BX,CX,DX DESTROYED
2014 ;-----+
2015 095E ;-----+
2016 095E 8B DF TURN_ON PROC NEAR
2017 0960 8A CB MOV BX,DI ; BX = DRIVE #
2018 0960 8A CB MOV CL,DL ; CL = DRIVE #
2019 0962 D0 C3 ROL BL,1 ; BL = DRIVE SELECT
2020 0964 D0 C3 ROL BL,1
2021 0964 D0 C3 ROL BL,1
2022 0964 D0 C3 ROL BL,1
2023 0964 D0 C3 CLI ;-----+
2024 0968 C4 06 0040 R FF MOV #MOTOR_COUNT,0FFH ; 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,000110000B ; GET DIGITAL OUTPUT REGISTER REFLECTION
2027 0975 B4 01 MOV AH,1 ; KEEP ONLY DRIVE SELECT BITS
2028 0977 D2 E4 SHL AH,CL ; MASK FOR DETERMINING MOTOR BIT
2029 ;-----+
2030 ; AL = DRIVE SELECT FROM #MOTOR_STATUS
2031 ; BL = DRIVE SELECT DESIRED
2032 ; AH = MOTOR ON MASK DESIRED
2033
2034 0979 3A C3 CMP AL,BL ; REQUESTED DRIVE ALREADY SELECTED ?
2035 097B 75 36 JNZ TURN_IT_ON ; IF NOT SELECTED JUMP
2036 097D 20 0003F 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 ;-----+
2041 0983 0A E3 MOV BH,#MOTOR_STATUS ; AH = DRIVE SELECT AND MOTOR ON
2042 0989 00 0F AND BH,00000111B ; SAVE COPY OF #MOTOR_STATUS BEFORE
2043 098C 80 26 003F R CO AND #MOTOR_STATUS,1000000B ; KEEP ONLY DRIVE SELECT BITS
2044 0991 08 26 003F R OR #MOTOR_STATUS,AH ; CLEAR OUT DRIVE SELECT AND MOTORS
2045 0994 A0 003F R MOV AL,#MOTOR_STATUS ; OR IN DRIVE SELECTED
2046 0994 8A D8 MOV BL,AL ; GET DIGITAL OUTPUT REGISTER REFLECTION
2047 0994 B0 E3 0F AND BL,00000111B ; BH=MOTOR STATUS AFTER, BH BEFORE
2048 0994 FB 00 AND AL,00000111B ; KEEP ONLY MOTOR BITS
2049 099E 3F RSTI ; 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 DC 0C OR AL,000001100B ;-----+
2055 09A8 00 03F2 MOV DX,03F2H ; NO RESET, ENABLE DMA/INTERRUPT
2056 09A0 EE 00 OUT DX,BH ; SELECT DRIVE AND TURN ON MOTOR
2057 09A4 3A DF CMP BL,BH ;-----+
2058 09B0 T4 02 JZ NO_MOT_WAIT ; NEW MOTOR TURNED ON ?
2059 09B2 F8 02 CLC ;-----+
2060 09B3 C3 RET ; NO_WAIT REQUIRED IF JUST SELECT
2061 ;-----+
2062 09B4 NO_MOT_WAIT: ; SET CARRY MEANING WAIT
2063 09B4 F9 STC ;-----+
2064 09B5 FB STI ;-----+
2065 09B5 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: ;-----+
2076 09B7 B2 09 PROC NEAR ;-----+
2077 09B8 E8 00FE R CALL GET_PARM ;-----+
2078 09B8 F6 06 003F R B0 TEST #MOTOR_STATUS,1000000B ;-----+
2079 09C0 74 09 JZ ISNT_WRITE ;-----+
2080 09C3 00 00010F CMP JAE DO_WAT ;-----+
2081 09C6 73 08 DO_WAT ;-----+
2082 09C4 B4 0F MOV AH,15 ;-----+
2083 09CA EB 04 JMP SHORT_DO_WAT ;-----+
2084 09C6 ISNT_WRITE: ;-----+
2085 09C6 0A E4 OR AH,AH ;-----+
2086 09C6 74 1F JZ HW_DONE ;-----+
2087 ;-----+ CHECK FOR WAIT TO BE ZERO
2088 ;-----+ IF NOT WRITE AND 0 THEN EXIT
2089 ;-----+ AH CONTAINS NUMBER OF MILLISECONDS TO WAIT
2090 09D0 DO_WAT: ;-----+
2091 09D0 8A C4 MOV AL,AH ;-----+
2092 09D4 32 E4 XOR AH,AH ;-----+
2093 09D4 50 PUSH DX ;-----+
2094 09D4 BA 03E8 MOV DX,1000 ;-----+
2095 09DA F7 E2 MUL DX ;-----+
2096 09DA BB CA MOV CX,DX ;-----+
2097 09DB BB D0 MOV DX,AH ;-----+
2098 09E0 B8 B6 MOV AH,B6H ;-----+
2099 09E0 CD 15 INT 1FH ;-----+
2100 09E2 58 00 POP AX ;-----+
2101 09E3 73 0A JNC HW_DONE ;-----+
2102 ;-----+ RESTORE HEAD SETTLE PARAMETER
2103 09E5 J29: ;-----+ CHECK FOR EVENT WAIT ACTIVE
2104 09E5 B9 0042 MOV CX,66 ;-----+
2105 09E8 EB 0000 E CALL J11F ;-----+
2106 09E8 FE C8 DEC AL ;-----+
2107 09E9 75 F6 JNZ J29 ;-----+
2108 09EF HW_DONE: ;-----+
2109 09EF C3 RET ;-----+
2110 09F0 HD_WAIT ENDP ;-----+
2111 ;-----+
2112 ;-----+ NEC_OUTPUT ;-----+
2113 ;-----+ 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,1000000B   ; STATUS 1 AND DIRECTION 0 ?
2139 09FD 74 0F  JZ   J27         ; 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   @DSKETTE_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   AX          ; INDICATE ERROR TO CALLER
2153 0A0D C3    RET
2154
2155 ;----- DIRECTION AND STATUS OK; OUTPUT BYTE
2156
2157 0A0E 8A C4  J27: 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
2163 0A14    NEC_OUTPUT ENDP
2164
2165 ;----- SEEK
2166 ; THIS ROUTINE WILL MOVE THE HEAD ON THE NAMED DRIVE
2167 ; TO THE NAMED TRACK. IF THE DRIVE HAS NOT BEEN ACCESSED
2168 ; SINCE THE DRIVE RESET COMMAND WAS ISSUED, THE DRIVE
2169 ; WILL BE RECALIBRATED.
2170
2171 ; ON ENTRY: DI = DRIVE #
2172 ; CH = TRACK #
2173
2174 ; ON EXIT: @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION.
2175 ; AX,BX,CX,DX DESTROYED
2176
2177 0A14    SEEK PROC NEAR
2178 0A14 BB 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 B0 01  MOV  AL,I       ; ESTABLISH MASK FOR RECALIBRATE TEST
2182 0A1B 80 CB  XCHG CL,CL      ; GET DRIVE VALUE INTO CL
2183 0A1E D2 00  ROL  AL,CL      ; SHIFT MASK BY THREE POSITION
2184 0A20 86 CB  XCHG CL,CL      ; RECOVER TRACK VALUE
2185 0A22 84 00 003E R  TEST AL,@SEEK_STATUS  ; TEST FOR RECALIBRATE REQUIRED
2186 0A26 75 21  JNZ  J28A      ; JUMP IF RECALIBRATE NOT REQUIRED
2187
2188 0A28 08 06 003E R  OR   @SEEK_STATUS,AL  ; TURN ON THE NO RECALIBRATE BIT IN FLAG
2189 0A2D 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  @DSKETTE_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 B3 FF 01  CMP  DI,!      ; IF REQUEST FOR DRIVE > 2
2200 0A40 77 21  JA   R8         ; DO SEEK EVERY TIME
2201 0A40 00 05 0094 R 00  MOV  DSK_TRK[DI],0  ; SAVE CYLINDER AS PRESENT POSITION
2202 0A45 04 ED  OR   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 B3 FF 01  CMP  DI,!      ; IF REQUEST FOR DRIVE > 2
2209 0A4C 77 13  JA   R8         ; DO SEEK EVERY TIME
2210 0A4E F6 85 0090 R 20  TEST @DSK_STATE[DI],DBL_STEP  ; CHECK FOR DOUBLE STEP REQUIRED
2211 0A53 74 02  JZ   R7         ; SINGLE STEP REQUIRED BYPASS DOUBLE
2212 0A55 DD E5  SHL  CH,!      ; DOUBLE NUMBER OF STEP TO TAKE
2213
2214 0A57 3A AD 0094 R  R7: CMP  CH,@DSK_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 B8 AD 0094 R  R8: MOV  @DSK_TRK[DI],CH  ; SAVE NEW CYLINDER AS PRESENT POSITION
2218
2219 0A61 51    PUSH CX          ; SAVE CYLINDER #
2220 0A62 B4 0F  MOV  AH,0FH      ; SEEK COMMAND TO NEC
2221 0A64 E8 09F0 R  CALL NEC_OUTPUT
2222 0A67 BB DF  MOV  BX,DI      ; BX = DRIVE #
2223 0A69 B8 E3  MOV  AH,BL      ; OUTPUT DRIVE NUMBER
2224 0A6A 80 00 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

```

```

2228
2229
2230
2231 0A75
2232 0A75 9C
2233 0A76 E8 09B7 R
2234 0A79 9D
2235 0A7A
2236 0A7B 58
2237 0A7B
2238 0A7B C3
2239 0A7C
2240
2241
2242
2243
2244
2245
2246
2247
2248 0A7C
2249 0A7C 51
2250 0A7D B8 0A91 R
2251 0A80 50
2252 0A81 B4 07
2253 0A83 E8 09F0 R
2254 0A84 E8 09E0
2255 0A88 B8 E3
2256 0A8A E8 09F0 R
2257 0A8D E8 0A93 R
2258 0A90 58
2259 0A91
2260 0A91 59
2261 0A92 C3
2262 0A93
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272 0A93
2273 0A93 B8 0A81 R
2274 0A96 50
2275 0A97 E8 0ABA R
2276 0A9A 72 14
2277 0A9C B8 06
2278 0A9D E8 09F0 R
2279 0A91 E8 0AE2 R
2280 0A44 72 0A
2281 0A66 A0 0042 R
2282 0A99 24 60
2283 0AAB 3C 60
2284 0A9B 74 03
2285 0AF F8
2286 0A80
2287 0A80 58
2288 0A81 C3
2289 0A81
2290
2291 0A82
2292 0A82 80 0E 0041 R 40
2293 0A87 F9
2294 0ABB EB F6
2295 0ABA
2296
2297
2298
2299
2300
2301
2302
2303
2304 0ABA
2305 0ABA FB
2306 0ABB F8
2307 0ACB BB 9001
2308 0ACD CD 15
2309 0AC1 T2 11
2310
2311 0AC3 B3 04
2312 0AC5 33 C9
2313
2314 0AC7 F6 06 003E R 80
2315 0AC7 T5 0C
2316 0ACE E2 F7
2317 0AD0 FE CB
2318 0AD2 T5 F3
2319
2320 0AD4 80 0E 0041 R 80
2321 0AD9 F9
2322 0ADA
2323 0ADA 9C
2324 0ADB 80 26 003E R 7F
2325 0AEO 9D
2326 0AEC C3
2327 0AE2
2328
2329
2330
2331
2332
2333
2334
2335
2336 0AE2
2337 0AE2 57
2338 0AE3 BF 0042 R
2339 0AE6 B8 07
2340 0AE8 BA 03F4
2341

;----- WAIT FOR HEAD SETTLE
DO_WAIT:
    PUSHF
    CALL    HD_WAIT
    POPF
    RB:
        POP    AX
        NEC_ERR: RET
        SEEK    ENDP
;----- RECALIBRATE DRIVE
;----- ON ENTRY DI = DRIVE #
;----- ON EXIT: CY REFLECTS STATUS OF OPERATION.
RECAL PROC NEAR
    PUSH   CX
    MOV    AX,OFFSET RC_BACK ; LOAD NEC_OUTPUT ERROR
    PUSH   AX
    CALL   NEC_OUTPUT
    MOV    AH,07H ; RECALIBRATE COMMAND
    CALL   NEC_OUTPUT
    MOV    BX,B1 ; BX = DRIVE #
    MOV    AL,0BL
    CALL   CHK_STAT_2 ; OUTPUT THE DRIVE NUMBER
    CALL   NEC_OUTPUT
    CALL   CHK_STAT_2 ; GET THE INTERRUPT AND SENSE INT STATUS
    POP    AX
    RC_BACK:
        POP    CX
        RET
        RECAL    ENDP
;----- CHK_STAT_2
;----- THIS ROUTINE HANDLES THE INTERRUPT RECEIVED AFTER
;----- RECALIBRATION. IT IS CALLED FROM THE ADAPTER. THE
;----- INTERRUPT IS WAITED FOR, THE INTERRUPT STATUS SENSED,
;----- AND THE RESULT RETURNED TO THE CALLER.
;----- ON EXIT: #DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION.
CHK_STAT_2 PROC NEAR
    MOV    AX,OFFSET CS_BACK ; LOAD NEC_OUTPUT ERROR ADDRESS
    PUSH   AX
    CALL   NEC_OUTPUT
    JC    J34 ; WAIT FOR THE INTERRUPT
    MOV    AH,0BH ; IF ERROR, RETURN IT
    CALL   NEC_OUTPUT
    CALL   CHK_STAT_2 ; SENSE INTERRUPT STATUS COMMAND
    JC    J34 ; READ IN THE RESULTS
    CALL   NEC_OUTPUT
    JC    J34 ; GET THE FIRST STATUS BYTE
    AND   AL,01100000B ; ISOLATE THE BITS
    CMP   AL,01100000B ; TEST FOR CORRECT VALUE
    JZ    J35 ; IF ERROR, SO MARK IT
    CLC
    J34: POP    AX
    CS_BACK: RET
;----- J35:
;----- OR    #DSKETTE_STATUS,BAD_SEEK
;----- STC
;----- JMP    SHORT J34 ; ERROR RETURN CODE
CHK_STAT_2 ENDP
;----- WAIT_INT
;----- THIS ROUTINE WAITS FOR AN INTERRUPT TO OCCUR A TIME OUT
;----- ROUTINE TAKES PLACE DURING THE WAIT, SO THAT AN ERROR
;----- MAY BE RETURNED IF THE DRIVE IS NOT READY.
;----- ON EXIT: #DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION.
WAIT_INT PROC NEAR
    STI
    CLC
    MOV    AX,09001H ; TURN ON INTERRUPTS, JUST IN CASE
    INT    15H ; CLEAR TIMEOUT INDICATOR
    JC    J36A ; LOAD WAIT CODE AND TYPE
    INT    15H ; PERFORM OTHER FUNCTION
    JC    J36A ; BYPASS TIMING LOOP IF TIMEOUT DONE
;----- J36:
;----- TEST  #SEEK_STATUS,INT_FLAG ; TEST FOR INTERRUPT OCCURRING
;----- JNZ    J37 ; COUNT DOWN WHILE WAITING
;----- LOOP   J36 ; SECOND LEVEL COUNTER
;----- DEC    BL
;----- JNZ    J36 ; CLEAR THE COUNTERS
;----- XOR    CX,CX ; FOR 2 SECOND WAIT
;----- J36A: OR    #DSKETTE_STATUS,TIME_OUT ; NOTHING HAPPENED
;----- STC
;----- J37: PUSHF
;----- AND    #SEEK_STATUS,NOT INT_FLAG ; SAVE CURRENT CARRY
;----- POPF
;----- RET    ; TURN OFF INTERRUPT FLAG
;----- J36A: RET    ; RECOVER CARRY
;----- J36A: RET    ; GOOD RETURN CODE
WAIT_INT ENDP
;----- RESULTS
;----- THIS ROUTINE WILL READ ANYTHING THAT THE NEC CONTROLLER
;----- RETURNS FOLLOWING AN INTERRUPT.
;----- ON EXIT: #DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION.
;----- AX,BX,CX,DX DESTROYED
RESULTS PROC NEAR
    PUSHF
    MOV    D1,OFFSET #NEC_STATUS ; POINTER TO DATA AREA
    MOV    BL,7 ; MAX STATUS BYTES
    MOV    DX,03F4H ; STATUS PORT

```

```

2342           I----- WAIT FOR REQUEST FOR MASTER
2343
2344 0AEB B7 02   R10:  MOV    BH,2          ; HIGH ORDER COUNTER
2345 0AED 33 C9   XOR    CX,CX        ; COUNTER
2346 0AEF         J39:  IN     AL,DX          ; WAIT FOR MASTER
2347 0AEF EC       AND    AL,1100000B    ; GET STATUS
2348 0AF0 24 C0   CMP    AL,1100000B    ; KEEP ONLY STATUS AND DIRECTION
2349 0AF2 3C C0   JZ     J42           ; STATUS I AND DIRECTION 0 ?
2350 0AF4 74 0E   LOOP   J39           ; STATUS AND DIRECTION OK
2351 0AF6 E2 F7
2352
2353 0AF8 FE CF   DEC    BH             ; DECREMENT HIGH ORDER COUNTER
2354 0AFA 75 F3   JNZ    J39           ; REPEAT TILL DELAY DONE
2355
2356 0AFC 80 0E 0041 R 80  OR    #DSKETTE_STATUS,TIME_OUT
2357 0B01 F9       STC    SC             ; SET ERROR RETURN
2358 0B02 EB 1B   JMP    SHORT POPRES  ; POP REGISTERS AND RETURN
2359
2360           I----- READ IN THE STATUS
2361
2362 0B04         J42:  INC    DX             ; POINT AT DATA PORT
2363 0B04 42       IN     AL,DX          ; GET THE DATA
2364 0B05 EC       MOV    [DI],AL        ; STORE THE BYTE
2365 0B06 88 05   INC    DI             ; INCREMENT THE POINTER
2366 0B08 47
2367
2368 0B09 B9 0002  MOV    CX,2          ; MINIMUM 12 MICROSECONDS FOR NEC
2369 0B09 00 0000 E CALL   M11F          ; WAIT 15 TO 30 MICROSECONDS
2370 0B0F 44       DEC    DX             ; POINT AT STATUS PORT
2371 0B10 EC       IN     AL,DX          ; GET STATUS
2372 0B11 A6 10   TEST   AL,0001000B    ; TEST FOR NEC STILL BUSY
2373 0B13 74 0A   JZ     POPRES        ; RESULTS DONE ?
2374
2375 0B15 FE CB   DEC    BL             ; DECREMENT THE STATUS COUNTER
2376 0B17 75 D2   JNZ    R10           ; GO BACK FOR MORE
2377 0B19 80 0E 0041 R 20  OR    #DSKETTE_STATUS,BAD_NEC
2378 0B1E F9       STC    SC             ; TOO MANY STATUS BYTES
2379
2380           I----- RESULT OPERATION IS DONE
2381
2382 0B1F         POPRES: POP   DI             ; RETURN WITH CARRY SET
2383 0B1F 5F
2384 0B20 C3
2385 0B21
2386
2387           I----- READ_DSKCHNG
2388           I----- READS THE STATE OF THE DISK CHANGE LINE.
2389
2390           I ON ENTRY: DI = DRIVE #
2391
2392           I ON EXIT: DI = DRIVE #
2393           I ZF = 0 DISK CHANGE LINE INACTIVE
2394           I ZF = 1 DISK CHANGE LINE ACTIVE
2395           I AX,CX,DX DESTROYED
2396
2397 0B21
2398 0B25 E8 0913 R READ_DSKCHNG PROC  NEAR
2399 0B24 B0 03FT H CALL   MOTOR_ON      ; TURN ON THE MOTOR
2400 0B27          MOV    DX,03F7H    ; ADDRESS DIGITAL INPUT REGISTER
2401 0B28 A6 80   IN    AL,DX          ; INPUT DIGITAL INPUT REGISTER
2402 0B2A C3   TEST   AL,DSK_CHG    ; CHECK FOR DISK CHANGE LINE ACTIVE
2403 0B2B          RET    SC             ; RETURN TO CALLER WITH ZERO FLAG SET
2404
2405
2406
2407
2408
2409
2410           I ON ENTRY: DI = DRIVE #
2411
2412 0B2B E8 0913 R DRIVE_DET PROC  NEAR
2413 0B2E E8 047C R CALL   MOTOR_ON      ; TURN ON MOTOR IF NOT ALREADY ON
2414 0B31 72 3E   CALL   RECAL          ; RECALIBRATE DRIVE
2415 0B33 B3 30   JC    DD_BAC        ; ASSUME NO DRIVE PRESENT
2416 0B35 E8 0414 R MOV    CH_TRL_SLAP  ; SEEK TO TRACK 48
2417 0B36 72 37   CALL   SEEK          ; *
2418 0B3A B5 0B   JC    DD_BAC        ; ERROR NO DRIVE
2419 0B3C          MOV    CH_QUIET_SEEK+I ; SEEK TO TRACK 10
2420 0B3C FE CD   SK_GIN: DEC   CH             ; DECREMENT TO NEXT TRACK
2421 0B3E 7E 26   JS    IS_40          ; END LOOP IF CYLINDER COUNT NEGATIVE
2422 0B40 51       PUSH  CX             ; SAVE TRACK
2423 0B41 E8 0414 R CALL   SEEK          ; SEEK
2424 0B44 72 2C   JC    DD_BAC        ; POP AND RETURN
2425 0B46 B6 0B71 R MOV    AX,0FFSET DD_BAC ; LOAD NEC OUTPUT ERROR ADDRESS
2426 0B49 50       PUSH  AX             ; SENSE DRIVE STATUS COMMAND BYTE
2427 0B4A B4 04   MOV    AH,SENSE_DRV_ST ; OUTPUT TO NEC
2428 0B4C E8 09FO R CALL   NEC_OUTPUT   ; OUTPUT TO NEC
2429 0B50 B8 0D    MOV    AX,1           ; LL = DRIVE
2430 0B51 E8 A0   MOV    CH_AL        ; AH = DRIVE
2431 0B53 E8 09FO R CALL   NEC_OUTPUT   ; OUTPUT TO NEC
2432 0B56 E8 0E2 R CALL   RESULTS        ; GO GET STATUS
2433 0B59 58       POP    AX             ; THROW AWAY ERROR ADDRESS
2434 0B5A 59       POP    CX             ; RESTORE TRACK
2435 0B5B 7E 10   TEST   AL,NEC_STATUS,HOME ; TRACK 0
2436 0B5C 7E 10   JZ    IS_40          ; OVERRIDE TRACK 0
2437 0B62 0A ED   OR    CH_CH        ; IS HOME AT TRACK 0 ?
2438 0B64 74 06   JZ    IS_80          ; MUST BE 80 TRACK DRIVE
2439
2440           I DRIVE IS A 3601 SET DRIVE TO DETERMINED!
2441           I SET MEDIA TO DETERMINED AT RATE 250.
2442 0B66          IS_40:  OR    #DSK_STATE[DI],DRV_DET+MED_DET+RATE_250
2443 0B66 80 0D 0090 R 94  RET    SC             ; ALL INFORMATION SET
2444 0B6B C3
2445
2446 0B7C          IS_80:  OR    #DSK_STATE[DI],TRK_CAPA ; SETUP 80 TRACK CAPABILITY
2447 0B7C 80 B0 0090 R 01  DD_BAC: RET
2448 0B71
2449 0B71 C3
2450
2451 0B72          POP_BAC: POP   CX             ; THROW AWAY
2452 0B72 59
2453 0B73 C3
2454
2455 0B74          DRIVE_DET: ENDP

```

```

2455
2457
2458 ;----- THIS ROUTINE HANDLES THE DISKETTE INTERRUPT.
2459
2460 ;----- ON EXIT: THE INTERRUPT FLAG IS SET IN *SEEK_STATUS.
2461
2462 0B74  DISK_INI_1 PROC FAR
2463 0B74    STI          ; ENTRY POINT FOR ORG 0EF5TH
2464 0B75  FB          ; RE-ENABLE INTERRUPTS
2465 0B75  50          ; SAVE WORK REGISTER
2466 0B77  E8 0000 E   ; SAVE REGISTERS
2467 0B7F  80 0E 003E R 80  CALL DDS      ; SETUP DATA ADDRESSING
2468 0B7F    OR  *SEEK_STATUS,INT_FLAG ; TURN ON INTERRUPT OCCURRED
2469 0B80  B0 20          RESTORE DS
2470 0B82  E6 20          SETUP_INTERRUPT_MARKER
2471 0B84  B8 9101        INT 15H      ; INTERRUPT CONTROL PORT
2472 0B87  CD 15          INTERRUPT_POST_CODE_AND_TYPE
2473 0B89  58          GO PERFORM OTHER TASK
2474 0B8A  C0          RECOVER_REGISTER
2475 0B8A  CF          RETURN FROM INTERRUPT
2476
2477 ;----- DISK_INI_1 ENDP
2478
2479 ;----- DSKETTE SETUP
2480 ;----- THIS ROUTINE DOES A PRELIMINARY CHECK TO SEE WHAT TYPE
2481 ;----- OF DISKETTE DRIVES ARE ATTACHED TO THE SYSTEM.
2482
2483 0BBB  DSKETTE_SETUP PROC NEAR
2484 0BBB  50          PUSH AX      ; SAVE REGISTERS
2485 0BBC  53          PUSH BX
2486 0BBD  51          PUSH CX
2487 0B8E  52          PUSH DX
2488 0B90  57          PUSH DI
2489 0B90  56          PUSH SI
2490 0B91  1E          PUSH DS
2491 0B92  E8 0000 E   CALL DDS      ; POINT DATA SEGMENT TO BIOS DATA AREA
2492 0B95  80 0E 0040 R 01 OR  *RTC_WAIT_FLAG,01 ; NO RTC WAIT, FORCE USE OF LOOP
2493 0B97  C7 06 0090 R 0000 MOV WORD PTR *DSK_STATE,0 ; INITIALIZE STATES
2494 0B9D  80 26 008B R 33 AND  *LASTRATE,NONEC_STAT,MSK+SENDISK ; CLEAR PART & SEND
2495 0B9E  80 26 008B R 33 OR  *RTC_WAIT_FLAG,01 ; INITIALIZE PART TO IMPOSSIBLE
2496 0BA0  C6 06 003E R 00  MOV *SEEK_STATUS,0 ; INDICATE RECALIBRATE NEEDED
2497 0BAF  C6 06 0040 R 00 MOV *MOTOR_COUNT,0 ; INITIALIZE MOTOR COUNT
2498 0B94  C6 06 003F R 00 MOV *MOTOR_STATUS,0 ; INITIALIZE DRIVES TO OFF STATE
2499 0B99  C6 06 0041 R 00 MOV *DSKETTE_STATUS,0 ; NO ERRORS
2500 0B9A  C6 06 0040 R 00 MOV AL,BYTE PTR *EQUIP_FLAG ; GET EQUIPMENT STATUS
2501 0BC1  D0 C0          ROL AL,1      ; SHIFT BITS 7,6 TO 1,0
2502 0BC3  D0 C0          ROL AL,1
2503 0BC5  24 03          AND AL,3      ; MASK DRIVE BITS
2504 0BC7  32 E4          XOR AH,AH ; AX=NUMBER OF DRIVES(RELATIVE ZERO)
2505 0BC9  33 FF          XOR DI,DI ; DI=INITIAL DRIVE TO BE ESTABLISHED
2506 0BCA  BE 0010        MOV SI,HOME ; SI=HOME MASK FOR ALL DRIVES
2507 0BCB  D0 C0          SUP0:
2508 0BCD  C6 85 0090 R 94 TEST *HF_CNTRL,DUAL ; TEST CONTROLLER TYPE
2509 0BCD  75 05          JNZ Supt
2510 0BCD  80 0E 0040 R 01 MOV *DSK_STATE[DI],DRV_DET+MED_DET-RATE_250
2511 0BCD  50          SUP1:
2512 0BDB  E8 0B2B R    PUSH AX      ; SAVE DRIVE COUNT
2513 0BDB  E8 0432 R    CALL DRIVE_DET ; DETERMINE DRIVE
2514 0BDB  E8 0432 R    CALL XLAT_DLD ; TRANSLATE STATE TO COMPATIBLE MODE
2515 0BDB  E8 0432 R    AND SI,WORD PTR *NEC_STATUS ; AND *NEC_STATUS WITH HOME MASK
2516 0BDB  E8 0432 R    INC DI      ; RESTORE DRIVE COUNT
2517 0BDB  E8 0432 R    CMP DI,AX ; POINT TO NEXT DRIVE
2518 0BEB  F9          JNA SUP0 ; REPEAT FOR EACH DRIVE
2519 0BEB  C6 06 003E R 00 MOV *SEEK_STATUS,0 ; FORCE RECALIBRATE
2520 0BF0  80 26 0040 R FE AND  *RTC_WAIT_FLAG,0FEH ; ALLOW FOR RTC WAIT
2521 0BF5  E8 0832 R    CALL SETUP_END_ ; VARIOUS CLEANUPS
2522 0BF5  E8 0832 R    JNZ HOME_OK ; EXIT WITH CY FLAG FROM SETUP_END
2523 0BF5  E8 0832 R    OR  SI,ST ; TEST HOME INDICATORS FOR ALL DRIVES
2524 0BF5  75 01          JNZ HOME_OK
2525 0BFE  F9          STC          ; ERROR-->HOME INDICATOR BAD
2526 0BF7  0F          HOME_OK:
2527 0BF7  1F          POP DS      ; RESTORE CALLERS RESISTERS
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  0F          DSKETTE_SETUP ENDP
2536 0C07  ENDS
2537 0C07  ENDN

```

```

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          :----- INPUT
7          | (AH)=0 READ THE NEXT ASCII CHARACTER STRUCK FROM THE KEYBOARD
8          | RETURN THE RESULT IN [AL], SCAN CODE IN [AH]
9          | (AH)=1 SET THE Z FLAG TO INDICATE IF AN ASCII CHARACTER IS
10         | AVAILABLE TO BE READ.
11         | (ZF)=1 -- NO CODE AVAILABLE
12         | (ZF)=0 -- CODE IS AVAILABLE
13         | [ZF] = 0, THE NEW CHARACTER IN THE BUFFER TO BE READ
14         | IS IN AX. [ZF] = 1, THE EXTENDED CHARACTERS IN THE BUFFER
15         | ARE IN AX. THE CURRENT SHIFT STATUS IN AL REGISTER
16         | THE BIT SETTINGS FOR THIS CODE ARE INDICATED IN THE
17         | THE EQUATES FOR KB FLAG
18         | (AH)=5 PLACE ASCII CHARACTER/SCAN CODE COMBINATION IN KEYBOARD
19         | BUFFER AS IF STRUCK FROM KEYBOARD
20
21         ENTRY: [CL] = ASCII CHARACTER
22         [CH] = SCAN CODE
23
24         EXIT: [AL] = 00H = SUCCESSFUL OPERATION
25         [AL] = 01H = UNSUCCESSFUL - BUFFER FULL
26
27
28         (AH)=10H EXTENDED READ INTERFACE FOR THE ENHANCED KEYBOARD
29         (AH)=11H EXTENDED ASCII STATUS FOR THE ENHANCED KEYBOARD,
30         OTHERWISE SAME AS FUNCTION AH=1
31         (AH)=12H RETURN THE EXTENDED SHIFT STATUS IN AX REGISTER
32         AL = BITS FROM KB FLAG, AH = BITS FOR LEFT AND RIGHT
33         CTL AND ALT KEYS FROM KB_FLAG_1 AND KB_FLAG_3
34
35         EXIT:
36
37         [T]6[5|4|3|2||0] AH REGISTER
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 0000          PUBLIC KEYBOARD_I0_I
77          PUBLIC KB_INT_1
78
79          .LIST
80
81          0000 FB          CODE SEGMENT BYTE PUBLIC
82          0001 IE          ASSUME CS:CODE, DS:DATA
83          0002 53          KEYBOARD_I0_I PROC FAR
84          0003 51          ST1 DS
85          0004 E8 0000 E  PUSH DS
86          0007 0A E4          PUSH BX
87          0009 74 26          PUSH CX
88          000C FE CC          CALL DDS
89          0011 T4 64          OR AH, AH
90          0013 80 E0 03          JZ K1
91          0016 T4 64          DEC AH
92          0018 80 EC 0B          SUB AH, 3
93          001B T4 OC          SUB AH, 0BH
94          001D FE CC          JZ K2
95          001F T4 1A          DEC AH
96          0021 FE CC          JZ K2E
97          0023 T4 39          DEC AH
98          0025 59          JZ K3E
99          0025 59          POP CX
100         0026 5B          POP BX
101         0027 1F          POP DS
102         0028 CF          IRET
103
104
105          ----- ASCII CHARACTER
106         0029 E8 009E R  K1E: CALL K1S
107         002C E8 00D1 R  CALL K10_E_XLAT
108         002F EB F4          JMP K10_EXIT
109
110         0031 E8 009E R  K1: CALL K1S
111         0034 E8 00DC R  CALL K10_S_XLAT
112         0037 72 F8          JC K1
113         0039 EB EA          JMP K10_EXIT
114

```

```

115          ;----- ASCII STATUS
116
117 003B EB 00C4 R K2E: CALL K25           ; TEST FOR CHARACTER IN BUFFER (EXTENDED)
118 003E T4 18   JZ K2B                   ; RETURN IF BUFFER EMPTY
119 0040          PUSHF                  ; SAVE ZF FROM TEST
120 0041 EB 00D1 R CALL K10_E_XLAT      ; ROUTINE TO XLATE FOR EXTENDED CALLS
121 0044 EB 11   JMP SHORT_K2A        ; GIVE IT TO THE CALLER
122
123 0046 EB 00C4 R K2: CALL K25           ; TEST FOR CHARACTER IN BUFFER
124 0049 T4 0D   JZ K2B                   ; RETURN IF BUFFER EMPTY
125 004B          PUSHF                  ; SAVE ZF FROM TEST
126 004C EB 00DC R CALL K10_S_XLAT      ; ROUTINE TO XLT FOR STANDARD CALLS
127 004F T3 06   JNC K2A                 ; CARRY CLEAR MEANS PASS VALID CODE
128 0051 9D     POPF                  ; INVALID CODE FOR THIS TYPE OF CALL
129 0052 EB 009E R CALL K15                 ; THROW THE CHARACTER AWAY
130 0055 EB EF   JMP K2                  ; GO LOOK FOR NEXT CHAR, IF ANY
131
132 0057 9D     K2A: POPF              ; RESTORE ZF FROM TEST
133 0058 59     K2B: POP CX            ; RECOVER REGISTER
134 0059 5B     POP BX                ; RECOVER SEGMENT
135 005A 1F     POP DS                ; THROW AWAY FLAGS
136 005B CA 0002 RET 2
137
138          ;----- SHIFT STATUS
139
140 005E
141 005E 8A 26 0018 R K3E: MOV AH, #KB_FLAG_1 ; GET THE EXTENDED SHIFT STATUS FLAGS
142 0062 80 E4 04   AND AH, SYS_SHIFT ; GET SYSTEM SHIFT KEY STATUS
143 0065 80 00 01   MOV CL, 5             ; MASK ALL BUT SYS KEY BIT
144 0067 D2 44   SHL AH, CL            ; SHIFT THE SYSTEM KEY BIT OVER TO
145 0069 A0 0018 R MOV AL, #KB_FLAG_1 ; GET SHIFT STATES BACK
146 006C 24 73   AND AL, 011T001B ; ELIMINATE SYS_SHIFT, HOLD STATE, AND INS_SHIFT
147 006E 0A E0   OR AH, AL              ; MERGE THE REMAINING BITS INTO AH
148 0070 A0 0096 R MOV AL, #KB_FLAG_3 ; GET RIGHT CTL AND ALT
149 0072 80 00 01   AND AL, 00000100B ; ELIMINATE LC_E0 AND LC_E1
150 0075 80 00    OR AH, AL              ; OR THE SHIFT FLAGS TOGETHER
151 0077 A0 0017 R MOV AL, #KB_FLAG_2 ; GET THE SHIFT STATUS FLAGS
152 007A EB A9   JMP K10_EXIT        ; RETURN TO CALLER
153
154          ;----- WRITE TO KEYBOARD BUFFER
155
156 007C
157 007C 56
158 007D FA
159 007E BB 1E 001C R K500: PUSH SI
160 0082 BB F3   MOV BX, [#BUFFER_TAIL]; GET THE "IN TO" POINTER TO THE BUFFER
161 0085 80 00 01   MOV SI, BX            ; SAVE A COPY IN CASE BUFFER NOT FULL
162 0087 BD 1E 001A R CALL K4               ; BUMP THE POINTER TO SEE IF BUFFER IS FULL
163 0088 BB 74 0B   CMP BX, [#BUFFER_HEAD]; TELL THE BUFFER OVERFLOW IF WE STORE THIS?
164 008D 89 0C   JE K502                ; YES - INVOKE CALLER OF ERROR
165 008F 89 1E 001C R MOV [SI].CX, K502 ; NO - PUT THE ASCII/SCAN CODE INTO BUFFER
166 0093 2A C0   MOV [#BUFFER_TAIL], BX; ADJUST "IN TO" POINTER TO REFLECT CHANGE
167 0096 EB 03 90  SUB AL, AL            ; TELL CALLER THAT OPERATION WAS SUCCESSFUL
168 0098          JMP K504                ; SUB INSTRUCTION ALSO RESETS CARRY FLAG
169 0098 BB 01   K502: MOV AL, 01H       ; BUFFER FULL INDICATION
170 009A          K504: STI
171 009A FB     POP SI
172 009B 5E     JMP K10_EXIT        ; RETURN TO CALLER WITH STATUS IN AL
173 009C EB 87
174
175 009E          KEYBOARD_10_! ENDP

```

```

176 PAGE
177 ;----- READ THE KEY TO FIGURE OUT WHAT TO DO -----
178
179 009E KIS PROC NEAR
180 009E BB IE 001A R MOV BX,*BUFFER_HEAD ; GET POINTER TO HEAD OF BUFFER
181 0042 3B IE 001C R CMP BX,*BUFFER_TAIL ; TEST END OF BUFFER
182 0064 75 05 JNE KIT ; IF ANYTHING IN BUFFER DONT DO INTERRUPT
183
184 0048 BB 9002 MOV AX,09002H ; MOVE IN WAIT CODE & TYPE
185 00AB CD 15 INT 15H ; PERFORM OTHER FUNCTION
186 00AD KIT: ; ASCII READ
187 00AD FB STI ; INTERRUPTS BACK ON DURING LOOP
188 00AD 90 NOP ; ALLOW AN INTERRUPT TO OCCUR
189 004F FA CLI ; INTERRUPT BACK OFF
190 0080 BB IE 001A R MOV BX,*BUFFER_HEAD ; GET POINTER TO HEAD OF BUFFER
191 0084 3B IE 001C R CMP BX,*BUFFER_TAIL ; TEST END OF BUFFER
192 0088 74 F3 JE KIT ; LOOP UNTIL SOMETHING IN BUFFER
193 008A BB 07 MOV AX,[BX] ; GET SCAN CODE AND ASCII CODE
194 008C E8 0114 R CALL K4 ; MOVE POINTER TO NEXT POSITION
195 0080 BB 89 IE 001A R MOV *BUFFER_HEAD,BX ; STORE VALUE IN VARIABLE
196 00C3 C3 RET ; RETURN
197 00C4 KIS ENDP ; RETURN
198
199
200 ;----- READ THE KEY TO SEE IF ONE IS PRESENT -----
201
202 00C4 K2S PROC NEAR
203 00C4 FA CLI ; INTERRUPTS OFF
204 00C5 BB IE 001A R MOV BX,*BUFFER_HEAD ; GET HEAD POINTER
205 00C9 3B IE 001C R CMP BX,*BUFFER_TAIL ; IF EQUAL (Z=1) THEN NOTHING THERE
206 00CD BB 07 MOV AX,[BX]
207 0000 FB STI ; INTERRUPTS BACK ON
208 0000 C3 RET ; RETURN
209 00D1 K2S ENDP ; RETURN
210
211
212 ;----- ROUTINE TO TRANSLATE SCAN CODE PAIRS FOR EXTENDED CALLS
213
214 00D1 K10_E_XLAT: ; IS IT ONE OF THE FILL-INs?
215 00D1 3C F0 CMP AL,0F0h ; NO, PASS IT ON
216 00D3 75 06 JNE K10_E_RET ; OR, AH,AH-
217 00D5 0A E4 OR AH,AH- ; AH = 0 IS SPECIAL CASE
218 00D7 74 02 JZ K10_E_RET ; PASS THIS ON UNCHANGED
219 00D8 32 C0 XOR AL,AL- ; OTHERWISE SET AL = 0
220 00DB K10_E_RET1 ; GO BACK
221 00DB C3 RET
222
223
224 ;----- ROUTINE TO TRANSLATE SCAN CODE PAIRS FOR STANDARD CALLS
225
226 00DC K10_S_XLAT: ; IS IT KEYPAD ENTER OR / ?
227 00DC 80 FC E0 CMP AH,0E0h ; NO, CONTINUE
228 00DF 75 12 JNE K10_S2 ; KEYPAD ENTER CODE?
229 00E1 3C 0D CMP AL,0Dh ; YES, MESSAGE A BIT
230 00E3 74 09 JE K10_S1 ; CTY MSG FOR ENTER CODE?
231 00E5 32 C0 CMP AL,0Ch ; YES, MESSAGE THE SAME
232 00E7 74 05 JE K10_S1 ; NO, MUST BE KEYPAD /
233 00E9 B4 35 MOV AH,35h ; GIVE TO CALLER
234 00EB E8 23 90 JMP K10_USE ; CONVERT TO COMPATIBLE OUTPUT
235 00EE B4 1C K10_S1: MOV AH,Tch ; GIVE TO CALLER
236 00F0 EB 1E 90 JMP K10_USE
237
238 00F3 80 FC B4 K10_S2: CMP AH,84h ; IS IT ONE OF THE EXTENDED ONES?
239 00F6 77 1A JA K10_DIS ; YES, THROW AWAY AND GET ANOTHER CHAR
240
241 00F8 3C F0 CMP AL,0F0h ; IS IT ONE OF THE FILL-INs?
242 00FA 75 07 JNE K10_S3 ; NO, TRY LAST TEST
243 00FB 0A E4 OR AH,AH- ; AH = 0 IS SPECIAL CASE
244 00FE 74 10 JZ K10_USE ; PASS THIS ON UNCHANGED
245 0100 EB 10 90 JMP K10_DIS ; THROW AWAY THE REST
246
247 0103 3C E0 K10_S3: CMP AL,0E0h ; IS IT AN EXTENSION OF A PREVIOUS ONE?
248 0105 09 E4 JNE K10_USE ; NO, MUST BE A STANDARD CODE
249 0107 0A E4 OR AH,AH- ; AH = 0 IS SPECIAL CASE
250 0109 74 05 TT K10_USE ; JUMP IF AH = 0
251 010B 32 C0 XOR AL,AL- ; CONVERT TO COMPATIBLE OUTPUT
252 010D EB 01 90 JMP K10_USE ; PASS IT ON TO CALLER
253
254 0110 K10_USE: ; CLEAR CARRY TO INDICATE GOOD CODE
255 0110 F8 CLC ; RETURN
256 0111 C3 RET
257 0112 K10_DIS: ; SET CARRY TO INDICATE DISCARD CODE
258 0112 F9 STC ; RETURN
259 0113 C3 RET

```

```

260 PAGE
261 ;-----+
262 ;----- INCREMENT BUFFER POINTER ROUTINE
263 ;-----+
264
265 0114 K4 PROC NEAR
266 0114 43 INC BX ; MOVE TO NEXT WORD IN LIST
267 0115 43 INC BX
268
269 0116 3B IE 0082 R CMP BX,ŹBUFFER_END ; AT END OF BUFFER?
270 0116 72 04 JB K5 ; NO, CONTINUE
271 0116 C8 BB IE 0080 R MOV BX,ŹBUFFER_START ; YES, RESET TO BUFFER BEGINNING
272 0120 C3 K5F: RET
273 0121 K4 ENDP
274
275
276 ;----- KEYBOARD INTERRUPT ROUTINE
277
278 0121 KB_INT_1 PROC FAR
279 0121 50 PUSH AX ; SAVE THE STI UNTIL AFTER KEYBOARD RESET
280 0122 53 PUSH BX
281 0123 51 PUSH CX
282 0124 52 PUSH DX
283 0125 56 PUSH SI
284 0126 57 PUSH DI
285 0127 1E PUSH DS
286 0128 06 PUSH ES
287 0129 FC CLD ; FORWARD DIRECTION
288 012A 00 0000 E CALL DDS ; SET UP ADDRESSING TO DATA SEGMENT
289 012D E4 60 IN AL,KB_DATA ; READ IN THE CHARACTER
290 012F 93 XCHG BX,AX ; SAVE IT
291
292 ;----- RESET THE SHIFT REGISTER ON THE PLANAR IF ENABLED, OR DO NOTHING IF
293 ;----- IT IS DISABLED
294
295 0130 E4 61 IN AL,KB_CTL ; GET THE CONTROL PORT
296 0132 8A E0 MOV AH,AL ; SAVE VALUE
297 0134 DC 80 OR AL,80H ; RESET BIT FOR KEYBOARD
298 0136 E6 61 OUT KB_CTL,AL
299 0138 86 E0 XCHG AH,AL ; GET BACK ORIGINAL CONTROL
300 013A 8A E0 OUT KB_CTL,AL ; KB HAS BEEN RESET
301 013C FB STI ; EXIT IF SYSTEM HANDLED SCAN CODE
302 013D 93 XCHG AX,BX ; RESTORE DATA IN
303
304 ;----- SYSTEM HOOK INT 15H - FUNCTION 4FH (ON HARDWARE INTERRUPT LEVEL 9H)
305
306 0140 B4 4F MOV AH,04FH ; SYSTEM INTERCEPT - KEY CODE FUNCTION
307 0140 F9 STC ; SET CY=1 (IN CASE OF IRET)
308 0141 CD 15 INT 15H ; CASSETTE CALL ((AL)= KEY SCAN CODE
309 ;-----+
310 0143 72 03 JC KB_INT_PC ; CONTINUE IF CARRY FLAG SET ((AL)=CODE)
311 0145 E9 02CA R JMP K26 ; EXIT IF SYSTEM HANDLED SCAN CODE
312
313 0148 KB_INT_PC: ; EXIT HANDLES HARDWARE EO1 AND ENABLE
314 0148 8A E0 MOV AH,AL ; SAVE SCAN CODE IN AH ALSO
315
316 ;----- TEST FOR OVERRUN SCAN CODE FROM KEYBOARD
317
318 014A 3C FF CMP AL,0FFFH ; IS THIS AN OVERRUN CHAR
319 014C 75 03 JNZ K16 ; NO, TEST FOR SHIFT KEY
320 014E E9 0540 R JMP K62 ; BUFFER_FULL_BEEP
321
322 0151 0E K16: PUSH CS ; ESTABLISH ADDRESS OF TABLES
323 0152 07 POP ES
324 0153 8A 3E 0096 R MOV BH,ŹKB_FLAG_3 ; LOAD FLAGS FOR TESTING
325
326 0157 TEST_E0: ;-----+
327 0157 3C E0 CMP AL,MC_E0 ; IS THIS THE GENERAL MARKER CODE?
328 0159 75 07 JNE TEST_E1 ; NO, TEST FOR SHIFT KEY
329 015B 80 0E 0096 R 12 OR #KB_FLAG_3,LC_E0+KBX ; SET FLAG BIT, SET KBX, AND
330 0160 EB 09 JMP SHORT EXIT_K ; THROW AWAY THIS CODE
331
332 0162 TEST_E1: ;-----+
333 0162 3C E1 CMP AL,MC_E1 ; IS THIS THE PAUSE KEY?
334 0163 80 0E 0096 R 11 JNE NOT_HC ; NO, TEST FOR SHIFT KEY
335 0166 80 0E 0096 R 11 OR #KB_FLAG_3,LC_E1+KBX ; SET FLAG, PAUSE KEY MARKER CODE
336 0168 E9 02CF R JMP K26A ; THROW AWAY THIS CODE
337
338 016E NOT_HC: ;-----+
339 016E 24 7F AND AL,07FH ; TURN OFF THE BREAK BIT
340 0170 F6 C7 02 TEST BH,LC_E0 ; WAS LAST CODE THE EO MARKER CODE?
341 0173 74 0C JZ NOT_LC_E0 ; JUMP IF NOT
342
343 0175 B9 0002 MOV CX,2 ; LENGTH OF SEARCH
344 0176 BF 0555 R MOV DI,ŹOFFSET K6+6 ; IS THIS A SHIFT KEY?
345 0178 F2/ AE REPNE SCASB ; CHECK IT
346 017D 75 54 JNE K16A ; NO, CONTINUE KEY PROCESSING
347 017F EB 3D JMP SHORT K16B ; YES, THROW AWAY & RESET FLAG
348
349 0181 NOT_LC_E0: ;-----+
350 0181 F6 C7 01 TEST BH,LC_E1 ; WAS LAST CODE THE E1 MARKER CODE?
351 0184 74 16 JZ T_SYS_KEY ; JUMP IF NOT
352
353 0186 B9 0004 MOV CX,4 ; LENGTH OF SEARCH
354 0186 BF 0553 R MOV DI,ŹOFFSET K6+4 ; IS THIS AN ALT, CTL, OR SHIFT?
355 018C F2/ AE REPNE SCASB ; CHECK IT
356 018E 74 DB JE EXIT_K ; THROW AWAY IF SO
357
358 0190 3C 45 CMP AL,NUM_KEY ; IS IT THE PAUSE KEY?
359 0192 75 2A JNE K16B ; NO, THROW AWAY & RESET FLAG
360 0194 F6 C4 80 TEST AH,80H ; YES, IS IT THE BREAK OF THE KEY?
361 0197 75 25 JNZ K16B ; YES, THROW THIS AWAY, TOO
362 0199 E9 03FF R JMP K39P ; NO, THIS IS THE REAL PAUSE STATE

```

```

363          PAGE
364          ;----- TEST FOR SYSTEM KEY
365
366          019C          T_SYS_KEY:
367          019C 3C 54      CMP    AL,SYNKEY
368          019E 75 33      JNE    K16A
369
370          01A0 F6 C4 80    TEST   AH,0B0H
371          01A3 75 1C      JNZ    K16C
372
373          01A5 F6 06 0018 R 04  TEST   KBFLAG_1,SYN_SHIFT
374          01AA T5 12      JNZ    K16B
375
376          01AC 80 0E 0018 R 04  OR     KBFLAG_1,SYN_SHIFT
377          01B1 B0 20      MOV    AL,E0I
378          01B3 E6 20      OUT   20H,AL
379
380          01B5 B8 8500      MOV    AX,0B500H
381          01B8 FB          STI    15H
382          01B9 CD 15      INT    K27
383          01B8 E9 02D4 R    JMP    K27
384
385          01BE E9 02CA R    K16B: JMP   K26
386
387          01C1 B0 26 0018 R FB  K16C: AND   KBFLAG_1,NOT SYN_SHIFT
388          01C6 B0 20      MOV    AL,E0I
389          01C8 E6 20      OUT   20H,AL
390
391          01CA B8 8501      MOV    AX,0B501H
392          01CD FB          STI    15H
393          01CE CD 15      INT    K27
394          01D0 E9 02D4 R    JMP    K27
395
396          ;----- TEST FOR SHIFT KEYS
397
398          01D3 8A IE 0017 R    K16A: MOV    BL,KBFLAG
399          01D7 BF 054F R    MOVI  DI,OFFSET K6
400          01DA BB 0008 90    MOV    CX,K6L
401          01DE F2 / AE    REPNE SCASB
402          01E0 8A C4      MOV    AL,AH
403          01E2 74 03      JE    K17
404          01E4 E9 02B6 R    JMP    K25
405
406          ;----- SHIFT KEY FOUND
407
408          01E7 81 EF 0550 R    K17: SUB   DI,OFFSET K6+1
409          01E9 25 04 A5 0557 R    MOV    AH,DI[K17][DI]
410          01F0 B1 02      MOVI  CL,2
411          01F2 AB 80      TEST   AL,1B0H
412          01F4 74 03      JZ    K17C
413          01F6 EB 6E 90      JMP    K23
414
415          ;----- SHIFT MAKE FOUND, DETERMINE SET OR TOGGLE
416
417          01F9 80 FC 10      K17C: CMP   AH,SCROLL_SHIFT
418          01FC 73 21      JAE    K18
419
420          ;----- PLAIN SHIFT KEY, SET SHIFT ON
421
422          01FE 08 26 0017 R    OR     KBFLAG,AH
423          0202 F6 C4 0C      TEST   AH,CTL_SHIFT+ALT_SHIFT
424          0205 75 03      JNZ    K17D
425          0207 E9 02CA R    JMP    K26
426          0204 FB F7 02      TEST   BH,LC_E0
427          0207 75 03      JZ    K17E
428          0208 0B 26 0096 R    OR     KBFLAG_3,AH
429          0213 E9 02CA R    JMP    K26
430          0216 02 EC          SHR   AH,CL
431          0218 08 26 0018 R    OR     KBFLAG_1,AH
432          021C E9 02CA R    JMP    K26
433
434          ;----- TOGGLED SHIFT KEY, TEST FOR 1ST MAKE OR NOT
435
436          021F                 K18: TEST   BL,CTL_SHIFT
437          021F F6 C3 04      JZ    K18A
438          0222 74 03      JMP    K25
439          0224 EB 0B 90      K18A: CMP   AL,INSKEY
440          0221 3C 52      JNE    K18B
441          0229 75 21      TEST   BL,ALT_SHIFT
442          0228 F6 C3 08      JZ    K18B
443          022E 74 03      JMP    K25
444          0230 E9 02B6 R    K18B: TEST   BH,LC_E0
445          0231 02 EC          JZ    K19
446          0234 75 14      JMP    K25
447          0238 F6 C3 20      K19: TEST   BL,NUM_STATE
448          023B 75 0A      JNZ    K21
449          023D F6 C3 03      TEST   BL,LEFT_SHIFT+RIGHT_SHIFT
450          0240 74 0A      JZ    K22
451          0241 02 EC          K20: MOV    AH,AL
452          0242 8A E0          JMP    K25
453          0244 EB 70 90      K21: TEST   BL,LEFT_SHIFT+RIGHT_SHIFT
454
455          0247 F6 C3 03      JZ    K20
456          0244 T4 F6          K21: TEST   BL,LEFT_SHIFT+RIGHT_SHIFT
457
458          024C                 K22: TEST   AH,KBFLAG_I
459          024C 84 26 0018 R    JZ    K22A
460          0250 74 03      JMP    K26
461          0252 EB 70 90      K22A: OR     KBFLAG_1,AH
462          0255 08 26 0018 R    XOR    KBFLAG,AH
463          0256 02 EC 0017 R    CMP    AL,INSKEY
464          0250 3C 52      JNE    K26
465          025F 75 69      MOV    AH,AL
466          0261 8A E0          JMP    K26
467          0263 EB 78 90      K22: TEST   AH,KBFLAG_I
468
469          ;----- BREAK SHIFT FOUND
470
471          0266                 K23: CMP   AH,SCROLL_SHIFT
472          0266 80 FC 10      NOT   AH
473          0269 F6 D0          AND   AH,NOT CTL_SHIFT
474          026A 74 03      CMP   AH,NOT CTL_SHIFT
475          0260 20 26 0017 R    CMP   AH,NOT CTL_SHIFT
476          0271 80 FC FB

```

```

477 0274 77 26          JA      K23D           ; NO, ALL DONE
478
479 0276 F6 CT 02        TEST   BH_LLC_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 EB 06          JMP    K23B           ; CONTINUE
483 0281 D2 FC          K23A: 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 0285 80 0001 R      MOV    AL,AL        ; SAVE SCAN CODE
486 0289 00 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 & CTL 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+B0H  ; IS THIS ALTERNATE SHIFT RELEASE
495 029E 75 2A          JNE    K26            ; INTERRUPT_RETURN
496
497
498 ;----- ALTERNATE SHIFT KEY RELEASED, GET THE VALUE INTO BUFFER
499 02A0 A0 0019 R      MOV    AL,@ALT_INPUT  ; SCAN CODE OF 0
500 02A3 B4 00          MOV    AH,0          ; ZERO OUT THE FIELD
501 02A5 88 26 0019 R    MOV    @ALT_INPUT,AH  ; WAS THE INPUT = 0?
502 02A9 3C 00          CMP    AL,0          ; INTERRUPT_RETURN
503 02AB 80 0010 R      JE    K26            ; IT WASN'T, SO PUT IN BUFFER
504 02AD E9 0519 R      JMP    K61            ; TEST FOR HOLD STATE
505
506 02B0
507 02B0 20 26 0018 R    K24:  AND    @KB_FLAG_1,AH  ; BREAK-TOGGLE
508 02B4 EB 14          JMP    SHORT_K26  ; INDICATE NO LONGER DEPRESSED
509
510 ;----- TEST FOR HOLD STATE
511
512 02B6 K25:  CMP    AL,B0H        ; AL, AH = SCAN CODE
513 02B6 3C 80          JAE    K26            ; NO-SHIFT-FOUND
514 02B8 73 10          TEST   @KB_FLAG_1,HOLD_STATE  ; TEST FOR BREAK KEY
515 02B9 06 0018 R 08    K25:  JZ     K26            ; NOTHING FOR BREAK CHARS FROM HERE ON
516 02BF 74 C           K25:  K26            ; 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  K25:  AND    @KB_FLAG_1,NOT HOLD_STATE  ; TURN OFF THE HOLD STATE BIT
520
521 02CA K26:  AND    @KB_FLAG_3,NOT LC_E0+LC_E1  ; RESET LAST CHAR H.C. FLAG
522 02CA 80 26 0096 R FC
523
524 02CF K26A: CLI
525 02CF FA          MOV    AL,E0I        ; INTERRUPT-RETURN
526 02D0 B0 20          OUT   020H,AL    ; TURN OFF INTERRUPTS
527 02D2 E6 20          ; END OF INTERRUPT COMMAND
528
529 02D4 K27: POP   ES            ; SEND COMMAND TO INTERRUPT CONTROL PORT
530 02D4 07          POP   DS            ; INTERRUPT-RETURN-NO-EOI
531 02D5 1F          POP   DI            ; RESTORE REGISTERS
532 02D6 5F
533 02D7 5E
534 02D8 5A          POP   SI
535 02D9 59          POP   DX
536 02DA 5B          POP   CX
537 02DB 58          POP   BX
538 02DC CF          POP   AX
539

```

IRET

; RETURN, INTERRUPTS BACK ON
; WITH FLAG CHANGE

```

540
541
542
543 02DD K28: TEST BL,ALT_SHIFT
544 02DD 3C 5B CMP AL,88
545 02DF 77 E9 JA K26
546
547 02E1 F6 C3 08 TEST BH,KBX
548 02E4 74 0C JZ K28A
549
550 02E6 F6 C7 10 TEST BH,KBX
551 02E9 74 0A JZ K29
552
553 02EB F6 06 0018 R 04 TEST KB_FLAG_1,.SYS_SHIFT
554 02F0 74 03 JZ K29_
555 02F2 E9 03CC R K28A: JMP K38
556
557
558 :----- TEST FOR RESET KEY SEQUENCE (CTL ALT DEL)
559
560 02F5 K29: TEST BL,CTL_SHIFT
561 02F6 F6 C3 04 JZ K31
562 02F8 74 37 CMP AL,DEL_KEY
563 02FA 3C 53 JNE K31
564 02FC 75 33
565
566 :----- CTL-ALT-DEL HAS BEEN FOUND, DO I/O CLEANUP
567
568 02FE C7 01 26 0072 R 1234 MOV PRESET_FLAG,1234H
569 0304 81 26 0072 R 0010 AND WORD PTR KB_FLAG_3,KBX
570 030A E9 0000 E JMP RESET
571
572 :----- ALT-INPUT-TABLE
573 030D K30: LABEL BYTE
574 030D 52 4F 50 51 4B DB 82,79,80,81,75
575 0312 4C 4D 47 48 49 DB 78,79,80,81,73
576
577 0317 10 11 12 13 14 15 :----- 10 NUMBERS ON KEYPAD
578 031D 16 17 18 19 1E 1F DB 16,17,18,19,20,21
579 0323 20 21 22 23 24 25 DB 22,23,24,25,30,31
580 0329 26 2C 2D 2E 2F 30 DB 32,33,34,35,36,37
581 032F 31 32 DB 38,44,45,46,47,48
582
583 :----- IN ALTERNATE SHIFT, RESET NOT FOUND
584
585 0331 K31: CMP AL,57
586 0331 3C 30 JNE K31_
587 0333 75 05 MOV AH,*
588 0335 B5 20 JMP K57
589 0337 E9 050D R
590 033A
591 033A 3C 0F K31_: TEST FOR TAB KEY
592 033B 75 06 JNE K312
593 033E B8 A500 MOV AX,0A500h
594 0341 E9 050D R JMP K57
595 0344
596 0344 3C 4A K312: TEST FOR KEYPAD -
597 0346 74 79 JE K37B GO PROCESS
598 0348 3C 4E CMP AL,78 TEST FOR KEYPAD +
599 034A 74 75 JE K37B GO PROCESS
600
601 :----- LOOK FOR KEY PAD ENTRY
602
603 034C K32: MOV DI,OFFSET K30
604 034C BF 030D R MOV CX,10
605 034F B9 0004 REPNE SCASB
606 0352 F2/ AE JNE K33
607 0354 75 1B TEST BH,LC_E0
608 0356 F7 C7 02 JNZ K37C
609 0359 75 6B SUB DI,OFFSET K30+1
610 035F B1 00 030E R MOV AH,*AL_INPUT
611 035F AA 0019 R ADD AH,10
612 0362 B0 0A MOV AH,10
613 0364 F6 E4 MUL AH
614 0366 03 C7 ADD AX,DI
615 0368 A2 0019 R MOV _ALT_INPUT,AL
616 036B E9 02CA R K32A: JMP K26
617
618
619 :----- LOOK FOR SUPERSHIFT ENTRY
620 036E K33: MOV _ALT_INPUT,0
621 036E C6 06 0019 R 00 :----- NO-ALT-KEYPAD
622 0373 B9 001A MOV CX,26
623 0376 F2/ AE REPNE SCASB
624 0378 74 42 JE K37A
625
626
627 :----- LOOK FOR TOP ROW OF ALTERNATE SHIFT
628 037A K34: CMP AL,2
629 037A 3C 02 JB K37B
630 037C 72 43 CMP AL,13
631 037E 3C 0D JA K35
632 0380 77 05 ADD AH,118
633 0382 80 C4 76 JMP SHORT K37A
634 0385 EB 35
635
636
637 :----- TRANSLATE ALTERNATE SHIFT PSEUDO SCAN CODES
638 0387 K35: CMP AL,F11_M
639 0387 3C 57 JB K35_
640 0389 72 09 CMP AL,F12_M
641 038B 3C 58 JA K35A
642 038D 77 05 ADD AH,52
643 038F 80 C4 34 JMP SHORT K37A
644 0392 EB 28
645
646 0394 F6 C7 02 K35A: TEST BH,LC_E0
647 0397 74 18 CMP AL,28
648 0399 3C 1C JNE K35B
649 039B T5 06 MOV AX,0A600h
650 039D B9 A600 JMP K57
651 03A1 E9 050D R
652 03A3 3C 53 K35B: CMP AL,B3
653 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 EO SPECIALS
656 03AB BB A400          MOV    AX,0A400h   ; SPECIAL CODE
657 03AE E9 050D R        JMP    K57           ; BUFFER FILL

658                                         ; TEST IN ON TABLE
659 03B1 3C 3B           K37:  CMP    AL,59          ; ALT-CONTINUE
660 03B3 72 0C           JB     K37B          ; IN KEYPAD REGION?
661 03B5 3C 44           CMP    AL,6B          ; OR NUMLOCK, SCROLLLOCK?
662                                         ; IF SO, IGNORE
663 03B7 77 B2           JA     K32A          ; CONVERT TO PSEUDO SCAN CODE
664 03B9 80 C4 2D        ADD    AH,45          ; ASCII CODE OF ZERO
665                                         ; PUT IT IN THE BUFFER

666 03BC BB 00           K37A: MOV    AL,0           ; USE SPECIAL ASCII CODE
667 03BE E9 050D R        JMP    K57           ; PUT IT IN THE BUFFER

668                                         ; CONVERT SCAN CODE (EDIT KEYS)
669 03C1 BB F0           K37B: MOV    AL,0F0h      ; (SCAN CODE NOT IN AH FOR INSERT)
670 03C3 E9 050D R        JMP    K57           ; PUT IT IN THE BUFFER

671                                         ; -----
672 03C6 04 50           K37C: ADD    AL,80          ; NOT IN ALTERNATE SHIFT
673 03C8 8A E0           MOV    AH,AL          ; NOT-ALT-SHIFT
674 03CA EB F0           JMP    K37A          ; BL STILL HAS SHIFT FLAGS
675                                         ; ARE WE IN CONTROL SHIFT?
676                                         ; YES, START PROCESSING
677                                         ; NOT-CTL-SHIFT

678 03CC                 K38:  TEST   BL,CTL_SHIFT  ; TEST FOR BREAK
679                                         ; JUMP, NO-BREAK
680 03CC F6 C3 04         TEST   BH,KBX        ; IS THIS THE ENHANCED KEYBOARD?
681 03CF 75 03           JNZ    K38A          ; NO, BREAK IS VALID
682 03D1 E9 0454 R        JMP    K44           ; YES, WAS LAST CODE AN EO?
683                                         ; NO-BREAK, TEST FOR PAUSE

684                                         ; -----
685                                         ; CONTROL SHIFT, TEST SPECIAL CHARACTERS
686                                         ; -----
687                                         ; TEST FOR BREAK
688 03D4 3C 46           K38A: CMP    AL,SCROLL_KEY  ; TEST FOR BREAK
689 03D6 75 1E           JNE    K39           ; JUMP, NO-BREAK
690 03D8 F6 C7 10         TEST   BH,KBX        ; IS THIS THE ENHANCED KEYBOARD?
691 03DB 74 05           JZ    K38B          ; NO, BREAK IS VALID
692 03D9 F6 C7 02         TEST   BH,LC_E0      ; YES, WAS LAST CODE AN EO?
693 03E0 74 14           JZ    K39           ; NO-BREAK, TEST FOR PAUSE

694                                         ; -----
695 03E2 BB 1E 001A R     K38B: MOV    BX,@BUFFER_HEAD  ; RESET BUFFER TO EMPTY
696 03E6 89 1E 001C R     MOV    @BUFFER_TAIL,BX
697 03EA C6 06 0071 R 80  MOV    @BIOS_BREAK,80H
698 03EF CD 1B           INT    1BH           ; TURN ON BIOS BREAK BIT
699 03F0 20 00           SUB    AX,AX          ; BREAK INTERRUPT VECTOR
700 03F3 E9 050D R        JMP    K57           ; PUT OUT DUMMY CHARACTER
701                                         ; BUFFER_FILL

702                                         ; -----
703                                         ; TEST FOR PAUSE
704 03F6                 K39:  TEST   BH,KBX        ; NO-BREAK
705 03F7 F6 C7 10         JNZ    K40           ; IS THIS THE ENHANCED KEYBOARD?
706 03F9 75 25           CMP    AL,NUM_KEY    ; YES, WHEN THIS CAN'T BE PAUSE
707 03FB 3C 45           CMP    AL,NUM_KEY    ; LOOK FOR PAUSE KEY
708 03FD 75 21           JNE    K41           ; NO-PAUSE
709 03FF 80 0E 0018 R 08  K39P: OR    @KB_FLAG_1,HOLD_STATE  ; TURN ON THE HOLD FLAG
710 0404 80 20           MOV    AL,E0I          ; END OF INTERRUPT TO CONTROL PORT
711 0405 80 20           OUT    020H,AL      ; ALLOW FURTHER KEYSTROKE INTS
712                                         ; -----
713                                         ; -----
714                                         ; DURING PAUSE INTERVAL, TURN CRT BACK ON
715 0408 80 3E 0049 R 07  K40:  CMP    @CRT_MODE,7  ; IS THIS BLACK AND WHITE CARD
716 0409 74 07           JE    K40           ; YES, NOTHING TO DO
717 040D 80 03D8          MOV    DX,03D8H      ; PORT FOR COLOR CARD
718 0412 00 0065 R        MOV    AL,CRT_MODE_SET  ; GET THE MODE OF THE CURRENT MODE
719 0415 EE               OUT    DX,AL          ; SET THE CRT MODE, SO THAT CRT IS ON
720 0416                 K40:  TEST   @KB_FLAG_1,HOLD_STATE  ; PAUSE-LOOP
721 0416 F6 00 0018 R 08  JNZ    K40           ; LOOP UNTIL FLAG TURNED OFF
722 0418 75 F9           JMP    K27           ; INTERRUPT_RETURN_NO_E0I
723 041D E9 02D4 R
724                                         ; -----
725                                         ; -----
726                                         ; TEST SPECIAL CASE KEY 55
727 0420                 K41:  CMP    AL,55          ; NO-PAUSE
728 0420 3C 37           JNE    K42           ; TEST FOR *PRTSC KEY
729 0421 74 14           CMP    AL,55          ; NOT-KEY-55
730 0424 F6 C7 10         TEST   BH,KBX        ; IS THIS THE ENHANCED KEYBOARD?
731 0427 74 05           JZ    K41           ; NO, NOT MORE SPECIAL CASES
732 0429 F6 C7 02         TEST   BH,LC_E0      ; YES, IS IT FROM THE KEYPAD?
733 042C 74 20           JZ    K42           ; NO, JUST TRANSLATE
734 042E BB 7200          MOV    AX,114*256   ; YES, SPECIAL CODE FOR THIS ONE
735 0431 E9 050D R        JMP    K57           ; BUFFER_FILL

736                                         ; -----
737                                         ; SET UP TO TRANSLATE CONTROL SHIFT
738                                         ; -----
739 0434                 K42:  CMP    AL,15          ; NOT-KEY-55
740 0434 3C 0F           JE    K42B          ; IS IT THE TAB KEY?
741 0436 74 14           CMP    AL,55          ; YES, XLATE TO FUNCTION CODE
742 0438 3C 35           JNE    K42A          ; IS THIS THE KEY?
743 043A 75 0B           CMP    AL,55          ; NO, NO MORE SPECIAL CASES
744 043C F6 C7 02         TEST   BH,LC_E0      ; YES, IS IT FROM THE KEYPAD?
745 043F 74 06           JZ    K42A          ; NO, JUST TRANSLATE
746 0441 BB 9500          MOV    AX,9500h      ; YES, SPECIAL CODE FOR THIS ONE
747 0444 E9 050D R        JMP    K57           ; BUFFER_FILL

748                                         ; -----
749 0447 BB 055F R        K42A: MOV    BX,OFFSET K8  ; SET UP TO TRANSLATE CTL
750 044A 3C 3B           CMP    AL,59          ; IS IT IN CHARACTER TABLE?
751 044C 72 57           JB     K45F          ; YES, GO TRANSLATE CHAR?
752 044E BB 055F R        K42B: MOV    BX,OFFSET K8  ; SET UP TO TRANSLATE CTL
753 0451 E9 04FC R        JMP    K64           ; NO, GO TRANSLATE_SCAN

754                                         ; -----
755                                         ; NOT IN CONTROL SHIFT
756                                         ; -----
757 0454 3C 37           K44:  CMP    AL,55          ; PRINT SCREEN KEY?
758 0456 74 1F           JNE    K45           ; NOT-PRINT-SCREEN
759 0458 F6 C7 10         TEST   BH,KBX        ; IS THIS ENHANCED KEYBOARD?
760 045B 74 07           JZ    K44           ; NO, TEST CODE A MARKER?
761 045D F6 C7 02         TEST   BH,LC_E0      ; YES, LAST CODE A MARKER?
762 0460 75 07           JNZ    K44B          ; YES, IS PRINT SCREEN
763 0462 EB 34           JMP    SHORT K45C   ; NO, XLATE TO *** CHARACTER
764 0464 F6 C3 03         K44A: TEST   BL,LEFT_SHIFT+RIGHT_SHIFT ; NOT 101 KBDR, SHIFT KEY DOWN?
765 0467 74 2F           JZ    K45C          ; NO, XLATE TO *** CHARACTER

766                                         ; -----
767                                         ; ISSUE INTERRUPT TO PERFORM PRINT SCREEN FUNCTION

```

```

768 0469 B0 20          K44B: MOV    AL,EO1      ; END OF CURRENT INTERRUPT
769 046B E6 20          OUT   020H,AL    ; SO FURTHER THINGS CAN HAPPEN
770 046D CD 05          INT    5H        ; ISSUE PRINT SCREEN INTERRUPT
771 046F 80 26 0096 R FC AND   #KB_FLAG_3,NOT LC_E0+LC_E1;ZERO OUT THESE FLAGS
772 0474 E9 02D4 R      JMP   K27       ; GO BACK WITHOUT EOI OCCURRING
773
774
775          ----- HANDLE THE IN-CORE KEYS
776 0477          K45:   CMP   AL,58      ; NOT-PRINT-SCREEN
777 0477 3C 3A          JA    K46       ; TEST FOR IN-CORE AREA
778 0479 77 2C          JNE   K45       ; JUMP IF NOT
779
780 047B 3C 35          CMP   AL,53      ; IS THIS THE /* KEY?
781 047D 75 05          JNE   K45A     ; NO, JUMP
782 047F F6 C7 02      TEST  BH,LC_E0    ; WAS LAST CODE THE MARKER?
783 0482 75 14          JNZ   K45C     ; YES, TRANSLATE TO CHARACTER
784
785 0484 B9 001A          K45A: MOV   CX,26      ; LENGTH OF SEARCH
786 0487 BF 0317 R      MOV   BX,OFFSET K30+10 ; POINT TO TABLE OF A-Z CHARS
787 048A F2/ AE          REPNE SCASB    ; IS THIS A LETTER KEY?
788 048C 75 05          JNE   K45B     ; NO, SYMBOL KEY
789
790 048E F6 C3 40          TEST  BL,CAPS_STATE ; ARE WE IN CAPS_LOCK?
791 0491 75 0A          JNZ   K45D     ; TEST FOR SUR
792 0493 F6 C3 03          K45B: TEST  BL,LEFT_SHIFT+RIGHT_SHIFT ; ARE WE IN SHIFT STATE?
793 0496 75 0A          JNZ   K45E     ; YES, UPPERCASE
794
795 0498 BB 05B7 R      K45C: MOV   BX,OFFSET K10    ; NO, LOWERCASE
796 049D EB 50          JMP   SHORT K56    ; TRANSLATE TO LOWERCASE LETTERS
797
798 049D F6 C3 03          K45D: TEST  BL,LEFT_SHIFT+RIGHT_SHIFT ; ALMOST-CAPS-STATE
799 04A0 75 F6          JNZ   K45F     ; TEST ON, IS SHIFT ON, TOO?
800 04A2 BB 060F R      K45E: MOV   BX,OFFSET K11    ; SHIFTED TEMP OUT OF CAPS STATE
801 04A5 EB 46          K45F: JMP   SHORT K56    ; TRANSLATE TO UPPERCASE LETTERS
802
803
804          ----- TEST FOR KEYS F1 - F10
805 04A7          K46:   CMP   AL,68      ; NOT IN-CORE AREA
806 04A7 3C 44          JA    K47       ; TEST FOR F1 - F10
807 04A9 77 02          JMP   SHORT K53    ; JUMP IF NOT
808 04AB EB 36
809
810
811          ----- HANDLE THE NUMERIC PAD KEYS
812
813 04AD          K47:   CMP   AL,63      ; NOT F1 - F10
814 04AD 3C 53          JA    K52       ; TEST FOR NUMPAD KEYS
815 04AF 77 2C          JNE   K48       ; JUMP IF NOT
816
817          ----- KEYPAD KEYS, MUST TEST NUM LOCK FOR DETERMINATION
818 04B1 3C 4A          K48:   CMP   AL,74      ; SPECIAL CASE FOR MINUS
819 04B3 74 ED          JE    K45E     ; GO TRANSLATE
820 04B5 3C 4E          CMP   AL,78      ; SPECIAL CASE FOR PLUS
821 04B7 74 E9          JE    K45E     ; GO TRANSLATE
822 04B9 F6 C7 02      TEST  BH,LC_E0    ; IS THIS ONE OF THE NEW KEYS?
823 04BC 75 0A          JNZ   K49       ; YES, TRANSLATE TO BASE STATE
824
825 04BE F6 C3 20          TEST  BL,NUM_STATE ; ARE WE IN NUM_LOCK?
826 04C1 75 13          JNZ   K50       ; TEST FOR SUR
827 04C3 F6 C3 03          TEST  BL,LEFT_SHIFT+RIGHT_SHIFT ; ARE WE IN SHIFT STATE?
828 04C6 75 13          JNZ   K51       ; IF SHIFTED, REALLY_NUM STATE
829
830          ----- BASE CASE FOR KEYPAD
831 04C8 3C 4C          K49:   CMP   AL,76      ; SPECIAL CASE FOR BASE STATE 5
832 04CA 75 05          JNE   K49A     ; CONTINUE IF NOT KEYPAD 5
833 04C9 75 05          MOV   AH,0Fh    ; SPECIAL ASCII CODE
834 04CE EB 3D 90          JMP   K52       ; BUFFER FILL
835 04D1 BB 05B7 R      K49A: MOV   BX,OFFSET K10    ; BASE CASE TABLE
836 04D4 EB 26          JMP   SHORT K64    ; CONVERT TO PSEUDO SCAN
837
838
839 04D6 F6 C3 03          K50:   TEST  BL,LEFT_SHIFT+RIGHT_SHIFT ; ALMOST_NUM_STATE
840 04D9 75 ED          JNZ   K49      ; SHIFTED TEMP OUT OF NUM STATE
841 04DB EB C5          K51:   JMP   SHORT K45E    ; REALLY_NUM_STATE
842
843
844          ----- TEST FOR THE NEW KEY ON WT KEYBOARDS
845
846 04DD          K52:   CMP   AL,86      ; NOT A NUMPAD KEY
847 04DD 3C 56          JNE   K53       ; IS IT THE NEW WT KEY?
848 04DF 75 02          JMP   SHORT K45B    ; JUMP IF NOT
849 04E1 EB B0          JNE   K56       ; HANDLE WITH REST OF LETTER KEYS
850
851
852          ----- MUST BE F11 OR F12
853
854 04E3 F6 C3 03          K53:   TEST  BL,LEFT_SHIFT+RIGHT_SHIFT ; F1 - F10 COME HERE, TOO
855 04E6 74 E0          JZ    K49       ; TEST SHIFT STATE
856
857 04E8 BB 060F R      MOV   BX,OFFSET K11    ; JUMP, LOWERCASE PSEUDO SC'S
858 04EB EB 0F          JMP   SHORT K64    ; TRANSLATE_SCAN
859
860          ----- TRANSLATE THE CHARACTER
861
862 04ED          K56:   DEC   AL      ; TRANSLATE-CHAR
863 04EF FE C8          XLAT  CS:K11    ; CONVERT ORIGIN
864 04EF 2E: D7          TEST  #KB_FLAG_3,LC_E0 ; CONVERT THE SCAN CODE TO ASCII
865 04F1 F6 06 0096 R 02 JZ    K57       ; IS THIS A NEW KEY?
866 04F6 74 15          MOV   AL,1      ; NO, GO FILL BUFFER
867 04F8 B4 E0          MOV   AH,MC_E0  ; YES, PUT SPECIAL MARKER IN AH
868 04FA EB 11          JMP   SHORT K57    ; PUT IT INTO THE BUFFER
869
870          ----- TRANSLATE SCAN FOR PSEUDO SCAN CODES
871
872 04FC          K64:   DEC   AL      ; TRANSLATE-SCAN-ORGD
873 04FC FE C8          XLAT  CS:K10    ; CONVERT ORIGIN
874 04FE 2E: D7          MOV   AL,1      ; CTL TABLE SCAN
875 0500 75 05          MOV   AL,1      ; PUT VALUE INTO AH
876 0502 B0 00          MOV   AL,1      ; ZERO ASCII CODE
877 0504 F6 06 0096 R 02 TEST  #KB_FLAG_3,LC_E0 ; IS THIS A NEW KEY?
878 0509 T4 02          JZ    K57       ; NO, GO FILL BUFFER
879 050B B0 E0          MOV   AL,MC_E0  ; YES, PUT SPECIAL MARKER IN AL
880
881          ----- PUT CHARACTER INTO BUFFER

```

```

882
883 050D          K57:    CMP    AL,-1      ; BUFFER-FILL
884 050D 3C FF     JE     K59      ; IS THIS AN IGNORE CHAR
885 050F T4 05     CMP    AH,-1      ; YES, DO NOTHING WITH IT
886 0511 80 FC FF   JNE    K61      ; LOOK FOR -1 PSEUDO SCAN
887 0514 75 03
888
889 0516          K59:    JMP    K26      ; NEAR_INTERRUPT_RETURN
890 0516 E9 02CA R
891
892 0519          K61:    CLI
893 0519 FA        MOV    BX,*BUFFER_TAIL ; GET THE END POINTER TO THE BUFFER
894 051A BB 0E 001C R   MOV    S1,BX      ; SAVE THE VALUE
895 051E BB F3        CALL   K4       ; ADVANCE THE TAIL
896 0520 E8 0114 R     CMP    BX,*BUFFER_HEAD ; HAS THE BUFFER WRAPPED AROUND
897 0523 3B IE 001A R   JE     K62      ; BUFFER_FULL_BEEP
898 0524 E8 74 07     MOV    AX,09102H ; STORE THE VALUE
899 0529 BB 00 04     MOV    BX,09102H ; MOVE THE POINTER UP
900 052B 89 1E 001C R   MOV    BX,BUFFER_TAIL,BX ; END OF INTERRUPT COMMAND
901 052F B0 20        MOV    AL,E01    ; SEND COMMAND TO INTERRUPT CONTROL PORT
902 0531 E6 20        OUT    020H,AL
903 0533 B8 9102     MOV    AX,09102H ; MOVE IN POST CODE & TYPE
904 0534 CD 15        INT    15H      ; PERFORM OTHER FUNCTION
905 0538 B0 28 0096 R   AND   0KB_FLAG_3,NOT LC_E0+LC_E1 ; RESET LAST CHAR H.C. FLAG
906 053D E9 02D4 R   JMP    K27      ; INTERRUPT_RETURN
907
908 :----- BUFFER IS FULL SOUND THE BEEPER
909
910 0540          K62:    MOV    AL,E01    ; ENABLE INTERRUPT CONTROLLER CHIP
911 0540 B0 20        OUT    INTA00,AL
912 0542 E6 20        MOV    CX,678    ; DIVISOR FOR 1760 Hz
913 0544 B9 02A6     MOV    BL,4     ; SHORT BEEP COUNT (1/16 + 1/64 DELAY)
914 0547 B3 04        CALL   BEEP    ; GO TO COMMON BEEP HANDLER
915 0549 E8 0000 E     JMP    K27      ; EXIT
916 054C E9 02D4 R
917
918 054F          KB_INT_I ENDP

```

```

PAGE :----- KEY IDENTIFICATION SCAN TABLES -----
:----- TABLE OF SHIFT KEYS AND MASK VALUES -----
K6 :----- KEY TABLE -----
LABEL BYTE
DB INS_KEY
DB CAPS_KEY,NUM_KEY,SCROLL_KEY,ALT_KEY,CTL_KEY
DB LEFT_KEY,RIGHT_KEY
EQU -$K6-
K6L
:----- MASK_TABLE -----
LABEL BYTE
DB INS_SHIFT
DB CAPS_SHIFT,NUM_SHIFT,SCROLL_SHIFT,ALT_SHIFT,CTL_SHIFT
DB LEFT_SHIFT,RIGHT_SHIFT
K7
:----- TABLES FOR CTRL CASE -----
K8 :----- LABEL BYTE -----
LABEL BYTE
DB 27,-1,00,-1,-1,-1 ; Esc, I, 2, 3, 4, 5
DB 30,-1,-1,-1,-1,-1 ; 6, 7, 8, 9, 0
DB 10,127,148,17,23,5 ; Bksp, Tab, Q, W, E
DB 16,29,110,09,55 ; R, T, Y, U, I, O
DB 16,29,110,09,55 ; S, D, F, G, H, J
DB 19,04,06,07,08,10 ; Enter, Ctrl, A
DB 11,12,-1,-1,-1,-1 ; K, L, ;, , , Lshift
DB 28,26,24,03,22,02 ; I, Z, X, C, V, B
DB 14,13,-1,-1,-1,-1 ; N, M, ., /, Rshift
DB 150,-1,-1,-1,-1,-1 ; Alt, Space, CL
:----- CHARACTERS -----
:----- FUNCTIONS -----
DB 94,95,96,97,98,99 ; F1 - F6
DB 100,101,102,97,103,-1,-1 ; F7 - F10, NL, SL
DB 119,141,132,142,115,143 ; Home, Up, PgUp, End, Down, PgDn, Ins
DB 116,144,117,145,118,146 ; Right, +, Left, -, Underline, Del, SysRq
DB 147,-1,-1,-1,137,138 ; SysReq, Under, WT, F11, F12
;----- TABLES FOR LOWER CASE -----
:----- K10 -----
LABEL BYTE
DB 27,-1,12345
DB *67890-
DB *,08,09,*qwe*
DB *r,t,y,u,i,o*
DB *p||,00H,-1,*a*
DB *s,d,f,g,h,j*
DB *k,l,;,.,,lshift*
DB *\zxcvb*
DB *nm,./
DB -*,-1,-1,-1,-1,-1 ; R Shift, *, Alt, Space, CL
:----- LC TABLE SCAN -----
DB 59,60,61,62,63 ; BASE STATE OF F1 - F10
DB 64,65,66,67,68 ; NL, SL
DB -1,-1
:----- K15 -----
LABEL BYTE
DB 71,72,73,-1,75,-1 ; BASE STATE OF KEYPAD KEYS
DB 77,-1,79,80,81,82
DB 83
DB -1,-1,\*,133,134 ; SysRq, Undef, WT, F11, F12
:----- KEYPAD TABLE -----
:----- K11 -----
LABEL BYTE
DB 27,'#*$%'
DB *`&(*'
DB *+,08,09,*QWE*
DB *R,T,Y,U,I,O*
DB *P||,00H,-1,*A*
DB *S,D,F,G,H,J*
DB *K,L,;,.,,Lshift*
DB *\ZXCVB*
DB *NM,>?*
DB -*,-1,-1,-1,-1,-1 ; R Shift, *, Alt, Space, CL
:----- UC TABLE SCAN -----
K12 :----- LABEL BYTE -----
LABEL BYTE
DB 84,85,86,87,88 ; SHIFTED STATE OF F1 - F10
DB 89,90,91,92,93 ; NL, SL
DB -1,-1
:----- NUM STATE TABLE -----
K14 :----- LABEL BYTE -----
LABEL BYTE
DB *789-456+1230.* ; NUMLOCK STATE OF KEYPAD KEYS
CODE ENDS
ENDS -1,-1,\*,135,136 ; SysRq, Undef, WT, F11, F12
1010
1003
1004 0655
1005 0656 37 38 39 2D 34 35
1006 0657 36 37 38 2D 34 35
1007 0658 2A 36
1008 0662 FF FF 7C 87 88
1009 0667
1010

```

```

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

0000      CODE     SEGMENT BYTE PUBLIC
0001      PUBLIC    PRINTER_10_
0002      EXTRN    DDS:NEAR

0003      ;--- INT 17 H ---
0004      PRINTER_10
0005      THIS ROUTINE PROVIDES COMMUNICATION WITH THE PRINTER
0006      INPUT
0007          (AH)= 00H PRINT THE CHARACTER IN (AL)
0008          ON RETURN, (AH)= 1 IF CHARACTER NOT PRINTED (TIME OUT)
0009          OTHER BITS SET AS ON NORMAL STATUS CALL
0010          (AH)= 01H INITIALIZE THE PRINTER PORT
0011          RETURN WITH (AH) SET WITH PRINTER STATUS
0012          (AH)= 02H READ THE PRINTER STATUS INTO (AH)
0013
0014          7   6   5   4   3   2-1   0
0015          |   |   |   |   |   |   |
0016          |   |   |   |   |   |   _ I = TIME OUT
0017          |   |   |   |   |   |   _ I = I/O ERROR
0018          |   |   |   |   |   |   _ I = SELECTED
0019          |   |   |   |   |   |   _ I = OUT OF PAPER
0020          |   |   |   |   |   |   _ I = ACKNOWLEDGE
0021          |   |   |   |   |   |   _ I = NOT BUSY
0022
0023          (DX) = PRINTER TO BE USED (0,1,2) CORRESPONDING TO ACTUAL VALUES
0024          IN PRINTER BASE AREA
0025          DATA AREA @PRINTER_BASE CONTAINS THE BASE ADDRESS OF THE PRINTER CARD(S)
0026          AVAILABLE (LOCATED AT BEGINNING OF DATA SEGMENT, 40BH ABSOLUTE, 3 WORDS)
0027
0028          DATA AREA @PRINT_TIM_OUT (BYTE) MAY BE CHANGE TO CAUSE DIFFERENT
0029          TIME OUT WAITS. DEFAULT=20
0030
0031          REGISTERS (AH) IS MODIFIED WITH STATUS INFORMATION
0032          ALL OTHERS UNCHANGED
0033
0034
0035
0036
0037
0038
0039
0040
0041
0042
0043
0044
0045
0046
0047
0048      ASSUME CS:CODE,DS:DATA
0049
0050      0000      PRINTER_10_ PROC FAR
0051          STI           ; ENTRY POINT FOR ORG 0EFD2H
0052          0000  FB      PUSH DX        ; INTERRUPTS BACK ON
0053          0001  52      PUSH BX        ; SAVE WORK REGISTERS
0054          0002  53      CMP  DX,03H    ; CHECK FOR PRINTER NUMBER VALID 0-3
0055          0003  83  FA  03  JA   B10       ; ERROR EXIT IF OUT OF RANGE
0056          0006  77  25
0057
0058          0008  8A  FB      MOV   BH,AL    ; SAVE CHARACTER TO BE PRINTED
0059          000A  1E      PUSH DS        ; SAVE SEGMENT
0060          000B  E0  0000  E  CALL DDS       ; ADDRESS DATA SEGMENT
0061
0062          000E  56      PUSH SI        ; SAVE WORK POINTER REGISTER
0063          000F  8B  F2      MOV   S1,DX    ; GET PRINTER PARAMETER
0064          0011  8A  9C  0078  R  MOV   BL,@PRINT_TIM_OUT[S1] ; LOAD TIMEOUT VALUE
0065          0015  D1  E6      SHL   S1,1      ; WORD OFFSET SET INTO TABLE INTO (S1)
0066          0017  8B  94  0008  R  MOV   DX,@PRINTER_BASE[S1] ; GET BASE ADDRESS FOR PRINTER CARD
0067          001B  5E      POP   S1        ; RECOVER CALLERS (S1) REGISTER
0068          001C  1F      POP   DS        ; AND (DS) SEGMENT REGISTER
0069
0070          001D  0B  D2      ASSUME DS:NOTHING
0071          001F  74  0C      OR    DX,DX    ; TEST DX = ZERO, INDICATING NO PRINTER
0072          0021  0A  E4      JZ   B10       ; EXIT, NO PRINTER ADAPTER AT OFFSET
0073          0023  74  0D
0074
0075          0025  FE  CC      DFC  AH        ; TEST FOR (AH)= 00H
0076          0027  74  4B      JZ   B80       ; PRINT CHARACTER IN (AL)
0077
0078          0029  FE  CC      DEC  AH        ; TEST FOR (AH)= 01H
0079          002B  74  39      JZ   B60       ; INITIALIZE PRINTER
0080
0081          002D
0082          002D  B4  29      B10: MOV  AH,029H  ; RETURN ERROR BITS FOR INVALID CALLS
0083
0084          002F
0085          002F  B20:      B20:  MOV  AH,029H  ; RETURN ERROR BITS FOR INVALID CALLS
0086          002F  5B      POP   BX        ; RETURN
0087          0030  5A      POP   DX        ; RECOVER REGISTERS
0088          0031  CF      IRET      ; RETURN TO CALLING PROGRAM (AH)= STATUS
0089
0090
0091          ;----- PRINT THE CHARACTER IN (AL)
0092          0092  EE      B30:  OUT  DX,AL    ; OUTPUT CHARACTER TO DATA PORT
0093          0092  EE      INC   DX        ; POINT TO STATUS PORT
0094          0093  42

```

```

96 PAGE
97 :----- 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 B0    TEST AL,80H        ; IS THE PRINTER CURRENTLY BUSY
102 0038 75 05    JNZ B40           ; SKIP SYSTEM DEVICE BUSY CALL IF NOT
103
104 :----- 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 :----- WAIT BUSY
110
111 003F          B40:             ; SAVE CALLERS (CX) REGISTER
112 003F 51      PUSH CX           ; INNER LOOP (64K)
113 0040 2B C9    SUB CX,CX
114 0042          B45:             ; GET STATUS
115 0042 EC      IN AL,DX          ; STATUS TO (AH) ALSO
116 0043 8A E0    MOV AH,AL
117 0045 A8 B0    TEST AL,80H        ; IS THE PRINTER CURRENTLY BUSY
118 0047 75 0F    JNZ B50           ; GO TO OUTPUT STROBE
119
120 0049 E2 F7    LOOP B45          ; LOOP IF NOT
121
122 004B FE CB    DEC BL           ; DECREMENT OUTER LOOP COUNT
123 004D 75 F3    JNZ B45           ; MAKE ANOTHER PASS IF NOT ZERO
124
125 004F 59      POP CX            ; RESTORE (CX) WITH CALLERS VALUE
126 0050 80 CC 01  OR AH,1           ; SET ERROR FLAG
127 0053 80 E4 F9  AND AH,OF9H        ; TURN OFF THE UNUSED BITS
128 0055 EB 15    JMP SHORT B70      ; RETURN WITH ERROR FLAG SET
129
130 0058          B50:             ; SEND STROBE PULSE
131 0058 59      POP CX            ; RESTORE (CX) WITH CALLERS VALUE
132 0059 B0 0D    MOV AL,0DH         ; SET THE STROBE LOW (BIT ON)
133 005B 42      INC DX           ; OUTPUT STROBE TO CONTROL PORT
134 005C FA      CLI              ; PREVENT INTERRUPT PULSE STRETCHING
135 005D EE      OUT DX,AL         ; OUTPUT STROBE BIT > 1us < 5us
136 005E EB 00    JMP $+2           ; I/O DELAY TO ALLOW FOR LINE LOADING
137
138 0060 B0 0C    MOV AL,0CH         ; AND FOR CORRECT PULSE WIDTH
139 0062 EE      OUT DX,AL         ; SET THE -STROBE HIGH
140 0063 FB      STI              ; INTERRUPTS BACK ON
141 0064 4A      DEC DX           ; ADJUST BACK TO BASE ADDRESS
142 0065 4A      DEC DX           ; FOR STATUS ROUTINE EXIT
143
144
145 :----- PRINTER STATUS
146
147 0066          B60:             ; POINT TO CONTROL PORT
148 0066 42      INC DX           ; PRE-CHARGE +BUSY LINE IF FLOATING
149 0067 EC      INC AL,DX         ; GET PRINTER STATUS HARDWARE BITS
150 0068 EC      IN AL,DX
151 0069 24 F8    AND AL,0F8H        ; TURN OFF UNUSED BITS
152 006B 8A E0    MOV AH,AL
153 006D          B70:             ; SAVE
154 006D B8 C7    MOV AL,BH          ; RECOVER CHARACTER INTO (AL) REGISTER
155 006F B0 F4 48  XOR AH,48H        ; FLIP A COUPLE OF BITS IN STATUS
156 0072 EB BB    JMP B20           ; RETURN FROM ROUTINE WITH STATUS IN AH
157
158
159 :----- INITIALIZE THE PRINTER PORT
160
161 0074          B80:             ; POINT TO OUTPUT PORT
162 0074 42      INC DX           ; SET INIT LINE LOW
163 0075 42      INC DX
164 0076 B0 08    MOV AL,8           ; ADJUST FOR INITIALIZATION DELAY LOOP
165 0077 00 EE 08  OUT DX,AL
166 0079 B8 03E8  MOV AX,1000
167 007C          B90:             ; DECREMENT DELAY COUNTER
168 007C 48      DEC AX           ; LOOP FOR RESET TO TAKE
169 007D 75 FD    JNZ B90
170
171 007F B0 0C    MOV AL,0CH         ; NO INTERRUPTS, NON AUTO LF, INIT HIGH
172 0081          OUT DX,AL         ; SET DEFAULT INITIAL OUTPUTS
173 0082 4A      DEC DX           ; ADJUST BACK TO BASE ADDRESS
174 0083 4A      DEC DX
175 0084 EB E0    JMP B60           ; FOR STATUS ROUTINE EXIT
176
177 0086          PRINTER_10_1  ENDP
178
179 0086          CODE ENDS
180

```

```

1 PAGE 118,121
2 TITLE RS232 ---- 01/10/86 COMMUNICATIONS BIOS (RS232)
3 .LIST
4 CODE SEGMENT BYTE PUBLIC
5
6 PUBLIC RS232_10_1
7 EXTRN A1:NEAR
8 EXTRN DDS:NEAR
9
10 .---- INT 14 H
11 RS232_10_1
12 ; THIS ROUTINE PROVIDES BYTE STREAM I/O TO THE COMMUNICATIONS
13 ; PORT ACCORDING TO THE PARAMETERS
14 ;
15 ; (AH)= 00H INITIALIZE THE COMMUNICATIONS PORT
16 ; (AL) HAS PARAMETERS FOR INITIALIZATION
17 ;
18 ;----- BAUD RATE -- -PARITY-- STOPBIT --WORD LENGTH--|
19 ;----- 7 6 5 4 3 2 1 0
20 ;----- 000 - 110 X0 - NONE 0 - 1 10 - 7 BITS
21 ;----- 001 - 150 01 - ODD 1 - 2 11 - 8 BITS
22 ;----- 010 - 300 11 - EVEN
23 ;----- 011 - 600
24 ;----- 100 - 1200
25 ;----- 101 - 2400
26 ;----- 110 - 4800
27 ;----- 111 - 9600
28 ;----- 111 - 0000
29 ;----- ON RETURN, CONDITIONS SET AS IN CALL TO COMMO STATUS (AH=03H)
30
31 ;(AH)= 01H SEND THE CHARACTER IN (AL) OVER THE COMM LINE
32 ;(AL) REGISTER IS PRESERVED
33 ;ON EXIT, BIT 7 OF AH IS SET IF THE ROUTINE WAS UNABLE TO
34 ;TO TRANSMIT THE BYTE OF DATA OVER THE LINE.
35 ;IF BIT 7 OF AH IS NOT SET, THE
36 ;REMAINDER OF (AH) IS SET AS IN A STATUS REQUEST,
37 ;REFLECTING THE CURRENT STATUS OF THE LINE.
38 ;(AH)= 02H RECEIVE A CHARACTER IN (AL) FROM COMMO LINE BEFORE
39 ;PUTTING IT INTO AL
40 ;ON EXIT, (AH) HAS THE CURRENT LINE STATUS, AS SET BY THE
41 ;THE STATUS ROUTINE, EXCEPT THAT THE ONLY BITS
42 ;LEFT ON ARE THE ERROR BITS (7,4,3,2,1)
43 ;IF (AH) HAS BIT 7 ON (TIME OUT) THE REMAINING
44 ;BITS ARE NOT PREDICTABLE.
45 ;THUS, (AH) IS NON ZERO ONLY WHEN AN ERROR OCCURRED.
46 ;(AH)= 03H RETURN THE COMM PORT STATUS IN (AX)
47 ;(AH) CONTAINS THE LINE CONTROL STATUS
48 ;BIT 7 = TIME OUT
49 ;BIT 6 = TRANSMIT SHIFT REGISTER EMPTY
50 ;BIT 5 = TRANSMIT HOLDING REGISTER EMPTY
51 ;BIT 4 = RECEIVE DECODED
52 ;BIT 3 = FRAMING ERROR
53 ;BIT 2 = PARITY ERROR
54 ;BIT 1 = OVERRUN ERROR
55 ;BIT 0 = DATA READY
56
57 ;(AL) CONTAINS THE MODEM STATUS
58 ;BIT 5 = RECEIVE LINE SIGNAL DETECT
59 ;BIT 4 = LINE 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 ;OUTPUT AX MODIFIED ACCORDING TO PARAMETERS OF CALL
74 ;ALL OTHERS UNCHANGED
75
76 ;ASSUME CS:CODE,DS:DATA
77
78 0000 RS232_10_1 PROC FAR
79 0000 FB ST1:           ; INTERRUPTS BACK ON
80 0001 1E PUSH DS          ; I SAVE SEGMENT
81 0002 52 PUSH DX
82 0003 66 PUSH SI
83 0004 57 PUSH DI
84 0005 51 PUSH CX
85 0006 53 PUSH BX
86 0007 83 FA 03 CMP DX,03H ; CHECK FOR ADAPTER NUMBER VALID 0-3
87 0008 3C 24 JZ A3E        ; TEST FOR 0 BASE ADDRESS
88 0009 BB F2 MOV SI,DX ; RETURN
89 000A BB FA MOV DI,DX ; RS232 VALUE TO (SI)
90 0010 D1 E6 SHL SI,1    ; AND TO (DI) (FOR TIMEOUTS)
91 0012 E8 0000 E CALL DDS ; WORD OFFSET
92 0015 BB 94 0000 R MOV DX,@RS232_BASE[SI] ; GET BASE ADDRESS
93 0016 3C 00 OR DX,DX ; TEST FOR 0 BASE ADDRESS
94 0018 74 33 JZ A3E        ; RETURN
95 001D 0A E4 OR AH,AH ; RS232
96 001F T4 18 JZ A4        ; TEST FOR (AH)= 00H
97 0021 FE CC DEC AH      ; COMMO INITIALIZATION
98 0023 74 4B JZ A5        ; TEST FOR (AH)= 01H
99 0025 FE CC DEC AH      ; SEND (AL)
100 0027 74 70 JZ A12       ; TEST FOR (AH)= 02H
101 0029          A2: DEC AH      ; RECEIVE INTO (AL)
102 0029 FE CC JNC A3E        ; TEST FOR (AH)= 03H
103 002B 75 03 JMP A18        ; ERROR IF BAD COMMAND
104 002D E9 00BB R          ; COMMUNICATION STATUS
105 0030 0000 R
106 0030 B4 80 A3E: MOV AH,080H ; SET ERROR RETURN CODE
107 0032          A3: POP BX      ; RETURN FROM RS232
108 0032 5B POP CX
109 0033 59 POP DI
110 0034 5F POP SI
111 0035 1E POP DX
112 0036 5A POP DS
113 0037 1F IRET          ; RETURN TO CALLER, NO ACTION
114 0038 CF

```

```

PAGE 1----- INITIALIZE THE COMMUNICATIONS PORT
116
117
118 0039
119 0009 8A E0 : SAVE INITIALIZATION PARAMETERS IN (AH)
120 0039 B3 C2 03
121 003E B0 80 : POINT TO 8250 CONTROL REGISTER
122 0040 EE : SET DLAB=1
123
124 1----- DETERMINE BAUD RATE DIVISOR
125
126 0041 8A D4 : GET PARAMETERS TO (DL)
127 0043 B1 04
128 0045 D2 C2
129 0047 B1 E2 000E : ISOLATE THEM
130 0048 BF 0000 E : BASE OF TABLE
131 0049 B3 FA : PUT INTO INDEX REGISTER
132 0050 B9 94 0000 R : POINT TO HIGH ORDER OF DIVISOR
133 0054 A2
134 0055 8E 8A 45 01 : GET HIGH ORDER OF DIVISOR
135 0059 EE : SET ms OF DIVISOR TO 0
136 005A 4A
137 005B 90
138 005C 8C 8A 05 : I/O DELAY
139 005F EE : GET LOW ORDER OF DIVISOR
140 0060 B3 C2 03 : SET LOW OF DIVISOR
141 0063 8A C4
142 0065 24 1F : GET PARAMETERS BACK
143 0066 01 F8 : STRIP OFF THE BAUD BITS
144 0068 4A : LINE CONTROL TO 8 BITS
145 0069 4A
146 006A 90
147 006B B0 00 : NOP
148 006D EE : I/O DELAY
149 006E EB 4B : INTERRUPT ENABLES ALL OFF
150
151
152 1----- SEND CHARACTER IN (AL) OVER COMMO LINE
153 0070
154 0070 50 : SAVE CHAR TO SEND
155 0071 B3 C2 04 : MODEM CONTROL REGISTER
156 0074 B0 03 : DTR AND RTS
157 0076 EE : DATA TERMINAL READY, REQUEST TO SEND
158 0077 42
159 0078 42
160 0079 B7 30 : MODEM STATUS REGISTER
161 007B E8 00CA R : DATA SET READY & CLEAR TO SEND
162 007C 80 74 06 : ARE BOTH TRUE
163 0080 : YES, READY TO TRANSMIT CHAR
164 0080 59
165 0081 8A C1
166 0083
167 0083 B0 CC 80 : RELOAD DATA BYTE
168 0086 EB AA : INDICATE TIME OUT
169
170 0088
171 0088 4A : RETURN
172 0089
173 0089 B7 20 : CLEAR TO SEND
174 008B E8 00CA R : LINE STATUS REGISTER
175 008E 75 F0 : WAIT SEND
176 0090 : IS TRANSMITTER READY
177 0090 B3 EA 05 : TEST FOR TRANSMITTER READY
178 0093 59 : RETURN WITH TIME OUT SET
179 0094 8A C1 : OUT_CHAR
180 0096 EE : DATA PORT
181 0097 EB 99 : RECEIVE IN CX TEMPORARILY
182
183 1----- RECEIVE CHARACTER FROM COMMO LINE
184
185 0099
186 0099 B3 C2 04 : MOVE TO AL
187 009C B0 01 : DATA TERMINAL READY
188 009E EE : INC DX
189 009F 42 : INC INC DX
190 00A0 42
191 00A1 4A : WAIT_DSR
192 00A1 B7 20 : DATA_SET_READY
193 00A3 E8 00CA R : TEST FOR DSR
194 00A5 75 DB : RETURN WITH ERROR
195 00A8
196 00A8 4A : WAIT_DSR_END
197 00A9 : LINE_STATUS_REGISTER
198 00A9 B7 01 : WAIT_DSR
199 00AB E8 00CA R : RECEIVE_BUFFER_FULL
200 00AE 75 D3 : TEST FOR RECEIVE_BUFFER_FULL
201 00B0 : SET_TIMEOUT_ERROR
202 00B0 B0 E4 IE : GET_CHAR
203
204 00B3 BB 94 0000 R : DATA_PORT
205 00B7 EC : GET_CHARACTER_FROM_LINE
206 00B8 E9 0032 R : RETURN
207
208 1----- COMMO PORT STATUS ROUTINE
209
210 00BB : MOV DX,RS232_BASE[S1] : CONTROL_PORT
211 00BB BB 94 0000 R : GET LINE CONTROL STATUS
212 00BB B3 C2 05 : PUT (AH) FOR RETURN
213 00C2 EC : POINT TO MODEM STATUS REGISTER
214 00C4 8A E0 : GET MODEM CONTROL STATUS
215 00C5 42 : RETURN
216 00C6 EC
217 00C7 E9 0032 R : RETURN

```

```

218 PAGE
219 ;-----[PAGE]-----;
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 = NOT FOUND. ;
225 ;        (AH) = LAST STATUS READ ;
226 ;-----[PAGE]-----;
227
228 00CA WAIT_FOR_STATUS PROC NEAR
229
230    00CA 8A 9D 007C R      WFS0: MOV BL, @RS232_TIM_OUT[D1]  ; LOAD OUTER LOOP COUNT
231    00CE                               ;-----[PAGE]-----;
232    00CE 2B C9      WFS1: SUB CX, CX
233    00D0
234    00D0 EC      IN AL, DX          ; GET STATUS
235    00D1 8A E0      MOV AH, AL          ; MOVE TO (AH)
236    00D3 22 C7      AND AL, BH          ; ISOLATE BITS TO TEST
237    00D5 3A C7      CMP AL, BH          ; EXACTLY = TO MASK
238    00D7 74 08      JE WFS_END        ; RETURN WITH ZERO FLAG ON
239
240    00D9 E2 F5      LOOP WFS1         ; TRY AGAIN
241
242    00D0 FE CB      DEC BL           ; DECREMENT LOOP COUNTER
243    00D0 75 EF      JNZ WFS0
244
245    00D6 0A FF      OR BH, BH        ; SET ZERO FLAG OFF
246    00E1
247    00E1 C3      WFS_END: RET
248
249    00E2 WAIT_FOR_STATUS ENDP
250
251    00E2 RS232_IO_I ENDP
252
253    00E2 CODE ENDS
254    00E2 END

```

```

PAGE 118,121
TITLE VIDEO ---- 01/10/86 VIDEO DISPLAY BIOS
.LIST
CODE SEGMENT BYTE PUBLIC
5
6    PUBLIC IC ACT_DISP PAGE
7    PUBLIC IC READ_AC_CURRENT
8    PUBLIC IC READ_CURSOR
9    PUBLIC IC READ_DOT
10   PUBLIC IC READ_LPEN
11   PUBLIC IC SCROLL_DOWN
12   PUBLIC IC SCROLL_UP
13   PUBLIC IC SET_CURSOR
14   PUBLIC IC SET_CPOS
15   PUBLIC IC SET_CTYPE
16   PUBLIC IC SET_MODE
17   PUBLIC IC WRITE_AC_CURRENT
18   PUBLIC IC WRITE_C_CURRENT
19   PUBLIC IC WRITE_DOT
20   PUBLIC IC WRITE_LPEN
21   PUBLIC IC VIDEO_IO_1
22   PUBLIC IC VIDEO_STATE
23
24   PUBLIC IC SET_MODE
25   PUBLIC IC SET_CTYPE
26   PUBLIC IC SET_CPOS
27   PUBLIC IC READ_CURSOR
28   PUBLIC IC READ_LPEN
29   PUBLIC IC ACT_DISP PAGE
30   PUBLIC IC SCROLL_UP
31   PUBLIC IC SCROLL_DOWN
32   PUBLIC IC READ_AC_CURRENT
33   PUBLIC IC WRITE_AC_CURRENT
34   PUBLIC IC WRITE_C_CURRENT
35   PUBLIC IC SET_COLOR
36   PUBLIC IC WRITE_DOT
37   PUBLIC IC READ_DOT
38   PUBLIC IC SET_MODE
39   PUBLIC IC VIDEO_STATE
40   PUBLIC IC VIDEO_RETURN
41   PUBLIC IC VIDEO_RETURN
42   PUBLIC IC VIDEO_RETURN
43   PUBLIC IC WRITE_STRING
44
45   EXTRN BEEP:Near           ; SPEAKER BEEP ROUTINE
46   EXTRN CRT_CHAR_GEN:Near   ; CHARACTER GENERATOR GRAPHICS TABLE
47   EXTRN DDSINEAR            ; LOAD (DS) WITH DATA SEGMENT SELECTOR
48   EXTRN M5IWORD             ; REGEN BUFFER LENGTH TABLE
49   EXTRN M6BYTE               ; COLUMNS PER MODE TABLE
50   EXTRN M7BYTE               ; MODE SET VALUE PER MODE TABLE
51
52
53   INT 10 H
54
55   VIDEO_IO
56   ; THESE ROUTINES PROVIDE THE CRT DISPLAY INTERFACE
57   ; THE FOLLOWING FUNCTIONS ARE PROVIDED:
58
59   (AH)= 00H SET_MODE (AL) CONTAINS MODE VALUE
60   (AL) = 00H 40X25 BW MODE (POWER ON DEFAULT)
61   (AL) = 01H 40X25 COLOR
62   (AL) = 02H 80X25 BW
63   (AL) = 03H 80X25 COLOR
64   (AL) = 04H 320X200 GRAPHS MODES
65   (AL) = 05H 320X200 BW MODE
66   (AL) = 06H 640X200 BW MODE
67   (AL) = 07H 80X25 MONOCHROME (USED INTERNAL TO VIDEO ONLY)
68   *** NOTES -BW MODES OPERATE SAME AS COLOR MODES, BUT COLOR
69   BURST IS NOT ENABLED
70   CURSOR IS NOT DISPLAYED IN GRAPHICS MODE
71
72   (AH)= 01H SET_CURSOR_TYPE
73   (CH) = BITS 4-0 = START LINE FOR CURSOR
74   ** HARDWARE WILL ALWAYS CAUSE BLINK
75   ** SETTING BIT 5 OR 6 WILL CAUSE ERRATIC BLINKING
76   OR NO CURSOR AT ALL
77   (CL) = BITS 1-3 = END LINE FOR CURSOR
78
79   (AH)= 02H SET_CURSOR_POSITION
80   (DH,DL) = ROW,COLUMN (00H,00H) = UPPER LEFT
81   (BH) = PAGE NUMBER (MUST BE 00H FOR GRAPHICS MODES)
82
83   (AH)= 03H READ_CURSOR_POSITION
84   (BH) = PAGE NUMBER (MUST BE 00H FOR GRAPHICS MODES)
85   ON EXIT (DH,DL) = ROW,COLUMN OF CURRENT CURSOR
86   (CH,CL) = CURSOR MODE CURRENTLY SET
87
88   (AH)= 04H READ_LIGHT_PEN POSITION
89   ON EXIT:
90   (AH) = 00H -- LIGHT PEN SWITCH NOT DOWN/NOT TRIGGERED
91   (AH) = 01H -- VALID LIGHT PEN VALUE IN REGISTERS
92   (DH,DL) = ROW,COLUMN OF CHARACTER LP POSITION
93   (CH,CL) = PAPER POSITION (0-19, 63-99)
94   (BX) = PIXEL COLUMN (0-319, 639)
95
96   (AH)= 05H SELECT_ACTIVE_DISPLAY_PAGE (VALID ONLY FOR ALPHA MODES)
97   (AL) = NEW PAGE VALUE (0-7 FOR MODES 041, 0-3 FOR MODES 243)
98
99   (AH)= 06H SCROLL_ACTIVE_PAGE_UP
100  (AL) = NUMBER OF LINES, INPUT LINES BLANKED AT BOTTOM OF WINDOW
101  (CH,CL) = ROW,COLUMN OF UPPER LEFT CORNER OF SCROLL
102  (DH,DL) = ROW,COLUMN OF LOWER RIGHT CORNER OF SCROLL
103  (BH) = ATTRIBUTE TO BE USED ON BLANK LINE
104
105  (AH)= 07H SCROLL_ACTIVE_PAGE_DOWN
106  (AL) = NUMBER OF LINES, INPUT LINES BLANKED AT TOP OF WINDOW
107  (CH,CL) = ROW,COLUMN OF UPPER LEFT CORNER OF SCROLL
108  (DH,DL) = ROW,COLUMN OF LOWER RIGHT CORNER OF SCROLL
109  (BH) = ATTRIBUTE TO BE USED ON BLANK LINE
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
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
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
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
635
636
637
638
639
639
640
641
642
643
644
644
645
646
647
648
648
649
649
650
651
652
653
654
655
655
656
657
658
658
659
659
660
661
662
663
663
664
665
665
666
666
667
667
668
668
669
669
670
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1540
1541
1541
1542
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1550
1551
1551
1552
1552
1553
1553
1554
1554
1555
1555
1556
1556
1557
1557
1558
1558
1559
1559
1560
1560
1561
1561
1562
1562
1563
1563
1564
1564
1565
1565
1566
1566
1567
1567
1568
1568
1569
1569
1570
1570
1571
1571
1572
1572
1573
1573
1574
1574
1575
1575
1576
1576
1577
1577
1578
1578
1579
1579
1580
1580
1581
1581
1582
1582
1583
1583
1584
1584
1585
1585
1586
1586
1587
1587
1588
1588
1589
1589
1590
1590
1591
1591
1592
1592
1593
1593
1594
1594
1595
1595
1596
1596
1597
```

```

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 1 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 MODES.
125          ; THE CHARACTERS ARE FORMED FROM 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 (0000H) TO POINT TO THE 1K BYTE TABLE CONTAINING
130          ; THE CODE POINTS FOR THE SECOND 128 CHARS.
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          ; MEETING THE COLOR ID REQUESTED.
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 ID INDICATES THE BORDER COLOR
149          ; TO BE USED. (VALUES 0-3) WHERE 16-31 SELECT
150          ; THE HIGH INTENSITY BACKGROUND SET.
151
152          ; (AH)= 0CH WRITE DOT
153          ; (DX) = ROW NUMBER
154          ; (CX) = COLUMN NUMBER
155          ; (AL) = COLOR VALUE
156          ; IF BIT 7 OF AL = 1, THEN THE COLOR VALUE IS EXCLUSIVE
157          ; ORed WITH THE CURRENT CONTENTS OF THE DOT
158
159          ; (AH)= 0DH READ DOT
160          ; (DX) = ROW NUMBER
161          ; (CX) = COLUMN NUMBER
162          ; (AL) RETURNS THE DOT READ
163
164          ; ASCII TELETYPE ROUTINE FOR OUTPUT
165
166          ; (AH)= 0EH WRITE TELETYPE TO ACTIVE PAGE
167          ; (AL) = CHAR TO WRITE
168          ; (BL) = FOREGROUND COLOR IN GRAPHICS MODE
169          ; NOTE: SCREEN WIDTH IS CONTROLLED BY PREVIOUS MODE SET
170          ; (AH)= 0FH CURRENT VIDEO STATE
171          ; RETURNS THE CURRENT VIDEO STATE
172          ; (AL) = MODE CURRENTLY SET (SEE (AH)= 0OH FOR EXPLANATION)
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          ;         - LENGTH OF CHARACTER STRING TO BE 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 OF 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          ; 0000 009F R
208          ; 0002 0146 R
209          ; 0004 0147 R
210          ; 0006 018F R
211          ; 0008 0785 R
212          ; 000A 0146 R
213          ; 000C 0200 R
214          ; 000D 014D R
215          ; 0010 02FF R
216          ; 0012 035C R
217          ; 0014 038E R
218          ; 0016 01C8 R
219          ; 0018 0450 R
220          ; 001A 0451 R
221          ; 001C 06FE R
222          ; 001E 01EE R
223          ; 0020 013D R
224          ; 0022 013D R
225          ; 0024 013D R
226          ; 0026 03BB R
227          ; 0028
228
229          ; M1 DW 0FFSET SET_MODE           ; TABLE OF ROUTINES WITHIN VIDEO I/O
230          ; DW 0FFSET SET_CTYPE
231          ; DW 0FFSET SET_CURSOR
232          ; DW 0FFSET READ_CURSOR
233          ; DW 0FFSET READ_LPEN
234          ; DW 0FFSET ACT_DISP_PAGE
235          ; DW 0FFSET SCROLL_UP
236          ; DW 0FFSET SCROLL_DWN
237          ; DW 0FFSET READ_AC_CURRENT
238          ; DW 0FFSET WRITE_AC_CURRENT
239          ; DW 0FFSET SET_COLOR
240          ; DW 0FFSET WRITE_DOT
241          ; DW 0FFSET READ_DOT
242          ; DW 0FFSET WRITE_TTY
243          ; DW 0FFSET VIDEO_STATE
244          ; DW 0FFSET VIDEO_RETURN ; I RESERVED
245          ; DW 0FFSET VIDEO_RETURN ; I RESERVED
246          ; DW 0FFSET VIDEO_RETURN ; I RESERVED
247          ; DW 0FFSET WRITE_STRING ; CASE 13H, WRITE STRING
248
249          ; M1L EQU $-M1

```

```
229 0028      VIDEO_10_I    PROC  NEAR   ; ENTRY POINT FOR ORG 0F065H
230 0028 FB    ST1    DS                 ; INTERRUPTS BACK ON
231 0029 FC    CLD    DS                 ; 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 11    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 0038 8E DE   MOV   DS,C1
245 003C 8B F0   MOV   SI,AX
246 003E A0 0010 R  MOV   AL,BYTE PTR EQUIP_FLAG ; SAVE COMMAND/DATA INTO (SI) REGISTER
247 0041 24 30   AND   AL,30H   ; GET THE EQUIPMENT FLAG VIDEO BITS
248 0043 3C 30   CMP   AL,30H   ; ISOLATE CRT SWITCHES
249 0045 BF B800  MOV   D1,0B800H ; IS SETTING FOR MONOCHROME CARD?
250 0046 45 03   JNE   M2
251 0044 BF B800  MOV   D1,0B800H ; GET SEGMENT FOR COLOR CARD
252 004D 44 04   JNE   M2,0B800H ; SKIP IF NOT MONOCHROME CARD
253 0044 8E C7   MOV   ES,DI
254 004F 8A C4   MOV   AL,AH
255 0051 98    CBW
256 0052 01 E0   SAL   AX,1
257 0054 96    XCHG  SI,AX
258
259 0055 8A 26 0049 R  MOV   AH,OCR_MODE ; MOVE CURRENT MODE INTO (AH) REGISTER
260
261 0059 2E: FF A4 0000 R  JMP   WORD PTR CS:[SI+OFFSET M1] ; GO TO SELECTED FUNCTION
262
263 005E          M4:    IRET   ; COMMAND NOT VALID
264 005E CF    VIDEO_10_I  ENDP
265 005F          ;-----;
266 005E          ;-----;
267 005E          ;-----;
268 005E          ;-----;
269 005E          ;-----;
270 005E          ;-----;
271 005E          ;-----;
272 005E          ;-----;
273 005E          ;-----;
274 005E          ;-----;
275 005E          ;-----;
276 005E          ;-----;
277 005E          ;-----;
278 005E          ;-----;
279 005F          ;-----;
280 005F 8A 03D4  SET_MODE PROC  NEAR   ; THIS ROUTINE INITIALIZES THE ATTACHMENT TO
281 0062 8B 3E 0010 R  MOV   DX,03D4H ; THE SELECTED MODE. THE SCREEN IS BLANKED.
282 0066 81 E7 0030  MOV   DS,EQUIP_FLAG ; INPUT
283 0064 83 FF 30   AND   DS,30H   ; EQUIP_FLAG BITS 5-4 = MODE/WIDTH
284 0060 75 06    CMP   DS,30H   ; 11 = MONOCHROME (FORCES MODE 7)
285 006F B0 07    JNE   M8C
286 0071 B2 B4   MOV   AL,7
287 0073 EB 0D   MOV   DL,0B4H
288 0075          M8C:    SHORT M8 ; DS = COLOR MODE REQUESTED ( RANGE 0 - 8 )
289 0075 3C 07   CMP   AL,7
290 0077 T2 09   JB    M8
291 0079 B0 00   MOV   AL,0
292 007A 00 20   CMP   AL,20H
293 007E T4 02   JE    M8
294 0080 B0 02   MOV   AL,2
295 0082          MB:    CMP   AL,7
296 0082 A0 0049 R  MOV   AL,7
297 0085 B9 16 0063 R  MOV   AL,0
298 0086 C6 06 0084 R 18  MOV   DS,ROWS,25-1
299 008E          PUSH  DS
300 008F F0    PUSH  AX
301 0090 98    CBW
302 0091 B8 F0   MOV   SI,AX
303 0092 2E: 8A 84 0000 E  MOV   AL,CS:[SI + OFFSET M7]
304 0092 A0 0065 R  MOV   AL,OCR_MODE_SET,AL
305 0098 40 37   AND   AL,037H
306 0090 52    PUSH  DX
307 009E B3 C2 04  ADD   DX,4
308 00A1 EE    OUT   DX,AL
309 00A2 5A    POP   DX
310          ASSUME DS:AB50
311 00A3 0043 29 DB  SUB   DS,BX
312 00A5 BE DB   MOV   DS,BX
313 00A7 C5 1E 0074 R  LDS   BX,PPARM_PTR
314          ASSUME DS:CODE_
315 00AB 58    POP   AX
316 00AC B9 0010  MOV   CX,16
317 00BD 00 02    CMP   AL,CX
318 00B1 72 0E   JC    M9
319 00B3 03 D9   ADD   BX,CX
320 00B3 3C 04   CMP   AL,4
321 00B7 T2 08   JC    M9
322 00B9 03 D9   ADD   BX,CX
323 00B9 00 17   CMP   AL,7
324 00BD T2 02   JC    M9
325 00BF 03 D9   ADD   BX,CX
326
327          ;----- BX POINTS TO CORRECT ROW OF INITIALIZATION TABLE
328
329 00C1          M9:    OUT   INIT
330 00C1 50    PUSH  AX
331 00C2 8B 47 0A  MOV   AX,[BX+10]
332 00C5 86 E0   XCHG  AH,AL
333 00C7 1E    PUSH  DS
334          ASSUME DS:DATA
335 00C8 E8 0000 E  CALL  DS
336 00CB A3 0060 R  MOV   DS,CURSOR_MODE,AX
337          ASSUME DS:CODE_
338 00CE 1F    POP   DS
339 00CF 32 E4   XOR   AH,AH
340
341          ;----- LOOP THROUGH TABLE, OUTPUTTING REGISTER ADDRESS, THEN VALUE FROM TABLE
342
```

```

343 00D1 M10:           ; INITIALIZATION LOOP
344 00D1 8A C4          ; GET 6845 REGISTER NUMBER
345 00D3 EE             ; OUT DX,AL
346 00D4 42             ; INC DX
347 00D5 FE C4          ; INC AH
348 00D7 8A 07          ; MOV AL,[BX]
349 00D9 EE             ; OUT DX,AL
350 00D9 E3             ; INC BX
351 00DB 4A             ; DEC DX
352 00DC E2 F3          ; LOOP M10
353 00DE 58             ; POP AX
354 00DF 1F             ; POP DS
355 ASSUME DS:DATA      ; RECOVER SEGMENT VALUE
356
357 ;----- FILL REGEN AREA WITH BLANK
358
359 00E0 33 FF          ; XOR DI,DI
360 00E2 89 3E 004E R   ; MOV *CRT_START,DI
361 00E6 C6 06 0062 R 00 ; MOV *ACTIVE_PAGE,0
362 00E8 B0 0000          ; MOV CX,8192
363 00EE 0C 04          ; CMP AL,4
364 00F0 72 0A          ; JC M12
365 00F2 3C 07          ; CMP AL,7
366 00F4 74 04          ; JE M11
367 00F6 33 C0          ; XOR AX,AX
368 00F8 EB 05          ; JMP SHORT M13
369 00FA
370 00FA B5 08          ; MOV CH,0BH
371 00FC
372 00FC BB 0720          ; MOV AX,'+'+7H
373 00FF
374 00FF F3 / AB          ; REP STOSW
375
376 ;----- ENABLE VIDEO AND CORRECT PORT SETTING
377
378 0101 8B 16 0063 R   ; MOV DX,*ADDR_6845
379 0105 83 C2 04          ; ADD DX,4
380 0108 A0 0065 R          ; MOV AL,*CRT_MODE_SET
381 010B EE             ; OUT DX,AL
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]
387 0111 99
388 0112 A3 004A R          ; CLEAR HIGH BYTE
389
390 ;----- SET CURSOR POSITIONS
391
392 0115 81 E6 000E          ; AND SI,00EH
393 0119 2E: BB 84 0000 E  ; MOV AX,CS:[SI + OFFSET M5]
394 011E A3 004C R          ; MOV *CRT_LEN,AX
395 0121 B9 0008          ; MOV CX,8
396 0124 BF 0050 R          ; MOV DI,OFFSET *CURSOR_POSN
397 0127 1E
398 0129 07
399 012B 93 C0
400 012B F3 / AB          ; PUSH DS
401
402 ;----- SET UP OVERSCAN REGISTER
403
404 012D 42             ; INC DX
405 012E B0 30             ; MOV AL,30H
406 0130 80 3E 0049 R 06   ; CMP *CRT_MODE,6
407 0135 75 02             ; JNZ M14
408 0137 B0 3F             ; MOV AL,3FH
409 0139
410 0139 EE             ; OUT DX,AL
411 013A A2 0066 R          ; MOV *CRT_PALETTE,AL
412
413 ;----- NORMAL RETURN FROM ALL VIDEO RETURNS
414
415 013D
416 013E 5D
417 013E 5F
418 013F 5E
419 0140 5B
420 0141
421 0141 59
422 0141 5A
423 0143 5F
424 0144 07
425 0145 CF
426 0146
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_MODE
435 0146 PROC NEAR
436 0146 B4 0A             ; MOV AH,10
437 0148 89 0E 0060 R          ; MOV *CURSOR_MODE,CX
438 014C E8 0151 R             ; CALL M16
439 014F EB EC             ; JMP VIDEO_RETURN
440
441 ;----- THIS ROUTINE OUTPUTS THE CX REGISTER TO THE 6845 REGISTERS NAMED IN (AH)
442
443
444 0151
445 0151 8B 16 0063 R          ; MOV DX,*ADDR_6845
446 0155 C4
447 0156 EE
448 0158 42
449 0159 8A C5
450 015B EE
451 015C 4A
452 015D C4
453 015F FE C0
454 0161 EE
455 0162 42
456 0163 8A C1
457
458 M16:           ; ADDRESS REGISTER
459 0151 8A AH             ; GET VALUE
460 0152 80 AL             ; REGISTER SET
461 0153 80 AL             ; DATA REGISTER
462 0154 80 AL             ; DATA
463 0155 80 AL             ; POINT TO OTHER DATA REGISTER
464 0156 80 AL             ; SET FOR SECOND REGISTER
465 0157 80 AL             ; SECOND DATA VALUE

```

```

457 0165 EE OUT DX,AL
458 0166 C3 RET ; ALL DONE
459 0167 SET_CTYPE ENDP

461 ;-----+
462 ; SET_CURSOR POSITION
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 016B F7 00 MOV [SI+OFFSET *CURSOR_POSN],DX ; USE INDEX REGISTER
476 016D 89 94 0050 R CMP AX,[SI] ; SAVE THE POINTER
477 0171 3E 0062 R JNZ M17 ; SET_CPOS RETURN
478 0175 75 05 MOV AX,DX ; GET ROW/COLUMN TO AX
479 0177 B8 C2 MOV M18 ; CURSOR SET
480 0178 E8 017E R CALL M18 ; CURSOR_SET
481 017C M17: ;-----+
482 017E EB BF JMP VIDEO_RETURN ; SET_CPOS_RETURN
483 017E SET_CPOS ENDP

485 ;-----+
486 ; SET CURSOR POSITION, AX HAS ROW/COLUMN FOR CURSOR
487 M18 PROC NEAR
488 017E E8 0200 R CALL POSITION ; DETERMINE LOCATION IN REGEN BUFFER
489 0181 B8 C8 MOV CX,AX
490 0183 03 00 004E R ADD CX,[CRT_START]
491 0187 D1 F9 SAR CX,1 ; ADD IN THE START ADDRESS FOR THIS PAGE
492 0189 B4 0E MOV AH,14 ; DIVIDE BY 2 FOR CHAR ONLY COUNT
493 018A E8 0151 R CALL M16 ; REGISTER NUMBER FOR CURSOR
494 018E C3 RET ;-----+
495 018F M18 ENDP

496 ;-----+
497 ; READ_CURSOR
498 ; THIS ROUTINE READS THE CURRENT CURSOR VALUE FROM THE
499 ; 6845, FORMATS IT, AND SENDS IT BACK TO THE CALLER
500 ;-----+
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 0191 32 FF XOR BH,BH
510 0193 D1 E3 SAL BX,1 ; WORD OFFSET
511 0195 B8 97 0050 R MOV DX,[BX+OFFSET *CURSOR_POSN]
512 0199 B8 0E 0060 R MOV CX,[CURSOR_MODE]
513 019D 50 POP BP
514 019E 51 POP DI
515 01A0 58 POP SI
516 01A1 58 POP BX ; 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 ; ACT_DISP_PAGE
524 ; THIS ROUTINE SETS THE ACTIVE DISPLAY PAGE, ALLOWING
525 ; THE FULL USE OF THE MEMORY SET ASIDE FOR THE VIDEO ATTACHMENT
526 ;-----+
527 ; INPUT
528 ; AL HAS THE NEW ACTIVE DISPLAY PAGE
529 ;-----+
530 ;-----+
531 01A6 ACT_DISP_PAGE PROC NEAR
532 01A6 A2 0062 R MOV AX,[ACTIVE_PAGE_AL] ; SAVE ACTIVE PAGE VALUE
533 01A9 98 CBW ; CONVERT (AL) TO WORD
534 01AA 50 PUSH AX ; SAVE PAGE VALUE
535 01AB F7 26 004C R MUL WORD PTR [CRT_LEN] ; DISPLAY PAGE TIMES REGEN LENGTH
536 01AF A3 004E R MOV [CRT_START_AX],DX ; SAVE START ADDRESS FOR LATER
537 01B2 B8 C8 MOV CX,AH ; SAVE ADDRESS FOR CX
538 01B5 D1 F9 SAR CX,1 ; START ADDRESS FOR 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 50 POP BX ; RECOVER PAGE VALUE
542 01BC D1 E3 SAL BX,1 ; +2 FOR WORD OFFSET
543 01BE B8 87 0050 R MOV AX,[BX + OFFSET *CURSOR_POSN] ; GET CURSOR FOR THIS PAGE
544 01C2 E8 017E R CALL M18 ;-----+
545 01C5 E9 013D R JMP VIDEO_RETURN ; SET THE CURSOR POSITION
546 01CB ACT_DISP_PAGE ENDP

547 ;-----+
548 ; SET COLOR
549 ; THIS ROUTINE WILL ESTABLISH THE BACKGROUND COLOR, THE OVERSCAN COLOR,
550 ; AND THE FOREGROUND COLOR SET FOR MEDIUM RESOLUTION GRAPHICS
551 ;-----+
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 HIGH BITS OF BH
558 ; 0 = GREEN, RED, YELLOW FOR COLORS 1,2,3
559 ; 1 = BLUE, CYAN, MAGENTA FOR COLORS 1,2,3
560 ;-----+
561 ; (BL) HAS THE COLOR VALUE TO BE USED
562 ;-----+
563 ;-----+
564 01C8 SET_COLOR PROC NEAR
565 01C8 B8 16 0063 R MOV DX,*ADDR_6845 ; I/O PORT FOR PALETTE
566 01CC 83 C2 05 ADD DX,5 ; OVERSCAN PORT
567 01CF A0 0066 R MOV AL,[CRT_PALETTE] ; GET THE CURRENT PALETTE VALUE
568 01D2 0A FF OR BH,BH ; IS THIS COLOR 0?
569 01D4 75 0E JNZ M20 ;-----+
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 ;-----+

```

```

571      01D6 24 E0          AND    AL,0E0H      ; TURN OFF LOW 5 BITS OF CURRENT
572      01D8 80 E3 IF      AND    BL,01FH      ; TURN OFF HIGH 3 BITS OF INPUT VALUE
574      01D8 0A C3          OR     AL,BL       ; PUT VALUE INTO REGISTER
575      01DD               M19:              ; OUTPUT THE PALETTE
576      01D8 EE              OUT   DX,AL       ; OUTPUT COLOR SELECTION TO 3D9 PORT
577      01DE A2 0066 R      MOV    *CRT_PALETTE,AL ; SAVE THE COLOR VALUE
578      01E1 E9 013D R      JMP    VIDEO_RETURN

579
580      ;----- HANDLE COLOR 1 BY SELECTING THE PALETTE TO BE USED
581
582      01E4               M20:              ;----- SET COLOR 1
583      01E4 24 DF          AND    AL,0DFH      ; TURN OFF PALETTE SELECT BIT
584      01E6 D0 EB          SHR    BL,1        ; TEST THE LOW ORDER BIT OF BL
585      01E8 73 F3          JNC    M19          ; ALREADY DONE
586      01E8 0C 20          OR     AL,20H      ; TURN ON PALETTE SELECT BIT
587      01EC EB EF          JMP    M19          ; GO DO IT
588      01EE               SET_COLOR_ENDP

589
590      ;----- VIDEO STATE
591      ;----- RETURNS THE CURRENT VIDEO STATE IN AX
592      ;----- AH = NUMBER OF COLUMNS ON THE SCREEN
593      ;----- AL = CURRENT VIDEO MODE
594      ;----- BH = CURRENT ACTIVE PAGE
595
596      01EE               VIDEO_STATE PROC NEAR
597      01EE BA 26 004A R    MOV    AH,BYTE PTR *CRT_COLS ; GET NUMBER OF COLUMNS
598      01F2 A0 0049 R      MOV    AL,*CRT_MODE      ; CURRENT MODE
599      01F5 8A 3E 0062 R    MOV    BH,*ACTIVE_PAGE ; GET CURRENT ACTIVE PAGE
600      01F6 00 00           POP    BP        ; RECOVER REGISTERS
601      01FA 5F              POP    DI        ; RECOVER REGISTERS
602      01FB 5E              POP    SI        ; RECOVER REGISTERS
603      01FC 59              POP    CX        ; DISCARD SAVED BX
604      01FD E9 0141 R      JMP    M15          ; RETURN TO CALLER
605      0200               VIDEO_STATE ENDP

606
607      ;----- POSITION
608      ;----- THIS SERVICE ROUTINE CALCULATES THE REGEN BUFFER ADDRESS
609      ;----- OF A CHARACTER IN THE ALPHA MODE
610
611      ;----- INPUT   AX = ROW, COLUMN POSITION
612
613      ;----- OUTPUT  AX = OFFSET OF CHAR POSITION IN REGEN BUFFER
614
615      0200               POSITION PROC NEAR
616      0200 53             PUSH   BX        ; SAVE REGISTER
617      0200 93             XCHG   BX,AX      ; SAVE ROW/COLUMN POSITION IN (BX)
618      0202 00 004A R      MOV    AL,BYTE PTR *CRT_COLS ; GET COLUMNS PER ROW COUNT
619      0205 F6 E7          MUL    BL,BL      ; DETERMINE BYTES TO ROW
620      0207 32 FF          XOR    BH,BH      ; ROW/COLUMN OF LOWER RIGHT CORNER
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        ; RECOVER REGISTERS
624      020E C3              RET
625      020F               POSITION ENDP

626
627      ;----- SCROLL_UP
628      ;----- THIS ROUTINE MOVES A BLOCK OF CHARACTERS UP
629      ;----- ON THE SCREEN
630
631      ;----- INPUT   (AH) = CURRENT CRT MODE
632      ;----- (AL) = NUMBER OF ROWS TO SCROLL
633      ;----- (CX) = ROW/COLUMN OF UPPER LEFT CORNER
634      ;----- (DX) = ROW/COLUMN OF LOWER RIGHT CORNER
635      ;----- (BH) = ATTRIBUTE TO BE USED ON BLANKED LINE
636      ;----- (DS) = DATA SEGMENT
637      ;----- (ES) = REGEN BUFFER SEGMENT
638
639      ;----- OUTPUT  NONE -- THE REGEN BUFFER IS MODIFIED
640
641      020F               SCROLL_UP ASSUME DS:DATA,ES:DATA
642      020F               SCROLL_UP PROC NEAR
643
644      020F E8 02EA R      CALL   TEST_LINE_COUNT
645      0212 80 FC 04          CMP    AH,4        ; TEST FOR GRAPHICS MODE
646      0215 72 08          JC    N1          ; HANDLE SEPARATELY
647      0217 80 FC 07          CMP    AH,7        ; TEST FOR BW CARD
648      0218 74 03          JE    N1          ; RECOVER ATTRIBUTES
649      021C E9 04AC R      JMP    GRAPHICS_UP
650
651      021F 53             PUSH   BX        ; SAVE FILL ATTRIBUTE IN BH
652      0220 8B C1          MOV    AX,CX      ; UPPER LEFT POSITION
653      0222 E8 025C R      CALL   SCROLL_POSITION
654      0224 00 00           ADD    DI,BP      ; DO SETUP FOR SCROLL
655      0231 03 00           ADD    SI,AX      ; BLANK FIELD
656      0229 8A E6          ADD    DH,DH      ; # ROWS IN BLOCK
657      0228 2A E3          SUB    AH,BL      ; # ROWS TO BE MOVED
658
659      022D E8 029D R      N2:              CALL   N10          ; ROW_LOOP
660      022E 00 00           ADD    SI,BP      ; MOVE ONE ROW
661      0232 03 FD          ADD    DI,BP      ; POINT TO NEXT LINE IN BLOCK
662      0234 FE CC          DEC    AH        ; COUNT OF LINES TO MOVE
663      0236 75 F5          JNZ    N2          ; ROW_LOOP
664      0238               N3:              CLEAR_ENTRY
665      0238 58              POP    AX        ; RECOVER ATTRIBUTE IN AH
666      0238 00 00           MOV    AL,' '
667      023B               N4:              CALL   N11          ; CLEAR_BLANKS
668      023B E8 02A6 R      ADD    DI,BP      ; CLEAR_LOOP
669      023E 03 FD          DEC    BL        ; POINT TO NEXT LINE
670      0240 FE CB          JNZ    N4          ; COUNTER OF LINES TO SCROLL
671      0242 75 F7          CALL   DDS          ; CLEAR_LOOP
672      0243 00 00           OUT   DX,AL      ; SCROLL_END
673      0244 E8 0000 E      N6:              CALL   DDS          ; VIDEO_RET_HERE
674      0247 80 E0 0049 R 07  CMP    *CRT_MODE,7
675      024C 74 07          JE    N6          ; IS THIS THE BLACK AND WHITE CARD
676      024E A0 0065 R      MOV    AL,*CRT_MODE_SET ; IF SO, SKIP THE MODE RESET
677      0251 8A 03D8          MOV    DX,03D8H ; GET THE VALUE OF THE MODE SET
678      0252 00 00           OUT   DX,AL      ; ALWAYS SET COLOR CARD PORT
679      0255
680      0255 E9 013D R      N7:              JMP    VIDEO_RETURN
681      0258               SCROLL_UP ENDP
682      0258 8A DE          MOV    BL,DH      ; BLANK_FIELD
683      025A EB DC          JMP    N2          ; GET ROW COUNT
684      025C               SCROLL_UP ENDP

```

```

685
686
687      ;----- HANDLE COMMON SCROLL SET UP HERE
688 025C      SCROLL_POSITION PROC NEAR
689 025C E8 0200 R    CALL POSITION          ; CONVERT TO REGEN POINTER
690 025F 03 004E R    ADD AX, *CRT_START    ; OFFSET OF ACTIVE PAGE
691 0243 BB F8        MOV DI, AX           ; TO ADDRESS FOR SCROLL
692 0245 BB F0        MOV SI, AX           ; FROM ADDRESS FOR SCROLL
693 0267 2B D1        SUB DX, CX           ; DX = #ROWS, #COLS IN BLOCK
694 0269 FE C6        INC DH
695 026B FE C2        INC DL
696 026D 3C ED        XOR CH, CH
697 026F BB 0B 004A R    MOV AL, *CRT_COLS    ; SET HIGH BYTE OF COUNT TO ZERO
698 0273 03 ED        ADD BP, BP
699 0275 A0 004A R    MOV AL, BYTE PTR *CRT_COLS    ; GET NUMBER OF COLUMNS IN DISPLAY
700 0278 F8 E3        MUL BL
701 027A 03 C0        ADD AX, AX
702 027D 50 00        PUSH AX
703 027D 00 0049 R    MOV AL, *CRT_MODE    ; INCREMENT FOR 0 ORIGIN
704 0280 06          PUSH AX
705 0281 1F          POP DS
706 0282 3C 02        CMP AL, 2
707 0284 72 13        JB N9
708 0286 3C 03        CMP AL, 3
709 0288 77 0F        JA N9
710
711 028A 52          PUSH DX
712 028B BB 03DA      MOV DX, 3DAH
713 028E
714 028F EC          N8: IN AL, DX
715 029F A5 08        TEST AL, RVRT
716 0291 74 F8        JZ N9
717 0293 B0 25        MOV AL, 25H
718 0295 BB D8        MOV DL, 0D8H
719 0297 EE          OUT DX, AL
720 0298 5A          POP DX
721 0299 56          ;----- 80X25 COLOR CARD SCROLL
722 029A 0A DB        PUSH AX
723 029C C3          OR BL, BL
724 029D RET
725 029D SCROLL_POSITION ENDP
726
727
728 029D PROC NEAR
729 029D 8A CA        MOV CL, DL
730 029F 56          PUSH SI
731 02A0 57          PUSH DI
732 02A1 F3 / A5      REP MOVSW
733 02A3 5C          POP DI
734 0244 5E          POP SI
735 0245 C3          RET
736 0246 ENDP
737
738
739 0246 PROC NEAR
740 0246 8A CA        MOV CL, DL
741 0248 57          PUSH DI
742 0249 F3 / AB      REP STOSW
743 024B 5F          POP DI
744 02AC C3          RET
745 02AD ENDP
746
747
748
749      ;----- SCROLL DOWN
750      ; THIS ROUTINE MOVES THE CHARACTERS WITHIN A DEFINED
751      ; BLOCK DOWN ON THE SCREEN, FILLING THE TOP LINES
752      ; WITH A DEFINED CHARACTER
753      ; INPUT
754      ; (AH) = CURRENT CRT MODE
755      ; (AL) = NUMBER OF LINES TO SCROLL
756      ; (CX) = UPPER LEFT CORNER OF REGION
757      ; (DX) = LOWER RIGHT CORNER OF REGION
758      ; (BH) = FILL CHARACTER
759      ; (DS) = DATA SEGMENT
760      ; (ES) = REGEN SEGMENT
761      ; OUTPUT
762      ; NONE -- SCREEN IS SCROLLED
763
764 02AD SCROLL_DOWN PROC NEAR
765 02AD FD          D
766 02AE E8 02EA R    CALL TEST_LINE_COUNT    ; DIRECTION FOR SCROLL DOWN
767 0281 80 FC 04    CMP AH, 4
768 0284 72 08        JC N12
769 0286 80 FC 07    CMP AH, 7
770 0289 74 03        JE N12
771 028A 0E 0503 R    JMP GRAPHICS_DOWN
772 028E
773 028E 53          N12: PUSH BX
774 028F BB C2        MOV AX, DX
775 02C1 E8 025C R    CALL SCROLL_POSITION    ; TEST FOR GRAPHICS
776 02C4 74 20        JZ N13
777 02C6 2B F0        SUB SI, AX
778 02C8 8A E6        MOV AH, DH
779 02CA 2A E3        SUB AH, BL
780 02CC
781 02CC E8 029D R    N13: PUSH N10
782 02CF 2B F5        SUB SI, BP
783 02D0 2B FD        SUB DI, BP
784 02D3 FE CB        DEC BH
785 02D5 75 F5        DEC AH
786 02D7 58          JNZ N14
787 02D8 BB 20        N14: POP AX
788 02D8 BB 20        MOV AL, * *
789 02D0 2B FD 02A6 R    CALL N11
790 02D0 2B FD        SUB DI, BP
791 02E1 75 F7        DEC BH
792 02E3 E9 0244 R    JNZ N15
793 02E3 E9 0244 R    JMP NS
794 02E6 8A DE        MOV BL, DH
795 02E8 EB ED        JMP N14
796 02EA 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 02EA      TEST_LINE_COUNT PROC NEAR
802
803 02EA 8A DB    MOV     BL,AL           ; SAVE LINE COUNT IN BL
804 02EC 0A C0    OR     AL,AL           ; TEST IF AL IS ALREADY ZERO
805 02F0 0E       JZ     BL,SET          ; IF IT IS THEN RETURN...
806 02F0 50       PUSH    AX             ; SAVE AX
807 02F1 8A C6    MOV     AL,DH           ; SUBTRACT LOWER ROW FROM UPPER ROW
808 02F3 2A C5    SUB     AL,CH           ; ADJUST DIFFERENCE BY 1
809 02F5 FE C0    INC     AL             ; LINE COUNT = AMOUNT OF ROWS IN WINDOW?
810 02F6 3A C3    CMP     AL,BL           ; RESTORE AX
811 02F9 91       POP    AX             ; IF NOT THEN WE'RE ALL SET
812 02FA 52       JNE     BL,SET          ; OTHERWISE SET BL TO ZERO
813 02FC 2A DB    SUB     BL,BL           ; RETURN
814 02FE          BL_SET:
815 02FE C3       RET             ; RETURN
816 02FF          TEST_LINE_COUNT ENDP

817
818
819          ----- READ_AC_CURRENT
820          THIS ROUTINE READS THE ATTRIBUTE AND CHARACTER AT THE CURRENT
821          CURSOR POSITION AND RETURNS THEM TO THE CALLER
822
823          ----- INPUT
824          (AH) = CURRENT CRT MODE
825          (BH) = DISPLAY PAGE ( ALPHA MODES ONLY )
826          (DS) = DATA SEGMENT
827          (ES) = REGEN SEGMENT
828
829          ----- OUTPUT
830          (AL) = CHARACTER READ
831          (AH) = ATTRIBUTE READ
832
833 02FF      READ_AC_CURRENT PROC NEAR
834 02FF 80 FC 04    CMP     AH,4           ; IS THIS GRAPHICS
835 0302 72 08    JC    P10
836
837 0304 80 FC 07    CMP     AH,7           ; IS THIS BW CARD
838 0307 T4 03    JE    P10
839
840 0309 E9 063E R   JMP     GRAPHICS_READ
841 030C          P10:   READ_AC_CONTINUE
842 030C E8 0328 R   CALL    FIND_POSITION
843 030F 8B F7       MOV     SI,DT           ; GET REGEN LOCATION AND PORT ADDRESS
844 0311 06         PUSH   ES             ; ESTABLISH ADDRESSING IN SI
845 0312 1F         POP    DS             ; GET REGEN SEGMENT FOR QUICK ACCESS
846
847          ----- WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
848
849 0313 0A DB       OR     BL,BL           ; CHECK MODE FLAG FOR COLOR CARD IN 80
850 0315 75 0D       JNZ    P13
851 0317          P11:   NOP
852 0317 FB       STI    CL              ; ELSE SKIP RETRACE WAIT - DO FAST READ
853 0318 00         NOP
854 0319 FA       CLW
855 031A EC       IN    AL,DX           ; WAIT FOR HORIZ RETRACE LOW OR VERTICAL
856 031B A8 01       TEST   AL,RHRZ          ; ENABLE INTERRUPT FIRST
857 031D 75 F8       JNZ    P11
858 031F EC       IN    AL,DX           ; GET STATUS FROM THE ADAPTER
859 0320 A8 09       TEST   AL,VRVT+RHRZ
860 0322 74 FB       JZ    P12
861
862 0324          P12:   IN    AL,DX           ; IS HORIZONTAL RETRACE LOW
863 0324 AD       LODSW
864 0325 E9 013D R   JMP    VIDEO_RETURN
865
866 0326          READ_AC_CURRENT ENDP

867
868
869          ----- FIND_POSITION
870 0328          FIND_POSITION PROC NEAR
871 0328 86 E3       XCHG   AH,BL           ; SETUP FOR BUFFER READ OR WRITE
872 032A 89 E8       MOV    BP,AX           ; SWAP MODE TYPE WITH ATTRIBUTE
873 032C 80 EB 02    SUB    BL,2            ; SAVE CHARACTER/ATTR IN (BP) REGISTER
874 032F D0 EB       SHR    BL,1            ; CONVERT DISPLAY MODE TYPE TO A
875 0331 8A C7       MOV    AL,BH           ; ZERO VALUE FOR COLOR IN 80 COLUMN
876 0332 98         CBW
877 0334 F8       MOV    D1,AX           ; MOVE DISPLAY PAGE TO LOW BYTE
878 0336 D1 E7       SAL    D1,I             ; CLEAR HIGH BYT OF PAGE OFFSET
879 0336 BB 95 0050 R  MOV    DX,[D1+OFFSET *CURSOR_POS] ; TIMES 2 FOR WORD OFFSET
880 033C 74 09       JZ    P21
881
882 033E 33 FF       XOR    D1,D1           ; GET ROW/COLUMN OF THAT PAGE
883 0340          P20:   XOR    D1,D1           ; SKIP BUFFER ADJUSTMENT IF PAGE ZERO
884 0340 03 3E 004C R  ADD    D1,*CRT_LEN
885 0344 48         DEC    AX             ; ELSE SET BUFFER START ADDRESS TO ZERO
886 0345 75 F9       JNZ    P20
887
888 0347          P21:   ADD    AL,BYTE PTR *CRT_COLS
889 0347 A0 004A R   MULT   DH             ; ADD LENGTH OF BUFFER FOR ONE PAGE
890 034A F6 E6       XOR    DH,DH           ; DECREMENT PAGE COUNT
891 034C 32 F6       ADD    AX,DX           ; DETERMINE BYTES TO ROW
892 034E 03 C2       ADD    AX,DX           ; ADD IN COLUMN VALUE
893 0350 D1 E0       SAL    AX,1             ; * 2 FOR ATTRIBUTE BYTES
894 0352 03 F8       ADD    D1,AX           ; ADD IN ROW COUNT OF REGEN PAGE
895 0356 03 16 0063 R  MOV    DX,*ADDR_6845
896 0358 83 C2 06     ADD    DX,b             ; GET BASE ADDRESS OF ACTIVE DISPLAY
897 035B C3         RET
898
899 035C          FIND_POSITION ENDP

```

```

900 PAGE
901
902 ;----- AC CURRENT
903 ;----- THIS ROUTINE WRITES THE ATTRIBUTE AND CHARACTER
904 ;----- AT THE CURRENT CURSOR POSITION
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 WRITE_AC_CURRENT PROC NEAR
918 038E 00 FC 04
919 038F 00 00
920 0361 80 FC 07
921 0364 74 03
922 0366 E9 058A R
923 0369
924 0367 E8 0328 R
925 0370 00
926 036C 0A DB
927 036E 74 06
928
929 0370 95 XCHG AX,BP
930 0371 F3 / AB REP STOSW
931 0373 EB 16 JMP SHORT P35
932
933 ;----- WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
934
935 0375 P31: XCHG BP,AX
936 0375 95 P32: IN AL,DX
937 0376 00 TEST AL,RVRT
938 0377 FB JNZ P34
939 0377 90 CLC
940 0378 FA IN AL,DX
941 0379 EC TEST AL,RVRT
942 037A A8 08 JNZ P34
943 037C T5 09 TEST AL,RHRZ
944 037D A8 01 JNZ P32
945 0380 T5 F4
946 0382 P33: IN AL,DX
947 0382 EC TEST AL,RVRT+RHZ
948 0383 A8 09 JZ P33
949 0385 T4 FB LOOP P31
950 0386 00
951 0387 95 XCHG AX,BP
952 0388 AB 01 STOSW
953 0389 E2 EA LOOP P31
954 038B
955 0389 E9 013D R P35: JMP VIDEO_RETURN
956
957 038E WRITE_AC_CURRENT ENDP
958
959 ;----- C CURRENT
960 ;----- THIS ROUTINE WRITES THE CHARACTER AT
961 ;----- THE CURRENT CURSOR POSITION, ATTRIBUTE UNCHANGED
962 ;----- INPUT
963 ;----- (AH) = CURRENT CRT MODE
964 ;----- (BH) = DISPLAY PAGE
965 ;----- (CX) = COUNT OF CHARACTERS TO WRITE
966 ;----- (AL) = CHAR TO WRITE
967 ;----- (DS) = DATA SEGMENT
968 ;----- (ES) = REGEN SEGMENT
969 ;----- OUTPUT
970 ;----- DISPLAY REGEN BUFFER UPDATED
971
972
973
974 038E WRITE_C_CURRENT PROC NEAR
975 038E 00 FC 04
976 038F 00 00
977 0393 80 FC 07
978 0396 74 03
979 0398 E9 058A R
980 039B
981 039B E8 0328 R
982
983
984 ;----- WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
985
986 039E P41: STI
987 039E FB OR BL,BL
988 039F 00 DB JNZ P43
989 03A1 05 0F CMP AH,7
990 03A3 FA JE P40
991 03A4 EC CLI
992 03A5 A8 08 IN AL,DX
993 03A7 T5 09 TEST AL,RVRT
994 03A8 A8 01 JNZ P43
995 03AB T5 F1 TEST AL,RHRZ
996 03AD JNZ P41
997 03AD EC IN AL,DX
998 03AE A8 09 TEST AL,RVRT+RHZ
999 03B0 T4 FB JZ P42
1000 03B2 MOV AX,BP
1001 03B3 00 BB C5 STOSB
1002 03B4 AA INC DI
1003 03B5 47 LOOP P41
1004 03B6 E2 E6
1005 03B8 E9 013D R P43: JMP VIDEO_RETURN
1006
1007
1008 03B9 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          ;   [OE] = SOURCE STRING SEGMENT (FOR USE IN (ES) IN STACK +14)
1021          ; OUTPUT
1022          ;   NONE
1023

1024 03BB      WRITE_STRING    PROC    NEAR
1025          ;-----+
1026          ;   BP,BH
1027          MOV     BP,BH
1028          MOV     ES,[BP]+14+2
1029          POP    BP
1030          CBW
1031          MOV     DI,AX
1032          CMP    AL,04
1033          JNB    P59
1034
1035          JCXZ  P59
1036          MOV     SI,BX
1037          MOV     BH,BH
1038          XOR    BH,BH
1039          XCHG  SI,BX
1040          SAL    SI,1
1041          PUSH   [SI+OFFSET_CURSOR_POSN]
1042          MOV     AX,0200H
1043          INT    10H
1044
1045 03E 26: 8A 46 00    P50:
1046          MOV     AL,ES:[BP]
1047          INC    BP
1048
1049          ;----- TEST FOR SPECIAL CHARACTER'S
1050 03E3 3C 08
1051 03E5 74 0C
1052 03E7 3C 0D
1053 03E9 74 08
1054 03EB 3C 0A
1055 03F1 75 04
1056 03EF 3C 07
1057 03F1 75 0A
1058 03F3
1059 03F3 84 0E
1060 03FF CD 00
1061 03E5 8B 00 0050 R
1062 03FB EB 2D
1063
1064 03FD
1065 03FD 51
1066 03FE 53
1067 03FE 89 0001
1068 0402 83 FF 02
1069 0405 72 05
1070 0407 26: 8A 5E 00
1071 040B 45
1072 040C
1073 040E 84 09
1074 040E CD 10
1075 0410 5B
1076 0411 59
1077 0412 FE C2
1078 0413 3E 00 004A R
1079 0418 72 10
1080 041A FE C6
1081 041C DA D2
1082 041E 80 FE 19
1083 0421 72 07
1084
1085 0423 88 0E0A
1086 0426 CD 10
1087 0428 FE CE
1088 042A
1089 042A 88 0200
1090 042D CD 10
1091 042F E2 AD
1092
1093 0431 5A
1094 0432 97
1095 0433 A8 01
1096 0435 75 05
1097 0436 8B 0200
1098 043A CD 10
1099 043C
1100 043C E9 013D R
1101
1102 043F

          CMP    AL,08H
          JE    P51
          CMP    AL,CR
          JE    P51
          CMP    AL,LF
          JE    P51
          CMP    AL,07H
          JE    P51
          JNE   P52
          CMP    AL,0EH
          INT   10H
          MOV   DX,[SI+OFFSET_CURSOR_POSN]
          JMP   SHORT P54
          PUSH   CX
          PUSH   BX
          MOV   DL,1
          CMP   DL,2
          JB    P53
          MOV   BL,ES:[BP]
          INC   BP
          MOV   AH,09H
          INT   10H
          POP   BX
          POP   CX
          INC   DL
          CMP   DL,BYTE PTR CRT_COLS
          JB    P54
          INC   DH
          SUB   DL,DL
          CMP   DH,25
          JB    P54
          MOV   AX,0E0AH
          INT   10H
          DEC   DH
          MOV   AX,0200H
          INT   10H
          LOOP  P50
          POP   DX
          XCHG  AX,DI
          TEST  AL,01H
          JNZ   P59
          MOV   AX,0200H
          INT   10H
          JMP   VIDEO_RETURN
          ENDP

          ;-----+
          ; SAVE BUFFER OFFSET (BP) IN STACK
          ; GET POINTER TO STACKED REGISTERS
          ; RECOVER ENTRY (ES) SEGMENT REGISTER
          ; RESTORE BUFFER OFFSET
          ; CLEAR (AH) REGISTER
          ; SAVE (AL) COMMAND IN (DI) REGISTER
          ; TEST FOR INVALID WRITE STRING OPTION
          ; IF OPTION INVALID THEN RETURN
          ;-----+
          ; IF ZERO LENGTH STRING THEN RETURN
          ; SAVE CURRENT CURSOR PAGE
          ; MOVE PTR TO LOW BYTE
          ; CLEAR HIGH BYTE
          ; MOVE OFFSET AND RESTORE PAGE REGISTER
          ; CONVERT TO PAGE OFFSET (SI= PAGE)
          ; SAVE CURRENT CURSOR POSITION IN STACK
          ; SET NEW CURSOR POSITION
          ;-----+
          ; GET CHARACTER FROM INPUT STRING
          ; BUMP POINTER TO CHARACTER
          ;-----+
          ; IS IT A BACKSPACE
          ; BACK SPACE
          ; IS IT CARRIAGE RETURN
          ; CAR RET
          ; IS IT A LINE FEED
          ; LINE FEED
          ; IS IT A BELL
          ; BELL
          ; IF NOT THEN DO WRITE CHARACTER
          ;-----+
          ; SET CHARACTER WRITE AMOUNT TO ONE
          ; IS THE ATTRIBUTE IN THE STRING
          ; IF NOT THEN SKIP
          ; ELSE GET NEW ATTRIBUTE
          ; BUMP STRING POINTER
          ;-----+
          ; GOT CHARACTER
          ; WRITE CHARACTER TO THE CRT
          ; RESTORE REGISTERS
          ;-----+
          ; INCREMENT COLUMN COUNTER
          ; IF COLS ARE WITHIN RANGE FOR THIS MODE
          ; THEN SET THE COLUMN SET
          ; BUMP ROW COUNTER BY ONE
          ; SET COLUMN COUNTER TO ZERO
          ; IF ROWS ARE LESS THAN 25 THEN
          ; GO TO ROWS_COLUMNS_SET
          ;-----+
          ; ELSE SCROLL SCREEN ONE LINE
          ; RESET ROW COUNTER TO 24
          ;-----+
          ; ROW_COLUMNS_SET
          ; SET NEW CURSOR POSITION COMMAND
          ; ESTABLISH NEW CURSOR POSITION
          ; DO IT ONCE MORE UNTIL (CX) = ZERO
          ;-----+
          ; RESTORE OLD CURSOR COORDINATES
          ; RECOVER WRITE STRING COMMAND
          ; IF CURSOR WAS NOT TO BE MOVED THEN
          ; THEN EXIT WITHOUT RESETING OLD VALUE
          ; ELSE RESTORE OLD CURSOR POSITION
          ;-----+
          ; DONE - EXIT WRITE STRING
          ; RETURN TO CALLER

```

1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121 043F
1122 043F E8 0473 R
1123 0442 26 8A 04
1124 0445 22 C4
1125 0446 22 00
1126 0449 8A CE
1127 044B D2 C0
1128 044D E9 013D R
1129 0450
1130
1131 0450
1132 0450 50
1133 0451 50
1134 0452 E8 0473 R
1135 0455 D2 E8
1136 0456 22 C4
1137 0459 8A 0C
1138 045C 5B
1139 045D F6 C3 80
1140 0460 75 00
1141 0462 F6 D4
1142 0464 22 CC
1143 0465 8A C1
1144 0468
1145 0468 26 88 04
1146 0468 5B
1147 046C E9 013D R
1148 046F
1149 0470 32 C1
1150 0471 EB F5
1151 0473
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165 0473
1166
1167
1168
1169
1170 0473 96
1171 0474 B0 28
1172 0476 F6 E2
1173 0477 00 08
1174 047A 74 03
1175 047C 05 IFD8
1176 047F
1177 047F 96
1178 0480 BB D1
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188 0482 BB 02C0
1189 0485 B9 0302
1190 0488 B0 3E 0049 R 06
1191 048F 72 08
1192 048F BB 0180
1193 0492 B9 0703
1194
1195
1196 0495
1197 0495 22 EA
1198
1199
1200
1201 0497 D3 EA
1202 0499 03 F7
1203 049B 8A F7
1204
1205
1206
1207 049D 2A C9
1208 049E 00 00
1209 049F D0 C8
1210 04A1 02 CD
1211 04A3 FE CF
1212 04A5 75 F8
1213 04A7 8A E3
1214 04A9 D2 EC
1215 04AC C3
1216 04AC
PAGE

; READ DOT -- WRITE DOT
; THESE ROUTINES WILL WRITE A DOT, OR READ THE
; DOT AT THE INDICATED LOCATION
; ENTRY --
; DX = ROW (0-199) (THE ACTUAL VALUE DEPENDS ON THE MODE)
; CX = COLUMN (0-639) (THE VALUES ARE NOT RANGE CHECKED)
; AL = DOT VALUE TO WRITE (1,2 OR 4 BITS DEPENDING ON MODE,
; REQUIRED FOR WRITE DOT ONLY, RIGHT JUSTIFIED)
; DS OF AL AND ES INDICATES XOR THE VALUE INTO THE LOCATION
; DS = DATA SEGMENT
; ES = REGEN SEGMENT
; EXIT
; AL = DOT VALUE READ, RIGHT JUSTIFIED, READ ONLY

ASSUME DS:DATA,ES:DATA
READ_DOT PROC NEAR
CALL R3 ; DETERMINE BYTE POSITION OF DOT
MOV AL,ES:[SI] ; GET THE BYTE
AND AL,AH ; MASK OFF THE OTHER BITS IN THE BYTE
SHR AL,CL ; LEFT JUSTIFY THE VALUE
MOV CL,DH ; GET NUMBER OF BITS IN RESULT
ROL AL,CL ; RIGHT JUSTIFY THE RESULT
JMP VIDEO_RETURN ; RETURN FROM VIDEO I/O
READ_DOT ENDP ;-----
WRITE_DOT PROC NEAR
PUSH AX ; SAVE DOT VALUE
PUSH AX ; TWICE
CALL R3 ; DETERMINE BYTE POSITION OF THE DOT
SHR AL,CL ; SHIFT TO SET UP THE BITS FOR OUTPUT
AND AL,AH ; STRIP OFF THE OTHER BITS
MOV CL,ES:[SI] ; GET THE CURRENT BYTE
POP BX ; RECOVER XOR FLAG
TEST BL,80H ; IS IT ON
JNZ R2 ; YES, XOR THE DOT
NOT AH ; SET MASK TO REMOVE THE INDICATED BITS
AND CL,AH ; OR IN THE NEW VALUE OF THOSE BITS
OR AL,CL ; FINISH DOT
MOV ES:[SI],AL ; RESTORE THE BYTE IN MEMORY
POP AX ;-----
JMP VIDEO_RETURN ; RETURN FROM VIDEO I/O
XOR DOT ;-----
R2: XOR AL,CL ; EXCLUSIVE OR THE DOTS
JMP R1 ; FINISH UP THE WRITING
WRITE_DOT ENDP ;-----
; THIS SUBROUTINE DETERMINES THE REGEN BYTE LOCATION OF THE
; INDICATED ROW/COLUMN VALUE IN GRAPHICS MODE.
; ENTRY --
; DX = ROW VALUE (0-199)
; CX = COLUMN VALUE (0-639)
; EXIT --
; SI = OFFSET INTO REGEN BUFFER FOR BYTE OF INTEREST
; AH = MASK TO STRIP OFF THE BITS OF INTEREST
; CX = # OF BITS TO SHIFT TO RIGHT JUSTIFY THE MASK IN AH
; DH = # BITS IN RESULT
; BX = MODIFIED

R3 PROC NEAR

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

XCHG SI,AX ; WILL SAVE AL AND AH DURING OPERATION
MOV AL,40
MUL DL ; AX= ADDRESS OF START OF INDICATED ROW
TEST AL,008H ; TEST FOR EVEN/ODD ROW CALCULATED
JZ R4 ; JUMP IF EVEN ROW
ADD AX,200H-40 ; OFFSET TO LOCATION OF ODD ROWS ADJUST
; EVEN ROW
XCHG SI,AX ; MOVE POINTED TO (SI) AND RECOVER (AX)
MOV DX,CX ; COLUMN VALUE TO DX

; DETERMINE GRAPHICS MODE CURRENTLY IN EFFECT

; SET UP THE REGISTERS ACCORDING TO THE MODE
; CH = MASK FOR LOW OF COLUMN ADDRESS (7/3 FOR HIGH/MED RES)
; CL = # OF ADDRESS BITS IN COLUMN VALUE (3/2 FOR H/M)
; BL = MASK TO SELECT BITS FROM POINTEO BYTE (80H/COH FOR H/M)
; BH = NUMBER OF VALID BITS IN POINTEO BYTE (1/2 FOR H/M)

MOV BX,2C0H
MOV CX,302H ; SET PARM FOR MED RES
CMP .ORCT_MODE,6
JC R5 ; HANDLE IF MED RES
MOV BX,180H
MOV CX,703H ; SET PARM FOR HIGH RES

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

; DETERMINE BYTE OFFSET FOR THIS LOCATION IN COLUMN
SHR DX,CL ; SHIFT BY CORRECT AMOUNT
ADD SI,DX ; INCREMENT THE POINTER
MOV DH,BH ; GET THE # OF BITS IN RESULT TO DH

; MULTIPLY BH (VALID BITS IN BYTE) BY CH (BIT OFFSET)
SUB CL,CL ; ZERO INTO STORAGE LOCATION

R6:
ROR AL,1 ; LEFT JUSTIFY VALUE IN AL (FOR WRITE)
ADD CL,CH ; ADD IN THE BIT OFFSET VALUE
DEC BH ; LOOP CONTROL
JNZ R6 ; ON EXIT, CL HAS COUNT TO RESTORE BITS
MOV AH,BL ; GET MASK TO AH
SHR AH,CL ; MOVE THE MASK TO CORRECT LOCATION
RET ; RETURN WITH EVERYTHING SET UP
R3 ENDP ;-----

```

1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231 04AC
1232 04AC 8A D8
1233 04E4 BB C1
1234
1235
1236
1237
1238 04B0 E8 06EC R
1239 04B3 BB F8
1240
1241
1242
1243 04B5 2B D1
1244 04B7 81 C2 0101
1245 04B8 D0 E6
1246 04BD D0 E6
1247
1248
1249
1250 04BF 80 3E 0049 R 06
1251 04C4 73 04
1252
1253
1254 04C6 D0 E2
1255 04C8 D1 E7
1256
1257
1258 04CA
1259 04CA 06
1260 04CB 1F
1261 04CC 2A ED
1262 04CD D0 E3
1263 04D0 D0 E3
1264 04D1 2B
1265 04D4 B0 E0
1266 04D6 F6 E3
1267 04D8 BB F7
1268 04DA 03 F0
1269 04DC BA E6
1270 04DE 2A E3
1271
1272
1273 04E0
1274 04E0 E8 0560 R
1275 04E3 B1 EE 1FB0
1276 04E5 B1 EF 1FB0
1277 04E8 FE CC
1278 04ED 75 F1
1279
1280
1281 04EF
1282 04F0 8A C7
1283
1284 04F1 E8 0579 R
1285 04F4 B1 EF 1FB0
1286 04F8 FE CB
1287 04FA 75 F5
1288 04FC E9 013D R
1289
1290 04FF
1291 04FF BA DE
1292 0501 EB EC
1293 0503
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309 0503
1310 0503 FD
1311 0504 8A D8
1312 0506 BB C2
1313
1314
1315
1316
1317 0508 E8 06EC R
1318 050B BB F8
1319
1320
1321
1322 050D 2B D1
1323 050F B1 C2 0101
1324 0513 D0 E6
1325 0515 D0 E6
1326
1327
1328
1329 0517 80 3E 0049 R 06
1330 051C 73 05

-----  

; SCROLL UP  

; THIS ROUTINE SCROLLS UP THE INFORMATION ON THE CRT  

; ENTRY --  

; CH,CL = UPPER LEFT CORNER OF REGION TO SCROLL  

; DH,DL = LOWER RIGHT CORNER OF REGION TO SCROLL  

; BOTH OF THE ABOVE ARE IN CHARACTER POSITIONS  

; BL = # LINES TO SCROLL (BL=0 MEANS BLANK THE ENTIRE FIELD)  

; DS = DATA SEGMENT  

; ES = REGEN SEGMENT  

; EXIT --  

; NOTHING, THE SCREEN IS SCROLLED  

-----  

GRAPHICS_UP PROC NEAR  

MOV BL,AL ; SAVE LINE COUNT IN BL  

MOV AX,CX ; GET UPPER LEFT POSITION INTO AX REG  

-----  

; USE CHARACTER SUBROUTINE FOR POSITIONING  

; ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE  

-----  

CALL GRAPH_POSN ; SAVE RESULT AS DESTINATION ADDRESS  

MOV DI,AX  

-----  

; DETERMINE SIZE OF WINDOW  

-----  

SUB DX,CX ; ADJUST VALUES  

ADD DX,101H ; MULTIPLY ROWS BY 4 AT 8 VERT DOTS/CHAR  

SAL DH,1 ; AND EVEN/ODD ROWS  

-----  

; DETERMINE CRT MODE  

-----  

CMP OCRT_MODE,6 ; TEST FOR MEDIUM RES  

JNC R7 ; FIND_SOURCE  

-----  

; MEDIUM RES UP  

SAL DH,1 ; # COLUMNS * 2, SINCE 2 BYTES/CHAR  

SAL DI,1 ; OFFSET * 2 SINCE 2 BYTES/CHAR  

-----  

; DETERMINE THE SOURCE ADDRESS IN THE BUFFER  

R7:  

PUSH ES ; FIND_SOURCE  

POP DS ; GET SEGMENTS BOTH POINTING TO REGEN  

SUB CH,CH ; ZERO TO HIGH OF COUNT REGISTER  

SAL BL,1 ; MULTIPLY NUMBER OF LINES BY 4  

SAL BL,1 ; IF ZERO, THEN BLANK ENTIRE FIELD  

JZ R11 ; DO BYTES/ROWS  

MOV AL,B0 ; DETERMINE OFFSET TO SOURCE  

MUL BL ; SET UP SOURCE  

MOV SI,DI ; ADD IN OFFSET TO IT  

ADD SI,AX ; NUMBER OF ROWS IN FIELD  

MOV AH,DH ; DETERMINE NUMBER TO MOVE  

SUB AH,BL  

-----  

; LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD FIELDS  

R8:  

CALL R17 ; ROW_LOOP  

SUB SI,2000H-80 ; MOVE ONE ROW  

SUB DI,2000H-80 ; MOVE TO NEXT ROW  

DEC AH ; NUMBER OF ROWS TO MOVE  

JNZ R8 ; CONTINUE TILL ALL MOVED  

-----  

; FILL IN THE VACATED LINE(S)  

R9:  

MOV AL,BH ; CLEAR_ENTRY  

-----  

R10:  

MOV AL,BH ; ATTRIBUTE TO FILL WITH  

-----  

CALL R18 ; CLEAR THAT ROW  

SUB DI,2000H-80 ; POINT TO NEXT LINE  

DEC BL ; NUMBER OF LINES TO FILL  

JNZ R10 ; CLEAR_LOOP  

JMP VIDEO_RETURN ; EVERYTHING DONE  

-----  

R11:  

MOV BL,DH ; BLANK_FIELD  

SET BLANK COUNT TO EVERYTHING IN FIELD  

JMP R9 ; CLEAR THE FIELD  

GRAPHICS_UP ENDP  

-----  

; SCROLL DOWN  

; THIS ROUTINE SCROLLS DOWN THE INFORMATION ON THE CRT  

; ENTRY --  

; CH,CL = UPPER LEFT CORNER OF REGION TO SCROLL  

; DH,DL = LOWER RIGHT CORNER OF REGION TO SCROLL  

; BOTH OF THE ABOVE ARE IN CHARACTER POSITIONS  

; BL = # LINES TO SCROLL (BL=0 MEANS BLANK THE ENTIRE FIELD)  

; DS = DATA SEGMENT  

; ES = REGEN SEGMENT  

; EXIT --  

; NOTHING, THE SCREEN IS SCROLLED  

-----  

GRAPHICS_DOWN PROC NEAR  

STD ; SET DIRECTION  

MOV BL,AL ; SAVE LINE COUNT IN BL  

MOV AX,DX ; GET LOWER RIGHT POSITION INTO AX REG  

-----  

; USE CHARACTER SUBROUTINE FOR POSITIONING  

; ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE  

-----  

CALL GRAPH_POSN ; SAVE RESULT AS DESTINATION ADDRESS  

MOV DI,AX  

-----  

; DETERMINE SIZE OF WINDOW  

-----  

SUB DX,CX ; ADJUST VALUES  

ADD DX,101H ; MULTIPLY ROWS BY 4 AT 8 VERT DOTS/CHAR  

SAL DH,1 ; AND EVEN/ODD ROWS  

-----  

; DETERMINE CRT MODE  

-----  

CMP OCRT_MODE,6 ; TEST FOR MEDIUM RES  

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 DL,1 ; OFFSET * 2 SINCE 2 BYTES/CHAR
1336 INC DI ; POINT TO LAST BYTE
1337
1338 0523 06 R12: ;---- DETERMINE THE SOURCE ADDRESS IN THE BUFFER
1339 0524 1F PUSH ES ; FIND_SOURCE_DOWN
1340 0524 IF POP DS ; BOTH SEGMENTS TO REGEN
1341 0525 2A ED SUB CH,CH ; ZERO TO HIGH OF COUNT REGISTER
1342 0527 81 C7 00F0 ADD D1,240 ; POINT TO LAST ROW OF PIXELS
1343 052B D0 E3 SAL BL,1 ; MULTIPLY NUMBER OF LINES BY 4
1344 052D D0 E3 SAL BL,1
1345 052E 29 R16 ; IF ZERO, THEN BLANK ENTIRE FIELD
1346 0531 B0 50 MOV AL,80 ; 80 BYTES/ROW
1347 0533 F6 E3 MUL BL,DI ; DETERMINE OFFSET TO SOURCE
1348 0535 BB F7 MOV SI,DI ; SET UP SOURCE
1349 0537 2B F0 SUB SI,AX ; SUBTRACT THE OFFSET
1350 0539 8A E6 MOV AH,DH ; NUMBER OF ROWS IN FIELD
1351 053B 2A E3 SUB AH,BL ; DETERMINE NUMBER TO MOVE
1352
1353 ;---- LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD FIELDS
1354 053D R13: CALL R17 ; ROW_LOOP_DOWN
1355 053D E8 0560 R SUB SI,2000H+80 ; MOVE ONE ROW
1356 0540 81 EE 2050 SUB DI,2000H+80 ; MOVE TO NEXT ROW
1357 0543 81 00 2050 DEC AH ; NUMBER OF ROWS TO MOVE
1359 0548 FE CC JNZ R13 ; CONTINUE TILL ALL MOVED
1360
1361 ;---- FILL IN THE VACATED LINE(S)
1362 054C R14: MOV AL,BH ; CLEAR_ENTRY_DOWN
1363 054C 8A C7 R15: MOV DI,CL,DL ; AT ENTRY, TO FILL WITH
1364 054E REP PUSH SI ; CLEAR_LOOP_DOWN
1365 054E E8 0579 R CALL R18 ; CLEAR A ROW
1366 0551 81 EF 2050 SUB DI,2000H+80 ; POINT TO NEXT LINE
1367 0555 FE CB DEC BL ; NUMBER OF LINES TO FILL
1368 0557 75 F5 JNZ R15 ; CLEAR_LOOP_DOWN
1369
1370 0559 E9 013D R JMP VIDEO_RETURN ; EVERYTHING DONE
1371
1372 055C R16: MOV BL,DH ; BLANK_FIELD_DOWN
1373 055C 8A DE JMP R14 ; SET BLANK COUNT TO EVERYTHING IN FIELD
1374 055E EB EC
1375 0560 GRAPHICS_DOWN ENDP
1376
1377 ;---- ROUTINE TO MOVE ONE ROW OF INFORMATION
1378
1379 0560 R17 PROC NEAR ; NUMBER OF BYTES IN THE ROW
1380 0562 8A CA MOV CL,DL ; SAVE POINTERS
1381 0570 56 PUSH SI ; MOVE THE EVEN FIELD
1382 0573 57 PUSH DI
1383 0564 F3/ A4 REP MOVSB ; SAVE THE POINTERS
1384 0566 5F POP DI ; COUNT BACK
1385 0567 5E POP SI ; MOVE THE ODD FIELD
1386 0568 81 C6 2000 ADD SI,2000H ; POINT TO THE ODD FIELD
1387 0570 CT 2000 ADD DI,2000H
1388 0570 56 PUSH SI ; SAVE THE POINTERS
1389 0571 57 PUSH DI ; COUNT BACK
1390 0572 8A CA MOV CL,DL ; MOVE THE ODD FIELD
1391 0574 F3/ A4 REP MOVSB ; POINTERS BACK
1392 0576 5F POP DI ; RETURN TO CALLER
1393 0577 5E POP SI
1394 0578 C3 RET
1395 0579 R17 ENDP
1396
1397 ;---- CLEAR A SINGLE ROW
1398
1399 0579 R18 PROC NEAR ; NUMBER OF BYTES IN FIELD
1400 0579 8A CA MOV CL,DL ; SAVE POINTER
1401 057B 57 PUSH DI ; STORE THE NEW VALUE
1402 057C F3/ AA REP STOSB ; POINTER BACK
1403 057E 5F POP DI ; POINT TO ODD FIELD
1404 057F 81 CT 2000 ADD DI,2000H
1405 0580 56 PUSH SI ; FILL THE ODD FIELD
1406 0584 8A CA MOV CL,DL
1407 0586 F3/ AA REP STOSB ; RETURN TO CALLER
1408 0588 5F POP DI
1409 0589 C3 RET
1410 058A R18 ENDP
1411 -----GRAPHICS WRITE----- ; THIS ROUTINE WRITES THE ASCII CHARACTER TO THE CURRENT POSITION ON THE SCREEN.
1412 ENTRY -- ; AL = CHARACTER TO WRITE
1413 ; BL = COLOR WORD TO BE USED FOR FOREGROUND COLOR
1414 ; IF BIT 7 IS SET, THE CHAR IS XOR'D INTO THE REGEN BUFFER
1415 ; IO IS USED FOR THE BACKGROUND COLOR)
1416 CX = NUMBER OF CHARS TO WRITE
1417 DS = DATA SEGMENT
1418 ES = REGEN SEGMENT
1419 EXIT -- ; NOTHING IS RETURNED
1420
1421 ; GRAPHICS READ
1422 ; THIS ROUTINE READS THE ASCII CHARACTER AT THE CURRENT CURSOR POSITION ON THE SCREEN BY MATCHING THE DOTS ON THE SCREEN TO THE CHARACTER GENERATOR CODE POINTS
1423 ENTRY -- ; NONE (0 IS ASSUMED AS THE BACKGROUND COLOR)
1424 ; EXIT -- ; AL = CHARACTER READ AT THAT POSITION (0 RETURNED IF NONE FOUND)
1425
1426 ; FOR BOTH ROUTINES, THE IMAGES USED TO FORM CHARS ARE CONTAINED IN ROM
1427 ; FOR THE 1ST 128 CHARS. TO ACCESS CHARS IN THE SECOND HALF, THE USER
1428 ; MUST INITIALIZE THE VECTOR AT INTERRUPT 1FH (LOCATION 0007CH) TO
1429 ; POINT TO THE USER SUPPLIED TABLE OF GRAPHIC IMAGES (8x8 BOXES).
1430 ; FAILURE TO DO SO WILL CAUSE IN STRANGE RESULTS
1431 -----GRAPHICS WRITE----- ; ASSUME DS:DATA, ES:DATA
1432 GRAPHICS_WRITE PROC NEAR ; ZERO TO HIGH OF CODE POINT
1433 MOV AH,0 ; SAVE CODE POINT VALUE
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443 058A B4 00
1444 058C 50

```

```

1445
1446
1447 ;----- DETERMINE POSITION IN REGEN BUFFER TO PUT CODE POINTS
1448 058D E8 06E9 R
1449 0590 BB F8
1450
1451 ;----- DETERMINE REGION TO GET CODE POINTS FROM
1452
1453 0592 58
1454 0593 3C 80
1455 0595 T3 06
1456
1457 ;----- IMAGE IS IN FIRST HALF, CONTAINED IN ROM
1458
1459 0597 BE 0000 E
1460 059A 0E
1461 059B EB 18
1462
1463 ;----- IMAGE IS IN SECOND HALF, IN USER MEMORY
1464
1465 059D
1466 059D 2C 80
1467 059F 1E
1468 05A0 2B F6
1469 05A2 8E DE
1470
1471 05A4 C5 36 007C R
1472 05AB 8C DA
1473
1474 05AA 1F
1475 05AB 52
1476 05AC 0B D6
1477 05AE T5 05
1478
1479 05B0 58
1480 05B1 BE 0000 E
1481 05B4 0E
1482
1483 ;----- DETERMINE GRAPHICS MODE IN OPERATION
1484
1485 05B5
1486 05B5 D1 E0
1487 05B7 D1 E0
1488 05B8 D1 E0
1489 05B9 03 F0
1490 05B0 80 3E 0049 R 06
1491 05C2 1F
1492 05C3 72 2C
1493
1494 ;----- HIGH RESOLUTION MODE
1495 05C5
1496 05C5 57
1497 05C6 56
1498 05C7 B6 04
1499 05C9
1500 05D1 AC
1501 05CA F6 C3 80
1502 05CD T5 16
1503 05CF AA
1504 05D0 AC
1505 05D1
1506 05D4 26: 88 85 IFFF
1507 05D4 83 C7 4F
1508 05D9 FE CE
1509 05DB 75 EC
1510 05DD 5E
1511 05DE 5F
1512 05E0 47
1513 05E0 E2 E3
1514 05E2 E9 013D R
1515
1516 05E5
1517 05E5 26: 32 05
1518 05E5 AC
1519 05E9 AC
1520 05E6 26: 32 85 IFFF
1521 05EF EB E0
1522
1523 ;----- MEDIUM RESOLUTION WRITE
1524 05F1
1525 05F1 8A D3
1526 05F3 D1 E7
1527
1528 05F5 80 E3 03
1529 05F5 B0 55
1530 05F6 76 E3
1531 05FC 8A DB
1532 05FE 8A F8
1533 0600
1534 0600 57
1535 0601 56
1536 0602 B6 04
1537 0604
1538 0604 AC
1539 0605 E8 06C0 R
1540 0608 23 C3
1541 060A 86 E0
1542 060B 86 C2 80
1543 060F T4 03
1544 0611 26: 33 05
1545 0614
1546 0614 26: 89 05
1547 0618 86 06C0 R
1548 061B E8 06C0 R
1549 061B 23 C3
1550 061D 86 E0
1551 061F F6 C2 80
1552 0622 T4 05
1553 0623 26: 33 85 2000
1554 0629
1555 0629 26: 89 85 2000
1556 062E 83 C7 50
1557 0631 FE CE
1558 0633 T5 CF

1445
1446
1447 ;----- DETERMINE POSITION IN REGEN BUFFER TO PUT CODE POINTS
1448 058D E8 06E9 R
1449 0590 BB F8
1450
1451 ;----- DETERMINE REGION TO GET CODE POINTS FROM
1452
1453 0592 58
1454 0593 3C 80
1455 0595 T3 06
1456
1457 ;----- IMAGE IS IN FIRST HALF, CONTAINED IN ROM
1458
1459 0597 BE 0000 E
1460 059A 0E
1461 059B EB 18
1462
1463 ;----- IMAGE IS IN SECOND HALF, IN USER MEMORY
1464
1465 S1: SUB AL, 80H           ; FIND LOCATION IN REGEN BUFFER
1466 PUSH DS                ; REGEN POINTER IN DI
1467 MOV DI, AX
1468
1469 ;----- DETERMINE REGION TO GET CODE POINTS FROM
1470
1471 POP AX                 ; RECOVER CODE POINT
1472 CMP AL, 80H             ; IS IT IN SECOND HALF
1473 JAE SI                 ; YES
1474
1475 ;----- IMAGE IS IN FIRST HALF, CONTAINED IN ROM
1476
1477 MOV SI, OFFSET CRT_CHAR_GEN ; OFFSET OF IMAGES
1478 PUSH CS                ; SAVE SEGMENT ON STACK
1479 JMP SHORT S2             ; DETERMINE_MODE
1480
1481 ;----- IMAGE IS IN SECOND HALF, IN USER MEMORY
1482
1483 S1: SUB AL, 80H           ; EXTEND CHAR
1484 PUSH DS                ; ZERO ORIGIN FOR SECOND HALF
1485 SUB SI, SI              ; SAVE DATA POINTER
1486 MOV DS, SI
1487 ASSUME DS:DS0             ; ESTABLISH VECTOR ADDRESSING
1488 LODS SI, _EXT_PTR        ; GET THE OFFSET OF THE TABLE
1489 ADD DS, SI               ; GET THE SEGMENT OF THE TABLE
1490 MOV DS, DS
1491 ASSUME DS:DATA            ; RECOVER DATA SEGMENT
1492 POP DS                 ; SAVE TABLE SEGMENT ON STACK
1493 OR DX, SI               ; CHECK FOR VALID TABLE DEFINED
1494 JNZ S2                  ; CONTINUE IF DS1SI NOT 0000:0000
1495
1496 POP AX                 ; ELSE SET (AX)= 0000 FOR "NULL"
1497 MOV SI, OFFSET CRT_CHAR_GEN ; POINT TO DEFAULT TABLE OFFSET
1498 PUSH CS                ; IN THE CODE SEGMENT
1499
1500 ;----- DETERMINE GRAPHICS MODE
1501 S2: SAL AX, 1             ; DETERMINE_MODE
1502 SAL AX, 1
1503 SAL AX, 1
1504 ADD SI, AX              ; MULTIPLY CODE POINT VALUE BY 8
1505 CMP _CRT_MODE, 6          ; SI HAS OFFSET OF DESIRED CODES
1506 POP DS
1507 JC S7                  ; RECOVER TABLE POINTER SEGMENT
1508 JC S7                  ; TEST FOR MEDIUM RESOLUTION MODE
1509
1510 ;----- HIGH RESOLUTION MODE
1511 S3: PUSH DI              ; HIGH CHAR
1512 PUSH SI                ; SAVE REGEN POINTER
1513 MOV DH, 4               ; SAVE CODE POINTER
1514
1515 S4: LODSB                ; NUMBER OF TIMES THROUGH LOOP
1516 TEST BL, 80H             ; GET BYTE FROM CODE POINTS
1517 JNZ S6                  ; SHOULD WE USE THE FUNCTION
1518 LODSB                ; TO PUT CHAR IN
1519 LODSB                ; STORE IN REGEN BUFFER
1520
1521 S5: MOV ES:[DI+2000H-1], AL ; STORE IN SECOND HALF
1522 ADD DI, 79               ; MOVE TO NEXT ROW IN REGEN
1523 DEC DI
1524 JNZ S6                  ; DONE WITH LOOP
1525
1526 S6: XOR AL, ES:[DI]       ; EXCLUSIVE OR WITH CURRENT
1527 STOSB                ; STORE THE CODE POINT
1528 LODSB                ; AGAIN FOR ODD FIELD
1529 XOR AL, ES:[DI+2000H-1] ; BACK TO MAINSTREAM
1530 JNP S5
1531
1532 ;----- MEDIUM RESOLUTION WRITE
1533 S7: MOV DL, BL             ; MED_RES_WRITE
1534 SAL DI, 1                ; SAVE_HIGR COLOR BIT
1535 LODSB                ; OFFSET*2 SINCE 2 BYTES/CHAR
1536 AND BL, 3                ; EXPAND BL TO FULL WORD OF COLOR
1537 MOV AL, 055H             ; ISOLATE THE COLOR BITS ( LOW 2 BITS )
1538 MUL BL                 ; GET BIT CONVERSION MULTIPLIER
1539 MOV BH, AL               ; EXPAND 2 COLOR BITS TO 4 REPLICATIONS
1540 MOV BH, AL               ; PLACE BACK IN WORK REGISTER
1541 LODSB                ; EXPAND TO 8 REPLICATIONS OF COLOR BITS
1542 AND BH, BL              ; MED_CHAR
1543 XOR BH, BH
1544 SAL DI, 1                ; SAVE'REGEN POINTER
1545 LODSB                ; SAVE THE CODE POINTER
1546 MOV DH, 4               ; NUMBER OF LOOPS
1547
1548 S8: PUSH DI              ; GET CODE POINT
1549 PUSH SI                ; DOUBLE UP ALL THE BITS
1550 MOV DH, 4
1551 LODSB                ; CONVERT TO FOREGROUND COLOR ( 0 BACK )
1552 AND AX, BX              ; SWAP HIGH/LOW BYTES FOR WORD MOVE
1553 XCHG AH, AL             ; IS THIS XOR FUNCTION
1554 TEST DL, 80H             ; NO, STORE IT IN AS IT IS
1555 JZ S11                 ; DO FUNCTION WITH HIGH/LOW
1556 XOR AX, ES:[DI]
1557 XOR AX, ES:[DI]
1558 LODSB                ; STORE FIRST BYTE HIGH, SECOND LOW
1559 CALL S21                ; GET CODE POINT
1560 CALL S21
1561 AND AX, BX
1562 XCHG AH, AL
1563 TEST DL, 80H
1564 JZ S11
1565 XOR AX, ES:[DI+2000H]
1566 LODSB                ; STORE SECOND PORTION HIGH
1567 ADD DI, 80
1568 DEC DH
1569 JNZ S9                  ; POINT TO NEXT LOCATION
1570
1571 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   $9
1564 063B E9 013D R    JMP    VIDEO_RETURN
1565 063E             GRAPHICS_WRITE ENDP

1566
1567
1568
1569 063E             ;-----; GRAPHICS READ
1570 063E E8 0E9 R    GRAPHICS_READ PROC NEAR
1571 0641 BB F0        CALL   DS
1572 0643 83 EC 08      MOV   SI,AX
1573 0646 BB EC        SUB   SP,B
1574                   MOV   BP,SP
1575
1576
1577 0648 80 3E 0049 R 06  ;-----; DETERMINE GRAPHICS MODES
1578 064D 06          CMP    *CRT_MODE,6
1579 064E 1F          PUSH   ES
1580 064F T2 19        POP    DS
1581                   JC    S13
1582
1583
1584 0651 B6 04        ;-----; HIGH RESOLUTION READ
1585 0652 8A 04        S12: GET VALUES FROM REGEN BUFFER AND CONVERT TO CODE POINT
1586 0653 8A 04        MOV    AL,[SI]
1587 0655 88 46 00      MOV    [BP],AL
1588 0658 45          INC    BP
1589 0659 8A 2000      MOV    AL,[SI+200H]
1590 065D 88 46 00      MOV    [BP],AL
1591 0660 45          INC    BP
1592 0661 8A 04        ADD    DH,80
1593 0662 8C 50        DEC    DH
1594 0664 FE CE        JNZ    S12
1595 0666 75 EB        JMP    SHORT S15
1596 0668 EB 16        JMP    SHORT S15
1597
1598
1599 066A             ;-----; MEDIUM RESOLUTION READ
1600 066A D1 E6        S13: MED RES READ
1601 066C B6 04        SAL    S1,I
1602 066E             MOV    DH,4
1603 066E EB 06CF R    S14: OFFSET$2 SINCE 2 BYTES/CHAR
1604 0671 B1 C6 1FFE    CALL   S23
1605 0672 80 6C 00      ADD    S1,200H-2
1606 0678 81 E1 IFBZ    CALL   S23
1607 067C FE CE        SUB    S1,200H-80+2
1608 067E 75 EE        DEC    DH
1609                   JNZ    S14
1610
1611 0680             ;-----; SAVE AREA HAS CHARACTER IN IT, MATCH IT
1612 0680 BF 0000 E    S15: FIND_CHAR
1613 0683 0E          PUSH   CS
1614 0684 07          POP    ES
1615 0685 83 ED 08      SUB    BP,B
1616 0688 BB F5        MOV    S1,BP
1617 0689 80 00        MOV    AL,0
1618 068C             S16: CURRENT CODE POINT BEING MATCHED
1619 068C 16          PUSH   SS
1620 068D 1F          POP    DS
1621 068E BA 0080      MOV    DX,128
1622 0691             S17: SAVE SAVE AREA POINTER
1623 0691 56          PUSH   SI
1624 0692 67          PUSH   DI
1625 0693 B9 0004      MOV    CX,4
1626 0696 F3 / A7      REPE  CMPSW
1627 0698 5F          POP    DI
1628 0699 5E          POP    SI
1629 069A FE IE        JZ    S18
1630 069C B6 CO        INC    AL
1631 069E 83 CT 08      ADD    AL,8
1632 06A1 4A          DEC    DX
1633 06A2 75 ED        JNZ    S17
1634
1635
1636
1637 06A4 3C 00        ;-----; CHAR NOT MATCHED, MIGHT BE IN USER SUPPLIED SECOND HALF
1638 06A6 T4 12          CMP    AL,0
1639 06A8 2B C0          JE    S18
1640 06AA BE D8          SUB    AX,AX
1641 06AC 00             ASSUME DS:AB50
1642 06AC C4 3E 007C R  LESS  DS:DATA_PTR
1643 06B0 8C C0          MOV    AX,ES
1644 06B2 0B C7          OR     AX,DI
1645 06B4 T4 04          JZ    S18
1646 06B6 B0 80          MOV    AL,128
1647 06B8 EB D2          JMP    S16
1648                   ASSUME DS:DATA
1649
1650
1651 06BA             ;-----; CHARACTER IS FOUND ( AL=0 IF NOT FOUND )
1652 06BA B3 C4 08        S18: READJUST THE STACK, THROW AWAY SAVE
1653 06BD E9 013D R    ADD    SP,B
1654 06C0             JMP    VIDEO_RETURN
1655
1656
1657
1658
1659
1660
1661 06C0             ;-----; EXPAND BYTE
1662 06C0 51          GRAPHICS_READ ENDP
1663 06C1 B9 0008
1664 06C6 00 00
1665 06C4 D0 C8
1666 06C6 D1 DD
1667 06C6 D1 FD
1668 06C6 E2 F8
1669
1670 06CC 95          XCHG   AX,BP
1671 06CD 59          POP    CX
1672 06CE C3          RET
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
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
3243
3244
3245
3246
3247
3248
3249
3250
3251
3252
3253
3254
3255
3256
3257
3258
3259
3260
3261
3262
3263
3264
3265
3266
3267
3268
3269
3270
3271
3272
3273
3274
3275
3276
3277
3278
3279
3280
3281
3282
3283
3284
3285
3286
3287
3288
3289
3290
3291
3292
3293
3294
3295
3296
3297
3298
3299
3300
3301
3302
3303
3304
3305
3306
3307
3308
3309
3310
3311
3312
3313
3314
3315
3316
3317
3318
3319
3320
3321
3322
3323
3324
3325
3326
3327
3328
3329
3330
3331
3332
3333
3334
3335
3336
3337
3338
3339
3340
3341
3342
3343
3344
3345
3346
3347
3348
3349
3350
3351
3352
3353
3354
3355
3356
3357
3358
3359
3360
3361
3362
3363
3364
3365
3366
3367
3368
3369
3370
3371
3372
3373
3374
3375
3376
3377
3378
3379
3380
3381
3382
3383
3384
3385
3386
3387
3388
3389
3390
3391
3392
3393
3394
3395
3396
3397
3398
3399
3400
3401
3402
3403
3404
3405
3406
3407
3408
3409
3410
3411
3412
3413
3414
3415
3416
3417
3418
3419
3420
3421
3422
3423
3424
3425
3426
3427
3428
3429
3430
3431
3432
3433
3434
3435
3436
3437
3438
3439
3440
3441
3442
3443
3444
3445
3446
3447
3448
3449
3450
3451
3452
3453
3454
3455
3456
3457
3458
3459
3460
3461
3462
3463
3464
3465
3466
3467
3468
3469
3470
3471
3472
3473
3474
3475
3476
3477
3478
3479
3480
3481
3482
3483
3484
3485
3486
3487
3488
3489
3490
3491
3492
3493
3494
3495
3496
3497
3498
3499
3500
3501
3502
3503
3504
3505
3506
3507
3508
3509
3510
3511
3512
3513
3514
3515
3516
3517
3518
3519
3520
3521
3522
3523
3524
3525
3526
3527
3528
3529
3530
3531
3532
3533
3534
3535
3536
3537
3538
3539
3540
3541
3542
3543
3544
3545
3546
3547
3548
3549
3550
3551
3552
3553
3554
3555
3556
3557
3558
3559
3560
3561
3562
3563
3564
3565
3566
3567
3568
3569
3570
3571
3572
3573
3574
3575
3576
3577
3578
3579
3580
3581
3582
3583
3584
3585
3586
3587
3588
3589
3590
3591
3592
3593
3594
3595
3596
3597
3598
3599
3600
3601
3602
3603
3604
3605
3606
3607
3608
3609
3610
3611
3612
3613
3614
3615
3616
3617
3618
3619
3620
3621
3622
3623
3624
3625
3626
3627
3628
3629
3630
3631
3632
3633
3634
3635
3636
3637
3638
3639
3640
3641
3642
3643
3644
3645
3646
3647
3648
3649
3650
3651
3652
3653
3654
3655
3656
3657
3658
3659
3660
3661
3662
3663
3664
3665
3666
3667
3668
3669
3670
3671
3672
3673
3674
3675
3676
3677
3678
3679
3680
3681
3682
3683
3684
3685
3686
3687
3688
3689
3690
3691
3692
3693
3694
3695
3696
3697
3698
3699
3700
3701
3702
3703
3704
3705
3706
3707
3708
3709
3710
3711
3712
3713
3714
3715
3716
3717
3718
3719
3720
3721
3722
3723
3724
3725
3726
3
```

```

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             ; GET FIRST BYTE AND SECOND BYTES
1689 06D0 86 C4        XCHG AL,AH           ; SWAP FOR COMPARE
1690 06D0 B9 C000        MOV CX,0C00H         ; 2 BIT MASK TO TEST THE ENTRIES
1691 06D0 B2 00        MOV DL,0            ; RESULT REGISTER
1692 06D1
1693 06D7 85 C1        S24:    TEST AX,CX           ; IS THIS SECTION BACKGROUND?
1694 06D9 74 01        JZ     S25              ; IF ZERO, IT IS BACKGROUND (CARRY=0)
1695 06DE F9        STC               ; WASN'T, SO SET CARRY
1696 06DC
1697 06E0 D0 D2        S25:    RCL   DL,1             ; MOVE THAT BIT INTO THE RESULT
1698 06E0 E9        SHR    CX,1             ; MOVE THE MASK TO THE RIGHT BY 2 BITS
1699 06E0 D1 E9        SHR    CX,1             ; DO IT AGAIN IF MASK DIDN'T FALL OUT
1700 06E2 73 F3        JNC    S24              ; STORE RESULT IN SAVE AREA
1701 06E4 88 56 00        MOV [BP],DL          ; ADJUST POINTER
1702 06E7 45        INC    BP               ; ALL DONE
1703 06E8 C3        RET               ; ENDP
1704 06E9
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  | 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          LABEL  NEAR
1719 06EC 53        PUSH   BX             ; SAVE REGISTER
1720 06E6 8B D8        MOV    BX,AX           ; SAVE A COPY OF CURRENT CURSOR
1721 06E7 A0 004A R      MOV    AL,BYTE PTR *CRT_COLS ; GET BYTES PER COLUMN
1722 06F2 F6 E4        MUL    AH             ; MULTIPLY BY ROWS
1723 06F2 E0          SHL    AX,1            ; ISOLATE COLUMN VALUE
1724 06F6 D1 E0        SUB    BH,BH           ; DETERMINE OFFSET
1725 06F8 2A FF        ADD    AX,BX           ; RECOVER POINTER
1726 06FA 03 C3        POP    BX             ; ALL DONE
1727 06FC 58        RET               ; ENDP
1728 06FC C3        S26    ENDP
1729 06FE          S26    ENDP
1730  |----- WRITE_TTY -----
1731
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 06FF 97        XCHG DI,AX           ; SAVE (AX) REGISTER IN (DI) FOR EXIT
1755 06FF B4 03        MOV    AH,03H          ; READ CURSOR POSITION
1756 0701 80 3E 0062 R      MOV    BH,ACTIVE_PAGE ; GET CURRENT PAGE SETTING
1757 0705 CD 10        INT    10H             ; READ THE CURRENT CURSOR POSITION
1758 0707 BB C7        MOV    AX,DI           ; RECOVER CHARACTER FROM (DI) REGISTER
1759
1760  |----- DX NOW HAS THE CURRENT CURSOR POSITION
1761
1762 0709 3C 0D        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 070D          UO:    MOV    AH,0AH          ; WRITE CHARACTER ONLY COMMAND
1767 070F B9 0001        MOV    CX,1            ; ONLY ONE CHARACTER
1768 0712 CD 10        INT    10H             ; WRITE THE CHARACTER
1769
1770  |----- POSITION THE CURSOR FOR NEXT CHAR
1771
1772 0714 FE C2        INC    DL             ; TEST FOR COLUMN OVERFLOW
1773 0716 8A 1A 004A R      CMP    DL,BYTE PTR *CRT_COLS ; SET_CURSOR
1774 071A 75 33        JNZ    U1              ; COLUMN FOR CURSOR
1775 071C B2 00        MOV    DL,0            ; CHECK FOR LAST ROW
1776 071E 80 FE 18        CMP    DH,25-1         ; SET_CURSOR_INC
1777 0721 75 2A        JNZ    U6              ; SET_CURSOR_INC
1778
1779  |----- SCROLL REQUIRED
1780 0723 B4 02        U1:    MOV    AH,02H          ; SET THE CURSOR
1781 0725 CD 10        INT    10H             ; SET THE CURSOR
1782
1783 0725 0000          U1:    INT    10H             ; SET THE CURSOR
1784
1785  |----- DETERMINE VALUE TO FILL WITH DURING SCROLL

```

```

1787 0727 A0 0049 R      MOV    AL, *CRT_MODE          ; GET THE CURRENT MODE
1788 072A 3C 04           CMP    AL, 4                ; READ_CURSOR
1789 072C 72 06           JC     U2                  ; FILL WITH BACKGROUND
1790 072E 3C 07           CMP    AL, 7                ; SCROLL UP ONE LINE
1791 0730 B7 00           MOV    BH, 0               ; READ_CURSOR
1792 0732 75 06           JNE    U3                  ; STORE IN BH
1793 0734                 U2:                ; READ_CURSOR
1794 0734 B4 08           MOV    AH, 0BH             ; SCROLL UP
1795 0736 CD 10           INT    10H                ; GET READ CURSOR COMMAND
1796 0738 8A FC           MOV    BH, AH              ; READ CHAR/ATTR AT CURRENT CURSOR
1797 073A                 U3:                ; STORE IN BH
1798 073A BB 0601          MOV    AX, 0601H           ; SCROLL UP ONE LINE
1799 073D B2 C9           SUB    CX, BX             ; UPPER LEFT CORNER
1800 073F B6 18           MOV    DH, 25-1           ; LOWER RIGHT ROW
1801 0741 16 004A R       MOV    DL, BYTE PTR *CRT_COLS ; LOWER RIGHT COLUMN
1802 0745 FE CA           DEC    DL                ; VIDEO-CALL-RETURN
1803 0747                 U4:                ; SCROLL UP
1804 0747 CD 10           INT    10H                ; SET-CURSOR-INC
1805 0749 19               INC    DH                ; NEXT ROW
1806 0749 97               U5:                ; SET-CURSOR
1807 074A E9 013D R       XCHG   AX, DI             ; RESTORE THE ENTRY CHARACTER FROM (DI)
1808                 JMP    VIDEO_RETURN          ; RETURN TO CALLER
1809 074D                 U6:                ; ESTABLISH THE NEW CURSOR
1810 074D FE C6           INC    DH                ; SET-CURSOR-INC
1811 074F 02               UT:                ; NEXT ROW
1812 074F B4 02           MOV    AH, 02H             ; SET-CURSOR
1813 0751 EB F4           JMP    U4                ; ESTABLISH THE NEW CURSOR
1814
1815                 I-----; CHECK FOR CONTROL CHARACTERS
1816 0753                 U8:                ; IF NOT A CONTROL, DISPLAY IT
1817 0753 74 13           JE     U9                  ; WAS IT A CARRIAGE RETURN
1818 0755 3C 04           CMP    AL, LF              ; IS IT A LINE FEED
1819 0757 74 13           JE     U10                ; GO TO LINE FEED
1820 0759 3C 07           CMP    AL, 07H             ; IS IT A BELL
1821 075B 74 16           JE     U11                ; GO TO BELL
1822 075D 3C 08           CMP    AL, 08H             ; IS IT A BACKSPACE
1823 075F 75 AC           JNE    U0                  ; IF NOT A CONTROL, DISPLAY IT
1824
1825                 I-----; BACK SPACE FOUND
1826
1827 0761 0A D2           OR    DL, DL             ; IS IT ALREADY AT START OF LINE
1828 0763 74 EA           JE     UT                  ; SET_CURSOR
1829 0765 4A               DEC    DX                ; NO -- JUST MOVE IT BACK
1830 0766 EB E7           JMP    UT                  ; SET_CURSOR
1831
1832                 I-----; CARRIAGE RETURN FOUND
1833
1834 0768                 U9:                ; MOVE TO FIRST COLUMN
1835 0768 B2 00           MOV    DL, 0               ; SET_CURSOR
1836 076A EB E3           JMP    UT
1837
1838                 I-----; LINE FEED FOUND
1839
1840 074C                 U10:               ; BOTTOM OF SCREEN
1841 074C 80 FE 18           CMP    DH, 25-1           ; YES, SCROLL THE SCREEN
1842 074F 75 DC           JNE    U4                ; NO, JUST SET THE CURSOR
1843 0771 EB B0           JMP    U1
1844
1845
1846
1847 0773                 U11:               ; 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 EE 0000 E        CALL   BEEP              ; SOUND THE POD BELL
1851 077B EB CC           JMP    U5                  ; TTY_RETURN
1852 077D                 WRITE_TTY            ENDP
1853
1854                 I-----; LIGHT PEN
1855 THIS ROUTINE TESTS THE LIGHT PEN SWITCH AND THE LIGHT
1856 PEN TRIGGER. IF BOTH ARE SET, THE LOCATION OF THE LIGHT
1857 PEN IS DETERMINED. OTHERWISE, A RETURN WITH NO INFORMATION
1858 IS MADE.
1859 ON EXIT:
1860 (AH) = 0 IF NO LIGHT PEN INFORMATION IS AVAILABLE
1861 (BX,CX,DX) ARE DESTROYED
1862 (AH) = 1 IF LIGHT PEN IS AVAILABLE
1863 (DH,DL) = ROW,COLUMN OF CURRENT LIGHT PEN POSITION
1864 (CH) = RASTER POSITION
1865 (BX) = BEST GUESS AT PIXEL HORIZONTAL POSITION
1866
1867
1868 077D 03 03 05 05 03 03 V1 ASSUME DS:DATA
1869 03 04 DB 3,3,5,5,3,3,3,4 ; SUBTRACT_TABLE
1870
1871                 I-----; WAIT FOR LIGHT PEN TO BE DEPRESSED
1872 0785                 READ_LPEN PROC NEAR
1873 0785 B4 00           MOV    AH, 0               ; SET NO LIGHT PEN RETURN CODE
1874 0787 B8 16 0063 R       MOV    DX, *ADDR_6845 ; GET BASE ADDRESS OF 6845
1875 078B B3 C2 06           ADD    DX, 6              ; POINT TO STATUS REGISTER
1876 078E EC               IN     AL, DX             ; GET STATUS REGISTER
1877 078F A0 04           TEST   AL, 004H           ; TEST LIGHT PEN SWITCH
1878 0791 14 03           JZ    V6-A              ; GO IF YES
1879 0793 E9 0816 R       JMP    V6                ; NOT SET, RETURN
1880
1881                 I-----; NOT TEST FOR LIGHT PEN TRIGGER
1882
1883 0796 A8 02           V6_A:  TEST   AL, 2              ; TEST LIGHT PEN TRIGGER
1884 0798 75 03           JNZ    V7A              ; RETURN WITHOUT RESETTING TRIGGER
1885 079A E9 0820 R       JMP    V7
1886
1887                 I-----; TRIGGER HAS BEEN SET, READ THE VALUE IN
1888
1889 079D                 VTA:  MOV    AH, 16              ; LIGHT PEN REGISTERS ON 6845
1890 079D B4 10           MOV    AH, 16
1891
1892                 I-----; INPUT REGISTERS POINTED TO BY AH, AND CONVERT TO ROW/COLUMN IN (DX)
1893
1894 079F 8B 16 0063 R       MOV    DX, *ADDR_6845 ; ADDRESS REGISTER FOR 6845
1895 07A3 8A C4           MOV    AL, AH              ; REGISTER TO READ
1896 07A5 EE               OUT   DX, AL             ; SET IT UP
1897 07A6 90               NOP
1898 07A7 42               INC    DX
1899 07A8 EC               IN     AL, DX             ; I/O DELAY
1900 07A9 8A E8           MOV    CH, AL             ; DATA REGISTER
1901

```

```

1901 0TAB 4A           DEC    DX          ; ADDRESS REGISTER
1902 0TAC FE C4         INC    AH          ; SECOND DATA REGISTER
1903 0TAE 8A C4         MOV    AL,AH
1904 0TB0 EE             OUT   DX,AL      ; POINT TO DATA REGISTER
1905 0TBI 42             INC    DX          ; I/O DELAY
1906 0TC0 90             NOP
1907 0TB3 CC             IN    AL,DX      ; GET SECOND DATA VALUE
1908 0TB4 E5             MOV    AH,CH      ; AX HAS INPUT VALUE
1909
1910 ;----- AX HAS THE VALUE READ IN FROM THE 6845
1911
1912 0TB6 8A IE 0049 R   MOV    BL,*CRT_MODE
1913 0TBA 2A FF           SUB   BX,BX
1914 0TCB 2E; 8A 9F 077D R MOV    BL,CSIV1[BX] ; MODE VALUE TO BX
1915 0TC1 2B C3           SUB   AX,BX      ; DETERMINE AMOUNT TO SUBTRACT
1916 0TC3 8B 1E 004E R   MOV    BX,*CRT_START ; TAKE IT AWAY
1917 0TC7 D1 EB           SHR   BX,1
1918 0TC9 2B C3           SUB   AX,BX      ; CONVERT TO CORRECT PAGE ORIGIN
1919 0TCB 19 02           JNS   V2          ; IF POSITIVE, DETERMINE MODE
1920 0TC0 2B C0           SUB   AX,AX      ; <0 PLAYS AS 0
1921
1922 ;----- DETERMINE MODE OF OPERATION
1923
1924 0TCF
1925 0TCF B1 03           V2:    MOV    CL,3        ; DETERMINE MODE
1926 0TD1 80 3E 0049 R 04   CMP   *CRT_MODE,4 ; SET SHIFT COUNT
1927 0TDA 72 2A           JB    V4          ; DETERMINE IF GRAPHICS OR ALPHA
1928 0TDE 80 3E 0049 R 07   CMP   *CRT_MODE,7 ; ALPHA_PEN
1929 0TDD 74 23           JE    V4          ; ALPHA_PEN
1930
1931 ;----- GRAPHICS MODE
1932
1933 0TDF B2 28           MOV    DL,40       ; DIVISOR FOR GRAPHICS
1934 0TE1 F6 F2           DIV   DL          ; DETERMINE ROW(AL) AND COLUMN(AH)
1935
1936 ;----- DETERMINE GRAPHIC ROW POSITION
1937
1938 0TE3 8A E8           MOV    CH,AL      ; SAVE ROW VALUE IN CH
1939 0TE5 02 ED           ADD   CH,CH      ; *2 FOR EVEN/ODD FIELD
1940 0TE7 8A DC           MOV    BL,AH      ; COLUMN VALUE TO BX
1941 0TE9 2A FF           SUB   BH,BH      ; MULTIPLY BY 8 FOR MEDIUM RES
1942 0TEB 80 3E 0049 R 06   CMP   *CRT_MODE,6 ; DETERMINE MEDIUM OR HIGH RES
1943 0TEC 75 04           JNE   V3          ; NOT HIGH RES
1944 0TF2 00 04           MOV   CL,4        ; HIGH VALUE FOR HIGH RES
1945 0TF4 D0 E4           SAL   AH,1
1946 0TF6
1947 0TF6 D3 E3           SHL   BX,CL      ; COLUMN VALUE TIMES 2 FOR HIGH RES
1948
1949 ;----- DETERMINE ALPHA CHAR POSITION
1950
1951 0TF8 8A D4           MOV    DL,AH      ; NOT HIGH RES
1952 0TF8 8A F0           MOV    DH,AL      ; COLUMN VALUE FOR RETURN
1953 0TFC D0 EE           SHR   DH,1
1954 0TFC D0 EE           SHR   DH,1      ; ROW VALUE
1955 0800 EB 12           JMP   SHORT V5  ; DIVIDE BY 4
1956
1957 ;----- ALPHA MODE ON LIGHT PEN
1958
1959 0802
1960 0802 F6 36 004A R   V4:    DIV   BYTE PTR *CRT_COLS ; FOR VALUE IN 0-24 RANGE
1961 0800 EA 00           MOV   DH,AH      ; LIGHT_PEN_RETURN_SET
1962 0808 EA D4           MOV   DH,AH
1963 080A D2 E0           MOV   AL,CL      ; DETERMINE ROW,COLUMN VALUE
1964 080C 8A E8           MOV   CH,AL      ; ROWS TO DIVIDE BY 8
1965 080E 8A DC           MOV   BL,AH      ; COLS TO DIVIDE BY 8
1966 0810 32 FF           XOR   BH,BH      ; GET RASTER VALUE TO RETURN REGISTER
1967 0810 D3 E3           SAL   BX,CL      ; COLUMN VALUE
1968 0814
1969 0814 B4 01           MOV   AH,1      ; TO BX
1970 0816
1971 0816 52             V6:    PUSH  DX        ; LIGHT PEN RETURN_SET
1972 0817 BB 16 0063 R   MOV   DX,*ADDR_6845 ; INDICATE EVERY THING SET
1973 0817 83 C2 07           ADD   DX,7
1974 081E 00 FF           OUT   DX,AL      ; LIGHT PEN RETURN
1975 081F 5A             POP   DX          ; SAVE RETURN VALUE (IN CASE)
1976 0820
1977 0820 5D             POP   BP          ; GET BASE ADDRESS
1978 0821 5F             POP   DI          ; POINT TO RESET PARM
1979 0822 5E             POP   SI          ; ADDRESS, NO DATA, IS IMPORTANT
1980 0823 F               POP   DS          ; RECLEAR VALUE
1981 0824 1F             POP   DS          ; RETURN_NO_RESET
1982 0825 1F             POP   DS
1983 0826 1F             POP   DS
1984 0827 07             POP   ES
1985 0828 CF             IRET
1986 0829           READ_LPEN
1987 0829           CODE ENDS
1988           ENDP

```

```

1 PAGE 118,121
2 TITLE BIOS1 ---- 01/10/86 INTERRUPT 15H BIOS ROUTINES
3 .LIST
4 0000 CODE SEGMENT BYTE PUBLIC
5
6 PUBLIC CASSETTE_10_I
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 RETURNS FOR THESE FUNCTIONS ALWAYS (AH) = 86H, CY = 1
19 IF CASSETTE PORT NOT PRESENT
20
21 INPUT - UNUSED FUNCTIONS
22 (AH) = 04H THROUGH 7FH
23 RETURNS FOR THESE FUNCTIONS ALWAYS (AH) = 86H, CY = 1
24 (UNLESS INTERCEPTED BY SYSTEM HANDLERS)
25 NOTE: THE KEYBOARD INTERRUPT HANDLER INTERRUPTS WITH AH=4FH
26
27 EXTENSIONS
28 (AH) = 80H DEVICE OPEN (NULL)
29 (BX) = DEVICE ID
30 (CX) = PROCESS ID
31
32 (AH) = 81H DEVICE CLOSE (NULL)
33 (BX) = DEVICE ID
34 (CX) = PROCESS ID
35
36 (AH) = 82H PROGRAM TERMINATION (NULL)
37 (BX) = DEVICE ID
38
39 (AH) = 83H EVENT WAIT (NULL)
40
41 (AH) = 84H JOYSTICK SUPPORT
42 (DX) = 00H - READ THE CURRENT SWITCH SETTINGS
43 RETURNS AL = SWITCH SETTINGS (BITS 7-4)
44 (DX) = 01H - READ THE RESISTIVE INPUTS
45 RETURNS AX = A(x) VALUE
46 (DX) = 02H - READ THE DYNAMIC JOYSTICK POSITION
47 CX = B(x) VALUE
48 DX = B(y) VALUE
49
50 (AH) = 88H EXTENDED MEMORY SIZE DETERMINE
51
52 (AH) = 91H INTERRUPT COMPLETE FLAG SET
53 (AL) TYPE CODE
54 00H -> 7FH
55 SERIALLY REUSABLE DEVICES
56 OPERATING SYSTEM MUST SERIALIZE ACCESS
57 80H -> BFH
58 SUBENTRANT DEVICES; ES:BX IS USED TO
59 DISTINGUISH DIFFERENT CALLS (MULTIPLE I/O
60 CALLS ARE ALLOWED SIMULTANEOUSLY)
61 COH -> FFH
62 WAIT ONLY CALLS -- THERE IS NO
63 COMPLEMENTARY POST* FOR THESE WAITS.
64 THESE ARE TIMEOUT ONLY. TIMES ARE
65 FUNCTION NUMBER DEPENDENT.
66
67 TYPE DESCRIPTION TIMEOUT
68
69 00H = DISK YES
70 01H = DISKETTE YES
71 02H = KEYBOARD NO
72 80H = NETWORK NO
73 (ES:BX -> NCB)
74 FDH = DISKETTE MOTOR START YES
75 FEH = PRINTER YES
76
77 (AH) = COH RETURN CONFIGURATION PARAMETERS POINTER
78 RETURNS
79 (AH) = 00H AND CY= 0 (IF PRESENT ELSE 86 AND CY= 1)
80 (ES:BX) = PARAMETER TABLE ADDRESS POINTER
81 WHERE:
82
83 DW 8 LENGTH OF FOLLOWING TABLE
84 DB MODEL_BYTE SYSTEM MODEL BYTE
85 DB TYPE_BYTE SYSTEM MODEL TYPE BYTE
86 DB BIOS_LEVEL BIOS REVISION LEVEL
87 DB ?
88 00000000 = DISK CHANNEL 3 USE BY BIOS
89 01000000 = DISK CHANNEL INTERRUPT LEVEL 2
90 00010000 = REAL TIME CLOCK AVAILABLE
91 00010000 = KEYBOARD SCAN CODE HOOK 1AH
92 DB 0 RESERVED
93 DB 0 RESERVED
94 DB 0 RESERVED
95 DB 0 RESERVED
96
97
98 ASSUME CS:CODE
99
100 0000 CASSETTE_10_I PROC FAR
101 0000 FB STI: ; ENABLE INTERRUPTS
102 0001 80 FC 80 CMP AH,080H ; CHECK FOR RANGE OF 00-7FH
103 0004 73 06 JAE CI_G ; SKIP AND HANDLE, ELSE RETURN ERROR
104
105 0006 CI_I: MOV AH,86H ; ERROR
106 0006 B4 86 STC ; SET BAD COMMAND
107 0008 F9 ; SET CARRY FLAG ON (CY=1)
108
109 0009 CI_F: RET 2 ; COMMON EXIT
110 0009 CA 0002 ; FAR RETURN EXIT FROM ROUTINES
111
112 000C CI_G: CMP AH,0C0H ; CONTINUE CHECKING FOR FUNCTION
113 000C 80 FC C0 JE CONF_PARMS ; CHECK FOR CONFIGURATION PARAMETERS
114 000F 74 2E

```

```

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 17          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    CI             ; 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             ; TURN CARRY OFF
143 003C EB CB          JMP    CI_F          ; RETURN WITH (AH= 00) AND CY=0
144
145 003E                 CASSETTE_10_1; ENDP
146
147
148
149
150
151
152
153
154
155 003E                 INT_COMPLETE PROC NEAR ; 
156 003E CF              IRET            ; RETURN
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    CI_F          ; EXIT THROUGH COMMON RETURN
165 0048
166
167
168
169
170
171
172
173
174
175
176
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
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204 004B    JOY_STICK    PROC NEAR
205      STI             ; INTERRUPTS BACK ON
206      MOV AX,DX        ; GET SUB FUNCTION CODE
207      MOV DX,201H       ; ADDRESS OF PORT
208      OR AL,AL          ;
209      JZ JOY_2          ; READ SWITCHES
210      DEC AL           ; AL=0
211      JZ JOY_3          ; READ RESISTIVE INPUTS
212      JMP C1            ; GO TO ERROR RETURN
213
214 005B    JOY_1:         STI             ; INTERRUPTS BACK ON
215      JMP C1_F          ; GO TO COMMON RETURN
216
217 005E    JOY_2:         IN  AL,DX        ; READ THE CURRENT SWITCHES
218      AND AL,0FH        ; RETURNS (AL)=SWITCH SETTINGS IN BITS 7-4
219      JMP JOY_1          ; (DX)=0 READ THE RESISTIVE INPUTS
220      OR AL,AL          ; RETURNS (AL)=(BX+A(X)) VALUE
221      SHR CX,1           ; (BX+A(Y)) VALUE
222      SHR CX,1           ; (CX)=B(X) VALUE
223      SHR CX,1           ; (DX)=B(Y) VALUE
224 0055    FE C8          DEC AL           ; AL=0
225 0053    74 09          JZ JOY_2          ; READ SWITCHES
226 0057    74 0A          JZ JOY_3          ; READ RESISTIVE INPUTS
227 0059    EB AB          JMP C1            ; GO TO ERROR RETURN
228
229 005B    JOY_1:         STI             ; INTERRUPTS BACK ON
230      JMP C1_F          ; GO TO COMMON RETURN
231
232 005E    EC              IN  AL,DX        ; STRIP UNWANTED BITS OFF
233 005F    24 F0          AND AL,0FH        ; FINISHED
234 0061    EB F8          JMP JOY_1          ;
235
236 0063    B3 01          MOV BL,1          ; SAVE A(X) VALUE
237 0065    E8 0081 R      CALL TEST_CORD
238 0066    51             PUSH CX
239 0069    03 02          MOV BL,2          ; SAVE A(Y) VALUE
240 006B    E8 0081 R      CALL TEST_CORD
241 006E    51             PUSH CX
242 006F    B3 04          MOV BL,4          ; SAVE B(X) VALUE
243 0071    E8 0081 R      CALL TEST_CORD
244 0074    51             PUSH CX
245 0075    B3 08          MOV BL,5          ; SAVE B(Y) VALUE
246 0077    E8 0081 R      CALL TEST_CORD
247 0078    51             PUSH CX
248 007A    BB D1          MOV DX,CX        ; GET B(X) VALUE
249 007C    59             POP CX
250 007D    5B             POP BX
251 007E    58             POP AX
252 007F    EB DA          JMP JOY_1          ; FINISHED - RETURN
253
254 0081    TEST_CORD:    PROC NEAR
255 0081    52              PUSH DX
256 0082    FA              CLI             ; SAVE
257 0083    00             MOV AL,0          ; BLOCK INTERRUPTS WHILE READING
258 0084    B0 00          OUT  AL,TIMER+3,AL ; SET UP TO LATCH TIMER 0
259 0085    6 43            JMP $-2
260 0087    EB 00          IN   AL,TIMER
261 0088    E8 00          JMP $-2
262 0089    E4 40          IN   AL,TIMER
263 008A    84 E0          MOV AH,AL
264 008B    E4 40          IN   AL,TIMER
265 008C    00               ; READ HIGH BYTE OF TIMER 0
266 008D    84 E0          AH,AL
267 008E    00               ; REARRANGE TO HIGH,LOW
268 008F    E4 40          XCHG AH,AL
269 0089    50             PUSH AX
270 0091    66 E0          MOV CX,4FFH
271 0093    50             OUT DX,AL
272 0094    B9 04FF          JMP $-2
273 0097    EE              OUT DX,AL
274 0098    EB 00          JMP $-2
275
276 009A    EC              TEST AL,DX        ; READ VALUES
277 009B    84 C3          TEST AL,BL        ; HAS PULSE ENDED?
278 009D    E0 FB          LOOPNZ TEST_CORD_1
279 009F    B3 F9 00          CMP CX,0-
280 00A2    59             POP CX
281 00A3    75 04          JNZ SHORT TEST_CORD_2
282 00A4    EB 00          SUB CX,CX
283 00A7    EB 2D          JMP SHORT TEST_CORD_3
284 00A9
285
286 00A9    B0 00          MOV AL,0          ; ORIGINAL COUNT
287 00AB    E4 40          OUT TIMER+3,AL ; SET 0 COUNT FOR RETURN
288 00AC    EB 00          JMP $-2
289 00AD    E4 40          IN  AL,TIMER
290 00AE    84 E0          MOV AH,AL
291 00AF    E4 40          JMP $-2
292 00B0    84 E0          IN  AL,TIMER
293 00B1    EB 00          XCHG AH,AL
294 00B2    84 E0          AH,AL
295 00B3    8B C8          CMP CX,AX
296 00B4    73 0B          JAE TEST_CORD_4
297 00B5    8B D2          PUSH DX
298 00B6    BA FFFF          MOV DX,-1
299
300 00C1    2B 00          SUB DX,AX
301 00C3    03 CA          ADD CX,DX
302 00C5    5A             POP DX
303 00C6    EB 02          JMP SHORT TEST_CORD_5
304
305 00C8    00               ; ADJUST FOR WRAP
306 00C9    81 E1 1FF0          TEST_CORD_5:
307 00CA    81 E1 1FF0          SUB CX,AX
308 00CB    81 E1 1FF0          AND CX,1FF0H
309 00CC    81 E1 1FF0          SHR CX,1
310 00CD    81 E1 1FF0          SHR CX,1
311 00DE    81 E1 1FF0          SHR CX,1
312 00DF    81 E1 1FF0          SHR CX,1
313
314 00D6    TEST_CORD_3:    STI             ; INTERRUPTS BACK ON
315 00D7    FB              MOV DX,201H
316 00D8    00               ; FLUSH OTHER INPUTS
317 00D9    51             PUSH CX
318 00DB    50             PUSH AX
319 00DC    B9 04FF          MOV CX,4FFH
320 00DD    00               ; COUNT
321 00DF    EC              IN  AL,DX

```

```
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
```

```
PAGE 118,121
TITLE POST ----- 01/10/86 SYSTEM POST AND BIOS PROCEDURES

1      PUBLIC A1
2      PUBLIC BEEP
3      PUBLIC CONF_TBL
4      PUBLIC CRT_CHAR_GEN
5      PUBLIC DDS_
6      PUBLIC DSK_BASE
7      PUBLIC M5
8      PUBLIC M6
9      PUBLIC M7
10     PUBLIC MD_TBL_1
11     PUBLIC MD_TBL_2
12     PUBLIC MD_TBL_3
13     PUBLIC MD_TBL_4
14     PUBLIC MD_TBL_5
15     PUBLIC MD_TBL_6
16     PUBLIC P_O_R
17     PUBLIC RESET
18     PUBLIC VIDEO_PARMS
19     PUBLIC WAIT

20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65 0000  CODE SEGMENT BYTE PUBLIC
66
67 0000 1FFF [ CC ]           DB 01FFH DUP (0CCH) ; FILL UNUSED LOCATIONS WITH INTERRUPT 3
68
69
70
71
72 0000           ; ORG 0E000H
73 0000 36 32 58 30 38 35   DB '62X0851 COPR. IBM 1986.' ; COPYRIGHT NOTICE
74 31 20 43 5F 50 52
75 25 20 42 4D 20
76 31 39 38 36
77
78
79
80
81
82
83 0016 00D5 R    C1 DW C11          ; RETURN ADDRESS
84 0018 0181 R    C2 DW C24          ; RETURN ADDRESS FOR DUMMY STACK
85 001A 20 4B 42 20 4F 4B  F3B DB 'KB OK',CR ; KB FOR MEMORY SIZE
86 0D
87
88
89
90
91
92
93
94
95
96
97
98
99
100 MFG_BOOT:
101 0021          CALL SP_TEST        ; GET COUNT LOW
102 0021 E8 19F0 R    MOV BH,BL       ; SAVE IT
103 0024 8A FB
104 0026 E8 19F0 R    CALL SP_TEST        ; GET COUNT HI
105 0028 8A EB
106 0028 8A CF
107 002D FC          CLD
108 002E FA          CLI
109 002F BF 0500      MOV DI,0500H      ; SET TARGET OFFSET (D5=0000)
110 0032 BD FD          MOV AL,0DH       ; UNMASK K/B INTERRUPT
111 0033 E6 11          OUT INT01,AL
112 0034 B0 0A          MOV AL,0AH
113 0038 E6 20          OUT INTA00,AL
114 003A BA 0061          MOV DX,PORT_B ; SET UP PORT B ADDRESS
```



```

228
229
230
231
232
233
234
235
236
237
238
239
240
241 00D7 B0 02      MOV    AL,02H          ;<><><><><><><>
242 00D9 E6 60      OUT    PORT_A,AL   ;<><>CHECKPOINT 2<><>
243 00DB B0 04      MOV    AL,04          ; DISABLE DMA CONTROLLER
244 00D0 E6 08      OUT    DMA08,AL
245
246
247
248 00DF B0 54      MOV    AL,54H          ; SEL TIMER 1,LSB,MODE 2
249 00E1 E6 43      OUT    TIMER+3,AL   ; SET INITIAL TIMER CNT TO 0
250 00E2 B0 C1      MOV    AL,CL          ; TIMER1_BITS_ON
251 00E3 B0 41      OUT    TIMER+1,AL   ; LATCH TIMER 1 COUNT
252 00E7             C12:   MOV    AL,40H          ; YES - SEE IF ALL BITS GO OFF
253 00E7 B0 40      OUT    TIMER+3,AL   ; TIMER1_BITS_OFF
254 00E9 E6 43      CMP    BL,0FFH        ; READ TIMER T COUNT
255 00E8 B0 FF      JE     C13           ; ALL BITS ON IN TIMER
256 00E6 74 07      IN     AL,TIMER+1
257 00E7 40 01      OR    BL,AL          ; TIMER1_BITS_ON
258 00F2 B0 08      LOOP   C12           ; TIMER T FAILURE, HALT SYS
259 00F4 E2 F1      HLT
260 00F6 F4
261 00F7             C13:   MOV    AL,BL          ; TIMER1_BITS_OFF
262 00F7 B0 C3      SUB    CX,CX        ; SET TIMER 1-CNT
263 00F8 28 B0       MOV    CX,CX
264 00F9 B0 41
265 00FD             C14:   MOV    AL,40H          ; TIMER_LOOP
266 00FF B0 40      OUT    TIMER+3,AL   ; LATCH TIMER 1 COUNT
267 00FF E6 43      NOP
268 0101 90         NOP
269 0102 00         NOP
270 0103 E3 41      IN     AL,TIMER+1
271 0105 22 D8      AND    BL,AL          ; WRAP_DMA_REG
272 0107 74 03      JZ    C15           ; TIMER_LOOP
273 0109 E2 F2      LOOP   C14           ; HALT SYSTEM
274 010B F4      HLT
275
276
277
278 010C B0 03      MOV    AL,03H          ;-----INITIALIZE TIMER 1 TO REFRESH MEMORY
279 010E E6 60      OUT    PORT_A,AL   ;<><><><><><>
280
281 0110 E6 0D      OUT    DMA+0DH,AL  ;<><>WRAP DMA CHANNELS ADDRESS AND COUNT REGISTERS
282
283
284
285 0112 B0 FF      MOV    AL,0FFH        ;<><><><><><>
286 0114 B0 D8      MOV    BH,AL          ;<><>SAVE PATTERN FOR COMPARE
287 0116 B0 F8
288 0118 B0 0008
289 011B B0 0000
290 011E EE
291 011F F0
292 0120 E0
293 0121 B0 01
294 0123 E0 C0
295 0124 B0 E0
296 0126 EC
297 0127 3B D8
298 0129 74 01
299 012B 28
300 012B F4
301 012C
302 012C 42
303 012D F9
304 012E E2 EE
305 0130 73 F9
306 0132 FE C0
307 0134 74 DE
308
309
310
311 0136 8E DB      MOV    DS,BX          ;-----INITIALIZE AND START DMA FOR MEMORY REFRESH.
312 0138 BE C3      MOV    ES,BX
313
314 013A B0 FF      ASSUME DS:AB50,ES:AB50
315 013C E6 01      MOV    AL,0FFH        ; SET CNT OF 64K FOR REFRESH
316 013E 50
317 013F E6 01
318 0140 74 B8
319 0143 E6 0B
320 0145 B0 00
321 0147 B0 E8
322 0149 E6 08
323 014B 50
324 014C 0A
325 014E B0 12
326 0150 E6 41
327 0152 B0 41
328 0154 E6 0B
329 0156 50
330 0157 74 08
331 0159 24 10
332 015B 74 01
333 015D F4
334 015E B0 42
335 0160 E6 0B
336 0162 B0 43
337 0164 E6 0B
311 0136 8E DB      MOV    DS,BX          ; SET UP AB50 INTO DS AND ES
312 0138 BE C3      MOV    ES,BX
313
314 013A B0 FF      ASSUME DS:AB50,ES:AB50
315 013C E6 01      MOV    AL,0FFH        ; SET CNT OF 64K FOR REFRESH
316 013E 50
317 013F E6 01
318 0140 74 B8
319 0143 E6 0B
320 0145 B0 00
321 0147 B0 E8
322 0149 E6 08
323 014B 50
324 014C 0A
325 014E B0 12
326 0150 E6 41
327 0152 B0 41
328 0154 E6 0B
329 0156 50
330 0157 74 08
331 0159 24 10
332 015B 74 01
333 015D F4
334 015E B0 42
335 0160 E6 0B
336 0162 B0 43
337 0164 E6 0B
311 0136 8E DB      MOV    DS,BX          ; GET DMA STATUS
312 0138 BE C3      AND    AL,000010000B
313
314 013A B0 FF      ASSUME DS:AB50,ES:AB50
315 013C E6 01      JZ    C18C           ; 15-TIMER REQUEST THERE?
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
311 0136 8E DB      MOV    DS,BX          ; (IT SHOULDN'T BE)
312 0138 BE C3      JZ    C18C           ; HALT SYS.(HOT TIMER 1 OUTPUT)
313
314 013A B0 FF      ASSUME DS:AB50,ES:AB50
315 013C E6 01      HLT
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
311 0136 8E DB      MOV    DS,BX          ; SET MODE FOR CHANNEL 2
312 0138 BE C3      OUT    DMA+OBH,AL
313
314 013A B0 FF      ASSUME DS:AB50,ES:AB50
315 013C E6 01      HLT
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
311 0136 8E DB      MOV    DS,BX          ; SET MODE FOR CHANNEL 3
312 0138 BE C3      OUT    DMA+OBH,AL

```

```

338
339
340
341
342
343
344
345
346
347 0166 AD LODSW I ALLOW RAM CHARGE TIME.
348 0167 AD LODSW
349 0168 AD LODSW
350 0169 AD LODSW
351
352
353 :---- DETERMINE MEMORY SIZE AND FILL MEMORY WITH DATA
354 016A BB IE 0472 R MOV BX,DATA_WORD[RESET_FLAG-DATA40] I SAVE 'RESET_FLAG' IN BX
355 016B 80 2E 0496 R MOV DX,DATA_WORD[KB_FLAG-DATA40] I SAVE KEYBOARD TYPE
356 0172 B9 00 0000H MOV CX,0000H
357 0175 11 FB 1234 CMP BX,1234H
358 0179 74 16 JE CLR_STG I WARM START?
359 017B BC 0018 R MOV SP,OFFSET C2
360 017E E9 0CCF R JMP STGST_CNT
361 0181 70 12 C24: JE HOW_BIG I STORAGE OK, DETERMINE SIZE
362 0182 B3 AA 05 MOV BL,BX I SAVE FAILING BIT PATTERN
363 0185 B0 04 MOV AX,04H I <-><-><-><-><-><->
364 0187 E6 60 C24A: OUT PORT_A,AL I <-><->CHECKPOINT <-><->
365 0189 2B C9 SUB CX,CX I BASE RAM FAILURE - HANG
366 018B E2 FE C24B: LOOP C24B I FLIPPING BETWEEN 04 AND
367 018D 86 D8 XCHG BL,AL I FAILING BIT PATTERN
368 018F EB F6 JMP C24A
369
370 0191 2B C0 CLR_STG: SUB AX,AX I MAKE AX=0000
371 0193 F3/ AB REP STOSW I STORE 32K WORDS OF 0000
372 0195 HOW_BIG: MOV DATA_WORD[RESET_FLAG-DATA40],BX I RESTORE RESET FLAG
373 0195 89 IE 0472 R MOV BP,BX I IS THE KBP DIR THE ONLY ONE ON
374 0196 03 FD 10 JE C24C I NOT THEN THIS MUST BE A P.O.R.
375 019C 74 02 SUB BP,BP I IF P.O.R. THEN INITIALIZE THIS TO ZERO
376 019E 2B ED
377 01A0
378 01A0 89 2E 0496 R C24C: MOV DATA_WORD[KB_FLAG_3-DATA40],BP I RESTORE RESET FLAG
379 01A4 2B ED SUB BP,BP I BP IS USED LATER AS AN ERROR INDICATOR
380 01A8 80 0000H MOV DX,0400H I SET POINTER TO JUST>16KB
381 01A9 BB 0010 MOV BX,16 I BASIC COUNT OF 16K
382 01AC FILL_LOOP: REP STOSW I SET COUNT FOR 8K WORDS
383 01AC 8E C2 MOV ES,DX I SET SEG. REG.
384 01AE 2B FF SUB DI,DI I TEST PATTERN
385 01B0 BB AA55 MOV AX,0AA55H I SAVE PATTERN
386 01B3 80 0000H MOV CX,0000H I SEND PATTERN TO MEM.
387 01B5 26 89 05 MOV ES:[DI],AX I PUT SOMETHING IN AL
388 01B8 B0 0F MOV AL,0FH I GET PATTERN
389 01B8 26 8B 05 MOV AX,ES:[DI]
390 01BD 33 C1 XOR AX,CX I COMPARE PATTERNS
391 01BF 75 11 JNZ HOW_BIG_END I GO END IF NO COMPARE
392 01C0 00000000 MOV CX,2000H I SET COUNT FOR 8K WORDS
393 01C4 F3/ AB REP STOSW I FILL 8K WORDS
394 01C6 81 C2 0400 ADD DX,400H I POINT TO NEXT 16KB BLOCK
395 01CA 83 C3 10 ADD BX,16 I BUMP COUNT BY 16KB
396 01CD 80 FE A0 CMP DH,0A0H I TOP OF RAM AREA YET? (A0000)
397 01D0 75 DA JNZ FILL_LOOP
398 01D2 0000 H
399 01D2 89 IE 0413 R HOW_BIG_END: MOV DATA_WORD[MEMORY_SIZE-DATA40],BX I SAVE MEMORY SIZE
400
401
402 :---- SETUP STACK SEG AND SP
403 01D6 BB 0030 MOV AX,STACK_SS I GET STACK VALUE
404 01D9 8E D0 MOV SS,AX I SET THE STACK UP
405 01DB BC 0100 MOV SP,TOS I STACK IS READY TO GO
406
407
408 :---- INITIALIZE THE 8259 INTERRUPT CONTROLLER CHIP :
409 01DE B0 13 C25: MOV AL,13H I ICW1 - EDGE, SNGL, ICW4
410 01E0 B6 20 OUT INTA00,AL I
411 01E2 B0 08 MOV AL,00H I SETUP ICW2 - INT TYPE 8 (8-F)
412 01E4 E4 E2 OUT INTA01,AL I
413 01E6 B0 09 MOV AL,9 I SETUP ICW4 - BUFFRD,8086 MODE
414 01E8 B6 E2 OUT INTA01,AL I
415 01EA B0 FF MOV AL,0FFH I MASK ALL INTS. OFF
416 01EC E6 21 OUT INTA01,AL I (VIDEO ROUTINE ENABLES INTS.)
417
418
419 :---- SET UP THE INTERRUPT VECTORS TO TEMP INTERRUPT
420 01EE IE PUSH DS I FILL_ALL_32_INTERRUPTS
421 01EF BB 0020 MOV CX,32 I FIRST INTERRUPT LOCATION
422 01F0 80 FF PUSH DI,I I SET ES:0000 ALSO
423 01F4 BE C7 D31: MOV DS,DI,I
424 01F6 BB 1F23 R MOV AX,OFFSET DII I MOVE ADDR OF INTR PROC TO TBL
425 01F9 AB STOSW I
426 01FC 8C C8 MOV AX,CS I GET ADDR OF INTR PROC SEG
427 01FD AB STOSW I
428 01FD E2 F7 LOOP D3 I VECTBLO
429
430
431 :---- ESTABLISH BIOS SUBROUTINE CALL INTERRUPT VECTORS
432 01FF BF 0040 R D3A: MOV DS,DI,I
433 0202 0E PUSH CS I SETUP ADDR TO INTR AREA
434 0204 0F POP DS I
435 0204 BE 1F03 R MOV DS,DI,I
436 0207 BB 0010 MOV SI,OFFSET VECTOR_TABLE+16 I SETUP ADDR OF VECTOR TABLE
437 020A A5 MOV CX,16 I START WITH VIDEO ENTRY
438 020B 47 INC DI,I
439 020C 47 INC DI,I
440 020D E2 FB LOOP D3A I MOVE VECTOR TABLE TO RAM
441
442
443 :---- DETERMINE CONFIGURATION AND MFG. MODE :
444
445 020F 1F POP DS I RECOVER DATA SEG
446 0210 IE PUSH DS I
447 0211 E4 62 IN AL,PORT_C I GET SWITCH INFO
448 0213 24 0F AND AL,0000111B I ISOLATE SWITCHES
449 0215 8A E0 MOV AH,AL I SAVE
450 0217 BB AD MOV AL,10101010B I ENABLE OTHER BANK OF SWS.
451 0219 E6 61 OUT PORT_B,AL

```

```

452 021B 90          NOP
453 021C E4 62          IN    AL,PORT_C
454 021E B1 04          MOV   CL,4
455 0220 D2 C0          ROL   AL,CL
456 0222 24 F0          AND   AL,1111000B      ; ROTATE TO HIGH NIBBLE
457 0224 0A C4          OR    AL,AH      ; ISOLATE
458 0226 0A C4          SUB   AH,AH      ; COMBINE WITH OTHER BANK
459 0228 80 99          MOV   BX,WORD[@EQUIP_FLAG-DATA40],AX      ; SAVE SWITCH INFO
460 022B 80 99          MOV   AL,99H
461 022D E6 63          OUT   CM0_PORT,AL
462 022E E8 19E3 R      CALL  KBD_RESET
463 0232 80 FB EA      CMP   BL,0EAH      ; SEE IF MFG. JUMPER IN
464 0234 75 08          JNE   KBX1      ; IS THIS THE EXTENDED KEYBOARD?
465 0237 C6 06 0496 R 10  MOV   DATA_AREA[@KB_FLAG_3-DATA40],KBX1      ; IF NOT THEN LEAVE THE FLAG ALONE
466 023C EB 22 90          JMP   E6      ; EXTENDED KEYBOARD
467 023F
468 023F 80 FB AA      KBX1:   CMP   BL,0AAH      ; KEYBOARD PRESENT?
469 0242 74 1C          JE    E6      ; YES
470 0244 80 FB 65          CMP   BL,065H      ; LOAD MFG. TEST REQUEST?
471 0247 74 03          JNE   D3B      ; NO
472 0249 E9 0021 R      JMP   MFG_BOOT      ; GO TO BOOTSTRAP IF SO
473 024C
474 024C 0A DB          D3B:   OR    BL,BL      ; MFG PLUG IN?
475 024C 75 10          JNZ   E6      ; YES
476 0251 0A 38          MOV   AL,38H
477 0252 E6 61          OUT   PORT_AL,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      ; NO
483 025C FE 06 0412 R  INC   DATA_AREA[@MFG_TST-DATA40]      ; SET MANUFACTURING TEST FLAG
484
485
486
487
488
489
490
491
492
493
494
495 0260
496 0260 A1 0410 R      E0:    MOV   AX,DATA_WORD[@EQUIP_FLAG-DATA40]      ; INITIALIZE AND START CRT CONTROLLER (6845)
497 0263 50          PUSH  AX      ; GET SENSE SWITCH INFO
498 0264 B0 30          MOV   AL,30H      ; SAVE IT
499 0266 A3 0410 R      MOV   DATA_WORD[@EQUIP_FLAG-DATA40],AX
500 0268 2A E4          SUB   AH,AR
501 026B 75 00          INT   10H      ; SEND INIT TO B/W CARD
502 026D B0 20          MOV   AL,20H
503 026F A3 0410 R      MOV   DATA_WORD[@EQUIP_FLAG-DATA40],AX
504 0272 2A E4          SUB   AH,AR      ; 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[@EQUIP_FLAG-DATA40],AX      ; RESTORE IT
508
509 027A 24 30          AND   AL,30H      ; AND CONTINUE
510 027C 75 0A          JNZ   ET      ; ISOLATE VIDEO SWS
511 027E BF 0040 R      MOV   DI,OFFSET @VIDEOINT      ; VIDEO SWS SET TO 0?
512 0281 C7 05 1F49 R 503 0281 E0 03BB R      MOV   WORD PTR [DI],OFFSET DUMMY_RETURN      ; SET INT 10H TO DUMMY
513 0285 E9 03BB R      JMP   E18_1      ; RET IF NO VIDEO CARD
514 0288
515 0288 3C 30          ET:    CMP   AL,30H      ; BYPASS VIDEO TEST
516 028A 74 08          JE    E8      ; IF NO VIDEO
517 028C FE C4          INC   AH      ; B/W CARD ATTACHED?
518 028E 3C 20          CMP   AL,20H      ; YES - SET MODE FOR B/W CARD
519 0291 0A 02          JNE   E8      ; SET COLOR MODE FOR COLOR CD
520 0292 B4 03          MOV   AH,3      ; 80X25 MODE SELECTED?
521 0294 86 E0          XCHG  AH,AL      ; NO - SET MODE FOR 40X25
522 0296 0A 02          MOV   AH,3      ; SET MODE FOR 80X25
523 0297 2A E4          E8:    PUSH  AX      ; SET MODE
524 0299 CD 10          SUB   AH,AH      ; SAVE VIDEO MODE ON STACK
525 029B 58          INT   10H      ; INITIALIZE TO ALPHANUMERIC MD
526 029C 00          POP   AX      ; CALL VIDEO IO
527 029D BB B000          PUSH  AX      ; RESTORE VIDEO SENSE SWS IN AH
528 02A0 EB 24          MOV   BX,0B000H      ; RESAVE VALUE
529
530
531
532
533 02C3
534 02C3
535 02C3 E9 185C R      NMI_INT:  JMP   NMI_INT_I      ;----- UNNATURAL ACT FOR ADDRESS COMPATIBILITY
536
537 02C6
538 02C6 BA 03BB
539 02C9 B9 0800
540 02CC B0 01
541 02CE 80 FC 30
542 02D1 74 09
543 02D3 74 09
544 02D5 BA 03D8
545 02D8 BB 20
546 02DA FE C8
547 02DC
548 02DC EE
549 02D9 74 0E 0472 R 1234
550 02E3 8E C3
551 02E5 74 07
552 02E7 8E DB
553
554 02E9 E9 0CCF R
555 02EC 75 33
556
557
558
559
560
561
562
563 02EE
564 02EE 58
565 02EF 50
      E8A:   MOV   DX,3B8H      ; MODE REG FOR B/W
      MOV   CX,2048      ; RAM WORD CNT FOR B/W CD
      MOV   AL,1      ; SET MODE FOR BW CARD
      CMP   AH,30H      ; B/W VIDEO CARD ATTACHED?
      JE    E9      ; YES - GO TEST VIDEO STG
      INT   1234H      ; BEG VIDEO RAM ADDR COLOR CD
      MOV   DX,0B8H      ; MODE REG FOR COLOR CD
      MOV   DS,BX      ; RAM WORD CNT FOR COLOR CD
      CH,20H
      DEC   AL      ; SET MODE TO 0 FOR COLOR CD
      E9:    OUT   DX,AL      ; TEST VIDEO STG;
      CMP   DATA_WORD[@RESET_FLAG-DATA40],1234H      ; DISABLE VIDEO FOR COLOR CD
      MOV   ES,DX      ; POINT ES TO COLOR RAM STG
      JE    E10      ; YES - SET COLOR RAM TEST
      MOV   DS,BX      ; POINT DS TO VIDEO RAM STG
      ASSUME DS:NOTHING,ES:NOTHING      ; GO TEST VIDEO R/W STG
      CALL  STGST_CNT      ; R/W STG FAILURE - BEEP SPK
      JNE   E17
      E10:   OUT   DX,AL      ;----- SETUP VIDEO DATA ON SCREEN FOR VIDEO
      LINE_TEST
      DESCRIPTION
      ENABLE VIDEO SIGNAL AND SET MODE
      DISPLAY A HORIZONTAL BAR ON SCREEN
      E10:   POP   AX      ; GET VIDEO SENSE SWS (AH)
      PUSH  AX      ; SAVE IT

```

```

566 02F0 B4 00      MOV    AH,0          ; ENABLE VIDEO AND SET MODE
567 02F2 CD 10      INT    10H          ; VIDEO
568 02F4 B0 7020    MOV    AX,7020H     ; WR BLANKS IN REVERSE VIDEO
569
570 02F7 2B FF      SUB    D1,D1        ; 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 :----- CRT INTERFACE LINES TEST
576 : DESCRIPTION
577 : SENSE ON/OFF TRANSITION OF THE
578 : VIDEO ENABLE AND HORIZONTAL
579 : SYNC LINES.
580
581 02FE 58          POP    AX          ; GET VIDEO SENSE SW INFO
582 02F2 50          PUSH   AX          ; SAVE IT
583 0300 80 FC 30    CMP    AH,30H      ; B/W CARD ATTACHED?
584 0303 BA 03BA    MOV    DX,03BAH    ; SETUP ADDR OF BW STATUS PORT
585 0306 74 03      JE    E11         ; YES - GO TEST LINES
586 0308 BA 03DA    MOV    DX,03DAH    ; COLOR CARD IS ATTACHED
587 030B B4 08      MOV    AH,8          ; LINE_TST1
588 030D             SUB    CX,CX      ; OFLOOP_CNT1
589 030D 2B C9      E11:  MOV    AH,8          ; READ CRT STATUS PORT
590 030F             E12:  SUB    CX,CX      ; CHECK VIDEO/HORZ LINE
591 0310 22 C4      E13:  IN    AL,DX        ; ITS ON - CHECK IF IT GOES OFF
592 0312 75 04      JNZ    E14         ; LOOP TILL ON OR TIMEOUT
593 0314 E2 F9      LOOP   E13         ; GO PRINT ERROR MSG
594 0316 EB 09      JMP    SHORT E17
595 0318
596 0319 2B C9      E14:  POP    DS          ; READ CRT STATUS PORT
597 031A             E15:  IN    AL,DX        ; CHECK VIDEO/HORZ LINE
598 031B 22 C4      AND    AL,AH      ; ITS ON - CHECK NEXT LINE
599 031C             JZ    E16         ; LOOP IF OFF TILL IT GOES ON
600 031D T4 11      LOOP   E15         ; CRT_ERR1
601 031F E2 F9      E17:  POP    DS          ; <><><>CRT ERR CHKPT. 06<><><>
602 0320             CALL   ERR_BEEP    ; GO BEEP SPEAKER
603 0321 EB 06      JMP    SHORT E18
604
605 0322 IE          E18:  POP    DS          ; NXT_LINE1
606 0323 CE 06 0015 R 06  IN    AL,DX        ; GET NEXT BIT TO CHECK
607 0328 BA 0102    AND    AL,AH      ; GO CHECK HORIZONTAL LINE
608 032B E8 19A5 R  CALL   INT10H      ; DISPLAY CURSOR
609 032E EB 06      INT    10H         ; GET VIDEO SENSE SWS (AH)
610 0330             E18:  MOV    CL,3          ; SET MODE AND DISPLAY CURSOR
611 0330 B1 03      SHR    AH,CL      ; CALL VIDEO I/O PROCEDURE
612 0332 D2 EC      JNZ    E12         ; IS PRESENT
613 0334 75 D7      E18:  POP    AX          ; GET FIRST 2 LOCATIONS
614 0336
615 0337 B4 08      E18:  MOV    AH,0          ; LET BUS SETTLE
616 0337 B4 00      INT    10H         ; PRESENT?
617 0339 CD 10      CALL   ROM_CHECK    ; NOT GO LOOK FOR OTHER MODULES
618 033B             JMP    SHORT E18C  ; GO SCAN MODULE
619 033B BA C000    E18A: MOV    DX,0C000H    ; SEE IF ADVANCED VIDEO CARD
620 033E
621 033E 8E DA      E18A: MOV    DS,DX        ; IS PRESENT
622 0340 2B DB      SUB    BX,BX      ; GET FIRST 2 LOCATIONS
623 0342 8B 07      MOV    AX,[BX]     ; LET BUS SETTLE
624 0344 53          PUSH   BX          ; PRESENT?
625 0345 5B          POP    BX          ; NOT GO LOOK FOR OTHER MODULES
626 0346 3D AA55    CMP    AX,0AA55H    ; GO SCAN MODULE
627 0347 75 D7      JNZ    E18B      ; READ IMR
628 0349 E8 1920 R  CALL   ROM_CHECK    ; ALL IMR BIT ON?
629 034E EB 04      JMP    SHORT E18C  ; NO - GO TO ERR ROUTINE
630 0350
631 0350 B1 C2 0080  E18B: ADD   DX,0080H    ; POINT TO NEXT 2K BLOCK
632 0354
633 0354 81 FA C800  E18C: CMP   DX,0C800H    ; TOP OF VIDEO ROM AREA YET?
634 0358 TC E4      JL    E18A         ; GO SCAN FOR ANOTHER MODULE
635
636
637 :----- 8259 INTERRUPT CONTROLLER TEST
638 : DESCRIPTION
639 : READ/WRITE THE INTERRUPT MASK REGISTER (IMR)
640 : WHICH DETERMINES AND ENFORCES WHICH SYSTEM
641 : INTERRUPTS CAN OCCUR. SETTING THE IMR TO 0
642 : FOR HOT INTERRUPTS (UNEXPECTED).
643
644 035A 1F          C21: ASSUME DS:AB50
645             POP   DS
646
647 :----- TEST THE IMR REGISTER
648 035B CE 06 0415 R 05  C21A: MOV    DATA_AREA[*MFG_ERR_FLAG-DATA40],05H
649
650
651 0360 B0 00      MOV    AL,0          ; <><><><><><><><><>
652 0362 6 21       OUT   INTA01,AL    ; <><><>CHECKPOINT 5<><>
653 0364 E4 21      IN    AL,INTA01    ; SET IMR TO ZERO
654 0366 0A C0      OR    AL,AL
655 0368 75 1B      JNZ    D6          ; READ IMR
656 036A B0 FF      MOV    AL,0FFH     ; IMR = 0?
657 036C E6 21      OUT   INTA01,AL    ; GO TO ERR ROUTINE IF NOT 0
658 036E 6 21       INT    AL,INTA01    ; DISABLE DEVICE INTERRUPTS
659 0370 04 01      ADD    AL,1          ; WRITE TO IMR
660 0372 75 11      JNZ    D6          ; READ IMR
661
662
663
664
665 :----- CHECK FOR HOT INTERRUPTS
666 0374 A2 046B R  MOV    DATA_AREA[*INTR_FLAG-DATA40],AL ; CLEAR INTERRUPT FLAG
667 0377 FB          STI    CX,CX      ; ENABLE EXTERNAL INTERRUPTS
668 0378 2B C9      SUB    CX,CX      ; WAIT 1 SEC FOR ANY INTRS THAT
669 0379             D4:  LOOP   D4          ; MIGHT OCCUR
670 037A E2 FE      D5:  LOOP   D5          ; ANY INTERRUPTS OCCUR?
671 037C             D5:  CMP    DATA_AREA[*INTR_FLAG-DATA40],00H ; NO - GO TO NEXT TEST
672 037C E2 FE      D6:  MOV    S1,OFFSET E0      ; DISPLAY 101 ERROR
673 037E 80 3E 046B R  CALL   E_MSG
674 0383 74 08      CLI    HLT         ; HALT THE SYSTEM
675
676 0385 BE 18CC R
677 0388 E8 1976 R
678 038B FA
679 038C F4

```

680 PAGE
681 :-----
682 : 8253 TIMER CHECKOUT
683 : DESCRIPTION
684 : VERIFY THAT THE SYSTEM TIMER (0) DOESN'T COUNT
685 : TOO FAST OR TOO SLOW.
686 :-----

687 038D D7: MOV DATA_AREA[0MFQ_ERR_FLAG-DATA40],02H
688 038D C6 06 0415 R 02
689 OUT AL,0FEH
690 INTA01,AL
691 MOV AL,00010000B
692 OUT TIM_CTL,AL
693 MOV CX,T6H
694 OUT AL,CL
695 MOV TIMER0,AL
696 OUT AL,0CL
697 MOV AL,0FFH
698 INTA01,AL
699 TEST DATA_AREA[0INTR_FLAG-DATA40],01H
700 JNZ D9 : DID TIMER 0 INTERRUPT OCCUR?
701 03AC T5 04 : YES - CHECK TIMER FOR SLOW TIME
702 03AC E2 F7 : LOOP D8 : WAIT FOR INTR FOR SPECIFIED TIME
703 03AC EB D9 : JMP D6 : TIMER 0 INTR DIDN'T OCCUR - ERR
704 03AC :-----
705 03AC B1 0C MOV CL,12 : SET PGM LOOP CNT
706 03AE B0 FF MOV AL,0FFH : WRITE TIMER 0 CNT REG
707 03B0 E6 40 OUT TIMER0,AL
708 03B2 C6 06 046B R 00 MOV DATA_AREA[0INTR_FLAG-DATA40],0 : RESET INTR RECEIVED FLAG
709 03B3 B0 FE MOV AL,0FEH : REENABLE TIMER 0 INTERRUPTS
710 03B9 E6 21 OUT INTA01,AL
711 03BB :-----
712 03B8 F6 06 046B R 01 TEST DATA_AREA[0INTR_FLAG-DATA40],01H : TIMER 0 INTERRUPT OCCUR?
713 03C0 T5 C3 JNZ D6 : YES - TIMER CNTING TOO FAST, ERR
714 03C2 E2 F7 LOOP D10 : WAIT FOR INTR FOR SPECIFIED TIME
715 :-----
716 :-----
717 :-----
718 03C4 B0 FF MOV AL,0FFH : DISABLE ALL DEVICE INTERRUPTS
719 03C5 E6 21 OUT INTA01,AL
720 03C8 B0 36 MOV AL,36H : SEL TIM 0,LSB,MSB,MODE 3
721 03C9 B0 43 OUT PORTR+3,AL : WRITE TIMER MODE REG
722 03CC B0 00 MOV AL,0
723 03CE E6 40 OUT TIMER_AL : WRITE LSB TO TIMER 0 REG
724 03D0 E6 40 OUT TIMER_AL : WRITE MSB TO TIMER 0 REG
725 :-----
726 :-----
727 :-----
728 :-----
729 :-----
730 :-----
731 :-----
732 03D2 TST1: MOV AL,99H : SET 8255 MODE A,C=IN B=OUT
733 03D2 B0 99 OUT CMD_PORT_AL
734 03D4 E6 63 MOV AL,DATA_AREA[0EQUIP_FLAG-DATA40]
735 03D6 A0 0410 R AND AL,01 : TEST CHAMBER?
736 03D8 24 01 JZ F7 : BYPASS IF SO
737 03D8 T4 30 CMP DATA_AREA[0MFQ_TST-DATA40],0 : MANUFACTURING TEST MODE?
738 03D8 B0 3E 0412 R 01 JE F7 : YES - SKIP KEYBOARD TEST
739 03D8 24 01 CALL KBD_RESET : ISSUE RESET TO KEYBOARD
740 03E4 E8 19E3 R JCXZ F6 : PRINT ERR MSG IF NO INTERRUPT
741 03E7 E3 IE MOV AL,49H : ENABLE KEYBOARD
742 03E9 B0 49 OUT PORT_B,AL
743 03EB E6 61 CMP BL,0AAH : SCAN CODE AS EXPECTED?
744 03EB B0 FB AA JNE F6 : NO - DISPLAY ERROR MSG
745 03F0 T5 15 :-----
746 :-----
747 :-----
748 :-----
749 03F2 B0 C8 MOV AL,OCBH : CLR KBD, SET CLK LINE HIGH
750 03F3 E6 61 OUT PORT_B,AL
751 03F5 B0 48 MOV AL,4BH : ENABLE KBD,CLK IN NEXT BYTE
752 03F8 E6 41 OUT PORT_B,AL
753 03FA 2B C9 SUB CX,CX
754 03FC :-----
755 03FC 2B FE F5: PUSH DS : KBD_WAIT:
756 03FE E4 60 LOOP F5 : DELAY FOR A WHILE
757 0400 00 00 IN AL,KB_DATA : CHECK FOR STUCK KEYS
758 0402 T4 09 CMP AL,0 : SCAN CODE = 0?
759 0404 E8 1958 R JE F7 : YES - CONTINUE TESTING
760 0407 CALL CPC_BYTEC : CONVERT AND PRINT
761 0407 BE 098A R MOV SI,OFFSET F1 : GET MSG ADDR
762 040A E8 1976 R CALL MSG : PRINT MSG ON SCREEN
763 :-----
764 :-----
765 040D F7: SETUP_HARDWARE_INT_VECTOR_TABLE :-----
766 040D IE PUSH DS : SETUP_INT_TABLE:
767 040E 2B C0 SUB AX,AX
768 040F 2B C0 MOV ES,AX
769 0412 B9 0008 MOV CS,08 : GET VECTOR CNT
770 0415 0E PUSH CS : SETUP DS SEG REG
771 0416 1F POP DS
772 0417 BE IEF3 R MOV SI,OFFSET VECTOR_TABLE
773 0418 BF 0020 R MOV DI,OFFSET *INT_PTR
774 041D 00 F7A: MOVSW
775 041D A5 INC DI : SKIP OVER SEGMENT
776 041E 47 INC DI
777 041F 47 LOOP F7A
778 0420 E2 FB POP DS
780 0422 1F :-----
781 :-----
782 :-----
783 :-----
784 0423 C7 06 0008 R 02C3 R MOV WORD PTR *NM1_PTR,OFFSET NM1_INT : NMI INTERRUPT
785 0429 C7 06 0014 R 1F54 R MOV WORD PTR *INT5_PTR,OFFSET PRINT_SCREEN : PRINT SCREEN
786 042F C7 06 0062 R F600 MOV WORD PTR *BASIC_PTR+2,0F600H : SEGMENT FOR CASSETTE BASIC
787 0435 C7 06 007E R 0000 MOV WORD PTR *EXT_PTR+2,0000 :SEGMENT FOR VIDEO EXTENSION
788 :-----
789 :-----
790 :-----
791 043B B0 3E 0412 R 01 CMP DATA_AREA[0MFQ_TST-DATA40],01H : MFG. TEST MODE?
792 0440 B5 DA EXP_TO
793 0442 C7 06 0070 1909 R MOV WORD PTR DS1[ICH*4],OFFSET BLINK_INT : SETUP TIMER TO BLINK LED

```

794 0448 BB FE          MOV    AL,0FEH           ; ENABLE TIMER INTERRUPT
795 044A E6 21          OUT   INTA01,AL
796
797
798
799
800
801
802
803
804
805 044C               MOV    DX,0210H          ; (CARD WAS ENABLED EARLIER)
806 044C BA 0210          MOV    AX,5555H          ; CONTROL PORT ADDRESS
807 044C BB 5555          OUT   DX,AH
808 0452 F0 00             MOV    AL,01H           ; SET DATA PATTERN
809 0453 B0 01             IN    AL,DX
810 0455 EC               CMP    AL,AH
811 0456 3A C4             JNE   E19
812 0458 75 43             NOT   AX
813 0459 F0 D0             OUT   DX,AL
814 045C EC               MOV    AL,01H           ; MAKE AL DIFFERENT
815 045D B0 01             IN    AL,DX
816 045F EC               CMP    AL,AH           ; RECOVER DATA
817 0460 3A C4             JNE   E19
818 0462 75 39             NOT   AX
819
820
821
822 0464               EXP_10:  MOV    BX,0001H          ; REPLY?
823 0464 BB 0001          MOV    DX,0215H          ; NO RESPONSE, GO TO NEXT TEST
824 0467 BA 0215          MOV    CX,0016
825 0468 B9 0010
826 046D
827 046D 2E 88 07          EXP_2:  MOV    CS:[BX],AL          ; MAKE DATA=AAAA
828 0470 90
829 0471 EC               NOP
830 0472 3A C7             CMP    AL,BH
831 0473 00 21             INC   DX
832 0476 42
833 0477 EC               JNE   EXP_ERR          ; GO ERROR IF MISCOMPARE
834 0478 3A C3             INC   DX
835 047A 75 1B             CMP    AL,BL
836 047C 4A               JNE   EXP_ERR          ; DX=216H (ADDR. LOW REG)
837 047D D1 E3             DEC   DX
838 047F E2 EC             SHL   BX,1
839
840
841
842 0481 B9 0008          EXP_3:  MOV    CS:[BX],AL          ; WRITE ADDRESS F000+BX
843 0484 B0 01             NOP
844 0486 4A               DEC   DX
845 0487
846 0487 8A E0             EXP_4:  MOV    AH,AL           ; READ ADDR. HIGH
847 0489 EE               OUT   DX,AL
848 048A B0 01             MOV    AL,01H
849 048B EC               IN    AL,DX
850 048D 3A C4             CMP    AL,AH
851 048F 75 06             JNE   EXP_ERR          ; COMPARE TO LOW ADDRESS
852 0491 D0 E0             INC   DX
853 0493 E2 F2             SHL   AL,1
854 0495 EB 06             LOOP  EXP_3
855 0497
856 0497 BE 1BDC R         EXP_ERR: MOV    SI,[OFFSET F3C          ; DX BACK TO 215H
857 049A EB 1976 R         CALL  E_MSC            ; LOOP TILL BIT WALKS ACROSS AL
858
859
860
861
862
863
864
865
866 049D
867 049D E8 1A12 R         EXP_5: ASSUME DS:DATA          ; GO ON TO NEXT TEST
868 04A0 1E               PUSH  DS
869 04A1
870 04A1 81 3E 0072 R 1234          E19:  CALL  DDS
871 04A4 75 03             E20:  CMP   $RESET_FLAG,1234H      ; WRITE/READ DATA PATTERNS TO ANY READ/WRITE
872 04A9 E9 054A R         JNE   E20A
873 04A9
874 04AC BB 0040          E20A:  JMP   ROM_SCAN          ; CONTINUE TEST IF NOT
875 04AF EB 28             E20B:  MOV   AX,64           ; GO TO NEXT ROUTINE IF SO
876 04B1
877 04B1 BB 1E 0013 R         E21:  MOV   BX,$MEMORY_SIZE      ; STARTING AMT. OF MEMORY OK
878 04B5 B3 40             ADD   BX,4
879 04B8 B3 40             SUB   BX,64
880 04B9 D3 EB             MOV   BX,CL
881 04BC BB CB             SHR   BX,CL
882 04BE BB 1000          MOV   BX,1000H          ; DIVIDE BY 16
883 04C1
884 04C1 8E DB             PRT_SIZ: MOV   DS,REG
885 04C5 B3 C3             ADD   ES,BX
886 04C5 B1 C3 0400          PUSH  BX,0400H          ; POINT TO NEXT 16K
887 04C9 52
888 04CA 51
889 04CB 53
890 04CC 50
891 04D0 88 2000          PUSH  DX
892 04D0 E8 0CCF R         PUSH  CX
893 04D3 75 4C             PUSH  BX
894 04D5 58
895 04D6 05 0010          PUSH  BX,1000H          ; SAVE COUNT OF 16K BLOCKS
896 04D9
897 04D9 50
898 04DA BB 000A          ADD   AX,16
899 04DD B9 0003          PRT_DEC_LOOP: MOV   BX,10           ; SET PTR. TO RAM SEGMENT>64K
900 04E0
901 04E0 33 D2             PUSH  CX,3
902 04E2 77 F3             DECIMAL_LOOP: XOR   DX,DX
903 04E4 B8 CA 30             DIV   BX
904 04E7 52
905 04E8 E2 F6             OR    DX,30H
906 04EA B9 0003             PUSH  DX
907 04ED

```

```

908 04ED 5B          POP    AX      ; RECOVER A NUMBER
909 04E8 E8 1969 R   CALL   PRT_HEX
910 04F1 E2 FA       LOOP   PRT_DEC_LOOP
911 04F3 B9 0007     MOV    CX,7
912 04F4 8D 00 01A    MOV    SI,OFFSET F3B      ; PRINT 'KB OK'
913 04F9
KB_LOOP:           MOV    AL,CS:[SI]
914 04F9 2E 8A 04     INC    SI
915 04FC 46          CALL   PRT_HEX
916 04F9 E8 1969 R   LOOP   KB_LOOP
917 0500 E2 F7       MOV    AX,58
918 0500 58          CMP    AX,58
919 0503 00 0040     JE    E20B      ; RECOVER WORK REGS
920 0506 74 A9       POP    BX
921 0509 5B          POP    CX
922 0509 59          POP    DX
923 050A 5A          LOOP   E21      ; FIRST PASS?
924 050B E2 B4       MOV    AL,10
925 050C 8D 00 01A    CALL   PRT_HEX      ; LOOP TILL ALL MEM. CHECKED
926 050F E8 1969 R   CALL   PRT_HEX      ; LINE FEED
927
928
;----- DMA TCO SHOULD BE ON BY NOW - SEE IF IT IS
929 0512 E4 08          IN     AL,DMA+08H
930 0512 81 00 01       AND   AL,00000001B      ; TCO STATUS BIT ON?
931 0512 81 00 02       JNZ   ROM_SCAN
932 0516 75 32          CALL   PRT_SCAN
933 0518 1F             DS
934 0519 C6 00 0015 R 03  POP   DS
935 0519 E8 1965 R   MOV    #MFG_ERR_FLAG,03H      ; <><><><><><><><>
936 051E E9 0385 R   JMP   D6      ; POST 101 ERROR MSG AND HALT
937
938
;----- PRINT FAILING ADDRESS AND XOR'ED PATTERN IF DATA COMPARE ERROR
939 0521 8A E8          E21A: MOV    CH,AL      ; SAVE FAILING BIT PATTERN
940 0523 B0 0D          MOV    AL,CR      ; CARRAGE RETURN
941 0525 E8 1969 R   CALL   PRT_HEX
942 0526 B0 0A          MOV    AL,_LF      ; LINE FEED
943 0528 E8 1969 R   CALL   PRT_HEX
944 052D 8D 00 01A    POP   AX
945 052E B2 C4 06     ADD   SP,4      ; RECOVER AMT. OF GOOD MEM.
946 0531 8C DA          MOV    DX,DS      ; BALANCE STACK
947 0533 1F             POP   DS
948 0534 1E             PUSH  DS
949 053A A3 0013 R   MOV    #MEMORY_SIZE,AX      ; LOAD MEM. SIZE WORD TO SHOW
950
951 0538 88 36 0015 R  MOV    #MFG_ERR_FLAG,DH      ; HOW MUCH MEM. WORKING
952
953 053C E8 OCBA R   CALL   PRT_SEG      ; <><><><><><><><><><>
954 053E 8A C5          MOV    AL,_CH      ; PRINT IT
955 0541 E8 1958 R   CALL   XPC_BYTEx      ; GET FAILING BIT PATTERN
956 0544 BE 18D1 R   MOV    SI,OFFSET E1      ; CONVERT AND PRINT CODE
957 0547 E8 1976 R   CALL   E_MSG      ; SETUP ADDRESS OF ERROR MSG
958
959
;----- CHECK FOR OPTIONAL ROM FROM C8000->F0000 IN 2K BLOCKS
960 ;----- (A VALID MODULE HAS "55AA" IN THE FIRST 2 LOCATIONS,
961 ;----- LENGTH INDICATOR (LENGTH/512) IN THE 3D LOCATION AND
962 ;----- TEST/INIT. CODE STARTING IN THE 4TH LOCATION.)
963
964 054A 00
965 054A BA C800      ROM_SCAN: MOV    DX,OC800H      ; SET BEGINNING ADDRESS
966 054D
967 054D BE DA          ROM_SCAN_1: MOV    DS,DX
968 054F 2B DB          SUB   BX,BX      ; SET BX=0000
969 0551 80 07          MO    AX,[BX]      ; GET 1ST WORD FROM MODULE
970 0553 53
971 0554 5B          PUSH  BX
972 0555 3D AA55      POP   BX
973 0556 75 06          CMP   AX,0AA55H      ; BUS SETTLING
974 0556 E8 1920 R   JNZ   NEXT_ROM      ; = TO ID WORD?
975 0556 E8 05 90       CALL   ROM_CHECK      ; PROCEED TO NEXT ROM IF NOT
976
977 0560 81 C2 0080      JMP   ARE_WE_DONE      ; GO CHECK OUT MODULE
978 0564
979 0564 81 FA F000      ARE_WE_DONE: ADD   DX,0080H      ; POINT TO NEXT 2K ADDRESS
980 0568 7C E3          CMP   DX,0F000H      ; AT F0000 YET?
981
982
;----- DISKETTE ATTACHMENT TEST
983 ;----- DESCRIPTION
984 ;----- CHECK IF IPL DISKETTE DRIVE IS ATTACHED TO SYSTEM. IF
985 ;----- ATTACHED, VERIFY STATUS OF NEC FDC AFTER A RESET. ISSUE
986 ;----- A RECAL AND SEEK CMD TO FDC AND CHECK STATUS. COMPLETE
987 ;----- SYSTEM INITIALIZATION THEN PASS CONTROL TO THE BOOT
988 ;----- LOADER PROGRAM.
989
990 056A
991 056A 1F
992 056B A0 0010 R   F9:   POP    DS      ; DISKETTE TEST?
993 056E 24 01          MOV    AL,BYTE PTR #EQUIP_FLAG      ; IS EQUIP PRESENT?
994 0571 74 5E          AND   AL,01H      ; N - BYPASS DISKETTE TEST
995
996 0572 BA 03F1      F10:  JZ    F15      ; I.D.-PORT
997 0575 EC          MOV    DX,3F1H
998 0576 90          IN    AL,DX
999 0577 BB FFFF      NOP
1000 0578 24 FF       MOV    BX,0FFFFH      ; BUS PRECHARGE
1001 057C 00 24 008F R FE  AND   AL,01BH      ; KEEP I.B. BITS
1002 0581 3C 00          AND   #0F_CTRNL,1111110B      ; RESET DUAL BIT
1003 0583 75 05          CMP   AL,CARD_ID
1004 0585 80 00 008F R 01  JNE   NO_ID      ; NO ID
1005 0586 00 00          OR    #0F_CTRNL,1      ; SET DUAL BIT
1006 0588 E4 21          NO_ID: IN    AL,INTAO1      ; ENABLE DISKETTE INTERRUPTS
1007 058C 24 BF          AND   AL,0FFH
1008 058E 6E 21          OUT   INTAO1,AL
1009 0590 B4 00          MOV    AH,0
1010 0592 8A D4          MOV    DL,AH      ; RESET NEC FDC
1011 0594 C0 13          INT   13H      ; SET FOR DRIVE 0
1012 0596 F6 C4 FF       TEST  AH,0FFH      ; VERIFY STATUS AFTER RESET
1013 0599 75 19          JNZ   F13      ; STATUS OK?
1014
1015
;----- TURN DRIVE 0 MOTOR ON
1016
1017 059B BA 03F2      F11:  MOV    DX,03F2H      ; GET ADDR OF FDC CARD
1018 059C B0 1C          MOV    AL,1CH      ; TURN MOTOR ON, EN DMA/INT
1019 05A0 EE          OUT   DX,AL      ; WRITE FDC CONTROL REG
1020 05A1 2B C9          SUB   CX,CX
1021 05A3

```

```

1022 05A3 E2 FE          LOOP    F11      ; WAIT FOR I SECOND
1023 05A4 E2 FE          F12:   LOOP    F12      ; MOTOR_WAIT1:
1024 05A5 E2 FE          XOR     DX,DX   ; SELECT DRIVE 0
1025 05A7 33 D2          MOV     CH,34   ; SELECT TRACK 34
1026 05A9 B5 22          MOV     @SEEK_STATUS_DL
1027 05AB 88 16 003E R   CALL    SEEK    ; RECALIBRATE DISKETTE AND SEEK TO 34
1028 05B2 T3 05          JNC    F14      ; OK--> GO TURN OF MOTOR
1029 05B3 00 00            TEST   AL,DX   ; DISKETTE ERROR?
1030 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
1035
1036 05B9                ;----- TURN DRIVE 0 MOTOR OFF
1037 05B9 33 F6          F14:   XOR    SI,SI   ; SEQUENCE END ENTRY IF NO ERROR
1038 05BB                ; ZERO SI IF NO ERROR
1039 05BB BD 0C          F14A:  MOV    AL,0CH   ; SEQUENCE END ENTRY IF ERROR
1040 05BD BA 03F2          MOV    DX,03F2H ; TURN DRIVE 0 MOTOR OFF
1041 05C0 EE              OUT    DX,AL   ; FDC CTL ADDRESS
1042
1043 ;-----SETUP DISKETTE STATES
1044
1045 05C1 E8 0000 E       CALL    DISKETTE_SETUP ; INITIALIZE DISKETTE PARMS
1046 05C4 T2 04          JC    F14B   ; CY-->DISKETTE SETUP ERROR
1047 05C6 0B F6          OR     SI,SI   ; PREVIOUS DISKETTE ERROR
1048 05C7 00 00            JZ    F15    ; NZ-->DISKETTE ERROR BEFORE SETUP
1049 05CA
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 BE 001E R       SI,OFFSET_0KB_BUFFER ; SETUP KEYBOARD PARAMETERS
1057 05D5 00 00            MOV    @BUFFER_HEAD,SI
1058 05D8 89 36 001A R   MOVS   @BUFFER_TAIL,SI
1059 05D9 89 36 001C R   MOVS   @BUFFER_START,SI
1060 05D9 89 36 0080 R   ADDS  SI,32H   ;DEFAULT BUFFER OF 32 BYTES
1061 05E4 83 C6 20          ADD    SI,32H
1062 05E7 89 36 0082 R   MOVS   @BUFFER_END,SI
1063 05EB BF 0078 R       MOVS   DI,OFFSET @PRINT_TIM_OUT ;SET DEFAULT PRINTER TIMEOUT
1064 05EE 1E
1065 05F0 00 07            PUSH   DS
1066 05F0 00 1414          POP    ES
1067 05F3 AB              MOV    AX,1414H ; DEFAULT=20
1068 05F4 AB              STOSW
1069 05F5 BB 0101          MOV    AX,0101H ;RS232 DEFAULT=01
1070 05F6 AB              STOSW
1071 05F7 AB              STOSW
1072 05F8 00 21            IN    AL,INTA0I
1073 05FC 24 FC            AND    AL,0FCH
1074 05FE E6 21            OUT   INTA0I,AL ; ENABLE TIMER AND KB INTS
1075
1076 0600 83 FD 00          CMP    BP,0000 ; CHECK FOR BP= NON ZERO
1077 0603 T4 18            JE    F15A_0 ; (ERROR HAPPENED)
1079 0605 BA 0002          MOV    DX,2      ; CONTINUE IF NO ERROR
1081 0608 EB 1945 R       CALL    ERR_BEEP ; 2 SHORT BEEPS (ERROR)
1080 0609 BE 0769 R       MOV    SI,OFFSET_F3D ; LOAD ERROR MSG
1082 060E EB 1997 R       CALL    P_MSG
1083 0611
1084 0612 B4 00            MOV    AH,00
1085 0613 CD 16            INT    16H ; WAIT FOR 'FI' KEY
1086 0615 80 FC 3B          CMP    AH,3BH
1087 0618 75 F7            JNE    ERR_WAIT ; BYPASS ERROR
1088 061A EB 0E 90          JMP    F15A_
1089 061D
1090 0620 00 3E 0012 R 01  F15A_0:  CMP    @MFG_TST,I ; MFG MODE
1091 0622 74 06            JE    F15A_1 ; BYPASS BEEP
1092 0624 BA 0001          MOV    DX,1      ; 1 SHORT BEEP (NO ERRORS)
1093 0627 EB 1945 R       CALL    ERR_BEEP
1094 062A A0 0010 R       F15A:  MOV    AL,BYTE PTR @EQUIP_FLAG ; GET SWITCHES
1095 062D 24 01            AND    AL,00000001B ; LOOP POST SWITCH_ON
1096 062E 00 00            JNZ    F15A_0 ; CONTINUE WITH BRING-UP
1097 0631 EB 005B R       F15B:  MOV    DX,1      ; CLEAR SCREEN
1098 0634 2A E4            SUB    AH,AH
1099 0636 A0 0049 R       MOV    AL,@CRT_MODE
1100 0639 CD 10            INT    10H
1101 063B
1102 0640 BD 1970 R       F15C:  MOV    BP,OFFSET F4 ; PRT_SRC_TBL
1103 0643 BE 0070          MOV    SI,0
1104 0641
1105 0641 2E: BB 56 00    F16:   MOV    DX,CS:[BP] ; GET_PRINTER_BASE_ADDR
1106 0645 BD AA            MOV    AL,0AAH ; WRITE DATA TO PORT A
1107 0647 EE
1108 0648 00
1109 0649 EC              OUT   DX,AL
1110 064A 1F              PUSH   DS
1111 064B 3C AA            IN    AL,DX ; BUS SETTLEING
1112 064D 75 05            CMP    AL,0AAH ; DATA PATTERN SAME
1113 064E 89 54 08          JNE    F17    ; NO - CHECK NEXT PRT_CD
1114 0650 00 00            MOV    [PRTBASE-DATA40][SI],DX ; YES - STORE PRT BASE ADDR
1115 0651 00 00            INC    SI
1116 0654 46              INC    SI
1117 0654 45              INC    BP
1118 0655 45              INC    BP
1119 0656 75 FD 1976 R   F17:   CMP    BP,OFFSET F4E ; ALL POSSIBLE ADDRS CHECKED?
1120 0657 75 E5            JNE    F18    ; PRT_CD = 1
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    ; POINT TO NEXT BASE ADDR
1126 0667 00 07 03F8          MOV    [RS232_BASE-DATA40][BX],3F8H ; SETUP RS232_CD #1 ADDR
1127 0668 A3
1128 066C 43
1129 066D
1130 066D BA 02FA          MOV    DX,2FAH ; CHECK IF RS232_CD 2 ATTCH
1131 0670 BB 0000          IN    AL,DX ; READ INTERRUPT ID REG
1132 0671 A8 F8            TEST   AL,0F8H
1133 0673 75 06            JNZ    F18    ; BASE_END
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
1139 ;----- SET UP EQUIP FLAG TO INDICATE NUMBER OF PRINTERS AND RS232 CARDS
1140 067B F19:          MOV     AX,51      ; BASE END!
1141 067B BB C6          MOV     CL,3       ; SI HAS 2nd NUMBER OF RS232
1142 067D B1 03          MOV     AL,CL      ; SHIFT COUNT
1143 067F D2 C8          ROR     AL,CL      ; ROTATE RIGHT 3 POSITIONS
1144 0680 A0 C3          OR      AL,BL      ; OR IN THE PRINTER COUNT
1145 0683 12 0011 R     MOV     BYTE PTR EQUIP_FLAG+1,AL ; STORE AS SECOND BYTE
1146 0686 BA 0201        MOV     DX,201H
1147 0689 EC             IN      AL,DX
1148 068A 90             NOP
1149 068B 90             NOP
1150 068C 90             NOP
1151 068D 18 0F           TEST    AL,0FH
1152 068F 75 05           TEST    AL,0FH
1153 0691 80 0E 0011 R 10 JNZ    F20:      ; NO_GAME_CARD
1154 0696                 OR      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 06A0 E6 61           OUT    PORT_B,AL
1163 06A0 BB 00           MOV     AL,80H      ; ENABLE NMI INTERRUPTS
1164 06A2 E6 A0           OUT    OA0H,AL
1165 06A4                 F21:      INT    19H      ; LOAD_BOOT_STRAP!
1166 06A4 CD 19           IN      AL,PORT_B      ; 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 TC00)      ;
1172 ;----- AND CONTROL IS TRANSFERRED THERE.          ;
1173 ;----- IF THERE IS A HARDWARE ERROR CONTROL IS   ;
1174 ;----- TRANSFERRED TO THE ROM BASIC ENTRY POINT. ;
1175
1176 ;----- ASSUME CS:CODE,DS:AB50
1177 ORG    0E6F2H
1178 ORG    006F2H
1179 06F2
1180
1181 04F2
1182 06F2 FB           PROC    NEAR      ; ENABLE_INTERRUPTS
1183 06F3 2B C0           STI
1184 06F5 8E D0           SUB    AX,AX      ; ESTABLISH ADDRESSING
1185
1186 ;----- RESET THE DISK PARAMETER TABLE VECTOR
1187
1188 06F7 C7 06 0078 R 0FC7 R     MOV     WORD PTR @DISK_POINTER,OFFSET DISK_BASE
1189 06FD 8C 0E 007A R     MOV     WORD PTR @DISK_POINTER+2,CS
1190
1191 ;----- LOAD SYSTEM FROM DISKETTE -- CX HAS RETRY COUNT
1192
1193 0701 B9 0004        MOV     CX,4       ; SET_RETRY_COUNT
1194 0704 H1:             HI
1195 0704 51             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 00 0B 201        MOV     AX,201H    ; IF_ERROR_TRY AGAIN
1200 0710 00 00 00          SUB    DX,DX    ; READ IN THE SINGLE SECTOR
1201 0710 BE C2           MOV     ES,DX    ; TO THE BOOT LOCATION
1202 0712 BB TC00 R      MOV     BX,OFFSET @BOOT_LOCN
1203
1204 0715 B9 0001        MOV     CX,1       ; DRIVE_0, HEAD_0
1205 0716 CD 13           INT    13H      ; SECTOR_1, TRACK_0
1206 071A H2:             HI
1207 071A 59             POP    CX       ; DISKETTE_IO
1208 071B T3 04           JNC    H4       ; RECOVER_RETRY_COUNT
1209 071D E2 E5           LOOP   H1       ; CF_SET_BY_UNSUCCESSFUL_READ
1210
1211 ;----- UNABLE TO IPL FROM THE DISKETTE
1212
1213 071F H3:             INT    18H      ; DO IT FOR RETRY TIMES
1214 071F CD 18
1215
1216 ;----- IPL WAS SUCCESSFUL
1217
1218 0721 H4:             INT    18H      ; GO TO RESIDENT BASIC
1219 0721 EA TC00 ----- R
1220 0726                 BOOT_STRAP
1221
1222 ;----- ORG 0E729H
1223 0729 0417           A1
1224 0728 0300           DW    1047      ; 110 BAUD      ; TABLE OF VALUES
1225 0728 0300           DW    768
1226 072D 0180           DW    384
1227 072F 00C0           DW    192
1228 0731 0060           DW    96
1229 0733 0070           DW    48
1230 0735 0018           DW    24
1231 0737 000C           DW    12
1232
1233 0739
1234 0739 E9 0000 E      JMP     RS232_10_
1235
1236 073C CONF_TBL_       DB      CONF_E-CONF_TBL-2 ; CONFIGURATION TABLE FOR THIS SYSTEM
1237 073C 0008           DB      LENGTH_OF_FOLLOWING_TABLE
1238 073E FB             DB      MODEL_BYTE
1239 073F 00             DB      SUB_MODEL_BYTE
1240 0740 01             DB      BIOS_LEVEL
1241 0741 50             DB      01010000B ; SYSTEM SUB MODEL TYPE BYTE
1242
1243
1244
1245 0742 00             DB      0 ; SYSTEM REVISON LEVEL
1246 0743 00             DB      0 ; 10000000 = SERIAL CHANNEL 3 USE BY BIOS
1247 0744 00             DB      0 ; 01000000 = CASCADE INTERRUPT LEVEL 2
1248 0745 00             DB      0 ; 00100000 = REAL TIME CLOCK AVAILABLE
1249 0746                 CONF_E EQU    $ ; 00010000 = KEYBOARD SCAN CODE HOOK IAH
1249 = 0746
1249
1249 ;----- RESERVED
1250 ;----- RESERVED
1251 ;----- RESERVED
1252 ;----- RESERVED
1253 ;----- RESERVED
1254 ;----- RESERVED
1255 ;----- RESERVED FOR EXPANSION

```

```

1250
1251
1252      ; PRINT ADDRESS AND ERROR MESSAGE FOR ROM CHECKSUM ERRORS
1253
1254 0746    ROM_ERR PROC NEAR
1255 0747 52    PUSH  DX          ; SAVE POINTER
1256 0747 50    PUSH  AX
1257 0748 8C DA    MOVS  DX,DS
1258 074A 26: 88 36 0015 R    MOV   ES:ROM_ERR_FLAG,DH
1259                      ; GET ADDRESS POINTER
1260 074F 81 FA C800    CMP   DX,0C800H
1261 0753 7C 0C    JL    ROM_ERR_BEEP
1262 0754 8B 0A    CALL  PRTEG
1263 0758 BE 18D1 R    MOV   SI,OFFSET F3A
1264 075B EB 1976 R    CALL  E_MSG
1265 075E    ROM_ERR_END:
1266 075E 58    POP   AX
1267 075F 54    POP   DX
1268 075F C3    RETF
1269 0761    ROM_ERR_BEEP:
1270 0761 BA 0102    MOV   DX,0102H
1271 0764 EB 19A5 R    CALL  ERR_BEEP
1272 0767 EB F5    JMP   SHORT_ROM_ERR_END
1273 0769    ROM_ERR_ENDP
1274
1275 0769 45 52 52 4F 52 2E    F3D  DB  'ERROR. (RESUME = "F1" KEY)',CR,LF
1276 0770 20 28 52 45 53 55
1277 0771 40 45 20 3D 20 22
1278 0772 46 31 22 20 4B 45
1279 0773 59 29 00 0A
1280
1281      ; ORG 0E82EH
1282 082E      ; ORG 0082EH
1283 082E    KEYBOARD_10:
1284 082E E9 0000 E    JMP   KEYBOARD_10_
1285
1286
1287 0987      ; ORG 0E987H
1288 0987    KB_INT:
1289 0987 E9 0000 E    JMP   KB_INT_
1290
1291 098A 20 33 30 31 0D 0A    F1  DB  ' 301',CR,LF
1292 098B 36 30 31 0D 0A    F3  DB  '601',CR,LF
1293
1294      ;-- INT 1A H -- SYSTEM AND REAL TIME CLOCK SERVICES --
1295
1296      ; THIS BIOS ROUTINE ALLOWS THE CLOCKS TO BE SET OR READ
1297
1298      ; PARAMETERS:
1299      ; (AH) = 00H READ THE CURRENT CLOCK SETTING AND RETURN WITH,
1300      ; (CX) = HIGH PORTION OF COUNT
1301      ; (DX) = LOW PORTION OF COUNT
1302      ; (AL) = 0 TIMER HAS NOT PASSED 24 HOURS SINCE LAST READ,
1303      ; I IF ON ANOTHER DAY. (RESET TO ZERO AFTER READ)
1304
1305      ; (AH) = 01H SET THE CURRENT CLOCK USING,
1306      ; (CX) = HIGH PORTION OF COUNT
1307      ; (DX) = LOW PORTION OF COUNT.
1308
1309      ; NOTE: COUNTS OCCUR AT THE RATE OF 1193180/65536 COUNTS/SECOND
1310      ; (OR ABOUT 18.2 PER SECOND -- SEE EQUATES)
1311
1312      ; (AH) = 0AH READ THE CURRENT COUNT OF DAYS AND RETURN WITH,
1313      ; (CX) = COUNT OF ELAPSED DAYS
1314
1315      ; (AH) = 0BH SET THE CURRENT COUNT OF DAYS USING,
1316      ; (CX) = COUNT OF ELAPSED DAYS
1317
1318      ; NOTES: FOR ALL RETURNS CY= 0 FOR SUCCESSFUL OPERATION.
1319      ; INTERRUPTS ARE DISABLED DURING DATA MODIFICATION.
1320      ; AH & AL ARE RETURNED MODIFIED AND NOT DEFINED EXCEPT WHERE INDICATED.
1321
1322      ASSUME CS:CODE,DS:DATA
1323
1324 0995    TIME_OF_DAY_!  PROC FAR
1325 0995    TIME_OF_DAY_!!:
1326 0995 FB    STI
1327 0996 80 FC 0C    CMP   AH,(RTC_TBE-RTC_TB)/2
1328 0999 F5    CMC
1329 099A 12 17    JC    TIME_9
1330
1331 099C 1E    PUSH  DS
1332 099D E8 1A12 R    CALL  DDS
1333 09A0 56    PUSH  SI
1334 09A1 80 C4    MOVS  AL,AH
1335 09A2 98    CBW
1336 09A4 03 C0    ADD   AX,AX
1337 09A6 BB F0    MOV   SI,AX
1338 09A8 FA    CLD
1339 09A9 2E: FF 94 09B6 R    CALL  CS:[SI]+OFFSET RTC_TB
1340
1341 09AE FB    STI
1342 09AF B4 00    MOV   AH,0
1343 09B1 5E    POP   SI
1344 09B2 1F    POP   DS
1345 09B3 CA 0002    TIME_9?:
1346 09B3 CA 0002    RET   2
1347
1348 09B6 09CE R    RTC_TB DW  RTC_00
1349 09B8 09DF R    DW  RTC_10
1350 09B8 09ED R    DW  RTC_N5
1351 09BC 09ED R    DW  RTC_N5
1352 09C0 09ED R    DW  RTC_N5
1353 09C0 09ED R    DW  RTC_N5
1354 09C2 09ED R    DW  RTC_N5
1355 09C4 09ED R    DW  RTC_N5
1356 09C6 09ED R    DW  RTC_N5
1357 09C8 09EF R    DW  RTC_N5
1358 09CA 09EF R    DW  RTC_A0
1359 09CC 09F4 R    DW  RTC_B0
1360 = 09CE
1361
1362 09CE    RTC_TBE EQU  $
1363
1364    TIME_OF_DAY_! ENDP

```

```

1364 09CE          RTC_00  PROC  NEAR
1365 09CE A0 0070 R  MOV    AL, @TIMER_OFLOW
1366 09D1 C6 06 0070 R 00  MOV    @TIMER_OFLOW, 0
1367 09D6 8B 0E 006E R  MOV    CX, @TIMER_HIGH
1368 09D4 BB 16 006C R  MOV    DX, @TIMER_LOW
1369 09DE C3        RET

1371 09DF          RTC_10: PROC  NEAR
1372 09DF 89 16 006C R  MOV    @TIMER_LOW, DX
1373 09E3 89 0E 006E R  MOV    @TIMER_HIGH, CX
1374 09E7 C6 06 0070 R 00  MOV    @TIMER_OFLOW, 0
1375 09EC C3        RET

1376 09E0          RTC_NS:  PROC  NEAR
1377 09E0 STC
1378 09E0 RET

1379 09EE C3        RTC_A0:  PROC  NEAR
1380 09EF           MOV    CX, @DAY_COUNT
1381 09EF           RET

1382 09EF 8B 0E 00CE R  RTC_B0:  PROC  NEAR
1383 09F3 C3        MOV    @DAY_COUNT, CX
1384 09F4           RET

1385 09F4           RTC_00: ENDP
1386 09F4 89 0E 00CE R
1387 09F8 C3
1388 09F9
1389 09F9
1390 09F9
1391 ; ORG 00C59H
1392 0C59 ORG 00C59H
1393 0C59 E9 0000 E  DISKETTE_10: JMP   DISKETTE_10_I

;---- BEEP -----
;----- ROUTINE TO SOUND THE BEEPER USING TIMER 2 FOR TONE -----
;----- ENTRY: -----
;----- (BL) = DURATION COUNTER ( 1 FOR 1/64 SECOND )
;----- (CX) = FREQUENCY DIVISOR ((193180/FREQUENCY) (1331 FOR 886 HZ) )
;----- EXIT: -----
;----- (AX), (BL), (CX) MODIFIED. -----
;----- BEEP -----
;----- SETUP TIMER 2 -----
;----- PUSHF
1404 0C56          BEEP  PROC  NEAR
1405 0C56 9C        CLI
1406 0C5D FA        MOV    AL, 10110110B
1407 0C5E B0 06      OUT   TIMER+3, AL
1408 0C60 E6 43      NOP
1409 0C62 90        MOV    AL, CL
1410 0C63 8A C1      OUT   TIMER+2, AL
1411 0C64 8A 42      NOP
1412 0C65 90        MOV    AL, CH
1413 0C68 8A C5      OUT   TIMER+2, AL
1414 0C6A E6 42      IN    AL, PORT_B
1415 0C6C E4 61      MOV    AH, AL
1416 0C6E 8A E0      OR    AL, GATE2+SPK2
1417 0C6F E0 C3      OUT   PORT_B, AL
1418 0C72 E6 61      POPF
1419 0C74 9D        POPPF
1420 0C75          G7:   MOV    CX, 1035
1421 0C75 B9 040B    CALL  WAITF
1422 0C78 E8 0CA0 R  DEC   BL
1423 0C7B FE CB    JNZ   G7
1424 0C7D 15 F6
1425
1426 0C7F 9C        PUSHF
1427 0C80 FA        CLI
1428 0C81 E4 61      IN    AL, PORT_B
1429 0C83 00 FC      OR    AL, NOT (GATE2+SPK2)
1430 0C84 00 23      AND   AL, AH
1431 0C87 8A C4      MOV    AH, AH
1432 0C89 24 FC      AND   AL, NOT (GATE2+SPK2)
1433 0C8B E6 61      OUT   PORT_B, AL
1434 0CBD 9D        POPF
1435 0C90 00 040B    MOV    CX, 1035
1436 0C91 E8 0CA0 R  CALL  WAITF
1437 0C94 9C        PUSHF
1438 0C95 FA        CLI
1439 0C96 E4 61      IN    AL, PORT_B
1440 0C98 24 03      AND   AL, GATE2+SPK2
1441 0C9A 00 2A C4    OR    AL, AH
1442 0C9C E8 61      OUT   PORT_B, AL
1443 0C9E 9D        POPF
1444 0C9F C3        RET

1446 0CA0          BEEP  ENDP
1447
1448 ;---- WAITF -----
;----- FIXED TIME WAIT ROUTINE (HARDWARE CONTROLLED - NOT PROCESSOR) -----
;----- ENTRY: -----
;----- (CX) = COUNT OF 15.085737 MICROSECOND INTERVALS TO WAIT
;----- MEMORY REFRESH TIMER 1 OUTPUT AT THE DMA CHANNEL 0
;----- ADDRESS REGISTER USED AS REFERENCE.
;----- EXIT: -----
;----- (CX) = 0 AFTER (CX) TIME COUNT (PLUS OR MINUS 31 MICROSECONDS) -----
;----- WAITF -----
;----- WAITF:  PROC  NEAR
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460 0CA0          WAITF: PROC  NEAR
1461 0CA0 50        PUSH  AX
1462 0CA1 D1 E9      SHR   CX, 1
1463 0CA3 E3 13      JCXZ WAITF9
1464
1465 0CA5 E6 0C      OUT   DMA+12, AL
1466 0CA7 9C        WAITF1: PROC  NEAR
1467 0CA8 FA        CLI
1468 0CA9 F4        WAITF3: PROC  NEAR
1469 0CA9 E4 00      IN    AL, DMA
1470 0CA9 FF 00      AND   AL, 1111110B
1471 0CA9 FE        CMP   AL, AL
1472 0CA9 3A E0      MOV   AH, AL
1473 0CAF 8A E0      IN    AL, DMA
1474 0CB1 E4 00      JE    WAITF3
1475 0CB3 74 F4      JE    WAITF3
1476
1477 0CBB 9D        POPF
1478
1479 ;----- WAITF -----
;----- DELAY FOR (CX)*15.085737 US
1480 ;----- SAVE WORK REGISTER (AH)
1481 ;----- DIVIDE 15us COUNT DOWN TO 30us COUNT
1482 ;----- EXIT IF COUNT WAS ZERO OR ONE
1483
1484 ;----- CLEAR THE DMA BYTE POINTER FLIP/FLOP
1485 ;----- SAVE INTERRUPT STATE
1486 ;----- BLOCK INTERRUPTS TILL NEXT CHANGE
1487 ;----- WAIT FOR REFRESH ADDRESS CHANGE
1488 ;----- READ CURRENT ADDRESS LOW BYTE
1489 ;----- DISCARD LOW BIT (30us)
1490 ;----- DIVIDE VALUE BY 15us
1491 ;----- SAVE NEW/OLD VALUE INCASE IT DID
1492 ;----- READ HIGH BYTE (AND IGNORE)
1493 ;----- WAIT FOR A CHANGE IN ADDRESS BITS
1494
1495 ;----- RESTORE INTERRUPTS

```

```

1478 0CB6 E2 EF      LOOP    WAITFI      ; DECREMENT CYCLES COUNT TILL COUNT END
1479 0CB8              WAITF9: POP     AX      ; RESTORE (AH)
1480 0CB8 58          RET      CX      ; RETURN (CX) = 0
1481 0CB9 C3
1482
1483 0CBA      WAITF  ENDP
1484
1485
1486
1487
1488
1489 0CBA      PRT_SEG PROC NEAR
1490 0CBA BA C6      MOV     AL,DI      ; GET MSB
1491 0CBC E8 1958 R   CALL    XPC_BYT
1492 0CDC C0 C2      MOV     AL,DL      ; LSB
1493 0CC1 E8 1958 R   CALL    XPC_BYT
1494 0CC4 B0 30      MOV     AL,“0”      ; PRINT A ‘0’
1495 0CC6 E8 1969 R   CALL    PRT_HEX
1496 0CC9 B0 20      MOV     AL,T      ; SPACE
1497 0CCE E8 1969 R   CALL    PRT_HEX
1498 0CCE C3      RET      CX
1499 0CCF  PRT_SEG ENDP
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
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      DI,DI      ; SET DIR FLAG TO INCREMENT
1519 0CD2 2B FF      SUB     DI,DI      ; SET DI=OFFSET0 REL TO ES REG
1520 0CD4 2B C0      SUB     AX,AX      ; SETUP FOR 0->FF PATTERN TEST
1521 0CD6
1522 0CD8 8A 05      C2_1:  MOV     [DI],AL      ; ON FIRST BYTE
1523 0CD8 8A 05      MOV     AL,[DI]
1524 0CDA 32 C4      XOR     AL,AH
1525 0CDC T5 79      JNZ     C7      ; O.K.?
1526 0CDE FE C4      INC     AH
1527 0CEO 8A C4      MOV     AL,AH
1528 0CE0 75 2        JNZ     C2_1      ; LOOP TILL WRAP THROUGH FF
1529 0CE0 54 55 AA    MOV     AL,055AAH
1530 0CE7 89 D0      MOV     DX,AX      ; GET INITIAL DATA PATTERN TO WRITE
1531 0CE9 F3 / AB    REP    STOSW
1532 0CED E4 61      IN      AL,PORT_B
1533 0CED OC 30      OR      AL,030H
1534 0CEC E6 61      OUT    PORT_B,AL
1535 0CEC E6 61      NOT    AL,0CFH
1536 0CF2 24 CF      AND    AL,0CFH
1537 0CF4 E6 61      OUT    PORT_B,AL
1538
1539 0CF6 4F
1540 0CF6 4F
1541 0CF8 FD      C3:   LODSW
1542 0CF9 BB F7      XOR     AX,DX      ; SET DIR FLAG TO GO BACKWARDS
1543 0CF9 BB CB      STD     CTX      ; INITIALIZE DESTINATION POINTER
1544 0CFD AD      MOV     SI,DI      ; SETUP WORD COUNT FOR LOOP
1545 0CFD AD      MOV     CX,BX      ; INNER TEST LOOP
1546 0D00 33 C2      LODSW
1547 0D00 75 43      XOR     AX,DX      ; READ OLD TEST WORD FROM STORAGE
1548 0D02 BB AA55    JNE     CTX      ; DATA READ AS EXPECTED ?
1549 0D05 AB      MOV     AX,0AA55H
1550 0D06 E2 F5      STOSW
1551
1552 0D08 FC      C4:   LODSW
1553 0D09 47      INC     DI      ; NO - GO TO ERROR ROUTINE
1554 0D0A 47      INC     DI      ; GET NEXT DATA PATTERN TO WRITE
1555 0D0B BB F7      MOV     SI,DI
1556 0D0D BB CB      MOV     CX,BX
1557 0D0F BB D0      MOV     DX,AX
1558
1559 0D11 AD      LODSW
1560 0D12 33 C2      XOR     AX,DX      ; WRITE INTO LOCATION JUST READ
1561 0D14 75 43      JNE     CTX      ; DECREMENT WORD COUNT AND LOOP
1562 0D16 BB FFFF    MOV     AX,0FFFFH
1563 0D19 AB      STOSW
1564 0D1A E2 F5      LOOP   C4
1565
1566 0D1C 4F
1567 0D1D 4F
1568 0D1E FD      C5:   LODSW
1569 0D1F BB F7      DEC     DI      ; POINT TO LAST WORD JUST WRITTEN
1570 0D20 BB CB      STD     CTX      ; SET DIR FLAG TO GO BACKWARDS
1571 0D22 BB D0      MOV     SI,DI      ; INITIALIZE DESTINATION POINTER
1572 0D25 AD      LODSW
1573 0D25 AD      MOV     CX,BX      ; SETUP WORD COUNT FOR LOOP
1574 0D26 BB D0      MOV     DX,AX      ; SETUP COMPARE PATTERN “0FFFH”
1575 0D28 75 2F      LODSW
1576 0D28 BB 0101H   XOR     AX,DX      ; INNER TEST LOOP
1577 0D2D BB D0      JNE     CTX      ; READ OLD TEST WORD FROM STORAGE
1578 0D2E E2 F5      MOV     AX,0101H
1579
1580 0D30 FC      LODSW
1581 0D31 47      INC     DI      ; DATA READ AS EXPECTED ?
1582 0D32 47      INC     DI      ; NO - GO TO ERROR ROUTINE
1583 0D33 BB F7      MOV     SI,DI
1584 0D35 BB CB      MOV     CX,BX
1585 0D37 BB D0      MOV     DX,AX
1586 0D39
1587 0D3A AD      LODSW
1588 0D3A 33 C2      XOR     AX,DX      ; WRITE ZERO INTO LOCATION READ
1589 0D3C 75 1B      JNE     CTX      ; DECREMENT WORD COUNT AND LOOP
1590 0D3D AB      STOSW
1591 0D3F E2 F6      LOOP   C6

```

```

1592          ; POINT TO LAST WORD JUST WRITTEN
1593 0D41 4F    DEC    DI      ; SET DIR FLAG TO GO BACKWARDS
1594 0D42 4F    DEC    DI      ; INITIALIZE DESTINATION POINTER
1595 0D43 FD    STD    SI,DI   ; SETUP WORD COUNT FOR LOD
1596 0D44 8B F7  MOV    SI,DI   ; SETUP COMPARE PATTERN "0000H"
1597 0D45 8B C0  MOV    CX,BX
1598 0D46 8B D0  MOV    DX,AX
1599 0D4A        C6X:    LODSW
1600 0D4A AD    LODSW
1601 0D4B 33 C2  XOR    AX,DX
1602 0D4D 75 0A  JNE    C7X
1603 0D4E F2 F9  LOOP   C6X
1604          ; DID A PARITY ERROR OCCUR ?
1605 0D51 E4 62  IN     AL,PORT_C
1606 0D53 24 C0  AND    AL,OCOH
1607 0D55 B0 00  MOV    AL,0
1608 0D57        C7:    SHORT C7
1609 0D58 FC    CLD
1610 0D58 C3    RET
1611 0D59        C7X:    CMP    AL,0
1612 0D59 3C 00  JNZ    C7
1613 0D5B 75 FA  MOV    AL,AH
1614 0D5D 8A C4  JMP    SHORT C7
1615 0D5F EB F6  STGTST_CNT ENDP
1616 0D61
1617
1618          ; SET DIRECTION FLAG TO INC
1619 0F57
1620 0F57 E9 0000 E
1621
1622
1623 0F79
1624
1625          ; MEDIA/DRIVE PARAMETER TABLES
1626
1627
1628
1629          ; 40 TRACK LOW DATA RATE MEDIA IN 40 TRACK LOW DATA RATE DRIVE :
1630 0F79 MD_TBL1 LABEL_BYT
1631 0F79 DF    DB     11011111B ; 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
1637 0F7F FF    DB     0FFH
1638 0F80 50    DB     050H
1639 0F80 06    DB     0F6H
1640 0F82 02    DB     15          ; HEAD SETTLE TIME (MILLISECONDS)
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_BYT
1648 0F86 DF    DB     11011111B ; 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 0F89 19    DB     09          ; EOT (LAST SECTOR ON TRACK)
1653 0F89 2A    DB     02AH
1654 0F8C FF    DB     0FFH
1655 0F8D 50    DB     050H
1656 0F8E F6    DB     0F6H
1657 0F8F 06    DB     15          ; HEAD SETTLE TIME (MILLISECONDS)
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_BYT
1665 0F93 DF    DB     11011111B ; 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 0F96 0F    DB     15          ; EOT (LAST SECTOR ON TRACK)
1670 0F98 1B    DB     01BH
1671 0F99 FF    DB     0FFH
1672 0F9A 54    DB     054H
1673 0F9B F6    DB     0F6H
1674 0F9C 0F    DB     15          ; HEAD SETTLE TIME (MILLISECONDS)
1675 0F9D 08    DB     8           ; MOTOR START TIME (1/8 SECONDS)
1676 0F9E 4F    DB     39          ; 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_BYT
1682 0FA0 DF    DB     11011111B ; 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 0FA4 2A    DB     02AH
1688 0FA4 FF    DB     0FFH
1689 0FA5 50    DB     050H
1690 0FA8 F6    DB     0F6H
1691 0FA9 0F    DB     15          ; HEAD SETTLE TIME (MILLISECONDS)
1692 0FAA 08    DB     8           ; MOTOR START TIME (1/8 SECONDS)
1693 0FAA 4F    DB     39          ; 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_BYT
1699 0FAD DF    DB     11011111B ; 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
1705 0FB3 FF    DB     0FFH

```

```

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 08          DB   8       ; MOTOR START TIME (1/8 SECONDS)
1710 0FB8 4F          DB   79     ; MAX. TRACK NUMBER
1709 0FB9 60          DB   RATE_250 ; DATA TRANSFER RATE
1712
1713
1714
1715 0FBA             DB   1010011B ; LABEL BYTE
1716 0FBC AF          DB   2       ; SRT=A, HD UNLOAD=0 - 1ST SPECIFY BYTE
1717 0FBD 02          DB   2       ; HD LOAD=1, MODE=DMA 2ND SPECIFY BYTE
1718 0FBC 25          DB   MOTOR_WAIT ; MOTOR_WAIT
1719 0FBD 02          DB   2       ; WAIT_TIME AFTER OPERATION TILL MOTOR OFF
1720 0FB2 12          DB   18      ; 512 BYTES/SECTOR
1721 0FBF 1B          DB   01BH   ; EOT (LAST SECTOR ON TRACK)
1722 0FC0 FF          DB   0FFH   ; DTL
1723 0FC0 4C          DB   05CH   ; 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
1732
1733
1734
1735
1736
1737 0FC7             DB   0EFC1H ; DISK_BASE
1738 0FC7             ORG  00FC1H
1739 0FC7 CF          DB   1000111B ; THIS IS THE SET OF PARAMETERS REQUIRED FOR DISKETTE OPERATION.
1740 0FC0 02          DB   2       ; THEY ARE POINTED AT BY THE DATA VARIABLE DISK_POINTER. TO
1741 0FC0 25          DB   MOTOR_WAIT ; MODIFY THE PARAMETERS, BUILD ANOTHER PARAMETER BLOCK AND POINT
1742 0FCA 02          DB   2       ; DISK_POINTER TO IT.
1743 0FC0 08          DB   5       ; 512 BYTES/SECTOR
1744 0FCC 2A          DB   02AH   ; EOT (LAST SECTOR ON TRACK)
1745 0FC0 FF          DB   0FFH   ; DTL
1746 0FCE 50          DB   050H   ; GAP LENGTH FOR FORMAT
1747 0FCE 56          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             DB   0EFD2H ; ORG
1753 0FD2             ORG  00FD2H
1754 0FDE E9 0000 E   PRINTER_IO; JMP PRINTER_IO_
1755
1756
1757 1045             DB   0F045H ; MI
1758 1045 0000 E       ORG  01045H ; TABLE OF ROUTINES WITHIN VIDEO !
1759 1045 0000 E       DW   OFFSET SET_MODE
1760 1045 0000 E       DW   OFFSET SET_TYPE
1761 1045 0000 E       DW   OFFSET SET_CPOS
1762 1045 0000 E       DW   OFFSET READ_CURSOR
1763 1045 0000 E       DW   OFFSET READ_LPEN
1764 1051 0000 E       DW   OFFSET ACT_DISP_PAGE
1765 1051 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 1050 0000 E       DW   OFFSET SET_COLOR
1770 1050 0000 E       DW   OFFSET WRITE_DOT
1771 1061 0000 E       DW   OFFSET READ_DOT
1772 1061 0000 E       DW   OFFSET WRITE_TTY
1773 1063 0000 E       DW   OFFSET VIDEO_STATE
1774 = 0020            MIL  EQU $-MI
1775
1776
1777 1065             DB   0F065H ; VIDEO_IO
1778 1065             ORG  01065H
1779 1065 E9 0000 E   VIDEO_IO; JMP VIDEO_IO_
1780
1781 ;----- VIDEO PARAMETERS --- INIT_TABLE
1782
1783
1784 10A4             DB   0F0A4H ; M4
1785
1786 10A4             DB   010A4H ; VIDEO_PARMS
1787 10A4 3B 28 2D 0A 1F 06   LABEL BYTE
1788 10A4 19             DB   38H,28H,2DH,0AH,1FH,6,19H ; SET UP FOR 40X25
1789 10AB 00 02 07 06 07   DB   ICH,2,7,6,7
1790 10B0 00 00 00 00 00   DB   0,0,0,0
1791 = 0010            M4  EQU $-VIDEO_PARMS
1792
1793 10B1 71 50 5A 0A 1F 06   DB   71H,50H,5AH,0AH,1FH,6,19H ; SET UP FOR 80X25
1794 19
1795 10B8 1C 02 07 06 07   DB   ICH,2,7,6,7
1796 10C0 00 00 00 00 00   DB   0,0,0,0
1797
1798 10C4 3B 28 2D 0A 1F 06   DB   38H,28H,2DH,0AH,7FH,6,64H ; SET UP FOR GRAPHICS
1799 64
1800 10C8 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 19 06   DB   61H,50H,52H,0FH,19H,6,19H ; SET UP FOR 80X25 B&W CARD
1804 19
1805 10D4 19 02 0D 0B 0C   DB   19H,2,DH,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
1814 10EC 28 28 50 50 28 28   M6  DB   40,40,80,80,40,40,80,80
1815 50 50
1816
1817 10F4 2C 28 2D 29 2A 2E   M7  C_REG_TAB ; TABLE OF MODE SETS
1818 0B00 00 00 00 00 00 00   DB   2CH,28H,2DH,29H,2AH,2EH,1EH,29H

```

```
1819  
1820  
1821  
1822 PAGE  
1823 :---- INT 12 ----  
1824 : MEMORY_SIZE_DET  
1825 : THIS ROUTINE DETERMINES THE AMOUNT OF MEMORY IN THE SYSTEM  
1826 : AS REPRESENTED BY THE SWITCHES ON THE PLANAR. NOTE THAT THE  
1827 : SYSTEM MAY NOT BE ABLE TO USE I/O MEMORY UNLESS THERE IS A FULL  
1828 : COMPLEMENT OF 64K BYTES ON THE PLANAR.  
1829 : INPUT  
1830 : NO REGISTERS  
1831 : THE MEMORY SIZE VARIABLE IS SET DURING POWER ON DIAGNOSTICS  
1832 : ACCORDING TO THE FOLLOWING HARDWARE ASSUMPTIONS:  
1833 : PORT 60 BITS 3,2 = 00 - 256K BASE RAM  
1834 : 01 - 512K BASE RAM  
1835 : 10 - 1024K BASE RAM  
1836 : 11 - 640K BASE RAM  
1837 : PORT 62 BITS 3,0 INDICATE AMOUNT OF I/O RAM IN 32K INCREMENTS  
1838 : E.G., 0000 - NO RAM IN I/O CHANNEL  
1839 : 0010 - 64K RAM IN I/O CHANNEL, ETC.  
1840 :  
1841 :-----  
1842 : OUTPUT  
1843 : (AX) = NUMBER OF CONTIGUOUS 1K BLOCKS OF MEMORY  
1844 :-----  
1845 : ASSUME CS:CODE,DS:DATA  
1846 : ORG 0F841H  
1847 : ORG 01841H  
1848 :-----  
1849 : MEMORY_SIZE_DET PROC FAR  
1850 : ST1  
1851 : PUSH DS : INTERRUPTS BACK ON  
1852 : CALL DDS : SAVE SEGMENT  
1853 : MOV AX,MEMORY_SIZE : GET VALUE  
1854 : POP DS : RECOVER SEGMENT  
1855 : IRET : RETURN TO CALLER  
1856 :-----  
1857 : MEMORY_SIZE_DET ENDP  
1858 :  
1859 :-----  
1860 : EQUIPMENT DETERMINATION  
1861 : THIS ROUTINE ATTEMPTS TO DETERMINE WHAT OPTIONAL  
1862 : DEVICES ARE ATTACHED TO THE SYSTEM.  
1863 : INPUT  
1864 : NO REGISTERS  
1865 : THE EQUIP FLAG VARIABLE IS SET DURING THE POWER ON  
1866 : DIAGNOSTICS USING THE FOLLOWING HARDWARE ASSUMPTIONS:  
1867 : PORT 60 = LOW ORDER BYTE OF EQUIPMENT  
1868 : PORT 3FA = EQUIPMENT IDENTIFIER OF 8250  
1869 : BITS 7-3 ARE UNKNOWN  
1870 : PORT 378 = OUTPUT PORT OF PRINTER -- B255 PORT THAT  
1871 : CAN BE READ AS WELL AS WRITTEN  
1872 :  
1873 : OUTPUT  
1874 : (AX) IS SET, BIT SIGNIFICANT, TO INDICATE ATTACHED I/O  
1875 : BIT 15 = NUMBER OF PRINTERS ATTACHED  
1876 : BIT 14 NOT USED  
1877 : BIT 12 = GAME I/O ATTACHED  
1878 : BIT 11,10,9 = NUMBER OF RS232 CARDS ATTACHED  
1879 : BIT 8 UNUSED  
1880 : BIT 7,6 = NUMBER OF DISKETTE DRIVES  
1881 : BIT 5,4 = INITIAL VIDEO MODE  
1882 : 00 - UNUSED  
1883 : 01 - 40X25 BY USING COLOR CARD  
1884 : 10 - 80X25 BY USING COLOR CARD  
1885 : 11 - 80X25 BY USING BW CARD  
1886 :  
1887 : BIT 3,2 = PLANAR RAM SIZE (00=256K,01=512K,10=576K,11=640K)  
1888 : BIT 1 = MATH COPROCESSOR  
1889 : BIT 0 = IPL FROM DISKETTE -- THIS BIT INDICATES THAT  
1890 : THERE ARE DISKETTE DRIVES ON THE SYSTEM  
1891 :  
1892 :-----  
1893 : NO OTHER REGISTERS AFFECTED  
1894 :  
1895 :-----  
1896 : ASSUME CS:CODE,DS:DATA  
1897 : ORG 0F840H  
1898 : ORG 01840H  
1899 :-----  
1900 : EQUIPMENT PROC FAR  
1901 : ST1  
1902 : PUSH DS : INTERRUPTS BACK ON  
1903 : CALL DDS : SAVE SEGMENT REGISTER  
1904 : MOV AX,EQUIP_FLAG : GET THE CURRENT SETTINGS  
1905 : POP DS : RECOVER SEGMENT  
1906 : IRET : RETURN TO CALLER  
1907 :-----  
1908 : EQUIPMENT ENDP  
1909 :  
1910 :-----  
1911 : INT 15  
1912 :  
1913 :-----  
1914 : CASSETTE_ID0_10_1  
1915 :  
1916 :-----  
1917 : NMI_INT_1 PROC NEAR  
1918 : ASSUME CS:DATA  
1919 : PUSH AX : SAVE ORIG CONTENTS OF AX  
1920 : IN AL,PORT_C :  
1921 : TEST AL,0COH : PARITY CHECK?  
1922 : JNZ NMI_1 :  
1923 : JMP D14 : NO, EXIT FROM ROUTINE  
1924 :  
1925 :-----  
1926 : NMI_1:  
1927 :  
1928 : D13:  
1929 : MOV DX,DATA : INIT AND SET MODE FOR VIDEO  
1930 : MOV DS,DX :  
1931 : MOV SI,OFFSET_D1 : ADDR OF ERROR MSG  
1932 : TEST AL,40H : I/O PARITY CHECK  
1933 : JNZ D13 : DISPLAY ERROR MSG  
1934 : MOV SI,OFFSET_D2 : MUST BE PLANAR  
1935 :  
1936 : D13:  
1937 : MOV AH,0 :  
1938 : MOV AL,0C0H : CALL VIDEO_I0 PROCEDURE  
1939 : INT 10H :  
1940 : CALL P_MSG : PRINT ERROR MSG  
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 :-----<
```

```

1933
1934
1935 ;----- SEE IF LOCATION THAT CAUSED PARITY CHECK CAN BE FOUND
1936 187F B0 00      MOV    AL,00H          ; DISABLE TRAP
1937 1881 E6 A0      OUT   0A0H,AL
1938 1883 E4 61      IN    AL,PORT_B
1939 1885 D0 30      OR    AL,0000000B
1940 1887 26 11      OUT   PORT_B,AL
1941 1889 24 CF      AND   AL,1011111B
1942 188B E6 61      OUT   PORT_B,AL
1943 188D BB 0E 0013 R MOV   BX,MEMORY_SIZE
1944 1891 FC          CLD
1945 1892 2B D2      SUB   DX,DX
1946
NMI_LOOP:           ; POINT DX AT START OF MEM
1947 1894 BE DA      MOV   DS,DX
1948 1896 BE C2      MOV   ES,DX
1949 1898 B9 4000     MOV   CX,4000H
1950 1898 2B F6      SUB   SI,SI
1951
1952 189D F3 AC      REP   LODSB
1953 189F E4 62      IN    AL,PORT_C
1954 18A1 24 C0      AND   AL,1100000B
1955 18A3 75 11      JNZ   PRT_NMI
1956 18A5 B1 C2 0400 ADD   DX,0400H
1957 18A9 B3 E0 10    SUB   BX,16D
1958 18B0 26 E6      JNZ   NMI_LOOP
1959 18B4 BE 1902 R  MOV   SI,TOFFSET D2A
1960 18B1 E0 1997 R  CALL  P_MSG
1961 18B4 FA          CLC
1962 18B5 F4          HLT
1963 18B6 00          ; HALT SYSTEM
1964 18B6 8C DA      MOV   DX,DS
1965 18B6 E8 0CBA R  CALL  PRT_SEG
1966 18B8 FA          CLI
1967 18BC F4          HLT
1968 18BD
1969 18B8 58
1970 18B8 CF
1971 18BF
1972
1973
1974
1975
1976 18BF B9 0000     ;----- ROS CHECKSUM SUBROUTINE
1977 18BF B9 0000     ROS_CHECKSUM PROC NEAR
1978 18C2              MOV   CX,0
1979 18C0 32 C0      ROS_CHECKSUM_CNT
1980 18C4              XOR   AL,AL
1981 18C2 02 07      C26:
1982 18C6 43          ADD   AL,DS:[BX]
1983 18C7 E2 FB      INC   BX
1984 18C9 0A C0      LOOP  C26
1985 18C8 C3          OR    AL,AL
1986 18CC              RET
ROS_CHECKSUM ENDP
1987
1988
1989
1990 18C C3 30 31 0D 0A  E0 DB  '101',CR,LF
1991 18D1 20 32 30 31 0D 0A  E1 DB  '201',CR,LF
1992 18D7 52 4F 0D 0A  F3A DB  'ROM',CR,LF
1993 18D6 31 38 30 0D 0A  F3C DB  '1801',CR,LF
1994 18E2 50 41 52 49 54 59  D1 DB  'PARITY CHECK 2',CR,LF
1995 20 43 45 43 48
1996 20 32 00 0A
1997 18F2 50 41 52 49 54 59  D2 DB  'PARITY CHECK 1',CR,LF
1998 20 43 43 45 43 48
1999 20 31 0D 0A
2000 1902 3F 3F 3F 3F 0D  D2A DB  '??????',CR,LF
2001 0A
2002
2003
2004
2005
2006
2007
2008 1909     ;----- BLINK LED PROCEDURE FOR MFG RUN-IN TESTS
2009 1904 FB          ASSUME DS:DATA
2010 190A 50          BLINK_INT PROC NEAR
2011 190B E4 61      ST1   PUSH AX
2012 190D E0          IN    AL,PORT_B
2013 190F F4 D0      MOV   AH,AL
2014 1911 24 40      NOT   AL
2015 1913 80 E4 BF  AND   AL,0100000B
2016 1916 0A C4      AND   AH,1011111B
2017 1918 E6 61      OR    AL,AH
2018 191A 00 20      OUT   PORT_B,AL
2019 191C E0 20      MOV   AX,ED1
2020 191E 58          OUT   INTA00,AL
2021 191F CF          POP   AX
2022 1920              IRET
BLINK_INT ENDP
2023
2024
2025
2026
2027
2028 1920     ;----- THIS ROUTINE CHECKSUMS OPTIONAL ROM MODULES AND
2029 1920 BB ----- R ; IF CHECKSUM IS OK, CALLS INIT/TEST CODE IN MODULE
2030 1923 8C C0          ROM_CHECK PROC NEAR
2031 1925 2A E4      MOV   AX,DATA
2032 1927 8A 47 02    MOV   ES,DX
2033 192A B1 09      SUB   AH,AH
2034 192C D3 E0      MOV   AL,[BX+2]
2035 192D 88 C8      SHL   AX,CL
2036 1930 00 00      MOV   CL,09H
2037 1931 B9 0004    PUSH  CX,AX
2038 1934 D3 E8      MOV   CX,4
2039 1936 03 D0      SHR   AX,CL
2040 1938 59          ADD   DX,AX
2041 193A 00 00      POP   CX
2042 193C 24 06      CALL  ROS_CHECKSUM_CNT
2043 193E EB 0746 R  JZ    ROM_CHECK_1
2044 1941 EB 14 90    CALL  ROM_ERR
2045 1944 JMP   ROM_CHECK_END
ROM_CHECK_1:          PUSH  DX
2046 1944 52          ;----- POINT ES TO DATA AREA
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
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
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2778
2779
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3039
3039
3040
3041
3042
3043
3044
3045
3046
3047
3048
3049
3049
3050
3051
3052
3053
3054
3055
3056
3057
3058
3059
3059
3060
3061
3062
3063
3064
3065
3066
3067
3068
3069
3069
3070
3071
3072
3073
3074
3075
3076
3077
3078
3079
3079
3080
3081
3082
3083
3084
3085
3086
3087
3088
3089
3089
3090
3091
3092
3093
3094
3095
3096
3097
3098
3099
3099
3100
3101
3102
3103
3104
3105
3106
3107
3108
3109
3109
3110
3111
3112
3113
3114
3115
3116
3117
3118
3119
3119
3120
3121
3122
3123
3124
3125
3126
3127
3128
3129
3129
3130
3131
3132
3133
3134
3135
3136
3137
3138
3139
3139
3140
3141
3142
3143
3144
3145
3146
3147
3148
3149
3149
3150
3151
3152
3153
3154
3155
3156
3157
3158
3159
3159
3160
3161
3162
3163
3164
3165
3166
3167
3168
3169
3169
3170
3171
3172
3173
3174
3175
3176
3177
3178
3179
3179
3180
3181
3182
3183
3184
3185
3186
3187
3188
3189
3189
3190
3191
3192
3193
3194
3195
3196
3197
3198
3199
3199
3200
3201
3202
3203
3204
3205
3206
3207
3208
3209
3209
3210
3211
3212
3213
3214
3215
3216
3217
3218
3219
3219
3220
3221
3222
3223
3224
3225
3226
3227
3228
3229
3229
3230
3231
3232
3233
3234
3235
3236
3237
3238
3239
3239
3240
3241
3242
3243
3244
3245
3246
3247
3248
3249
3249
3250
3251
3252
3253
3254
3255
3256
3257
3258
3259
3259
3260
3261
3262
3263
3264
3265
3266
3267
3268
3269
3269
3270
3271
3272
3273
3274
3275
3276
3277
3278
3279
3279
3280
3281
3282
3283
3284
3285
3286
3287
3288
3289
3289
3290
3291
3292
3293
3294
3295
3296
3297
3298
3299
3299
3300
3301
3302
3303
3304
3305
3306
3307
3308
3309
3309
3310
3311
3312
3313
3314
3315
3316
3317
3318
3319
3319
3320
3321
3322
3323
3324
3325
3326
3327
3328
3329
3329
3330
3331
3332
3333
3334
3335
3336
3337
3338
3339
3339
3340
3341
3342
3343
3344
3345
3346
3347
3348
3349
3349
3350
3351
3352
3353
3354
3355
3356
3357
3358
3359
3359
3360
3361
3362
3363
3364
3365
3366
3367
3368
3369
3369
3370
3371
3372
3373
3374
3375
3376
3377
3378
3379
3379
3380
3381
3382
3383
3384
3385
3386
3387
3388
3389
3389
3390
3391
3392
3393
3394
3395
3396
3397
3398
3399
3399
3400
3401
3402
3403
3404
3405
3406
3407
3408
3409
3409
3410
3411
3412
3413
3414
3415
3416
3417
3418
3419
3419
3420
3421
3422
3423
3424
3425
3426
3427
3428
3429
3429
3430
3431
3432
3433
3434
3435
3436
3437
3438
3439
3439
3440
3441
3442
3443
3444
3445
3446
3447
3448
3449
3449
3450
3451
3452
3453
3454
3455
3456
3457
3458
3459
3459
3460
3461
3462
3463
3464
3465
3466
3467
3468
3469
3469
3470
3471
3472
3473
3474
3475
3476
3477
3478
3479
3479
3480
3481
3482
3483
3484
3485
3486
3487
3488
3489
3489
3490
3491
3492
3493
3494
3495
3496
3497
3498
3499
3499
3500
3501
3502
3503
3504
3505
3506
3507
3508
3509
3509
3510
3511
3512
3513
3514
3515
3516
3517
3518
3519
3519
3520
3521
3522
3523
3524
3525
3526
3527
3528
3529
3529
3530
3531
3532
3533
3534
3535
3536
3537
3538
3539
3539
3540
3541
3542
3543
3544
3545
3546
3547
3548
3549
3549
3550
3551
3552
3553
3554
3555
3556
3557
3558
3559
3559
3560
3561
3562
3563
3564
3565
3566
3567
3568
3569
3569
3570
3571
3572
3573
3574
3575
3576
3577
3578
3579
3579
3580
3581
3582
3583
3584
3585
3586
3587
3588
3589
3589
3590
3591
3592
3593
3594
3595
3596
3597
3598
3599
3599
3600
3601
3602
3603
3604
3605
3606
3607
3608
3609
3609
3610
3611
3612
3613
3614
3615
3616
3617
3618
3619
3619
3620
3621
3622
3623
3624
3625
3626
3627
3628
3629
3629
3630
3631
3632
3633
3634
3635
3636
3637
3638
3639
3639
3640
3641
3642
3643
3644
3645
3646
3647
3648
3649
3649
3650
3651
3652
3653
3654
3655
3656
3657
3658
3659
3659
3660
3661
3662
3663
3664
3665
3666
3667
3668
3669
3669
3670
3671
3672
3673
3674
3675
3676
3677
3678
3679
3679
3680
3681
3682
3683
3684
3685
3686
3687
3688
3689
3689
3690
3691
3692
3693
3694
3695
3696
3697
3698
3699
3699
3700
3701
3702
3703
3704
3705
3706
3707
3708
3709
3709
3710
3711
3712
3713
3714
3715
3716
3717
3718
3719
3719
3720
3721
3722
3723
3724
3725
3726
3727
3728
3729
3729
3730
3731
3732
3733
3734
3735
3736
3737
3738
3739
3739
3740
3741
3742
3743
3744
3745
3746
3747
3748
3749
3749
3750
3751
3752
3753
3754
3755
3756
3757
3758
3759
3759
3760
3761
3762
3763
3764
3765
3766
3767
3768
3769
3769
3770
3771
3772
3773
3774
3775
3776
3777
3778
3779
3779
3780
3781
3782
3783
3784
3785
3786
3787
3788
3789
3789
3790
3791
3792
3793
3794
3795
3796
3797
3798
3799
3799
3800
3801
3802
3803
3804
3805
3806
3807
3808
3809
3809
3810
3811
3812
3813
3814
3815
3816
3817
3818
3819
3819
3820
3821
3822
3823
3824
3825
3826
3827
3828
3829
3829
3830
3831
3832
3833
3834
3835
3836
3837
3838
3839
3839
3840
3841
3842
3843
3844
3845
3846
3847
3848
3849
3849
3850
3851
3852
3853
3854
3855
3856
3857
3858
3859
3859
3860
3861
3862
3863
3864
3865
3866
3867
3868
3869
3869
3870
3871
3872
3873
3874
3875
3876
3877
3878
3879
3879
3880
3881
3882
3883
3884
3885
3886
3887
3888
3889
3889
3890
3891
3892
3893
3894
3895
3896
3897
3898
3899
3899
3900
3901
3902
3903
3904
3905
3906
3907
3908
3909
3909
3910
3911
3912
3913
3914
3915
3916
3917
3918
3919
3919
3920
3921
3922
3923
3924
3925
3926
3927
3928
3929
3929
3930
3931
3932
3933
3934
3935
3936
3937
3938
39
```

```
2047 1945 26: CT 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   DWDR PTR ES:10_ROM_INIT ; CALL INIT./TEST ROUTINE
2050 1952 5A                POP    DX
2051 1957 ROM_CHECK END:      RET    ; RETURN TO CALLER
2052 1957 C3
2053 1958 ROM_CHECK ENDP
2054
2055
2056
2057
2058
2059
2060 1958 26: CT 06 0067 R 0003    MOV    ES:10_ROM_INIT,0003H ; LOAD OFFSET
2061 1958 26: 8C 1E 0069 R    MOV    ES:10_ROM_SEG,DS ; LOAD SEGMENT
2062 1959 B1 04                CALL   DWDR PTR ES:10_ROM_INIT ; CALL INIT./TEST ROUTINE
2063 1958 D2 E8
2064 195D E8 1963 R
2065 1960 5B
2066 1961 24 0F
2067
2068 1963 XLAT_PR PROC NEAR
2069 1963 04 90                PUSH   AX ; SAVE FOR LOW NIBBLE DISPLAY
2070 1965 27                ADD    AL,090H ; SHL COUNT BY 4
2071 1966 14 40                SHR    CL ; HIGH NIBBLE SWAP
2072 1967 27                CALL   XLAT_PR ; DO THE HIGH NIBBLE DISPLAY
2073 1967 ADC    AL,040H ; RECOVER THE NIBBLE
2074 1968 27                AND    AL,0FH ; ISOLATE TO LOW NIBBLE
2075
2076 1963 PRT_HEX PROC NEAR
2077 1967 B4 0E                ADD    AH,14 ; FALL INTO LOW Nibble CONVERSION
2078 1968 B7 00                MOV    BH,0 ; CONVERT 00H TO ASCII CHARACTER
2079 196D CD 10                INT    10H ; ADD FIRST CONVERSION FACTOR
2080
2081 1967 C3                RET    ; ADJUST FOR NUMERIC AND ALPHA RANGE
2082 1970 PRT_HEX ENDP
2083 1970 XLAT_PR ENDP
2084 1970 XPC_BYTc ENDP
2085 1974 0278
2086 1976 F4 LABEL WORD ; PRINTER SOURCE TABLE
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096 1976 : THIS SUBROUTINE WILL PRINT A MESSAGE ON THE DISPLAY
2097 1976 BB EE
2098 1976 00 9997 R
2099 1978 1E
2100 197C E8 1A12 R
2101 197F A0 0010 R
2102 1982 24 01
2103 1984 75 0F
2104 1985 MFG_HALT PROC NEAR
2105 1986 FA
2106 1987 B0 89
2107 1988 E6 63
2108 1988 B0 85
2109 198D E6 61
2110 1990 A0 0015 R
2111 1992 00 60
2112 1994 F4
2113 1995
2114 1995 1F
2115 1995 C3
2116 1997
2117
2118 1997 P_MSG PROC NEAR
2119 1997 GT2A: ; PUT CHAR IN AL
2120 1997 2E: 8A 04 ; POINT TO NEXT CHAR
2121 199A 46 ; HAVE PRINT CHAR
2122 199B 40 ; CALL VIDEO_10
2123 199C E8 1969 R ; DISABLE KB
2124 199F 58 ; PORT_B_AL
2125 19A0 3C 0A ; AL,_MFG_ERR_FLAG
2126 19A2 75 F3 ; PORT_A_AL
2127 19A4 C3 ; HALT
2128 19A5 ; G12: ; YES - HALT SYSTEM
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139 1945 : THIS PROCEDURE WILL ISSUE LONG TONES (1-3/4 SECONDS) AND ONE OR
2140 1945 9C ; MORE SHORT TONES (9/32 SECOND) TO INDICATE A FAILURE ON THE
2141 1946 FA ; PLANAR BOARD, A BAD MEMORY MODULE, OR A PROBLEM WITH THE CRT.
2142 1947 0A F6 ; ENTRY PARAMETERS:
2143 1949 74 1E ; DH = NUMBER OF LONG TONES TO BEEP,
2144 194B ; DS = NUMBER OF SHORT TONES TO BEEP.
2145 194B B3 T0 ; G1: ; LONG BEEPS
2146 194B 0500 ; PUSH DS ; SAVE FLAGS
2147 1980 E8 0C5C R ; CLI ; DISABLING SYSTEM INTERRUPTS
2148 1983 B9 C233 ; OR DH,DH ; ANY LONG ONES TO BEEP
2149 1984 B8 0CA0 R ; JZ G3 ; NO, DO THE SHORT ONES
2150 1987 FE CE ; DEC DH ; LONG BEEPS
2151 1988 75 EE ; JNZ G1 ; COUNTER FOR LONG BEEPS (1-3/4 SECONDS)
2152 198D 40 0000 ; MOV BL,112 ; DIVIDER FOR 932 Hz
2153 198E E8 1A12 R ; CALL BEEP ; ONE-TIME BEEP
2154 19C1 80 3E 0012 R 01 ; MOV CX,49715 ; 2/3 SECOND DELAY AFTER LONG BEEP
2155 19C8 1F ; CALL WAITF ; DELAY BETWEEN BEEPS
2156 19C8 74 BD ; JE MFG_HALT ; ANY MORE LONG BEEPS TO DO
2157 19C8 40 0000 ; LOOP TILL DONE
2158 19C9 B3 12 ; PUSH DS ; SAVE DS REGISTER CONTENTS
2159 19C8 B9 0488 ; CALL DDS ; MANUFACTURING TEST MODE?
2160 19C8 E8 0C5C R ; CMP _MFG_TST,01H ; RESTORE ORIGINAL CONTENTS OF (DS)
                                ; POP DS ; YES - STOP BLINKING LED
                                ; JE MFG_HALT ; SHORT BEEPS
                                ; CALL BEEP ; COUNTER FOR SHORT BEEP (9/32)
                                ; DIVIDER FOR 967 Hz
                                ; CALL BEEP ; DO THE SOUND
ERR_BEEP PROC NEAR
```

```

2161 19D1 B9 8178      MOV    CX,33144        ; 1/2 SECOND DELAY AFTER SHORT BEEP
2162 19D0 EB 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 19D8 B9 8178      MOV    CX,33144        ; 1/2 SECOND DELAY AFTER LAST BEEP
2166 19E0 EB 0CA0 R    CALL   WAITF         ; MAKE IT ONE SECOND DELAY BEFORE RETURN
2167 19E1 9D            RET    PF             ; RESTORE FLAGS TO ORIGINAL SETTINGS
2168 19E2 C3            RET    RET             ; RETURN TO CALLER
2169 19E3          ERR_BEEP    ENDP

2170
2171
2172
2173
2174
2175 19E3          KBD_RESET    PROC NEAR
2176          ASSUME DS:AB50
2177 19E3 BD 08        MOV    AL,08H        ; SET KBD CLK LINE LOW
2178 19E3 E6 01        OUT   PORT_B,AL     ; WRITE 8255 PORT B
2179 19E7 B9 2956      MOV    CX,10582       ; HOLD KBD CLK LOW FOR 20 MS
2180 19E4          G8:           LOOP  G8          ; LOOP FOR 20 MS
2181 19E4 E2 FE        MOV    AL,OCBH       ; SET CLK, ENABLE LINES HIGH
2182 19E5 B0 C8        OUT   PORT_B,AL     ; ENTRY FOR MANUFACTURING TEST 2
2183 19E5 E6 61        MOV    AL,0FDH       ; SET KBD CLK HIGH, ENABLE LOW
2184 19E5 F0 D0        OUT   PORT_B,AL     ; ENABLE KEYBOARD INTERRUPTS
2185 19F0 B0 48        MOV    AL,48H        ; WRITE 8259 IMR
2186 19F2 E6 61        OUT   PORT_B,AL     ; READ SCAN CODE RETURNED
2187 19F4 B0 FD        MOV    AL,0FDH       ; ENABLE INTERRUPTS
2188 19F6 E6 21        OUT   INTA01,AL     ; SETUP INTERRUPT TIMEOUT CNT
2189 19F8 C6 06 046B R 00  MOV    DATA_AREA[INTR_FLAG-DATA40],0 ; RESET INTERRUPT INDICATOR
2190 19F9 B0 FB        STI    CX,CX         ; ENABLE INTERRUPTS
2191 19F9 C8 29        SUB   CX,CX         ; SETUP INTERRUPT TIMEOUT CNT
2192 1A00          G9:           TEST  DATA_AREA[INTR_FLAG-DATA40],02H ; DID A KEYBOARD INTN OCCUR?
2193 1A00 F6 06 046B R 02  JNZ   G10          ; YES - READ SCAN CODE RETURNED
2194 1A05 T5 02        LOOP  G9          ; NO - LOOP TILL TIMEOUT
2195 1A05 E2 F7        G10:           TEST  DATA_AREA[INTR_FLAG-DATA40],02H ; DID A KEYBOARD INTN OCCUR?
2196 1A09              JNZ   G10          ; YES - READ SCAN CODE JUST READ
2197 1A09 E4 60        IN    AL,PORT_A     ; CLEAR KEYBOARD
2198 1A09 B8 D8        MOV    BL,AL         ; READ KEYBOARD SCAN CODE
2199 1A09 B0 C8        MOV    AL,OCBH       ; SAVE SCAN CODE JUST READ
2200 1A09 E6 61        OUT   PORT_B,AL     ; RETURN TO CALLER
2201 1A11 C3            RET    RET             ; RETURN TO CALLER
2202 1A12          KBD_RESET    ENDP

2203
2204 1A12 DDS          PROC NEAR
2205 1A12 2E: 8E 1E 1A18 R  DDS  DS,CS:DDSDATA ; LOAD (DS) TO DATA AREA
2206 1A17 C3            MOV    DS,CS:DDSDATA ; PUT SEGMENT VALUE OF DATA AREA INTO DS
2207
2208 1A18 ---- R        DDSDATA DW DATA      ; RETURN TO USER WITH (DS)= DATA
2209
2210 1A1A DDS          ENDP
2211
2212          ;-- HARDWARE INT 08 H -- ( IRQ LEVEL 0 ) --
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228          ASSUME CS:CODE,DS:DATA
2229 1A1A          TIMER_INT_1 PROC NEAR
2230 1A1A FB          STI    INT             ; INTERRUPTS BACK ON
2231 1A1B 1E          PUSH  DS             ; SAVE MACHINE STATE
2232 1A1C 00          PUSH  AX             ; GET ADDRESS OF DATA SEGMENT
2233 1A1D 52          PUSH  DX             ; ESTABLISH ADDRESSTABILITY
2234 1A1E B8 ---- R  MOV    AX,DATA        ; INCREMENT TIME
2235 1A21 8E D8        MOV    DS,AX         ; GO TO TEST_DAY
2236 1A23 FF 06 006C R INC   @TIMER_LOW    ; INCREMENT DAY WORD OF TIME
2237 1A29 FF 06 006E R JNZ   T4             ; TEST DAY
2238 1A2D              INC   @TIMER_HIGH   ; TEST FOR COUNT EQUALING 24 HOURS
2239 1A2D              T4:           CMP   @TIMER_HIGH,01BH ; GO TO DISKETTE_CTRL
2240 1A2D 83 3E 0006E R 18  JNZ   T5             ; GO TO DISKETTE_CTRL
2241 1A32 75 19        CMP   @TIMER_LOW,0B0H ; TEST FOR COUNT NOT OUT
2242 1A34 81 3E 0006C R 00B0 JNZ   T5             ; TURN OFF MOTOR RUNNING BITS
2243 1A3A 75 11        INC   @DAY_COUNT    ; SET TIMER ELAPSED 24 HOURS FLAG
2244
2245          ;---- TEST FOR DISKETTE TIME OUT
2246
2247 1A3C 2B C0        SUB   AX,AX         ; INCREMENT ELAPSED DAY COUNTER
2248 1A3E A3 006E R    MOV    @TIMER_HIGH,AX ; CLEAR TIMER COUNT HIGH
2249 1A41 A3 006C R    MOV    @TIMER_LOW,AX ; AND LOW
2250 1A44 C6 06 0070 R 01  MOV    @TIMER_OFL,! ; SET TIMER ELAPSED 24 HOURS FLAG
2251 1A49 FF 06 00CE R INC   @DAY_COUNT    ; INCREMENT ELAPSED DAY COUNTER
2252
2253
2254
2255 1A40          T5:           DEC   #MOTOR_COUNT ; DECREMENT DISKETTE MOTOR CONTROL
2256 1A40 FE 0E 0040 R JNZ   T6             ; RETURN IF COUNT NOT OUT
2257 1A51 75 0B        AND   #MOTOR_STATUS,0F0H ; TURN OFF MOTOR RUNNING BITS
2258 1A53 B0 26 003F R F0  MOV    AL,0CH-        ; FDC CTL PORT
2259 1A58 B0 0C        MOV    DX,03F2H       ; TURN OFF THE MOTOR
2260 1A5A BA 03F2      OUT   DX,AL         ; TRANSFER CONTROL TO A USER ROUTINE
2261 1A5D EE            INT    ICH            ; TIMER TICK INTERRUPT
2262
2263 1A5E CD IC        T6:           INT    ICH            ; DISABLE INTERRUPTS TILL STACK CLEARED
2264 1A5E CD IC        CLI             ; GET END OF INTERRUPT MASK
2265
2266 1A60 FA            MOV    AL,E0I         ; END OF INTERRUPT TO 8259 - 1
2267 1A61 B0 20          OUT   INTA00,AL     ; RESTORE (DX)
2268 1A63 20            POP   DX             ; RESET MACHINE STATE
2269 1A65 5A            POP   AX             ; RETURN FROM INTERRUPT
2270 1A66 58            POP   DS             ; RESTORE (DX)
2271 1A67 1F            IRET
2272 1A68 CF            RET    PF             ; RETURN FROM INTERRUPT
2273
2274 1A69          TIMER_INT_1 ENDP

```

PAGE		CHARACTER GENERATOR GRAPHICS FOR 320X200 AND 640X200 GRAPHICS								
2275		I								
2276		I								
2277		I								
2278		I								
2279		I								
2280	I4E6	I								
2281	I4E6	CRT_CHAR	GEN	LABEL	BYTE					
2282	I4E6	00	00	00	00	00	00	D_00		
2283	00	00						BLANK		
2284	I4E7	TE	B1	A5	B1	BD	99	DB		
2285	TE	FF	FE	DB	FF	C3	E7	DB		
2287	FT	FE						07EH,0F0H,0FFH,0BCH,0FFH,0C3H,0E7H,0FFH,0TEH		
2288	I4E8	6C	FE	FE	TC	3C		DB		
2289	10	00						06CH,0FEH,0FEH,0FEH,0TC,03H,010H,000H		
2290	I4E8	10	3C	TC	FE	TC	36	DB		
2291	10	00						010H,038H,07CH,0FEH,07CH,038H,010H,000H		
2292	I4E8	38	TC	38	FE	FE	TC	DB		
2293	38	TC						038H,07CH,038H,0FEH,0FEH,07CH,038H,07CH		
2294	I4E8	10	10	38	TC	FE	TC	DB		
2295	38	TC						010H,010H,038H,07CH,0FEH,07CH,038H,07CH		
2296	I4E8	00	10	18	3C	3C	18	DB		
2297	00	00						000H,000H,018H,03CH,03CH,018H,000H,000H		
2298	I4E8	FF	FF	FE	E7	C3	E7	DB		
2299	00							0FFH,0FFH,0E7H,0C3H,03CH,0E7H,0FFH,0FFH		
2300	I4E8	00	3C	66	42	42	66	DB		
2301	3C	00						000H,03CH,066H,042H,042H,066H,03CH,000H		
2302	I4E8	00	3C	63	99	BD	99	DB		
2303	00	00						0FFH,0C3H,099H,0BDH,0BDH,099H,0C3H,0FFH		
2304	I4CF	0F	07	0F	7D	CC	CC	DB		
2305	CC	78						00FH,007H,00FH,07DH,0CCH,0CCH,0CCH,078H		
2306	I4E8	3C	66	66	66	3C	18	DB		
2307	3C	18						03CH,066H,066H,066H,03CH,018H,0TEH,018H		
2308	I4D6	00	00	00	3F	30	30	70	DB	
2309	00	00						03FH,033H,0FH,030H,030H,070H,0F0H,0E0H		
2310	I4E7	7F	63	7F	63	63	67	DB		
2311	E6	C0						07FH,063H,07FH,063H,063H,067H,0E6H,0C0H		
2312	I4E8	99	5A	3C	ET	E7	3C	DB		
2313	5A	99						099H,05AH,03CH,0E7H,0E7H,03CH,05AH,099H		
2314								D_OF		
2315	I4E8	00	E0	F8	FE	F8	E0	DB		
2316	00	00						080H,0E0H,0F8H,0FEH,0F8H,0E0H,080H,000H		
2317	I4E8	02	00	00	3E	3E	0E	DB		
2318	02	00						002H,00EH,03EH,0FEH,03EH,00EH,002H,000H		
2319	I4E8	00	00	00	37	7E	18	7E	DB	
2320	00	00						018H,03CH,07EH,018H,018H,07EH,03CH,018H		
2321	I4B6	66	66	66	66	66	66	00	DB	
2322	66	00						066H,066H,066H,066H,066H,066H,000H,066H		
2323	I4B7	FD	DB	7B	1B	1B	1B	DB		
2324	1B	00						07FH,0DBH,0DBH,07BH,01BH,01BH,01BH,000H		
2325	I4B6	00	00	00	38	6C	3C	38	DB	
2326	00	00						03EH,063H,038H,06CH,06CH,038H,0CCH,078H		
2327	I4B6	00	00	00	00	00	TE	7E	DB	
2328	TE	00						000H,000H,000H,000H,000H,000H,000H,000H		
2329	I4B6	18	3C	18	TE	18	3C	18	DB	
2330	18	FF						018H,03CH,07EH,018H,018H,07EH,03CH,018H		
2331	I4B6	00	00	00	00	00	TE	18	18	DB
2332	18	00						018H,03CH,07EH,018H,018H,018H,018H,000H		
2333	I4B6	18	18	18	18	TE	3C	18	DB	
2334	18	00						018H,01BH,01BH,01BH,01BH,01BH,01BH,000H		
2335	I4B6	00	00	00	00	00	TE	18	18	DB
2336	00	00						000H,01BH,000H,000H,000H,000H,000H,000H		
2337	I4B6	00	00	00	00	60	60	30	DB	
2338	00	00						000H,030H,060H,0F0H,060H,030H,000H,000H		
2339	I4B6	00	00	00	00	00	00	00	DB	
2340	00	00						000H,000H,000H,000H,000H,000H,000H,000H		
2341	I4B5	00	00	24	66	FF	66	24	DB	
2342	00	00						000H,024H,066H,0FFH,066H,024H,000H,000H		
2343	I4B5	00	00	18	3C	TE	FF	FF	DB	
2344	00	00						000H,018H,03CH,07EH,0FFH,0FFH,000H,000H		
2345	I4B6	00	00	FF	FF	7E	3C	18	DB	
2346	00	00						000H,0FFH,0FFH,07EH,03CH,018H,000H,000H		
2347								D_IF		
2348	I4B6	00	00	00	00	00	00	00	DB	
2349	00	00						000H,000H,000H,000H,000H,000H,000H,000H		
2350	I4B6	30	78	78	30	30	00	00	DB	
2351	30	00						030H,078H,078H,030H,030H,000H,030H,000H		
2352	I4B6	1C	6C	6C	00	00	00	00	DB	
2353	00	00						06CH,06CH,06CH,000H,000H,000H,000H,000H		
2354	I4B6	00	00	00	00	00	00	00	DB	
2355	I4B6	3C	7C	7C	00	00	00	00	DB	
2356	30	00						030H,07CH,0C0H,078H,0C0H,0FFH,030H,000H		
2357	30	00						000H,000H,000H,000H,000H,000H,000H,000H		
2358	I4B6	00	00	00	00	00	00	00	DB	
2359	C0	00						000H,00CH,00CH,018H,030H,066H,0C6H,000H		
2360	I4B6	00	00	00	00	00	00	00	DB	
2361	I4B6	00	00	00	38	76	DC	CC	DB	
2362	I4B6	60	60	60	00	00	00	00	DB	
2363	00	00						060H,060H,0C0H,000H,000H,000H,000H,000H		
2364	I4B6	18	30	60	60	60	30	30	DB	
2365	30	00						018H,030H,060H,060H,060H,030H,018H,000H		
2366	I4B6	00	00	00	00	18	18	30	DB	
2367	00	00						060H,030H,018H,018H,018H,030H,060H,000H		
2368	I4B6	00	00	00	00	3C	FF	3C	66	DB
2369	00	00						000H,066H,03CH,0FFH,03CH,066H,000H,000H		
2370	I4B6	00	00	00	30	30	30	30	DB	
2371	I4B6	00	00	00	00	00	00	00	DB	
2372	I4CE	00	00	00	00	00	00	30	DB	
2373	30	60						000H,000H,000H,000H,000H,030H,030H,060H		
2374	I4B6	00	00	00	00	FC	00	00	DB	
2375	00	00						000H,000H,000H,000H,000H,000H,000H,000H		
2376	I4B6	00	00	00	00	00	00	30	DB	
2377	I4B6	00	00	00	00	00	00	00	DB	
2378	I4B6	00	00	00	18	30	60	C0	DB	
2379	80	00						006H,00CH,018H,030H,060H,0C0H,080H,000H		
2380								D_2F / SLASH		
2381	I4B6	TC	C6	CE	D6	E6			DB	
2382	TC	00	00	00	00	00	00	00	DB	
2383	I4B6	30	70	30	30	30	30	30	DB	
2384	FC	00						030H,070H,030H,030H,030H,030H,030H,030H		
2385	I4B6	TC	00	CC	30	60	CC		DB	
2386	FC	00						078H,0CCH,0CCH,038H,060H,0CCH,0CCH,0FFH		
2387	I4C6	TC	00	CC	30	0C	CC		DB	
2388	I4B6	TC	C6	CE	D6	E6			07CH,0C6H,0CEH,0DEH,0F6H,06EH,07CH,000H	
2389	TC	00	00	00	00	00	00	00	DB	
2390	I4B6	30	70	30	30	30	30	30	DB	
2391	FC	00						030H,070H,030H,030H,030H,030H,030H,030H		
2392	I4B6	TC	00	CC	30	60	CC		DB	
2393	FC	00						078H,0CCH,0CCH,038H,060H,0CCH,0CCH,0FFH		
2394	I4C6	TC	00	CC	30	0C	CC		DB	
2395	I4B6	TC	C6	CE	D6	E6			07CH,0C6H,0CEH,0DEH,0F6H,06EH,07CH,000H	
2396	TC	00	00	00	00	00	00	00	DB	
2397	I4B6	TC	C6	CE	D6	E6			07CH,0C6H,0CEH,0DEH,0F6H,06EH,07CH,000H	
2398	TC	00	00	00	00	00	00	00	DB	
2399	I4B6	TC	C6	CE	D6	E6			07CH,0C6H,0CEH,0DEH,0F6H,06EH,07CH,000H	
2400	TC	00	00	00	00	00	00	00	DB	
2401	I4B6	TC	C6	CE	D6	E6			07CH,0C6H,0CEH,0DEH,0F6H,06EH,07CH,000H	
2402	TC	00	00	00	00	00	00	00	DB	
2403	I4B6	TC	C6	CE	D6	E6			07CH,0C6H,0CEH,0DEH,0F6H,06EH,07CH,000H	
2404	TC	00	00	00	00	00	00	00	DB	
2405	I4B6	TC	C6	CE	D6	E6			07CH,0C6H,0CEH,0DEH,0F6H,06EH,07CH,000H	
2406	TC	00	00	00	00	00	00	00	DB	
2407	I4B6	TC	C6	CE	D6	E6			07CH,0C6H,0CEH,0DEH,0F6H,06EH,07CH,000H	
2408	TC	00	00	00	00	00	00	00	DB	
2409	I4B6	TC	C6	CE	D6	E6			07CH,0C6H,0CEH,0DEH,0F6H,06EH,07CH,000H	
2410	TC	00	00	00	00	00	00	00	DB	
2411	I4B6	TC	C6	CE	D6	E6			07CH,0C6H,0CEH,0DEH,0F6H,06EH,07CH,000H	
2412	TC	00	00	00	00	00	00	00	DB	
2413	I4B6	TC	C6	CE	D6	E6			07CH,0C6H,0CEH,0DEH,0F6H,06EH,07CH,000H	
2414	TC	00	00	00	00	00	00	00	DB	
2415	I4B6	TC	C6	CE	D6	E6			07CH,0C6H,0CEH,0DEH,0F6H,06EH,07CH,000H	
2416	TC	00	00	00	00	00	00	00	DB	
2417	I4B6	TC	C6	CE	D6	E6			07CH,0C6H,0CEH,0DEH,0F6H,06EH,07CH,000H	
2418	TC	00	00	00	00	00	00	00	DB	
2419	I4B6	TC	C6	CE	D6	E6			07CH,0C6H,0CEH,0DEH,0F6H,06EH,07CH,000H	
2420	TC	00	00	00	00	00	00	00	DB	
2421	I4B6	TC	C6	CE	D6	E6			07CH,0C6H,0CEH,0DEH,0F6H,06EH,07CH,000H	
2422	TC	00	00	00	00	00	00	00	DB	
2423	I4B6	TC	C6	CE	D6	E6			07CH,0C6H,0CEH,0DEH,0F6H,06EH,07CH,000H	
2424	TC	00	00	00	00	00	00	00	DB	
2425	I4B6	TC	C6	CE	D6	E6			07CH,0C6H,0CEH,0DEH,0	

```

2389 IC0E IC 3C 6C CC FE 0C      DB    01CH,03CH,06CH,0CCH,0FEH,00CH,01EH,000H ; D_34 *
2390   IE 00                         DB    0FCH,0C0H,0F8H,0C0H,0CCH,0CCH,078H,000H ; D_35 5
2391 IC16 FC C0 F8 0C 0C CC      DB    038H,060H,0C0H,0F8H,0CCH,0CCH,078H,000H ; D_36 6
2392   78 00                         DB    0FCH,0CCH,0CCH,018H,030H,030H,030H,000H ; D_37 7
2393 IC1E 00 00 C0 F8 CC CC      DB    078H,0CCH,0CCH,078H,0CCH,0CCH,078H,000H ; D_38 8
2394   78 00                         DB    078H,0CCH,0CCH,07CH,0CCH,018H,070H,000H ; D_39 9
2395 IC26 FC CC 0C 18 30 30      DB    000H,030H,030H,000H,000H,030H,030H,000H ; D_3A 1 COLON
2396   30 00                         DB    000H,030H,030H,000H,000H,030H,030H,000H ; D_3B 1 SEMICOLON
2397 IC2E 78 CC CC 78 CC CC      DB    018H,030H,060H,0C0H,060H,030H,018H,000H ; D_3C < LESS THAN
2398   78 00                         DB    000H,000H,0FCFH,000H,000H,0FCFH,000H,000H ; D_3D = EQUAL
2399 IC36 78 CC CC 7C OC 18      DB    060H,030H,018H,0CCH,018H,030H,060H,000H ; D_3E > GREATER THAN
2400   78 00                         DB    078H,0CCH,0CCH,018H,030H,000H,030H,000H ; D_3F ? QUESTION MARK
2401 IC3E 00 30 30 00 00 30      DB    07CH,0C6H,0DEH,0DEH,0DEH,0CCH,078H,000H ; D_40 * AT
2402   30 00                         DB    030H,078H,0CCH,0CCH,0CCH,0CCH,0CCH,000H ; D_41 A
2403 IC46 00 30 30 00 00 30      DB    0FCFH,066H,066H,07CH,066H,066H,0FCFH,000H ; D_42 B
2404   30 60                         DB    03CH,066H,066H,0C0H,0C0H,066H,03CH,000H ; D_43 C
2405 IC4E 00 60 C0 60 30          DB    0F8H,06CH,066H,066H,066H,06CH,0F8H,000H ; D_44 D
2406   18 00                         DB    0FEH,062H,068H,078H,068H,062H,0FEH,000H ; D_45 E
2407 IC56 00 00 FC 00 00 FC      DB    0F8H,060H,060H,060H,060H,060H,060H,000H ; D_46 F
2408   00 00                         DB    03CH,066H,0C0H,0C0H,0CEH,066H,03EH,000H ; D_47 G
2409 IC5E 60 30 18 OC 18 30      DB    0CCH,0CCH,0CCH,0FCFH,0CCH,0CCH,0CCH,000H ; D_48 H
2410   60 00                         DB    078H,030H,030H,030H,030H,030H,078H,000H ; D_49 I
2411 IC66 78 CC OC 18 30 00      DB    01EH,00CH,00CH,00CH,0CCH,0CCH,078H,000H ; D_4A J
2412   30 00                         DB    0E6H,066H,06CH,078H,066H,06EH,000H ; D_4B K
2413 IC6E 7C OC DE DE C0          DB    0F0H,060H,060H,060H,062H,066H,0FEH,000H ; D_4C L
2414 IC76 30 00 CC CC FC CC      DB    0C6H,0EEH,0FEH,0FEH,0D6H,0C6H,0C6H,000H ; D_4D M
2415   78 00                         DB    0C6H,06EH,0F6H,0DEH,0CEH,0C6H,0C6H,000H ; D_4E N
2416 IC86 3C 60 C0 C0 66          DB    038H,0C6H,0C6H,0C6H,0C6H,0C6H,038H,000H ; D_4F O
2417 IC8E 78 66 66 66 66          DB    07CH,0C6H,066H,07CH,060H,060H,0F0H,000H ; D_50 P
2418 IC96 62 68 78 68 62          DB    078H,0CCH,0CCH,0CCH,0CCH,0CCH,078H,01CH,000H ; D_51 Q
2419 ICCE F0 00 60 C0 CC EE      DB    0FCFH,066H,066H,07CH,06CH,066H,06EH,000H ; D_52 R
2420 ICAC 00 60 CC CC FC CC      DB    078H,0CCH,0E0H,070H,01CH,0CCH,078H,000H ; D_53 S
2421 ICBE 78 30 30 30 30          DB    0FCFH,0B4H,030H,030H,030H,030H,078H,000H ; D_54 T
2422 ICCE F0 60 60 60 62 66      DB    0CCH,0CCH,0CCH,0CCH,0CCH,0CCH,0FCFH,000H ; D_55 U
2423 ICDE C6 E6 FE FE D6 C6      DB    0CCH,0CCH,0CCH,0CCH,0CCH,0CCH,078H,030H,000H ; D_56 V
2424 ICDE C6 00 F6 DE CE C6      DB    0C6H,0C6H,0C6H,0D6H,0FEH,0EEH,0C6H,000H ; D_57 W
2425 IC6E 38 6C C6 C6 6C          DB    0C6H,0C6H,0C6H,038H,0C6H,0C6H,0C6H,000H ; D_58 X
2426   38 00                         DB    0CCH,0CCH,0CCH,0CCH,0CCH,0CCH,078H,030H,030H,078H,000H ; D_59 Y
2427 ICCE F0 66 66 TC 60 60      DB    0FEH,0C6H,08CH,018H,032H,066H,0FEH,000H ; D_5A Z
2428 IC6E 78 00 60 CC CC CC      DB    078H,060H,060H,060H,060H,060H,060H,000H ; D_5B [ LEFT BRACKET
2429 IC6E 78 00 60 CC CC CC 78      DB    0C0H,060H,030H,018H,00CH,006H,002H,000H ; D_5C \ BACKSLASH
2430 IC6E 78 00 60 CC CC CC 78      DB    078H,018H,018H,018H,018H,018H,078H,000H ; D_5D ] RIGHT BRACKET
2431 IC6E 10 38 6C C6 00 00      DB    010H,038H,0C6H,0C6H,000H,000H,000H,000H,000H ; D_5E ^ CIRCUMFLEX
2432 ID66 00 00 00 00 00 00      DB    000H,000H,000H,000H,000H,000H,000H,0FFH ; D_5F _ UNDERSCORE
2433 ID66 00 00 FF 00 00 00      DB    030H,030H,018H,000H,000H,000H,000H,000H ; D_60 ' APOSTROPHE REV
2434 ID6E 30 30 18 00 00 00      DB    000H,000H,078H,00CH,07CH,0CCH,076H,000H ; D_61 *
2435 ID76 00 00 78 0C 7C CC      DB    0E0H,060H,060H,07CH,066H,066H,0DCH,000H ; D_62 b
2436 ID7E E0 60 60 7C 66 66      DB    000H,000H,078H,0CCH,0C0H,0CCH,078H,000H ; D_63 c
2437 ID86 DC 00 00 7C CC 0C CC      DB    01CH,00CH,00CH,07CH,0CCH,0CCH,076H,000H ; D_64 d
2438 ID8E 1C 00 0C 7C CC CC      DB    000H,000H,078H,0CCH,0FCFH,0C0H,078H,000H ; D_65 e
2439 ID96 00 00 78 CC FC C0      DB    038H,0C6H,060H,0F0H,060H,060H,0F0H,000H ; D_66 f
2440 ID9E 38 6C 60 F0 60 60      DB    000H,000H,076H,0CCH,0CCH,07CH,00CH,0F8H ; D_67 g
2441 IDA6 00 00 76 CC CC 7C      DB    0E0H,060H,06CH,076H,066H,066H,0E6H,000H ; D_68 h
2442 IDAE F0 60 6C 76 66 66      DB    030H,000H,070H,030H,030H,030H,078H,000H ; D_69 i
2443 IDBE 78 00 60 CC CC CC      DB    00CH,000H,00CH,00CH,00CH,00CH,0CCCH,078H ; D_6A j
2444 IDC1 CC 78 00 00 00 00      DB    0E0H,060H,066H,06CH,078H,06CH,0E6H,000H ; D_6B k

```

2503 E6 00
2504 1DCE 70 30 30 30 30 30 DB 070H,030H,030H,030H,030H,030H,078H,000H ; D_6C i
2505 78 00
2506 1DDE 00 00 CC FE FE D6 DB 000H,000H,0CCH,0FEH,0FEH,0D6H,0C6H,000H ; D_6D m
2507 C6 00
2508 1DDE 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,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 00
2515 1DF6 00 00 76 CC CC 7C DB 000H,000H,076H,0CCH,0CCH,07CH,00CH,01EH ; D_71 q
2516 OC 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 1E04 00 00 7C C0 78 0C DB 000H,000H,07CH,0C0H,078H,00CH,0F8H,000H ; D_73 s
2520 7C 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,076H,000H ; D_75 u
2524 76 00
2525 1E1E 00 00 CC CC CC 78 DB 000H,000H,0CCH,0CCH,0CCH,0CCH,078H,030H,000H ; D_76 v
2526 6C 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 1E22 00 00 C6 6C 38 6C DB 000H,000H,0C6H,06CH,038H,06CH,0C6H,000H ; D_78 x
2530 C6 00
2531 1E30 00 00 CC CC CC 7C DB 000H,000H,0CCH,0CCH,0CCH,07CH,00CH,0F8H ; D_79 y
2532 78 00
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 IC 00
2537 1E4E 18 18 18 00 18 18 DB 018H,018H,018H,000H,018H,018H,018H,000H ; D_7C | BROKEN STROKE
2538 00 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

```

PAGE
2545      ;----- INT 1A -----
2546      ; TIME OF DAY
2547      ; THIS ROUTINE ALLOWS THE CLOCK TO BE SET/READ
2548
2549      ; INPUT
2550      ; (AH) = 0      READ THE CURRENT CLOCK SETTING
2551      ; RETURNS DS = HIGH PORTION OF COUNT
2552      ;          DX = LOW PORTION OF COUNT
2553      ;          AL = 0 IF TIMER HAS NOT PASSED
2554      ;          AL = 1 IF ON ANOTHER DAY
2555      ;          24 HOURS SINCE LAST READ
2556      ;          >>0 IF ON ANOTHER DAY
2557
2558      ; (AH) = 1      SET THE CURRENT CLOCK
2559      ; CX = HIGH PORTION OF COUNT
2560      ; DX = LOW PORTION OF COUNT
2561      ; NOTE: COUNTS OCCUR AT THE RATE OF
2562      ; 1193180/65536 COUNTS/SEC
2563      ; (OR ABOUT 18.2 PER SECOND -- SEE EQUATES BELOW)
2564
2565      ;----- ASSUME CS:CODE,DS:DATA
2566      ; ORG 0FE6EH
2567      ; ORG 01E6EH
2568      ; JMP TIME_OF_DAY_11
2569
2570      ;----- ORG 0FEA8H
2571      ; ORG 01EA8H
2572      ;----- TIMER_INT:
2573      ; JMP TIMER_INT_I
2574
2575
2576
2577      ;----- THESE ARE THE VECTORS WHICH ARE MOVED INTO
2578      ; THE 8086 INTERRUPT AREA DURING POWER ON.
2579      ; ONLY THE OFFSETS ARE DISPLAYED HERE, CODE
2580      ; SEGMENT WILL BE ADDED FOR ALL OF THEM, EXCEPT
2581      ; WHERE NOTED.
2582
2583      ;----- ASSUME CS:CODE
2584      ; ORG 0FEF3H
2585      ; ORG 01EF3H
2586      ;----- VECTOR_TABLE LABEL WORD
2587      ; DW OFFSET_TIMER_INT    I VECTOR TABLE VALUES FOR POST TESTS
2588      ; DW OFFSET_KB_INT       I INT 0BH - HARDWARE TIMER 0   IRQ 0
2589      ; DW OFFSET_DISK_INT     I INT 0CH - KEYBOARD             IRQ 1
2590      ; DW OFFSET_D11           I INT 0DH - DISKETTE            IRQ 2
2591      ; DW OFFSET_D11           I INT 0BH -                         IRQ 3
2592      ; DW OFFSET_D11           I INT 0CH -                         IRQ 4
2593      ; DW OFFSET_DISK_INT     I INT 0DH -                         IRQ 5
2594      ; DW OFFSET_DISKETTE     I INT 0EH - DISKETTE            IRQ 6
2595      ; DW OFFSET_D11           I INT 0FH -                         IRQ 7
2596
2597      ;----- SOFTWARE INTERRUPTS ( BIOS CALLS AND POINTERS )
2598      ;----- INT 10H -- VIDEO DISPLAY
2599      ;----- INT 11H -- GET EQUIPMENT FLAG WORD
2600      ;----- INT 12H -- GET REAL MODE MEMORY SIZE
2601      ;----- INT 13H -- GET DISKETTE_10
2602      ;----- INT 14H -- COMMUNICATION ADAPTER
2603      ;----- INT 15H -- EXPANDED BIOS FUNCTION CALL
2604      ;----- INT 16H -- KEYBOARD INPUT
2605      ;----- INT 17H -- PRINTER OUTPUT
2606      ;----- INT 18H -- OF6000H INSERTED FOR BASIC
2607      ;----- INT 19H -- FDISK SYSTEM MEDIA
2608      ;----- INT 1AH -- TIME OF DAY
2609      ;----- INT 1BH -- KEYBOARD BREAK ADDRESS
2610      ;----- INT 1CH -- TIMER BREAK ADDRESS
2611      ;----- INT 1DH -- VIDEO PARAMETERS
2612      ;----- INT 1EH -- DISKETTE PARAMETERS
2613      ;----- INT 1FH -- POINTER TO VIDEO EXTENSION
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624      ;----- TEMPORARY INTERRUPT SERVICE ROUTINE
2625      ;----- 1. THIS ROUTINE IS ALSO LEFT IN PLACE AFTER THE
2626      ;----- POWER ON DIAGNOSTIC TO SERVICE UNUSED
2627      ;----- INTERRUPT VECTORS. ITS LOCATION 'INTR_FLAG' WILL
2628      ;----- CONTAIN EITHER: 1. LEVEL OF HARDWARE INT. THAT
2629      ;----- CAUSED CODE TO BE EXEC.,
2630      ;----- 2. 'FF' FOR NON-HARDWARE INTERRUPTS THAT WAS
2631      ;----- EXECUTED ACCIDENTALLY.
2632
2633      ;----- D11 PROC NEAR
2634      ;----- ASSUME DS:DATA
2635      ;----- PUSH DS
2636      ;----- CALL DDS
2637      ;----- PUSH AX
2638      ;----- MOV AL,0BH
2639      ;----- OUT AL,INTA00,AL
2640      ;----- NOP
2641      ;----- IN AL,INTA00
2642      ;----- MOV AH,AL
2643      ;----- OR AL,AH
2644      ;----- JNZ HW_INT
2645      ;----- MOV AH,0FFH
2646      ;----- JMP SHORT_SET_INTR_FLAG
2647      ;----- HW_INT:
2648      ;----- IN AL,INTA01
2649      ;----- OR AL,AH
2650      ;----- OUT AL,INTA01,AL
2651      ;----- MOV AL,0EH
2652      ;----- OUT AL,INTA00,AL
2653      ;----- SET_INTR_FLAG:
2654      ;----- MOV 0INTR_FLAG,AH
2655      ;----- POP AX
2656      ;----- POP DS
2657      ;----- DUMMY_RETURN:
2658      ;----- IRET
2659      ;----- D11 ENDP
2660
2661      ;----- DUMMY RETURN FOR ADDRESS COMPATIBILITY
2662
2663      ;----- ORG 0FF53H
2664      ;----- ORG 01F53H
2665
2666      ;----- IRET

```

```
2658 PAGE
2659 I--- INT 05 H ---
2660 I PRINT_SCREEN_I
2661 I   LOGIC WILL BE INVOKED BY INTERRUPT 05H TO PRINT THE SCREEN.
2662 I   THE CURSOR POSITION AT THE TIME THIS ROUTINE IS INVOKED WILL BE
2663 I   SAVED AND RESTORED UPON COMPLETION. THE ROUTINE IS INTENDED TO
2664 I   RUN WITH INTERRUPTS ENABLED. IF A SUBSEQUENT PRINT SCREEN KEY
2665 I   IS DEPRESSED WHILE THIS ROUTINE IS PRINTING IT WILL BE IGNORED.
2666 I   THE BASE PRINTERS STATUS IS CHECKED FOR NOT BUSY AND NOT OUT OF
2667 I   PAPER. AN INITIAL STATUS ERROR WILL ABEND THE PRINT REQUEST.
2668 I   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 I ORG 0F54H
2676 I ORG 0IF54H
2677
2678 I IF54
2679
2680 I PRINT_SCREEN_I PROC FAR
2681 IF54 IE
2682 IF55 E6 1A12 R
2683 IF56 80 3E 0100 R 01
2684 IF57 T4 7C
2685 IF58 C6 06 0100 R 01
2686 IF59 F4 50
2687 IF66 53
2688 IF67 51
2689 IF68 52
2690 IF69 B4 0F
2691 IF6B CD 10
2692
2693 IF6D 8A CC
2694 IF6F 8A 2E 0084 R
2695 IF73 FE C5
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705 IF75 33 D2
2706 IF77 B4 02
2707 IF77 CD 17
2708 IF78 B0 F4 80
2709 IF7E T6 C0 A0
2710 IF81 75 4E
2711
2712 IF83 E8 1FDD R
2713 IF86 51
2714 IF87 B4 03
2715 IF89 00 10
2716 IF8B 59
2717 IF8C 52
2718 IF8D 33 D2
2719
2720
2721
2722
2723 IF8F PRI0:
2724 IF8F B4 02
2725 IF91 CD 10
2726 IF91 00 08
2727 IF95 CD 10
2728 IF97 04 C0
2729 IF99 T5 02
2730 IF9B 80 20
2731 IF9D
2732 IF9E 52
2733 IF9E 33 D2
2734 IFAO 32 E4
2735 IFAC CD 17
2736 IFAC 5A
2737 IFAC F6 C4 29
2738 IFBD 75 22
2739 IFBD 00 02
2740 IFAC 3A CA
2741 IFAE T5 DF
2742 IFBD 32 D2
2743 IFBD 8A E2
2744 IFBD 52
2745 IFBD 00 1FDD R
2746 IFBD 5A
2747 IFBD FE C6
2748 IFBD 3A EE
2749 IFBD 75 DO
2750
2751 IFBD 5A
2752 IFCD B4 02
2753 IFCD CD 10
2754 IFCD FA
2755 IFCD C6 06 0100 R 00
2756 IFCA EB 0B
2757
2758 PAGE
2759 I--- INT 05 H ---
2760 I PRINT_SCREEN_I
2761 I   LOGIC WILL BE INVOKED BY INTERRUPT 05H TO PRINT THE SCREEN.
2762 I   THE CURSOR POSITION AT THE TIME THIS ROUTINE IS INVOKED WILL BE
2763 I   SAVED AND RESTORED UPON COMPLETION. THE ROUTINE IS INTENDED TO
2764 I   RUN WITH INTERRUPTS ENABLED. IF A SUBSEQUENT PRINT SCREEN KEY
2765 I   IS DEPRESSED WHILE THIS ROUTINE IS PRINTING IT WILL BE IGNORED.
2766 I   THE BASE PRINTERS STATUS IS CHECKED FOR NOT BUSY AND NOT OUT OF
2767 I   PAPER. AN INITIAL STATUS ERROR WILL ABEND THE PRINT REQUEST.
2768 I   ADDRESS 0050:0000 CONTAINS THE STATUS OF THE PRINT SCREEN!
2769
2770 :      50:0 = 0 PRINT SCREEN HAS NOT BEEN CALLED OR UPON RETURN
2771 :           FROM A CALL THIS INDICATES A SUCCESSFUL OPERATION.
2772 :      = 1 PRINT SCREEN IS IN PROGRESS - IGNORE THIS REQUEST.
2773 :      = 255 ERROR ENCOUNTERED DURING PRINTING.
2774
2775 I ORG 0F54H
2776 I ORG 0IF54H
2777
2778 I PRINT_SCREEN_I PROC FAR
2779
2780 PUSH DS
2781 CALL DDS
2782 CMP $STATUS_BYTE,1
2783 JE PRI190
2784 MOV $STATUS_BYTE,1
2785 ST DS
2786 PUSH AX
2787 PUSH BX
2788 PUSH CX
2789 PUSH DX
2790 MOV AH,0FH
2791 INT 10H
2792
2793 MOV CL,AH
2794 MOV CH,$ROWS
2795 INC CH
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
3243
3244
3245
3246
3247
3248
3249
3250
3251
3252
3253
3254
3255
3256
3257
3258
3259
3260
3261
3262
3263
3264
3265
3266
3267
3268
3269
3270
3271
3272
3273
3274
3275
3276
3277
3278
3279
3280
3281
3282
3283
3284
3285
3286
3287
3288
3289
3290
3291
3292
3293
3294
3295
3296
3297
3298
3299
3300
3301
3302
3303
3304
3305
3306
3307
3308
3309
3310
3311
3312
3313
3314
3315
3316
3317
3318
3319
3320
3321
3322
3323
3324
3325
3326
3327
3328
3329
3330
3331
3332
3333
3334
3335
3336
3337
3338
3339
3340
3341
3342
3343
3344
3345
3346
3347
3348
3349
3350
3351
3352
3353
3354
3355
3356
3357
3358
3359
3360
3361
3362
3363
3364
3365
3366
3367
3368
3369
3370
3371
3372
3373
3374
3375
3376
3377
3378
3379
3380
3381
3382
3383
3384
3385
3386
3387
3388
3389
3390
3391
3392
3393
3394
3395
3396
3397
3398
3399
3400
3401
3402
3403
3404
3405
3406
3407
3408
3409
3410
3411
3412
3413
3414
3415
3416
3417
3418
3419
3420
3421
3422
3423
3424
3425
3426
3427
3428
3429
3430
3431
3432
3433
3434
3435
3436
3437
3438
3439
3440
3441
3442
3443
3444
3445
3446
3447
3448
3449
3450
3451
3452
3453
3454
3455
3456
3457
3458
3459
3460
3461
3462
3463
3464
3465
3466
3467
3468
3469
3470
3471
3472
3473
3474
3475
3476
3477
3478
3479
3480
3481
3482
3483
3484
3485
3486
3487
3488
3489
3490
3491
3492
3493
3494
3495
3496
3497
3498
3499
3500
3501
3502
3503
3504
3505
3506
3507
3508
3509
3510
3511
3512
3513
3514
3515
3516
3517
3518
3519
3520
3521
3522
3523
3524
3525
3526
3527
3528
3529
3530
3531
3532
3533
3534
3535
3536
3537
3538
3539
3540
3541
3542
3543
3544
3545
3546
3547
3548
3549
3550
3551
3552
3553
3554
3555
3556
3557
3558
3559
3560
3561
3562
3563
3564
3565
3566
3567
3568
3569
3570
3571
3572
3573
3574
3575
3576
3577
3578
3579
3580
3581
3582
3583
3584
3585
3586
3587
3588
3589
3590
3591
3592
3593
3594
3595
3596
3597
3598
3599
3600
3601
3602
3603
3604
3605
3606
3607
3608
3609
3610
3611
3612
3613
3614
3615
3616
3617
3618
3619
3620
3621
3622
3623
3624
3625
3626
3627
3628
3629
3630
3631
3632
3633
3634
3635
3636
3637
3638
3639
3640
3641
3642
3643
3644
3645
3646
3647
3648
3649
3650
3651
3652
3653
3654
3655
3656
3657
3658
3659
3660
3661
3662
3663
3664
3665
3666
3667
3668
3669
3670
3671
3672
3673
3674
3675
3676
3677
3678
3679
3680
3681
3682
3683
3684
3685
3686
3687
3688
3689
3690
3691
3692
3693
3694
3695
3696
3697
3698
3699
3700
3701
3702
3703
3704
3705
3706
3707
3708
3709
3710
3711
3712
3713
3714
3715
3716
3717
3718
3719
3720
3721
3722
3723
3724
3725
3726
3727
3728
3729
3730
3731
3732
3733
3734
3735
3736
3737
3738
3739
3740
3741
3742
3743
3744
3745
3746
3747
3748
3749
3750
3751
3752
3753
3754
3755
3756
3757
3758
3759
3760
3761
3762
3763
3764
3765
3766
3767
3768
3769
3769
3770
3771
3772
3773
3774
3775
3776
3777
3778
3779
3779
3780
3781
3782
3783
3784
3785
3786
3787
3788
3789
3789
3790
3791
3792
3793
3794
3795
3796
3797
3798
3799
3799
3800
3801
3802
3803
3804
3805
3806
3807
3808
3809
3809
3810
3811
3812
3813
3814
3815
3816
3817
3818
3819
3819
3820
3821
3822
3823
3824
3825
3826
3827
3828
3829
3829
3830
3831
3832
3833
3834
3835
3836
3837
3838
3839
3839
3840
3841
3842
3843
3844
3845
3846
3847
3848
3849
3849
3850
3851
3852
3853
3854
3855
3856
3857
3858
3859
3859
3860
3861
3862
3863
3864
3865
3866
3867
3868
3869
3869
3870
3871
3872
3873
3874
3875
3876
3877
3878
3879
3879
3880
3881
3882
3883
3884
3885
3886
3887
3888
3889
3889
3890
3891
3892
3893
3894
3895
3896
3897
3898
3899
3899
3900
3901
3902
3903
3904
3905
3906
3907
3908
3909
3909
3910
3911
3912
3913
3914
3915
3916
3917
3918
3919
3919
3920
3921
3922
3923
3924
3925
3926
3927
3928
3929
3929
3930
3931
3932
3933
3934
3935
3936
3937
3938
3939
3939
3940
3941
3942
3943
3944
3945
3946
3947
3948
3949
3949
3950
3951
3952
3953
3954
3955
3956
3957
3958
3959
3959
3960
3961
3962
3963
3964
3965
3966
3967
3968
3969
3969
3970
3971
3972
3973
3974
3975
3976
3977
3978
3979
3979
3980
3981
3982
3983
3984
3985
3986
3987
3988
3989
3989
3990
3991
3992
3993
3994
3995
3996
3997
3998
3999
3999
4000
4001
4002
4003
4004
4005
4006
4007
4008
4009
4009
4010
4011
4012
4013
4014
4015
4016
4017
4018
4019
4019
4020
4021
4022
4023
4024
4025
4026
4027
4028
4029
4029
4030
4031
4032
4033
4034
4035
4036
4037
4038
4039
4039
4040
4041
4042
4043
4044
4045
4046
4047
4048
4049
4049
4050
4051
4052
4053
4054
4055
4056
4057
4058
4059
4059
4060
4061
4062
4063
4064
4065
4066
4067
4068
4069
4069
4070
4071
4072
4073
4074
4075
4076
4077
4078
4079
4079
4080
4081
4082
4083
4084
4085
4086
4087
4088
4089
4089
4090
4091
4092
4093
4094
4095
4096
4097
4098
4099
4099
4100
4101
4102
4103
4104
4105
4106
4107
4108
4109
4109
4110
4111
4112
4113
4114
4115
4116
4117
4118
4119
4119
4120
4121
4122
4123
4124
4125
4126
4127
4128
4129
4129
4130
4131
4132
4133
4134
4135
4136
4137
4138
4139
4139
4140
4141
4142
4143
4144
4145
4146
4147
4148
4149
4149
4150
4151
4152
4153
4154
4155
4156
4157
4158
4159
4159
4160
4161
4162
4163
4164
4165
4166
4167
4168
4169
4169
4170
4171
4172
4173
4174
4175
4176
4177
4178
4179
4179
4180
4181
4182
4183
4184
4185
4186
4187
4188
4189
4189
4190
4191
4192
4193
4194
4195
4196
4197
4198
4199
4199
4200
4201
4202
4203
4204
4205
4206
4207
4208
4209
4209
4210
4211
4212
4213
4214
4215
4216
4217
4218
4219
4219
4220
4221
4222
4223
4224
4225
4226
4227
4228
4229
4229
4230
4231
4232
4233
4234
4235
4236
4237
4238
4239
4239
4240
4241
4242
4243
4244
4245
4246
4247
4248
4249
4249
4250
4251
4252
4253
4254
4255
4256
4257
4258
4259
4259
4260
4261
4262
4263
4264
4265
4266
4267
4268
4269
4269
4270
4271
4272
4273
4274
4275
4276
4277
4278
4279
4279
4280
4281
4282
4283
4284
4285
4286
4287
4288
4289
4289
4290
4291
4292
4293
4294
4295
4296
4297
4298
4299
4299
4300
4301
4302
4303
4304
4305
4306
4307
4308
4309
4309
4310
4311
4312
4313
4314
4315
4316
4317
4318
4319
4319
4320
4321
4322
4323
4324
4325
4326
4327
4328
4329
4329
4330
4331
4332
4333
4334
4335
4336
4337
4338
4339
4339
4340
4341
4342
4343
4344
4345
4346
4347
4348
4349
4349
4350
4351
4352
4353
4354
4355
4356
4357
4358
4359
4359
4360
4361
4362
4363
4364
4365
4366
4367
4368
4369
4369
4370
4371
4372
4373
4374
4375
4376
4377
4378
4379
4379
4380
4381
4382
4383
4384
4385
4386
4387
4388
4389
4389
4390
4391
4392
4393
4394
4395
4396
4397
4398
4399
4399
4400
4401
4402
4403
4404
4405
4406
4407
4408
4409
4409
4410
4411
4412
4413
4414
4415
4416
4417
4418
4419
4419
4420
4421
4422
4423
4424
4425
4426
4427
4428
4429
4429
4430
4431
4432
4433
4434
4435
4436
4437
4438
4439
4439
4440
4441
4442
4443
4444
4445
4446
4447
4448
4449
4449
4450
4451
4452
4453
4454
4455
4456
4457
4458
4459
4459
4460
4461
4462
4463
4464
4465
4466
4467
4468
4469
4469
4470
4471
4472
4473
4474
4475
4476
4477
4478
4479
4479
4480
4481
4482
4483
4484
4485
4486
4487
4488
4489
4489
4490
4491
4492
4493
4494
4495
4496
4497
4498
4499
4499
4500
4501
4502
4503
4504
4505
4506
4507
4508
4509
4509
4510
4511
4512
4513
4514
4515
4516
4517
4518
4519
4519
4520
4521
4522
4523
4524
4525
4526
4527
4528
4529
4529
4530
4531
4532
4533
4534
4535
4536
4537
4538
4539
4539
4540
4541
4542
4543
4544
4545
4546
4547
4548
4549
4549
4550
4551
4552
4553
4554
4555
4556
4557
4558
4559
4559
4560
4561
4562
4563
4564
4565
4566
4567
4568
4569
4569
4570
4571
4572
4573
4574
4575
4576
4577
4578
4579
4579
4580
4581
4582
4583
4584
4585
4586
4587
4588
4589
4589
4590
4591
4592
4593
4594
4595
4596
4597
4598
4599
4599
4600
4601
4602
4603
4604
4605
4606
4607
4608
4609
4609
4610
4611
4612
4613
4614
4615
4616
4617
4618
4619
4619
4620
4621
4622
4623
4624
4625
4626
4627
4628
4629
4629
4630
4631
4632
4633
4634
4635
4636
4637
4638
4639
4639
4640
4641
4642
4643
4644
4645
4646
4647
4648
4649
4649
4650
4651
4652
4653
4654
4655
4656
4657
4658
4659
4659
4660
4661
4662
4663
4664
4665
4666
4667
4668
4669
4669
4670
4671
4672
4673
4674
4675
4676
4677
4678
4679
4679
4680
4681
4682
4683
4684
4685
4686
4687
4688
4689
4689
4690
4691
4692
4693
4694
4695
4696
4697
4698
4699
4699
4700
4701
4702
4703
4704
4705
4706
4707
4708
4709
4709
4710
4711
4712
4713
4714
4715
4716
4717
4718
4719
4719
4720
4721
4722
4723
4724
4725
4726
4727
4728
4729
4729
4730
4731
4732
4733
4734
4735
4736
4737
4738
4739
4739
4740
4741
4742
4743
4744
4745
4746
4747
4748
4749
4749
4750
4751
4752
4753
4754
4755
4756
4757
4758
4759
4759
4760
4761
4762
4763
4764
4765
4766
4767
4768
4769
4769
4770
4771
4772
4773
4774
4775
4776
4777
4778
4779
4779
4780
4781
4782
4783
4784
4785
4786
4787
4788
4789
4789
4790
4791
4792
4793
4794
4795
4796
4797
4798
4799
4799
4800
4801
4802
48
```

2757 PAGE
2758 PR160:
2759 IFCC 5A POP DX I ERROR EXIT
2760 IFCD B4 02 MOV AH,02H I GET CURSOR POSITION
2761 CD 10 INT 10H I INDICATE REQUEST CURSOR SET
2762 IFDI PRI70:
2763 IFDI FA CLI I CURSOR POSITION RESTORED
2764 IFD2 C6 06 0100 R FF MOV STATUS_BYTE,0FFH I BLOCK INTERRUPTS TILL STACK CLEARED
2765 IFDT PR180:
2766 IFD7 5A POP DX I SET ERROR FLAG
2767 IFD9 59 POP CX I EXIT ROUTINE
2768 IFD9 5B POP BX I RESTORE ALL THE REGISTERS USED
2769 IFDA 58 POP AX
2770 IFDB PR190:
2771 IFDB 1F POP DS I ROUTINE BUSY EXIT
2772 IFDC CF IRET I RETURN WITH INITIAL INTERRUPT MASK
2773 IFDD PRINT_SCREEN_I ENDP
2774
2775 I----- CARRIAGE RETURN, LINE FEED SUBROUTINE
2776
2777 IFDD CRLF PROC NEAR
2778 XOR DX,DX I SEND CR,LF TO FIRST PRINTER
2779 IFDD 33 D2 MOV AX,CR I ASSUME FIRST PRINTER (DX=0100)
2780 IFDF B8 000D INT 17H I GET THE PRINT CHARACTER COMMAND AND
2781 IFE2 CD 17 MOV AX,LF I THE CARRIAGE RETURN CHARACTER
2782 IFE4 B8 000A INT 17H I NOW GET THE LINE FEED AND
2783 IFE7 CD 17 RET I SEND IT TO THE BIOS PRINTER ROUTINE
2784 IFE9 C3 CRLF ENDP
2785 IFEA
2786
2787
2788 I----- POWER ON RESET VECTOR I
2789
2790 I-----
2791 IFF0 ORG OFFFOH
2792 ORG 01FFOH
2793 I----- POWER ON RESET
2794 IFF0 P_O_R LABEL FAR
2795 IFF0 EA DB 0EAH
2796 IFF1 E05B DW 0E05BH I LOW WORD OF RESET
2797 IFF3 F000 DW 0F000H I SEGMENT
2798
2799 IFF5 30 31 2F 31 30 2F DB '01/10/86' I RELEASE MARKER
2800 38 36
2801
2802
2803 IFFE I-----
2804 IFFE MODEL: ORG 01FFEH
2805 IFFE DB MODEL_BYTEx
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:

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

```

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 ; NO REFERENCES SHOULD BE MADE WHICH REFERENCE ;
8 ; ABSOLUTE ADDRESSES WITHIN THE CODE SEGMENT ;
9 ; VIOLATE THE STRUCTURE AND DESIGN OF BIOS.
10 ;-----;

11 ;-----;
12 ;-----; EQUATES ;-----;
13 ;-----;
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
19 CMD_PORT EQU 63H
20 INTA00 EQU 20H ; 8259 PORT
21 INTA01 EQU 21H ; 8259 PORT
22 EO1 EQU 20H
23 TIMER EQU 40H
24 TIM_CTL EQU 43H ; 8253 TIMER CONTROL PORT ADDR
25 TIMERO EQU 40H ; 8253 TIMER/CNTNR 0 PORT ADDR
26 TMINT EQU 01H ; TIMER 0 INT/RECDV MASK
27 DMA08 EQU 08H ; DMA STATUS REG PORT ADDR
28 DMA EQU 00H ; DMA CH.0 ADDR. REG PORT ADDR
29 MAX_PERIOD EQU 540H
30 MIN_PERIOD EQU 410H
31 KBD_IN EQU 60H ; KEYBOARD DATA IN ADDR PORT
32 KBINTR EQU 02H ; KEYBOARD INTNTR
33 KB_DATA EQU 60H ; KEYBOARD SCAN CODE PORT
34 KB_CTL EQU 61H ; CONTROL BITS FOR KEYBOARD SENSE DATA
35 ;-----;
36 ;-----; 8088 INTERRUPT LOCATIONS ;-----;
37 ;-----;
38 ;-----;
39 ABS0 SEGMENT AT 0
40 STG_LOC0 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 WORD
48 _ORG 10H*4
49 VIDEO_INTR LABEL WORD
50 _ORG 11H*4
51 PARM_PTR LABEL DWORD ; POINTER TO VIDEO PARMS
52 _ORG 18H*4
53 BASIC_PTR LABEL WORD ; ENTRY POINT FOR CASSETTE BASIC
54 _ORG 01EH*4 ; INTERRUPT IEH
55 DISK_POINTER LABEL DWORD
56 _ORG 01H*4
57 EXT_PTR LABEL DWORD ; LOCATION OF POINTER
58 _ORG 400H ; POINTER TO EXTENSION
59 DATA_AREA LABEL BYTE ; ABSOLUTE LOCATION OF DATA SEGMENT
60 DATA_WORD LABEL WORD
61 _ORG 0500H
62 MFG_TEST_PRN LABEL FAR
63 _ORG TCOOH
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 TOS LABEL WORD
74 STACK ENDS
75
76 ;-----;
77 ;-----; ROM BIOS DATA AREAS ;-----;
78 ;-----;
79 ;-----;
80 DATA SEGMENT AT 40H
81 RS232_BASE DW 4 DUP(?) ; ADDRESSES OF RS232 ADAPTERS
82 PRINTER_BASE DW 4 DUP(?) ; ADDRESSES OF PRINTERS
83 EQUIP_FLAG DW ? ; INSTALLED HARDWARE
84 MFG_TST DB ? ; INITIALIZATION FLAG
85 MEMORY_SIZE DW ? ; MEMORY SIZE IN K BYTES
86 MFG_ERR_FLAG DB ? ; SCRATCHPAD FOR MANUFACTURING
87 _ERR_CD DB ? ; ERROR CODES
88
89 ;-----;
90 ;-----; KEYBOARD DATA AREAS ;-----;
91 ;-----;
92
93 KB_FLAG DB ?
94
95 ;-----; SHIFT FLAG EQUATES WITHIN KB_FLAG ;-----;
96
97 INS_STATE EQU 80H ; INSERT STATE IS ACTIVE
98 CAPS_STATE EQU 40H ; CAPS LOCK STATE HAS BEEN TOGGLED
99 NUM_STATE EQU 20H ; NUM LOCK STATE HAS BEEN TOGGLED
100 SCROLL_STATE EQU 10H ; SCROLL LOCK STATE HAS BEEN TOGGLED
101 ALT_SHIFT EQU 08H ; ALTERNATE SHIFT KEY DEPRESSED
102 CTL_SHIFT EQU 04H ; CONTROL SHIFT KEY DEPRESSED
103 LEFT_SHIFT EQU 02H ; LEFT SHIFT KEY DEPRESSED
104 RIGHT_SHIFT EQU 01H ; RIGHT SHIFT KEY DEPRESSED
105

```

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
0080		107	
0000		108	INS_SHIFT EQU 80H ; INSERT KEY IS DEPRESSED
0020		109	CAPS_SHIFT EQU 40H ; CAPS LOCK KEY IS DEPRESSED
0010		110	NUM_LOCK SHIFT EQU 20H ; NUM LOCK KEY IS DEPRESSED
0008		111	SCROLL_LOCK SHIFT EQU 10H ; SCROLL LOCK KEY IS DEPRESSED
0019 ??		112	HOLD_STATE EQU 08H ; SUSPEND KEY HAS BEEN TOGGLED
001A ????		113	
001C ???		114	ALT_INPUT DB ? ; STORAGE FOR ALTERNATE KEYPAD ENTRY
001E ???		115	BUFFER_HEAD DW ? ; POINTER TO HEAD OF KEYBOARD BUFFER
001E {16}		116	BUFFER_TAIL DW ? ; POINTER TO TAIL OF KEYBOARD BUFFER
???		117	KB_BUFFER DW 16 DUP(?) ; ROOM FOR 15 ENTRIES
003E		118	KB_BUFFER_END LABEL WORD
		119	----- HEAD = TAIL INDICATES THAT THE BUFFER IS EMPTY
		120	
0045		121	
0046		122	NUM_KEY EQU 69 ; SCAN CODE FOR NUMBER LOCK
0038		123	SCROLL_KEY EQU 70 ; SCROLL LOCK KEY
001D		124	ALT_KEY EQU 56 ; ALTERNATE SHIFT KEY SCAN CODE
003A		125	CTL_KEY EQU 29 ; SCAN CODE FOR CONTROL KEY
002A		126	CAPS_KEY EQU 58 ; SCAN CODE FOR CAPS SHIFT
0036		127	LEFT_SHIFT EQU 42 ; SCAN CODE FOR LEFT SHIFT
0052		128	RIGHT_KEY EQU 54 ; SCAN CODE FOR RIGHT SHIFT
0053		129	INS_KEY EQU 82 ; SCAN CODE FOR INSERT KEY
		130	DEL_KEY EQU 83 ; SCAN CODE FOR DELETE KEY
		131	
		132	-----
		133	; DISKETTE DATA AREAS
003E ??		134	-----
		135	SEEK_STATUS DB ? ; DRIVE RECALIBRATION STATUS
		136	BIT 3-0 = DRIVE 3-0 NEEDS RECAL
		137	BEFORE NEXT SEEK IF BIT IS = 0
		138	
0080		139	INT_FLAG EQU 080H ; INTERRUPT OCCURRENCE FLAG
003F ??		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 FAILED TO RESPOND
0040		151	BAD_SEEK EQU 40H ; SEEK OPERATION FAILED
0000		152	BAD_WRITE EQU 20H ; WRITE OPERATION FAILED
0010		153	BAD_CRC EQU 10H ; BAD COMMAND 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 {7}		161	NEC_STATUS DB ? 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
004B ????		168	CRT_LEN DW ? ; LENGTH OF REGEN 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
???		171	
0060 ????		171	CURSOR_MODE DW ? ; CURRENT CURSOR MODE SETTING
0062 ???		172	ACTV_PAGE DB ? ; CURRENT PAGE BEING DISPLAYED
0063 ????		173	ADDR_68 DW ? ; BASE ADDRESS FOR ACTIVE DISPLAY CARD
0065 ???		174	CRT_MODE_SET DB ? ; CURRENT SETTING OF THE 3XB REGISTER
0066 ??		175	CRT_PALETTE DB ? ; CURRENT PALETTE SETTING COLOR CARD
		176	
		177	-----
		178	; POST DATA AREA
		179	-----
0067 ????		180	10_ROM_INIT DW ? ; PTRN TO OPTIONAL I/O ROM INIT ROUTINE
0069 ????		181	10_ROM_SEG DW ? ; PTRN TO 10 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 ? ; WORD=1234H IF KEYBOARD RESET UNDERWAY
		200	-----
0074 ????		201	; FIXED DISK DATA AREAS
0076 ????		202	-----
		203	
		204	
		205	-----
0078 {4}		206	; PRINTER AND RS232 TIME-OUT VARIABLES
???		207	-----
007C {4}		208	PRINT_TIM_OUT DB 4 DUP(?)
???		209	RS232_TIM_OUT DB 4 DUP(?)

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

```

210 ;----- ADDITIONAL KEYBOARD DATA AREA -----
213 BUFFER_START DW ?
214 BUFFER_END DW ?
215 DATA ENDS
216 ;
217 ;----- EXTRA DATA AREA -----
218 XXDATA SEGMENT AT 50H
220 STATUS_BYT DB ?
221 XXDATA ENDS
222 ;
223 ;----- VIDEO DISPLAY BUFFER -----
224 XXDATA SEGMENT AT 0B800H
226 REGEN LABEL BYTE
227 REGNEW LABEL WORD
228 DB 16384 DUP(?)
229 VIDEO_RAM ENDS
230 ;
231 ;----- ROM RESIDENT CODE -----
232 ;
233 CODE SEGMENT AT 0F000H
234 DB 57344 DUP(?) ; FILL LOWEST 56K
235
236 DB *1501512 COPR. IBM 1982* ; COPYRIGHT NOTICE
E000 31353031353132
20434F5052E20
49424D20313938
32
237
238
239 ;----- INITIAL RELIABILITY TESTS -- PHASE I -----
240
241
242 ASSUME CS:CODE,SS:CODE,ES:ABS0,DS:DATA
243
244
245 ;----- DATA DEFINITIONS -----
246
247
248
E016 D7E0 C1 DW C11 ; RETURN ADDRESS
E018 TEE1 C2 DW C24 ; RETURN ADDRESS FOR DUMMY STACK
E01A 20A842204F4B E020 0D 253
254 F3B DB 'KB OK',13 ; KB FOR MEMORY SIZE
255 ;----- LOAD A BLOCK OF TEST CODE THROUGH THE KEYBOARD PORT
256 ; FOR MANUFACTURING TEST.
257 ; THIS ROUTINE WILL LOAD A TEST (MAX LENGTH:FAFFFH) THROUGH
258 ; THE KEYBOARD PORT. THE CODE WILL BE LOADED AT LOCATION
259 ; 0000:0500. AFTER LOADING, CONTROL WILL BE TRANSFERED
260 ; TO LOCATION 0000:0500. 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
268 MFG_BOOT:
269 CALL SP_TEST ; GET COUNT LOW
270 MOV BH,CL ; SAVE IT
271 CALL SP_TEST ; GET COUNT HI
272 MOV CH,BL
273 MOV CL,BH ; CX NOW HAS COUNT
274 CLD ; SET DIR. FLAG TO INCREMENT
275 CLI ; SET TARGET OFFSET (IDS=0000)
276 MOV DI,0500H ; UNMASK K/B INTERRUPT
277 MOV AL,0FDH
278 OUT INTA01,AL ; SEND READ INT. REQUEST REG. CMD
279 MOV AL,0AH
280 OUT INTA00,AL ; SET UP PORT B ADDRESS
281 MOV DX,61H ; CONTROL BITS FOR PORT B
282 MOV BX,4CCCH ; K/B REQUEST PENDING MASK
283 MOV AH,02H
284 TST: ;----- POINT DX AT ADDR. 60 (KB DATA)
285 MOV AL,BL
286 OUT DX,AL ; TOGGLE K/B CLOCK
287 MOV AL,BH
288 OUT DX,AL
289 DEC DX ; POINT DX AT ADDR. 60 (KB DATA)
290 TST: ;----- GET IRR REG
291 IN AL,INTA00 ; KB REQUEST PENDING?
292 AND AL,AH ; LOOP TILL DATA PRESENT
293 JZ TST1 ; GET DATA
294 IN AL,DX ; STORE IT
295 STOSB ;----- POINT DX BACK AT PORT B (61)
296 INC DX ;----- LOOP TILL ALL BYTES READ
297 LOOP TST
298
E054 EA00050000 299 JMP MFG_TEST_RTN ; FAR JUMP TO CODE THAT WAS JUST
300
301

```

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

```

302 ;----- 8088 PROCESSOR TEST
303 ; DESCRIPTION
304 ; VERIFY 8088 FLAGS, REGISTERS
305 ; AND CONDITIONAL JUMPS
306 ;
307 ;----- 308 ASSUME CS:CODE,DS:NOTHING,ES:NOTHING,SS:NOTHING
309 ORG 0E05BH
310 RESET LABEL FAR
311 START: 312 MOV AH,0D5H ; DISABLE INTERRUPTS
312 ; SET SF, CF, ZF, AND AF FLAGS ON
313 SAHF
314 JNC ERROR1 ; GO TO ERR ROUTINE IF CF NOT SET
315 JNZ ERROR1 ; GO TO ERR ROUTINE IF ZF NOT SET
316 JNP ERROR1 ; GO TO ERR ROUTINE IF PF NOT SET
317 JNS ERROR1 ; GO TO ERR ROUTINE IF SF NOT SET
318 LAHF ; LOAD FLAG IMAGE TO AH
319 MOV CL,5 ; LOAD CNT REG WITH SHIFT CNT
320 SHR AH,CL ; SHIFT AF INTO CARRY BIT POS
321 JNC ERROR1 ; GO TO ERR ROUTINE IF AF NOT SET
322 MOV AL,1AH ; SET THE OF FLAG ON
323 SHL AL,1 ; SETUP FOR TESTING
324 JNO ERROR1 ; GO TO ERR ROUTINE IF OF NOT SET
325 XOR AH,AH ; SET AH = 0
326 SAHF ; CLEAR SF, CF, ZF, AND PF
327 JBE ERROR1 ; GO TO ERR ROUTINE IF CF ON
328 ; GO TO ERR ROUTINE IF OF ON
329 JS ERROR1 ; GO TO ERR ROUTINE IF SF ON
330 JP ERROR1 ; GO TO ERR ROUTINE IF PF ON
331 LAHF ; LOAD FLAG IMAGE TO AH
332 MOV CL,5 ; LOAD CNT REG WITH SHIFT CNT
333 SHR AH,CL ; SHIFT 'AF' INTO CARRY BIT POS
334 JC ERROR1 ; GO TO ERR ROUTINE IF ON
335 SHL AH,1 ; CHECK THAT 'OF' IS CLEAR
336 JO ERROR1 ; GO TO ERR ROUTINE IF ON
337
338 ;----- READ/WRITE THE 8088 GENERAL AND SEGMENTATION REGISTERS
339 ; WITH ALL ONE'S AND ZEROES'S.
340
341 MOV AX,0FFFFH ; SETUP ONE'S PATTERN IN AX
342 STC ; WRITE PATTERN TO ALL REGS
343 CB: MOV DS,AX
344 MOV BX,DS
345 MOV ES,BX
346 MOV CX,ES
347 MOV SS,CX
348 MOV DX,SS
349 MOV SP,DX
350 MOV BP,SP
351 MOV SI,BP
352 MOV DI,SI
353 JNC C9 ; TSTIA
354 XOR AX,DI ; PATTERN MAKE IT THRU ALL REGS
355 JNZ ERROR1 ; NO - GO TO ERR ROUTINE
356 CLR C ; TSTIA
357 JMP C8 ; NO - GO TO NEXT TEST
358 C9: OR AX,DI ; ZERO PATTERN MAKE IT THRU?
359 JZ C10 ; YES - GO TO NEXT TEST
360 ERROR1: HLT ; HALT SYSTEM
361
362 ;----- ROS CHECKSUM TEST I
363 ; DESCRIPTION
364 ; A CHECKSUM IS DONE FOR THE BK
365 ; ROS MODULE CONTAINING POD AND
366 ; BIOS.
367
368 ;----- E0AE
369 C10: ; ZERO IN AL ALREADY
370 OUT 0A0H,AL ; DISABLE NMI INTERRUPTS
371 OUT 83H,AL ; INITIALIZE DMA PAGE REG
372 MOV DX,3D8H
373 OUT DX,AL ; DISABLE COLOR VIDEO
374 OUT DX,AL
375 INC AL
376 MOV DL,0B8H ; DISABLE B/W VIDEO,EN HIGH RES
377 OUT DX,AL ; SET B255 FOR B,A=OUT, C=IN
378 MOV AL,89H
379 OUT CMD_PORT_AL ; BANK OF SWITCHES->PORT C(0-3)
380 MOV AL,T0100101B ; PULL KB CLOCK HI, TRI-STATE
381 ; KEYBOARD INPUTS,ENABLE HIGH
382 OUT PORT_B,AL ; BANK OF SWITCHES->PORT C(0-3)
383 ; PULL KB CLOCK HI, TRI-STATE
384 ; KEYBOARD INPUTS,ENABLE HIGH
385 MOV AL,01H ; BANK OF SWITCHES->PORT C(0-3)
386 OUT PORT_A,AL ; PULL KB CLOCK HI, TRI-STATE
387 MOV AX,C5 ; KEYBOARD INPUTS,ENABLE HIGH
388 MOV SS,AX ; BANK OF SWITCHES->PORT C(0-3)
389 MOV DS,AX ; SETUP SS SEG REG
390 ; SET UP DATA SEG TO POINT TO
391 CLD ; ROM ADDRESS
392 ASSUME SS:CODE ; SET DIRECTION FLAG TO INC.
393 MOV BX,0E000H ; BANK OF SWITCHES->PORT C(0-3)
394 MOV SP,OFFSET CI ; SETUP STARTING ROS ADDR
395 JMP ROS_CHECKSUM ; SETUP RETURN ADDRESS
396 C11: JNE ERROR1 ; HALT SYSTEM IF ERROR
397 ;----- 398 ;----- B237 DMA INITIALIZATION CHANNEL REGISTER TEST
399 ; DESCRIPTION
400 ; DISABLE THE B237 DMA CONTROLLER. VERIFY THAT
401 ; TIMER I FUNCTIONS OK. WRITE/READ THE CURRENT
402 ; ADDRESS AND WORD COUNT REGISTERS FOR ALL
403 ; CHANNELS. INITIALIZE AND START DMA FOR MEMORY
404 ; REFRESH.
405
406
407 ;----- DISABLE DMA CONTROLLER
408
409 MOV AL,02H ; <><><><><><><>
410 OUT PORT_A,AL ; <><><>CHECKPOINT 2<><>
411 MOV AL,04 ; <><><><><><><>
412 OUT DMA08,AL ; DISABLE DMA CONTROLLER
413
414 ;----- VERIFY THAT TIMER I FUNCTIONS OK
415
416 MOV AL,54H ; SEL TIMER 1,LSB,MODE 2
417 OUT TIMER+3,AL

```

LOC	OBJECT	LINE	SOURCE	BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82		
E0E5	8AC1	418	MOV	AL,CL	; SET INITIAL TIMER CNT TO 0	
E0E7	E641	419	OUT	TIMER+1,AL		
E0E9		420	C12:			
E0E9	B040	421	MOV	AL,40H	; TIMER1_BITS_ON	
E0EB	E0F3	422	OUT	TIMER+3,AL	; LATCH TIMER I COUNT	
E0EF	00FF	423	CMP	BL,0FFH		
E0F0	7407	424	JZ	C13	; YES - SEE IF ALL BITS GO OFF	
E0F2	E441	425	IN	AL,TIMER+1	; ALLBITS1S_ON	
E0F4	0AD8	426	OR	BL,AL	; READ TIMER COUNT	
E0F6	E2F1	427	LOOP	C12	; ALL BITS ON IN TIMER	
E0F8	F4	428	HLT		; TIMER1_BITS_ON	
E0F9		429	C13:			
E0F9	8AC3	430	MOV	AL,BL	; TIMER1_LOOP	
E0FB	2BC9	431	SUB	CX,CX	; LATCH_TIMER I COUNT	
E0FD	E641	432	OUT	TIMER+1,AL		
E0FF		433	C14:			
E0FF	B040	434	MOV	AL,40H	; SET TIMER I COUNT	
E101	E643	435	OUT	TIMER+3,AL		
E103	90	436	NOP			
E104	90	437	NOP			
E105	E441	438	IN	AL,TIMER+1	; DELAY FOR TIMER	
E107	22D8	439	AND	BL,AL		
E109	7403	440	JZ	C15	; WRAP_DMA_REG	
E10B	E2F2	441	LOOP	C14	; TIMER_LOOP	
E10D	F4	442	HLT		; HALT SYSTEM	
		443				
		444	-----	INITIALIZE TIMER I TO REFRESH MEMORY		
		445				
E10E	B003	446	C15:	MOV	AL,03H	; <><><><><><><><>
E110	E660	447	OUT	PORT_A,AL	; <><><>CHECKPOINT 3<><>	
		448	OUT		; WRAP_DMA_REG	
E112	E60D	449	OUT	DMA+0DH,AL	; SEND_MASTER CLEAR TO DMA	
		450				
		451	-----	WRAP DMA CHANNELS ADDRESS AND COUNT REGISTERS		
		452				
E114	B0FF	453	MOV	AL,0FFH	; WRITE PATTERN FF TO ALL REGS	
E116	8A08	454	MOV	BL,AL	; SAVE PATTERN FOR COMPARE	
E118	8A8F	455	MOV	BH,AL		
E11A	B90800	456	MOV	CX,B		
E11D	BA0000	457	MOV	DX,DMA		
E120	E0	458	C17:	OUT	DX,AL	
E121	50	459	PUSH	AX	; SETUP LOOP_CNT	
E122	E	460	OUT	DX,AL	; SETUP I/O PORT ADDR OF REG	
E123	B001	461	MOV	AL,01H	; WRITE PATTERN TO REG, LSB	
E125	EC	462	IN	AL,DX	; SATISFY 8237 I/O TIMINGS	
E126	8AE0	463	MOV	AH,AL	; MSB OF 16-BIT REG	
E128	EC	464	IN	AL,DX	; TO ANOTHER REG BEFORE RD	
E129	3B08	465	CMP	BX,AX	; READ 16-BIT DMA CH REG, LSB	
E130	7401	466	JE	C18	; SAVE LSB OF 16-BIT REG	
E12D	F4	467	HLT		; READ MSB OF DMA CH REG	
E12E		468	C18:		; PATTERN READ AS WRITTEN?	
E12E	42	469	INC	DX	; YES - CHECK NEXT REG	
E12F	E2EF	470	LOOP	C17	; NO - HALT THE SYSTEM	
E131	FE00	471	INC	AL	; NXT_DMA CH REG	
E133	74E1	472	JZ	C16	; SET I/O PORT TO NEXT CH REG	
		473			; SET PATTERN TO 0	
		474	-----	WRITE TO CHANNEL REGS		
		475				
E135	8EDB	476	MOV	DS,BX	; WRITE TO CHANNEL REG	
E137	E8C3	477	MOV	E5,BX	; SET UP ABS0 INTO DS AND ES	
		478	ASUMME	DS:AB50,ES:AB50		
E139	B0FF	479	MOV	AL,0FFH		
E13B	E601	480	OUT	DMA+0FH	; SET CNT OF 64K FOR REFRESH	
E13D	50	481	PUSH	AX		
E13E	E601	482	OUT	DMA+1,AL		
E140	B0	483	MOV	AL,05BH		
E142	60B	484	OUT	DMA+0BH,AL		
E144	B000	485	MOV	AL,00H		
E146	8A88	486	MOV	CH,AL		
E148	E608	487	OUT	DMA+8,AL		
E14A	50	488	PUSH	AX		
E14B	E60A	489	OUT	DMA+10,AL		
E14D	6012	490	MOV	AL,01H		
E14F	E641	491	OUT	TIMER+1,AL	; ENABLE DMA CH 0	
E151	B041	492	MOV	AL,41H	; START TIMER 1	
E153	E60B	493	OUT	DMA+0BH,AL		
E155	50	494	PUSH	AX		
E156	E408	495	IN	AL,DMA+0B		
E158	6010	496	AND	AL,00010000B		
E15A	7401	497	JZ	C18C		
E15C	F4	498	HLT			
		499	C18C:	MOV	AL,42H	
E15D	B042	500	OUT	DMA+0BH,AL		
E15F	E60B	501	MOV	AL,43H		
E161	B043	502	OUT	DMA+0BH,AL		
E163	E60B	503	OUT	DMA+0BH,AL		
		504	-----	SET MODE FOR CHANNEL 3		
		505				
		506	BASE_16K_READ/VERIFY_STORAGE_TEST			
		507	DESCRIPTION			
		508	WRITE/READ/VERIFY DATA PATTERNS			
		509	AA,55,FF,01, AND 00 TO 1ST 32K OF			
		510	STORAGE, VERIFY STORAGE ADDRESSABILITY.			
		511	-----	DETERMINE MEMORY SIZE AND FILL MEMORY WITH DATA		
		512				
E165	BA1302	513	MOV	DX,0213H		
E168	B001	514	MOV	AL,01H		
E16A	EE	515	OUT	DX,AL		
		516				
E16B	8B1E7204	517	MOV	BX,DATA_WORD[OFFSET RESET_FLAG];	; SAVE 'RESET_FLAG' IN BX	
E16F	B90020	518	MOV	CX,2000H	; SET FOR 16K WORDS	
E172	81B3412	519	CMP	BX,1234H	; WARM START?	
E174	7401	520	JE	CLR_STG		
E178	BC18E0	521	MOV	SP,DATA[SET_C2		
E17B	E9F104	522	JMP	STGSTD_CNT		
E17E	7412	523	C24:	JE	HOW_BIG	
E180	8A88	524	MOV	BL,AL		
E182	B008	525	MOV	AL,04H		
E184	EE	526	C24A:	OUT	PORT_A,AL	
E186	2BC9	527	SUB	CX,CR		
E188	E2FE	528	C24B:	LOOP	C24B	
E18A	86D8	529	XCHG	BL,AL		
E18C	EBF6	530	JMP	C24A		

LOC	OBJECT	LINE	SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82	
E18E		531	CLR_STG:	
E18F 28C0		532	SUB AX,AX	; MAKE AX=0000
E190 F3		533	REP STOSW	; STORE 8K WORDS OF 0000
E191 AB				
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 8E00		538	FILL_LOOP:	
E19C 8E02		539	MOV ES,DX	; SET SEG. REG.
E19E 2BFF		540	SUB DI,DI	
E1A0 B855AA		541	MOV AX,0AA55H	; TEST PATTERN
E1A3 8C8C		542	MOV CX,AX	; SAVE PATTERN
E1A5 268905		543	MOV ES:[DI],AX	; SEND PATTERN TO MEM.
E1A7 BFF		544	MOV AL,0FFH	; PUT SOMETHING IN AL
E1A8 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 FB		549	REP STOSW	; FILL 8K WORDS
E1B5 8B				
E1B6 812C0004		550	ADD DX,400H	; POINT TO NEXT 16KB BLOCK
E1B8 83C310		551	ADD BX,16	; BUMP COUNT BY 16KB
E1BD 80FEAO		552	CMP DH,0400H	; TOP OF RAM AREA YET? (A0000)
E1C0 75DA		553	JNZ FILL_LOOP	
E1C2		554	MOV DATA_WORD[OFFSET MEMORY_SIZE],BX	; SAVE MEMORY SIZE
E1C2 891E1304		555		
		556	----- SETUP STACK SEG AND SP	
		558		
E1C6 B83000		559	MOV AX,STACK	; GET STACK VALUE
E1C9 8ED0		560	MOV SS,AX	; SET THE STACK UP
E1CB 00001		561	MOV SP,OFFSET TOS	; STACK IS READY TO GO
		562	-----	
		563	INITIALIZE THE 8259 INTERRUPT CONTROLLER CHIP :	
		564		
E1CE B013		565	C25: MOV AL,13H	; ICW1 - EDGE, SNGL, ICW4
E1D0 E620		566	OUT INTA00,AL	
E1D4 E605		567	MOV AL,00H	; SETUP ICW2 - INT TYPE 8 (8-F)
E1D4 E621		568	OUT INTA01,AL	
E1D6 B009		569	MOV AL,9	; SETUP ICW4 - BUFRD,8086 MODE
E1D8 E621		570	OUT INTA01,AL	
E1DA B0FF		571	MOV AL,0FFH	; MASK ALL INTS. OFF
E1DC E621		572	OUT INTA01,AL	; (VIDEO ROUTINE ENABLES INTS.)
		573		
		574	----- SET UP THE INTERRUPT VECTORS TO TEMP INTERRUPT	
		575		
E1E4 IE		576	PUSH DS	
E1F0 B92000		577	MOV CX,32	; FILL ALL 32 INTERRUPTS
E1E2 2BFF		578	SUB DI,DI	; FIRST INTERRUPT LOCATION
E1E4 8C7C		579	MOV ES,DI	; SET ES=0000 ALSO
E1E6 B823FF		580	D3: MOV AX,OFFSET DII	; MOVE ADDR OF INTR PROC TO TBL
E1E7 AB		581	STOSW	
E1E8 8C8C		582	MOV AX,CS	; GET ADDR OF INTR PROC SEG
E1EC AB		583	STOSW	
E1ED E2F7		584	LOOP D3	; VECTBL0
		585		
		586	----- ESTABLISH BIOS SUBROUTINE CALL INTERRUPT VECTORS	
		587		
E1EF BF4000		588	MOV DI,OFFSET VIDEO_INT	; SETUP ADDR TO INTR AREA
E1F2 0E		589	PUSH CS	
E1F3 1F		590	POP DS	; SETUP ADDR OF VECTOR TABLE
E1F4 8CD8		591	MOV AX,DS	; SET AX=SEGMENT
E1F6 BE03FF90		592	MOV SI,OFFSET VECTOR_TABLE+16	; START WITH VIDEO ENTRY
E1F7 1000		593	MOV CX,16	
E1FD A5		594	D3A: MOV SW	; MOVE VECTOR TABLE TO RAM
E1FE A7		595	INC DI	; SKIP SEGMENT POINTER
E1FF 47		596	INC DI	
E200 E2FB		597	LOOP D3A	
		598	-----	
		599	DETERMINE CONFIGURATION AND MFG. MODE :	
		600		
		601		
E202 1F		602	POP DS	
E203 E0		603	PUSH DS	
E204 E462		604	IN AL,PORT_C	
E205 0000		605	AND AL,0000111B	; GET SWITCH INFO
E208 8AE0		606	MOV AH,AL	; SAVE SWITCHES
E20A BOAD		607	MOV AL,10101010B	; ENABLE OTHER BANK OF SWS.
E20C E661		608	OUT PORT_B,AL	
E20E 90		609	NOP	
E20F E462		610	IN AL,PORT_C	
E211 0041		611	MOV CL,4	
E213 D200		612	ROL AL,CL	
E215 24F0		613	AND AL,1111000B	; ROTATE TO HIGH NIBBLE
E217 0AC4		614	OR AL,AH	; ISOLATE
E219 2AE4		615	SUB AH,AH	; COMBINE WITH OTHER BANK
E21B A31004		616	MOV DATA_WORD[OFFSET EQUIP_FLAG],AX	; SAVE SWITCH INFO
E21D B095		617	MOV A,99H	
E220 0000		618	OUT CMOS_RESET,AL	
E222 E00518		619	CALL KBD_RESET	; SEE IF MFG. JUMPER IN
E225 80FBAA		620	CMP BL,0AAH	; KEYBOARD PRESENT?
E226 7418		621	JE E6	
E22A 80FB65		622	CMP BL,065H	; LOAD MFG. TEST REQUEST?
E22D 7503		623	JNE D3B	
E22E FFDD		624	JMP WORD BOOT	; GO TO BOOTSTRAP IF SO
E232 B038		625	D3B: MOV AL,38H	
E234 E661		626	OUT PORT_B,AL	
E236 90		627	NOP	
E237 90		628	NOP	
E238 E460		629	IN AL,PORT_A	
E239 00FF		630	AND AL,0FFFH	; WAS DATA LINE GROUNDED
E23C 7504		631	JNZ E6	
E23E FE061204		632	INC DATA_AREA[OFFSET MFG_TST]	; SET MANUFACTURING TEST FLAG
		633		

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

```

634 :-----+
635 : INITIALIZE AND START CRT CONTROLLER (16845)
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 : ADDRESSABLE CARD
642 : ERROR = 1 LONG AND 2 SHORT BEEPS
643 :-----+
644 E6:
645 MOV AX,DATA_WORD[OFFSET EQUIP_FLAG]; GET SENSE SWITCH INFO
646 PUSH AX ; SAVE IT
647 MOV AL,30H
648 MOV DATA_WORD[OFFSET EQUIP_FLAG],AX
649 SUB AH,AH
650 INT 10H ; SEND INIT TO B/W CARD
651 MOV AL,20H
652 MOV DATA_WORD[OFFSET EQUIP_FLAG],AX
653 SUB AH,AH ; AND INIT COLOR CARD
654 INT 10H
655 POP AX ; RECOVER REAL SWITCH INFO
656 MOV DATA_WORD[OFFSET EQUIP_FLAG],AX ; RESTORE IT
657 :-----+
658 AND AL,30H ; AND CONTINUE
659 JNZ E7 ; ISOLATE VIDEO SWS
660 MOV DI,OFFSET VIDEO_INT ; VIDEO SWS SET TO 0?
661 MOV [DI],OFFSET DUMMY_RETURN ; SET INT 10H TO DUMMY
662 JMP E18_1 ; RETURN IF NO VIDEO CARD
663 E7:
664 CMP AL,30H ; BYPASS VIDEO TEST
665 JE E8 ; TEST VIDEO:
666 INC AH ; B/W CARD ATTACHED?
667 CMP AL,20H ; YES - SET MODE FOR B/W CARD
668 JNE E8 ; SET COLOR MODE FOR COLOR CD
669 MOV AH,3 ; 80X25 MODE SELECTED?
670 E8: XCHG AH,AL ; NO - SET MODE FOR 40X25
671 PUSH AX ; SET MODE FOR 80X25
672 SUB AH,AH ; SET MODE
673 INT 10H ; SAVE VIDEO MODE ON STACK
674 POP AX ; INITIATE THE ALPHANUMERIC MD
675 PUSH AX ; CALL VIDEO_IO
676 MOV BX,0B000H ; RESTORE VIDEO SENSE SWS IN AH
677 MOV DX,3B8H ; RESAVE VALUE
678 MOV CX,048 ; BEG VIDEO RAM ADDR B/W CD
679 MOV AL,1 ; MODE REG FOR B/W CD
680 CMP AH,30H ; MODE REG FOR COLOR CD
681 JE E9 ; MODE REG FOR COLOR CD
682 MOV BH,0B8H ; MODE REG FOR COLOR CD
683 MOV DX,3D8H ; MODE REG FOR COLOR CD
684 MOV CH,20H ; MODE REG FOR COLOR CD
685 DEC AL ; SET MODE TO 0 FOR COLOR CD
686 E9: TEST VIDEO_STG;
687 OUT DX,AL ; DISABLE VIDEO FOR COLOR CD
688 CMP DATA_WORD[OFFSET RESET_FLAG],1234H ; POD INIT BY KBD RESET?
689 MOV ES,BX ; POINT ES TO VIDEO RAM STG
690 JE E10 ; TEST VIDEO RAM TEST
691 MOV DS,BX ; POINT DS TO VIDEO RAM STG
692 ASSUME DS:NOTHING,ES:NOTHING ; POINT DS TO VIDEO RAM STG
693 CALL STGTST_CNT ; GO TEST VIDEO R/W STG
694 JNE E17 ; R/W STG FAILURE - BEEP SPK
695 :-----+
696 :-----+ SETUP VIDEO DATA ON SCREEN FOR VIDEO
697 :-----+ LINE TEST.
698 :-----+ DESCRIPTION
699 :-----+ ENABLE VIDEO SIGNAL AND SET MODE.
700 :-----+ DISPLAY A HORIZONTAL BAR ON SCREEN.
701 :-----+
702 E10:
703 POP AX ; GET VIDEO SENSE SWS (AH)
704 PUSH AX ; SAVE IT
705 MOV AH,0 ; ENABLE VIDEO AND SET MODE
706 INT 10H ; VIDEO
707 MOV AX,7020H ; WRT BLANKS IN REVERSE VIDEO
708 :-----+ UNNATURAL ACT FOR ADDRESS COMPATIBILITY
709 :-----+
710 E11: JMP SHORT E10A
711 T12 ORG 0E2C3H
712 JMP NM1_INT
713 :-----+
714 :-----+ E10A:
715 T16 SUB DI,DI ; SETUP STARTING LOC
716 MOV CX,40 ; NO. OF BLANKS TO DISPLAY
717 REP STOSW ; WRITE VIDEO STORAGE
718 :-----+
719 :-----+ CRT INTERFACE LINES TEST
720 :-----+ DESCRIPTION
721 :-----+ SENSE ON/OFF TRANSITION OF THE
722 :-----+ VIDEO ENABLE AND HORIZONTAL
723 :-----+ SYNC LINES.
724 :-----+
725 :-----+
726 POP AX ; GET VIDEO SENSE SW INFO
727 PUSH AX ; SAVE IT
728 CMP AH,30H ; B/W CARD ATTACHED?
729 MOV DX,03BAH ; SETUP ADDR OF BW STATUS PORT
730 JE E11 ; YES - GO TEST LINES
731 MOV DX,03DAH ; COLOR CARD IS ATTACHED
732 E11: MOV AH,8 ; LINE_TST?
733 E12: MOV AX,CX ; OFLLOOP_CNT:
734 E12: SUB CX,CX
735 E13: IN AL,DX ; READ CRT STATUS PORT
736 E13: AND AL,AH ; CHECK VIDEO/HORZ LINE
737 E13: JNZ E14 ; ITS ON - CHECK IF IT GOES OFF
738 E14: LOOP E13 ; LOOP TILL ON OR TIMEOUT
739 E14: JMP SHORT E17 ; GO PRINT ERROR MSG
740 E14: :-----+
741 E14: IN AL,DX ; READ CRT STATUS PORT
742 E14: AND AL,AH ; CHECK VIDEO/HORZ LINE
743 E14: JNZ E15 ; ITS ON - CHECK NEXT LINE
744 E14: LOOP E15 ; LOOP IF OFF TILL IT GOES ON
745 E15: IN AL,DX ; READ CRT STATUS PORT
746 E15: AND AL,AH ; CHECK VIDEO/HORZ LINE
747 E15: JZ E16 ; ITS ON - CHECK NEXT LINE
748 E15: LOOP E15 ;-----+

```

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

```

E3F0          749 E17:    POP      DS                  ; CRT_ERR:
E2F0  IF       750 PUSH    DS
E2F1  IE       751 MOV     DS:MF6_ERR_FLAG,06H ; <><>CRT_ERR CHKPT. 06<><>
E2F2 C606150006 752 MOV     DX,102H
E2F7 BA0201   753 CALL    ERR_BEEP
E2FA EBD816   754 JMP     SHORT E18
E2FB EBD806   755
E2FF          756 E16:    MOV     CL,3
E2F8 B103     757 MOV     AH,CL
E301 D2EC     758 SHR     AH,CL
E303 75D7     759 JNZ     E12
E305          760 E18:    POP     AX
E306 58       761 MOV     AH,0
E308 CD10     762 INT     10H
E30A          763
E30A BA00C0   764 E18_1: MOV     DX,0C000H
E30D          765
E30F 8EDA     766 E18A:  MOV     DS,DX
E30F 2B08     767 SUB     BX,DX
E311 8807     768 MOV     AX,[BX]
E313 53       769 PUSH    BX
E314 5B       770 POP     BX
E315 3D55AA   771 CMP     AX,0AA55H
E318 7505     772 JNZ     E18B
E319 E83616   773 CALL    ROM_CHECK
E31D EB04     774 JMP     SHORT E18C
E31F          775
E31F 81C28000 776 E18B:  ADD     DX,0006H
E323          777
E323 81FA00C8 778 E18C1: CMP     DX,0C000H
E327 TCE4     779 JL      E18A
E327          780
E327          781 ;----- 8259 INTERRUPT CONTROLLER TEST
E329 1F       782 ; DESCRIPTION
E329          783 : READ/WRITE THE INTERRUPT MASK REGISTER (IMR)
E329          784 : WITH THE LINES AND FIELDS ENCODED SYSTEM
E329          785 : INTERRUPTS. MASK DEVICE INTERRUPTS OFF. CHECK
E329          786 : FOR HOT INTERRUPTS (UNEXPECTED).
E329          787 :----- 1
E329          788 :----- 2
E32A C606150405 789 C21:  ASSUME DS:AB50
E32A          790 C21:  POP     DS
E32A          791
E32A          792 ;----- TEST THE IMR REGISTER
E32A          793
E32F B000     794 C21A: MOV     DATA_AREA[OFFSET MFR_ERR_FLAG],05H
E331 E621     795 ; <><><><><><><><>
E333 E421     796 ; SET IMR TO ZERO
E335 0AC0     797 MOV     AL,0
E337 751B     798 OUT    INTA01,AL
E339 B0FF     799 IN     AL,INTA01
E340 E421     800 OR     AL,AL
E341 E421     801 JNZ     D6
E342 8005     802 MOV     AL,0FFH
E343 E421     803 OUT    INTA01,AL
E344 E421     804 IN     AL,INTA01
E345 0401     805 ADD     AL,1
E346 T511     806 JNZ     D6
E346          807
E346          808 ;----- CHECK FOR HOT INTERRUPTS
E346          809
E346          810 ;----- INTERRUPTS ARE MASKED OFF. CHECK THAT NO INTERRUPTS OCCUR.
E346          811
E347 A26B04   812 MOV     DATA_AREA[OFFSET INTR_FLAG],AL ; CLEAR INTERRUPT FLAG
E347 FB       813 STI
E347 2BC9     814 SUB     CX,CX
E349          815
E349 E2FE     816 D4:    LOOP    D4
E349          817 : MIGHT OCCUR
E349 E2FE     818 D5:    LOOP    D5
E349 8036E6B0400 819 CMP     DATA_AREA[OFFSET INTR_FLAG],00H ; DID ANY INTERRUPTS OCCUR?
E352 7409     820 JZ      D7
E354          821
E354 BEFFF890 822 MOV     SI,OFFSET E0 ; DISPLAY 101 ERROR
E355 E84E16   823 CALL    E_MSG
E356 FA       824 CLI
E356 F4       825 HLT
E356          826 ;----- HALT THE SYSTEM
E356          827 ;----- 8259 TIMER CHECKOUT
E356          828 ; DESCRIPTION
E356          829 : VERIFY THAT THE SYSTEM TIMER (0) DOESN'T COUNT
E356          830 : TOO FAST OR TOO SLOW.
E356          831
E35D C606150402 832 D7:    MOV     DATA_AREA[OFFSET MFR_ERR_FLAG],03H
E35D          833
E35D          834
E35D          835 ;----- 1
E35D          836 : MASK ALL INTRS EXCEPT LVL 0
E35D          837 : WRITE THE 8259 IMR
E35D          838 : SEL TIM 0, LSB, MODE 0, BINARY
E35D          839 : SET PGM LOOP CNT
E35D          840 : SET PGM LOOP CNT
E35D          841 : SET TIMER 0 CNT REG
E35D          842 : WRITE TIMER CNT REG
E35D          843 D8:    OUT    TIMER0,AL
E35D          844 TEST   DATA_AREA[OFFSET INTR_FLAG],01H
E35D          845 : SET PGM LOOP CNT
E35D          846 JNZ     D9
E35D          847 LOOP   D8
E35D          848 JMP     D6
E35D          849 D9:    ; TIMER 0 INTERRUPT OCCUR?
E35D          850 MOV     CL,12
E35D          851 MOV     AL,0FEH
E35D          852 OUT    INTA00,AL
E35D          853 MOV     DATA_AREA[OFFSET INTR_FLAG],0 ; RESET INTR RECEIVED FLAG
E35D          854 MOV     AL,0FEH
E35D          855 OUT    INTA01,AL
E35D          856 D10:   TEST   DATA_AREA[OFFSET INTR_FLAG],01H ; DID TIMER 0 INTERRUPT OCCUR?
E35D          857 JNZ     D6
E35D          858 D10:   ; YES - TIMER COUNTING TOO FAST, ERR
E35D          859 LOOP   D10
E35D          860 ;----- 2
E376 T504     861
E378 E2F7     862
E378 EBD8     863
E37C          864
E37C B10C     865
E37E B0FF     866
E37E 8040     867
E382 C6066B0400 868
E383 B0FF     869
E383 E621     870
E388          871
E388 F6066B0401 872
E390 T5C2     873
E392 E2FT     874

```

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

```

861 ;---- SETUP TIMER 0 TO MODE 3
862
E394 B0FF 863 MOV AL,0FFH ; DISABLE ALL DEVICE INTERRUPTS
E396 E621 864 OUT INTA01,AL
E398 B036 865 MOV AL,36H ; SEL TIM 0,L5B,MSB,MODE 3
E39A E643 866 OUT TIMER+3,AL ; WRITE TIMER MODE REG
E39C B000 867 MOV AL,0
E39E E640 868 OUT TIMER,AL ; WRITE LSB TO TIMER 0 REG
E3A0 E640 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 ;
876 ;---- TST12:
877 TST12: MOV AL,99H ; SET 8255 MODE A,C=IN B=OUT
878
E3A2 B099 879 OUT CMD_PORT_AL
E3A4 E663 880 MOV AL,DATA_AREA[OFFSET EQUIP_FLAG]
E3A6 B0304 881 ANI AL,01 ; TEST CHAMBER?
E3A9 2401 882 JZ F7 ; BYPASS IF ?
E3AB 7431 883 CMP DATA_AREA[OFFSET MFG_TST] ; I ; MANUFACTURING TEST MODE?
E3B2 7424 884 JE F7 ; YES - SKIP KEYBOARD TEST
E3B4 E87316 885 CALL KBD_RESET ; ISSUE RESET TO KEYBD
E3B5 E80400 886 JCXZ F6 ; PRINT ERR MSG IF NO INTERRUPT
E3B9 B049 887 MOV AL,49H ; ENABLE KEYBOARD
E3B8 E661 888 OUT PORT_B,AL
E3BD B0FBAA 889 CMP BL,0AAH ; SCAN CODE AS EXPECTED?
E3C0 T515 890 JNE F6 ; NO - DISPLAY ERROR MSG
891
892 ;---- CHECK FOR STUCK KEYS
893
E3C2 B0CB 894 MOV AL,0CBH ; CLR KBD, SET CLK LINE HIGH
E3C4 E661 895 OUT PORT_B,AL
E3C6 B045 896 MOV AL,4BH ; ENABLE KBD,CLK IN NEXT BYTE
E3C8 E661 897 OUT PORT_B,AL
E3D0 2BC9 898 SUB CX,CX
E3C9 899 F5: ; KBD_WAIT:
E3C C2FE 900 LOOP F5 ; DELAY FOR A WHILE
E3C E460 901 IN AL,KBD_IN ; CHECK FOR STUCK KEYS
E3D0 3C00 902 CMP AL,0 ; SCAN CODE = ??
E3D2 740A 903 JE F7 ; YES - CONTINUE TESTING
E3D4 E8B415 904 CALL XPC_BYT E ; CONVERT AND PRINT
E3D5 F6: ; GET MSG ADDR
E3D7 B84CEC90 905 MOV SI,OFFSET FI ; PRINT MSG ON SCREEN
E3DB E8CB15 906 CALL E_MSG
907
908 ;---- SETUP HARDWARE INT. VECTOR TABLE
909 ;-
910 F7: ;-
911 ;-
912 PUSH DS ; SETUP_INT_TABLE:
913 SUB AX,AX
914 MOV ES,AX
E3E3 B90800 915 MOV CX,08 ; GET VECTOR CNT
E3E5 0E 916 PUSH CS ; SETUP DS SEG REG
E3E7 1F 917 POP DS
E3E8 BEF3FE90 918 MOV SI,OFFSET VECTOR_TABLE
E3EC BF2000 919 MOV D1,OFFSET INT_PTR
E3EF 920 F7A: ;-
921 MOVWS ; SKIP OVER SEGMENT
922 INC DI
923 INC DI
E3F2 E2FB 924 LOOP F7A
E3F4 1F 925 POP DS
926
927 ;---- SET UP OTHER INTERRUPTS AS NECESSARY
928
E3F5 C70608005FF8 929 MOV NMI_PTR,OFFSET NMI_INT ; NMI INTERRUPT
E3FB C706140054FF 930 MOV INT5_PTR,OFFSET PRINT_SCREEN ; PRINT SCREEN
E401 C706620000F6 931 MOV BASIC_PTR+2,0F600H ; SEGMENT FOR CASSETTE BASIC
932
933 ;---- SETUP TIMER 0 TO BLINK LED IF MANUFACTURING TEST MODE
934
E407 B03E120401 935 CMP DATA_AREA[OFFSET MFG_TST],01H ; MFG. TEST MODE?
E40C 750A 936 JNZ EXP_TO
E40E C70670003CF9 937 MOV WORD PTR [ICH*4],OFFSET BLINK_INT; SETUP TIMER INTR TO BLINK LED
E414 B0FE 938 MOV AL,0FEH ; ENABLE TIMER INTERRUPT
E416 E621 939 OUT INTA01,AL

```

LOC	OBJECT	LINE	SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82
E418		940	; -----
E418 BA1002		941	; EXPANSION I/O BOX TEST
E418 BB5555		942	; CHECK TO SEE IF EXPANSION BOX PRESENT - IF INSTALLED,
E41E EE		943	; TEST DATA AND ADDRESS BUSES TO I/O BOX
E41F B001		944	; ERROR='1801'
E421 EC		945	; -----
E424 3AC4		946	; DETERMINE IF BOX IS PRESENT
E424 544		947	; -----
E426 F7D0		948	EXP_IO:
E428 EE		950	MOV DX,0210H ; (CARD WAS ENABLED EARLIER)
E429 B001		951	MOV AX,5555H ; CONTROL PORT ADDRESS
E42B EC		952	OUT DX,AL ; SET DATA PATTERN
E42C 3AC4		953	MOV AL,01H ; MAKE AL DIFFERENT
E42E 753A		954	IN AL,DX ; RECOVER DATA
		955	CMP AL,AH ; REPLY?
		956	JNE E19 ; NO RESPONSE, GO TO NEXT TEST
		957	NOT AX ; MAKE DATA=AAAA
		958	OUT DX,AL
		959	MOV AL,01H
		960	IN AL,DX ; RECOVER DATA
		961	CMP AL,AH
		962	JNE E19
		963	
		964	; ----- CHECK ADDRESS BUS
		965	
E430		966	EXP2:
E430 BB0100		967	MOV BX,0001H ; LOAD HI ADDR. REG ADDRESS
E433 BA1502		968	MOV DX,0215H ; GO ACROSS 16 BITS
E436 B91000		969	MOV CX,0016
E439		970	EXP3:
E439 2E8607		971	MOV CS:[BX],AL ; WRITE ADDRESS F0000+BX
E43C 90		972	NOP
E43E EC		973	IN AL,DX ; READ ADDR. HIGH
E43E 7AC7		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
E444 51B		979	TEST DX
E448 44		980	DEC DX ; DX BACK TO 215H
E449 D1E3		981	SHL BX,1
E44B E2EC		982	LOOP EXP3 ; LOOP TILL '*' 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 ; SAVE DATA BUS VALUE
E455 EE		991	OUT DX,AL ; SEND VALUE TO REG
E456 B001		992	MOV AL,01H
E458 C		993	IN AL,DX
E459 3AC4		994	CMP AL,BH ; RETRIEVE VALUE FROM REG
E45B 7506		995	JNE SHORT EXP_ERR ; = TO SAVED VALUE
E45D D0E0		996	SHL AL,1
E45F E2F2		997	LOOP EXP4 ; FORM NEW DATA PATTERN
E461 EB07		998	JMP SHORT E19 ; LOOP TILL BIT WALKS ACROSS AL
E463		999	
E463 BE0FF990		1000	MOV SI,OFFSET F3C ; GO ON TO NEXT TEST
E467 E83F15		1001	CALL E_MSG

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

```

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 E19:  

1011 CALL DDS  

1012 PUSH DS  

1013 E20:  

1014 CMP RESET_FLAG,1234H ; WARM START?  

1015 JNE E20A ; CONTINUE TEST IF NOT  

1016 JMP ROM_SCAN ; GO TO NEXT ROUTINE IF SO  

1017 E20A:  

1018 MOV AX,16 ; STARTING AMT. OF MEMORY OK  

1019 JMP SHORT_PRT_SIZE ; POST MESSAGE  

1020 E20B:  

1021 MOV BX,MEMORY_SIZE ; GET MEM. SIZE WORD  

1022 SUB BX,16 ; 1ST 16K ALREADY DONE  

1023 MOV CL,00H  

1024 SHL BX,CL ; DIVIDE BY 16  

1025 MOV CX,BX ; SAVE COUNT OF 16K BLOCKS  

1026 MOV BX,0400H ; SET PTR. TO RAM SEGMENT>16K  

1027 E21:  

1028 MOV DS,BX ; SET SEG. REG  

1029 MOV ES,BX  

1030 ADD BX,0400H ; POINT TO NEXT 16K  

1031 PUSH DX  

1032 PUSH CX ; SAVE WORK REGS  

1033 PUSH BX  

1034 PUSH AX  

1035 MOV CX,2000H ; SET COUNT FOR 8K WORDS  

1036 CALL STGTS_CNT  

1037 JNZ E21A ; GO PRINT ERROR  

1038 POP AX ; RECOVER TESTED MEM NUMBER  

1039 ADD AX,16  

1040 PRT_SIZE:  

1041 PUSH AX  

1042 MOV BX,10 ; SET UP FOR DECIMAL CONVERT  

1043 MOV CX,3 ; OF 3 NIBBLES  

1044 DECIMAL_LOOP:  

1045 XOR DX,DX ; DIVIDE BY 10  

1046 DIV BX ; MAKE INTO ASCII  

1047 OR DL,30H ; SAVE  

1048 PUSH DX  

1049 LOOP DECIMAL_LOOP  

1050 MOV CX,3  

1051 PRT_DEC_LOOP:  

1052 POP AX ; RECOVER A NUMBER  

1053 CALL PRT_HEX  

1054 LOOP PRT_DEC_LOOP  

1055 MOV CX,7  

1056 MOV SI,OFFSET F3B ; PRINT ' KB OK'  

1057 KB_LOOP:  

1058 MOV AL,CS:[SI]  

1059 INC SI  

1060 CALL PRT_HEX  

1061 LOOP KB_LOOP ; RECOVER WORK REGS  

1062 POP AX  

1063 CMP AX,16 ; FIRST PASS?  

1064 JE E20B  

1065 POP BX  

1066 POP DX  

1067 POP DS  

1068 LOOP E21 ; LOOP TILL ALL MEM. CHECKED  

1069 MOV AL,10 ; LINE FEED  

1070 CALL PRT_HEX  

1071  

1072 ;----- DMA TCO SHOULD BE ON BY NOW - SEE IF IT IS  

1073  

E4DF E408 1074 IN AL,DMA+08H ; TCO STATUS BIT ON?  

E4E1 2401 1075 AND AL,00000001B ; GO ON WITH NEXT TEST IF OK  

E4E3 7533 1076 JNZ ROM_SCAN  

E4E5 5B4F 1077 POP DS  

E4E6 C606150003 1078 MOV MFG_ERR_FLAG,03H ; <><><><><><><><><>  

E4EB E966FE 1079 JMP D6 ; POST 101 ERROR MSG AND HALT  

1080  

1081 ;----- PRINT FAILING ADDRESS AND XOR'ED PATTERN IF DATA COMPARE ERROR  

1082  

E4EE 8AE8 1083 E21A: MOV CH,AL ; SAVE FAILING BIT PATTERN  

E4F0 B000 1084 MOV AL,13 ; CARRAGE RETURN  

E4F2 E8A714 1085 CALL PRT_HEX  

E4F5 B00A 1086 MOV AL,TO  

E4F7 E8A214 1087 CALL PRT_HEX  

E4FA 58 1088 POP AX ; RECOVER AMT. OF GOOD MEM.  

E4FB 82C406 1089 ADD SP,6 ; BALANCE STACK  

E4FC 02DA 1090 MOV DX,DS ; GET FAILING SEGMENT  

E500 1F  

E501 1E 1091 POP DS  

E502 A31300 1092 PUSH DS  

1093 MOV MEMORY_SIZE,AX ; LOAD MEM. SIZE WORD TO SHOW  

E505 88361500 1094 MOV MFG_ERR_FLAG,DH ; HOW MUCH MEM. WORKING  

1095  

E509 E8CE1A 1097 CALL PRT_SEG ; <><><><><><><><><>  

E50C 8AC5 1098 MOV AL,CH ; PRINT IT  

E50E E87A14 1099 CALL XPC_BYTTE ; GET FAILING BIT PATTERN  

E511 BE04F990 1100 MOV SI,0FFSET EI ; CONVERT AND PRINT CODE  

E515 E89114 1101 CALL E_MSG ; SETUP ADDRESS OF ERROR MSG  

1102 ;-----  


```

```

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:
E518 BA00C8 1108 MOV DX,0C800H ; SET BEGINNING ADDRESS
E51B 8EDA 1109 MOV DS,DX
E51B 8EDA 1110 MOV BX,BX ; SET BX=0000
E51B 8DBB 1111 SUB DS,BX
E51F 8B07 1112 MOV AX,[BX] ; GET 1ST WORD FROM MODULE
E521 53 1113 PUSH BX
E522 5B 1114 POP BX ; BUS SETTLING
E523 D355AA 1115 CMP AX,0AA55H ; = TO ID WORD?
E526 7506 1116 JNZ NEXT_ROM ; PROCEED TO NEXT ROM IF NOT
E529 E82814 1117 CALL ROM_CHECK ; GO CHECK OUT MODULE
E529 EB0590 1118 JMP ARE_WE_DONE ; CHECK FOR END OF ROM SPACE
E52E 81C28000 1119 ADD DX,0080H ; POINT TO NEXT 2K ADDRESS
E532 81FA00F6 1120 ARE_WE_DONE: CMP DX,0F600H ; AT F6000 YET?
E536 TCE3 1121 ADD DS,DX ; DO CHECK ANOTHER ADD. IF NOT
E538 EB0190 1122 JMP BASE_ROM_CHK ; GO CHECK BASIC ROM
E538 ;-----+
E53B BASE_ROM_CHK: 1123 ; A CHECKSUM IS DONE FOR THE 4 ROM MODULES CONTAINING BASIC CODE
E53B B404 1124 ;-----+
E53D 8ED4 1125 ;-----+
E53D 8ED4 1126 ;-----+
E53F 8EDA 1127 ;-----+
E541 E8AEI3 1128 ;-----+
E544 7403 1129 BASE_ROM_CHK: MOV AH,4 ; NO. OF ROM MODULES TO CHECK
E546 E68201 1130 JE E5 ;-----+
E549 1131 CALL ROM_ERR ; POST ERROR
E549 81C20002 1132 SUB BX,BX ; SETUP STARTING ROM ADDR
E54D FEC0 1133 MOV DS,DX ;-----+
E54F 75EC 1134 CALL ROS_CHECKSUM ; CHECK ROM
E54F 75EC 1135 CALL ROS_CHECKSUM ; CONTINUE IF OK
E54F 75EC 1136 CALL ROM_ERR ; POST ERROR
E54F 81C20002 1137 ADD DX,0200H ; POINT TO NEXT 8K MODULE
E54F 75EC 1138 E5: DEC AH ; ANY MORE TO DO?
E54F 75EC 1139 JNZ E4 ; YES - CONTINUE
E54F 75EC 1140 ;-----+
E54F 75EC 1141 ;-----+
E54F 75EC 1142 ;-----+
E54F 75EC 1143 ; DISKETTE ATTACHMENT TEST
E54F 75EC 1144 ;-----+
E54F 75EC 1145 ;-----+
E54F 75EC 1146 ;-----+
E54F 75EC 1147 ;-----+
E54F 75EC 1148 ;-----+
E54F 75EC 1149 ;-----+
E54F 75EC 1150 ;-----+
E551 F9: 1151 ;-----+
E551 IF 1152 POP DS ;-----+
E552 A01000 1153 MOV AL,BYTE PTR EQUIP_FLAG ; DISKETTE PRESENT?
E555 2401 1154 AND AL,01H ; NO - BYPASS DISKETTE TEST
E555 743E 1155 JZ F15 ;-----+
E559 4241 1156 ;-----+
E55B 424F 1157 IN AL,INTA01 ; DISK_TEST:
E55D E621 1158 AND AL,0BFH ;-----+
E55F B400 1159 OUT INTA01,AL ; ENABLE DISKETTE INTERRUPTS
E561 AD4 1160 MOV AH,0 ;-----+
E563 CD13 1161 MOV DL,AH ; RESET NEC FDC
E565 F6CAFF 1162 INT 13H ; SET FOR DRIVE 0
E568 7520 1163 TEST AL,0FFH ; VERIFY STATUS AFTER RESET
E568 7520 1164 JNZ F13 ;-----+
E568 7520 1165 ;-----+
E568 7520 1166 ;-----+ TURN DRIVE 0 MOTOR ON
E568 7520 1167 ;-----+
E568 7520 1168 MOV DX,03F2H ;-----+
E568 7520 1169 MOV AL,1CH ; TURN MOTOR ON, EN DMA/INT
E568 7520 1170 OUT DX,AL ;-----+
E570 2BC9 1171 SUB CX,CX ; WRITE FDC CONTROL REG
E572 F10: 1172 ;-----+
E572 E2FE 1173 LOOP F11 ;-----+
E572 E2FE 1174 F12: ;-----+
E576 33D2 1175 LOOP F12 ;-----+
E578 B501 1176 XOR DX,DX ;-----+
E578 B501 1177 MOV CH,1 ;-----+
E578 B501 1178 MOV SEEK_STATUS,DL ;-----+
E578 E8FC08 1179 CALL SEEK- ;-----+
E578 E8FC08 1180 JC F13 ;-----+
E583 B522 1181 MOV CH,34 ;-----+
E585 E8F508 1182 CALL SEEK ;-----+
E588 7307 1183 JNC F14 ;-----+
E58A ;-----+
E58A BE52EC90 1184 F13: ;-----+
E58A BE52EC90 1185 MOV SI,OFFSET F3 ;-----+
E58E E81814 1186 CALL E_MSG ;-----+
E58E E81814 1187 ;-----+
E588 ;-----+ TURN DRIVE 0 MOTOR OFF
E588 ;-----+
E591 F14: 1189 ;-----+
E591 B00C 1190 MOV AL,0CH ;-----+
E593 BA0F203 1191 MOV DX,03F2H ;-----+
E596 EE 1192 OUT DX,AL ;-----+
E596 EE 1193 ;-----+
E595 ;-----+ SETUP PRINTER AND R5232 BASE ADDRESSES IF DEVICE ATTACHED
E595 ;-----+
E597 F15: 1194 ;-----+
E597 C6066B0000 1195 MOV INTR_FLAG,00H ;-----+
E59C BE1A00 1196 MOV SI,OFFSET BUFFER ;-----+
E59F 89361A00 1197 MOV BUFFER_HEAD,$1 ;-----+
E5A3 89361C00 1200 MOV BUFFER_TAIL,$1 ;-----+
E5A7 89368000 1201 MOV BUFFER_START,$1 ;-----+
E5A7 83C620 1202 ADD SI,32 ;-----+
E5A7 83C68200 1203 MOV BUFFER_END,SI ;-----+
E5B2 BF1800 1204 MOV DS,OFFSET PRINT_TIM_OUT ;-----+
E5B5 1E 1205 PUSH DS ;-----+
E5B6 07 1206 MOV AX,1414H ;-----+
E5B7 BB1414 1207 POP ES ;-----+
E5B8 AB 1208 MOV AX,1414H ;-----+
E5B8 AB 1209 STOSW ;-----+
E5B8 AB 1210 STOSW ;-----+
E5BC B60101 1211 MOV AX,0101H ;-----+
E5BF AB 1212 STOSW ;-----+
E5C0 AB 1213 STOSW ;-----+
E5C1 E421 1214 IN AL,INTA01 ;-----+
E5C3 244C 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 T419 1218 JL F15A_0 ; CONTINUE IF NO ERROR
E5CC BA0200 1220 MO DX,2 ; 2 SHORT BEEPS (ERROR)
E5CF E0614 1221 CALL ERR_BEEP
E5D0 BE09E890 1222 MOV SI,OFFSET F3D ; LOAD ERROR MSG
E5D6 EB113 1223 CALL P_MSG
E5D9 B400 1224 ERR_WAIT: ; BYPASS ERROR
E5DB CD14 1225 MOV AH,00 ; WAIT FOR 'F1' KEY
E5DD 80FC3B 1226 INT 16H
E5E0 75F7 1227 CMP AH,3BH
E5E2 EB0E90 1228 JNE ERR_WAIT
E5E3 F15A_0 1229 JMP F15A_0 ; BYPASS ERROR
E5E5 803E120001 1230 F15A_0: ; MFG MODE
E5E6 7406 1231 CMP MFC TST,T ; BYPASS BEEP
E5E7 BA0100 1232 JE F15B ; I SHORT BEEP (NO ERRORS)
E5E9 E8E13 1233 MOV DX,1
E5F2 A01000 1234 CALL ERR_BEEP
E5F5 2401 1235 F15A: MOV AL,_BYTE PTR EQUIP_FLAG ; GET SWITCHES
E5F7 7503 1236 AND AL,00000001B ; 'LOOP POST' SWITCH ON
E5F9 805FFA 1237 JNZ F15B ; CONTINUE WITH BRING-UP
E5FC 2AE4 1238 JMP START
E5FE A04900 1239 F15B: SUB AL,AH
E601 CD10 1240 MOV AL,CRT_MODE
E602 803E120001 1241 INT 10H ; CLEAR SCREEN
E603 BDA3F990 1242 F15C: ; PRT_SRC_TBL
E607 BE0000 1243 MOV BP,OFFSET F4 ; PRT_SRC_TBL
E60A 1244 MOV SI,0 ; PRT_BASE
E60A 1245 F16: ; GET_PRINTER_BASE_ADDR
E60B BOAA 1246 MOV DX,CS:[BP] ; WRITE DATA TO PORT A
E610 EE 1247 MOV AL,0AAH
E611 1E 1248 OUT DX,AL ; BUS SETTLING
E612 EC 1249 PUSH DS ; READ PORT A
E613 1F 1250 IN AL,DX
E614 3CAA 1251 POP DS
E616 7505 1252 CMP AL,0AAH ; DATA PATTERN SAME
E618 B95408 1253 JNE F17 ; NO - CHECK NEXT PRT_CD
E61B 46 1254 MOV PRINTER_BASE[SI],DX ; YES - STORE PRT BASE ADDR
E61C 46 1255 INC SI ; INCREMENT TO NEXT WORD
E61D 46 1256 INC SI
E61D 45 1257 F17: ; POINT TO NEXT BASE ADDR
E61E 45 1258 INC BP
E61F 81FDA9F9 1259 INC BP ; ALL POSSIBLE ADDRS CHECKED?
E622 75E5 1260 CMP BP,OFFSET F4E ; PRT_BASE
E625 803E120001 1261 JNE F16 ; POINTER TO RS232 TABLE
E628 BAFA03 1262 MOV BX,0 ; CHECK IF RS232 CD 1 ATTCH?
E62B EC 1263 MOV DX,2FAH ; READ INTR ID REG
E62C ABF8 1264 IN AL,DX
E62E 7506 1265 TEST AL,0FH ; READ INTERRUPT ID REG
E630 C707FB03 1266 JNZ F18 ; BASE END
E631 43 1267 MOV RS232_BASE[BX],3F8H ; SETUP RS232 CD #1 ADDR
E635 43 1268 INC BX
E636 1269 INC BX
E638 BAFA02 1270 F18: ; BASE END
E639 EC 1271 MOV DX,2FAH ; CHECK IF RS232 CD 2 ATTCH
E640 803E120001 1272 IN AL,DX ; READ INTERRUPT ID REG
E641 43 1273 TEST AL,0FH ; BASE END
E643 7505 1274 JNZ F19 ; SETUP RS232 CD #2
E642 43 1275 MOV RS232_BASE[BX],2F8H
E643 43 1276 INC BX
E644 1277 INC BX
E645 1278 F19: ;---- SET UP EQUIP FLAG TO INDICATE NUMBER OF PRINTERS AND RS232 CARDS
E646 BBC6 1279 MOV AX,SI ; SI HAS 28 NUMBER OF RS232
E646 B103 1280 MOV CL,3 ; SHIFT COUNT
E648 D2C8 1281 ROR AL,CL ; ROTATE RIGHT 3 POSITIONS
E64A 04C3 1282 OR AL,BL ; OR IN THE PRINTER COUNT
E64C A21100 1283 MOV BYTE PTR EQUIP_FLAG+I,AL ; STORE AS SECOND BYTE
E64F E00102 1284 INT DX,201H
E652 EC 1285 IN AL,DX
E653 90 1286 NOP
E654 90 1287 NOP
E655 90 1288 NOP
E656 800F 1289 TEST AL,0FH ; NO_GAME_CARD
E658 7605 1290 JNZ F20 ; NO_GAME_CARD
E65A 800E110010 1291 OR BYTE PTR EQUIP_FLAG+I,16 ; NO_GAME_CARD
E65F 1292 F20: ;---- ENABLE NMI INTERRUPTS
E65F E461 1293 IN AL,PORT_B ; RESET CHECK ENABLES
E661 0C30 1294 OR AL,30H
E663 E661 1295 OUT PORT_B,AL
E665 24CF 1296 AND AL,0CFH
E667 E661 1297 OUT PORT_B,AL ; ENABLE NMI INTERRUPTS
E668 B080 1298 MOV AL,80H
E66B E6A0 1299 OUT 0A0H,AL
E66D CD19 1300 F21: ; LOAD BOOT STRAP;
E66D CD19 1301 INT 19H ; GO TO THE BOOT LOADER
E66D CD19 1302
E66D CD19 1303
E66D CD19 1304
E66D CD19 1305
E66D CD19 1306
E66D CD19 1307
E66D CD19 1308

```

```

1309 ;-----+
1310 ; THIS SUBROUTINE PERFORMS A READ/WRITE STORAGE TEST ON A BLOCK :
1311 ; OF STORAGE.
1312 ; ENTRY REQUIREMENTS:
1313 ;   ES = ADDRESS OF STORAGE SEGMENT BEING TESTED
1314 ;   DX = ADDRESS OF STORAGE SEGMENT BEING TESTED
1315 ;   CX = WORD COUNT OF STORAGE BLOCK TO BE TESTED
1316 ; EXIT PARAMETERS:
1317 ;   ZERO FLAG = 0 IF STORAGE ERROR (DATA COMPARE OR PARITY
1318 ;   CHECK, AL=0 DENOTES A PARITY CHECK. ELSE AL=XOR'ED
1319 ;   BIT PATTERN OF THE EXPECTED DATA PATTERN VS THE ACTUAL
1320 ;   DATA READ.
1321 ; AX,BX,CX,DX,DI, AND SI ARE ALL DESTROYED.
1322 ;-----+
1323

E66F  STGTST_CNT    PROC  NEAR
E670  CLD             ; SET DIR FLAG TO INCREMENT
E671  SUB  DI,DI      ; SET DI=OFFSET 0 REL TO ES REG
E672  SUB  AX,AX      ; SETUP FOR 0->FF PATTERN TEST
E674
E674  8005            ;-----+
1324  MOV  [DI],AL      ; ON FIRST BYTE
E676  8A05            ; MOV  AL,[DI]
1325  XOR  AL,AH      ; O.K.?
E678  32C4            ; INC  AH
1326  JNZ  CX          ; GO ERROR IF NOT
E67C  7500            ;-----+
1327  INC  AH
E67E  8AC4            ;-----+
1328  C2_1             ;-----+
1329  MOV  [DI],AL      ; ON FIRST BYTE
E67F  8005            ;-----+
1330  MOV  AL,[DI]
E680  75F2            ;-----+
1331  XOR  AL,AH      ; O.K.?
E682  8BD9            ; INC  AH
1332  JNZ  C2_1         ; LOOP TILL WRAP THROUGH FF
E684  D1E3            ;-----+
1333  MOV  AX,0AAAHH    ; SAVE WORD COUNT OF BLOCK TO TEST
E685  B8AAA           ;-----+
1334  MOV  BX,CX         ; CONVERT TO A BYTE COUNT
E689  0FF55FF          ;-----+
1335  MOV  DX,0FF55H    ; GET INITIAL DATA PATTERN TO WRITE
E68C  F3               ;-----+
1336  REP  STOSW        ; SETUP OTHER DATA PATTERNS TO USE
E68D  AB               ;-----+
1337  SHL  BX,1          ; FILL STORAGE LOCATIONS IN BLOCK
E68E  E61              ;-----+
1338  MOV  AX,0AAAHH    ;-----+
E690  0C30            ;-----+
1339  MOV  DX,0FF55H    ;-----+
E691  E661            ;-----+
1340  REP  STOSW        ;-----+
E692  AB               ;-----+
E693  24CF            ;-----+
1341  IN   AL,PORT_B    ; POINT TO LAST BYTE JUST WRITTEN
E694  8D61            ;-----+
1342  OR   AL,00010000B ; SET DIR FLAG TO GO BACKWARDS
E695  90               ;-----+
1343  NOP             ;-----+
E696  94               ;-----+
1344  NOP             ;-----+
E697  E661            ;-----+
1345  AND  AL,1000111B ;-----+
E698  FD               ;-----+
1346  OUT  PORT_B,AL   ;-----+
E699  4F               ;-----+
1347  C3:              ;-----+
E69A  FD               ;-----+
1348  DEC  DI          ;-----+
E69B  4C:              ;-----+
1349  STD             ;-----+
E69C  8B07            ;-----+
1350  C4:              ;-----+
1351  MOV  SI,DI        ;-----+
E69D  8BCB            ;-----+
1352  MOV  CX,BX        ;-----+
E69F  AC               ;-----+
1353  C5:              ;-----+
1354  LODSB            ;-----+
E6A0  32C4            ;-----+
1355  XOR  AL,AH        ;-----+
E6A2  7525            ;-----+
1356  JNE  C7          ;-----+
E6A4  8AC2            ;-----+
1357  MOV  AL,DL        ;-----+
E6A6  AA               ;-----+
1358  STOSB            ;-----+
E6A7  E2F6            ;-----+
1359  LOOP  C5          ;-----+
E6A8  22E4            ;-----+
1360  AND  AH,AH        ;-----+
E6A9  T416            ;-----+
1361  JZ   C6X           ;-----+
E6AA  8AE0            ;-----+
1362  MOV  AH,AL        ;-----+
E6AF  E6E2            ;-----+
1363  XCHG  DL,DL        ;-----+
E6B1  22E4            ;-----+
1364  AND  AH,AH        ;-----+
E6B3  7504            ;-----+
1365  JNZ  C6           ;-----+
E6B5  8AD4            ;-----+
1366  MOV  DL,AH        ;-----+
E6B7  EBE0            ;-----+
1367  JMP  C3          ;-----+
E6B8  0000             ;-----+
1368  C6:              ;-----+
E6B9  FC               ;-----+
1369  CLD             ;-----+
E6BA  47               ;-----+
1370  INC  DI          ;-----+
E6BB  74DE            ;-----+
1371  JZ   C4          ;-----+
E6BD  4F               ;-----+
1372  DEC  DI          ;-----+
E6BE  BA0100          ;-----+
1373  JMP  C3          ;-----+
E6C1  EBD6            ;-----+
1374  C6X:             ;-----+
E6C2  0000             ;-----+
1375  INC  DI          ;-----+
E6C3  E462            ;-----+
1376  C6X:             ;-----+
E6C5  24C0            ;-----+
1377  IN   AL,PORT_C    ;-----+
E6C7  B000            ;-----+
1378  AND  AL,0C0H      ;-----+
E6C9
E6C9  FC               ;-----+
1379  MOV  AL,000H      ;-----+
E6CA  C3               ;-----+
1380  C7:              ;-----+
E6CB  0000             ;-----+
1381  CLD             ;-----+
E6CC  52               ;-----+
1382  RET             ;-----+
E6CD  BCD4            ;-----+
1383  STGTST_CNT        ;-----+
E6CF  2688361500      ;-----+
E6D0  81F0A0CB          ;-----+
1384  ROM_ERR PROC  NEAR ;-----+
E6D0  TC00             ;-----+
1385  CMP  DX,0C800H    ;-----+
E6D0  CALL  ROM_BEEP   ;-----+
1386  POP  AX           ;-----+
E6D0  E8FD18            ;-----+
1387  CALL  PRT_SEC     ;-----+
E6D0  BE0AF990          ;-----+
1388  MOV  SI,OFFSET F3A ;-----+
E6E1  E8C512            ;-----+
1389  MOV  ES:MFG_ERR_FLAG,DH ;-----+
E6E4
E6E4  ROM_ERR END:      ;-----+
1390  CALL  E_MSG         ;-----+
E6E4  58               ;-----+
1391  CMP  DX,0C800H    ;-----+
E6E5  5A               ;-----+
1392  JZ   ROM_BEEP      ;-----+
E6E5  4000             ;-----+
1393  CALL  PRT_SEC     ;-----+
E6E5  C3               ;-----+
1394  MOV  SI,OFFSET F3A ;-----+
E6E7
E6E7  1402  ROM_ERR_BEEP: ;-----+
1395  CALL  E_MSG         ;-----+
E6E7  BA0201          ;-----+
1396  MOV  DX,0102H      ;-----+
E6E8  E8EB12            ;-----+
1397  CALL  ERR_BEEP    ;-----+
E6E9  EBF5               ;-----+
1398  JMP  SHORT ROM_ERR_END ;-----+
E6E9  1405  ROM_ERR ENDP ;-----+
1399  POP  AX           ;-----+
E6E9  1406  ROM_ERR ENDP ;-----+

```

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 ; IF THERE IS A HARDWARE ERROR CONTROL IS : 
1414 ; TRANSFERRED TO THE ROM BASIC ENTRY POINT. : 
1415 ; 
1416 ;-----  

E6F2 1417 ASSUME CS:CODE,DS:ABS0  

1418 ORG 0E6F2H  

E6F2 1419  

E6F2 FB 1420 BOOT_STRAP PROC NEAR  

1421 STI : ENABLE INTERRUPTS  

E6F3 2BC0 1422 SUB AX,AX : ESTABLISH ADDRESSING  

E6F5 8ED8 1423 MOV DS,AX  

1424  

1425 ;----- RESET THE DISK PARAMETER TABLE VECTOR  

1426 E6F7 C7067800C7EF 1427 MOV WORD PTR DISK_POINTER, OFFSET DISK_BASE  

E6FD 8C0E7A00 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: PUSH CX : IPL SYSTEM  

E705 51 1434 MOV AH,0 : SAVE RETRY COUNT  

E706 C400 1435 INT 13H : LOAD THE DISKETTE SYSTEM  

E707 CD13 1436 : DISKETTE_10  

E709 720F 1437 JC H2 : IF ERROR, TRY AGAIN  

E70B B80102 1438 MOV AX,201H : READ IN THE SINGLE SECTOR  

E70E 2BD2 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_10  

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 CD18 1452 H3: INT 1BH : GO TO RESIDENT BASIC  

1453  

1454  

1455 ;----- IPL WAS SUCCESSFUL  

1456  

E721 1457 H4:  

E721 EA007C0000 1458 JMP BOOT_LOCN  

1459 BOOT_STRAP ENDP  

1460

```

```

1461: -----INT 14-----
1462: ; RS232_I/O
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: ; 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) REGISTERS ARE 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 FROM THE LINE 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 COMM 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 = RING 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 COMM LINE 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 = DELTA RECEIVE RING DETECTOR
1513: ; BIT 1 = DELTA DATA SET READY
1514: ; BIT 0 = DELTA CLEAR TO SEND
1515:
1516: ; (DX) = PARAMETER INDICATING WHICH RS232 CARD (0,1 ALLOWED)
1517:
1518: ; DATA AREA RS232 BASE CONTAINS THE BASE ADDRESS OF THE 8250 ON THE
1519: ; CARD LOCATION 400H CONTAINS UP TO 4 RS232 ADDRESSES POSSIBLE
1520: ; DATA AREA LABEL RS232_TIM_OUT (BYTE) CONTAINS OUTER LOOP COUNT
1521: ; VALUE FOR TIMEOUT (DEFAULT=1)
1522: ; OUTPUT
1523: ; AX MODIFIED ACCORDING TO PARM 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 ; 1152 BAUD
1530: DW 168 ; 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: RS232_I/O 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 ; RS232 VALUE TO SI
1550: MOV DI,DX
1551: SHL SI,1 ; WORD OFFSET
1552: CALL DOS ; GET BASE ADDRESS
1553: MOV DX,RS232_BASE[SI] ; TEST FOR 0 BASE ADDRESS
1554: OR DX,DX ; RETURN
1555: JZ A1 ; TEST FOR (AH)=0
1556: OR AH,AH ; TEST FOR (AH)=1
1557: JZ A4 ; COMMUN INIT
1558: DEC AH ; TEST FOR (AH)=2
1559: JZ A5 ; SEND AL
1560: DEC AH ; TEST FOR (AH)=3
1561: JZ A12 ; RECEIVE INTO AL
1562: A2: ; TEST FOR (AH)=3
1563: DEC AH ; TEST FOR (AH)=0
1564: JNZ A3 ; COMMUN INIT
1565: JMP A18 ; RETURN FROM RS232
1566: A3: ; COMMUNICATION STATUS
1567: POP BX ; RETURN TO CALLER, NO ACTION
1568: POP CX
1569: POP DI
1570: POP SI
1571: POP DX
1572: POP DS
1573: IRET
1574:

```

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

```

1575 ;----- INITIALIZE THE COMMUNICATIONS PORT
1576
1577 A4:           MOV    AH,AL      ; SAVE INIT PARMS IN AH
1578 ADD    DX,3      ; POINT TO 8250 CONTROL REGISTER
1579 ADD    AL,80H    ; SET DLAB=1
1580 OUT   DX,AL
1581
1582 ;----- DETERMINE BAUD RATE DIVISOR
1583
1584
1585 MOV    DL,AH      ; GET PARMs TO DL
1586 MOV    CL,CL
1587 ROL    DL,CL
1588 AND    DX,0EH    ; ISOLATE THEM
1589 MOV    DI,OFFSET AI ; BASE OF TABLE
1590 ADD    DI,DX      ; PUT INTO INDEX REGISTER
1591 MOV    DX,RS232_BASE[SI] ; POINT TO HIGH ORDER OF DIVISOR
1592 INC    DX
1593 MOV    AL,CS:[DI]+1 ; GET HIGH ORDER OF DIVISOR
1594 OUT   DX,AL      ; SET MS OF DIV TO 0
1595 DEC    DX
1596 MOV    AL,SC:[DI]  ; GET LOW ORDER OF DIVISOR
1597 OUT   DX,AL      ; SET LOW OF DIVISOR
1598 ADD    DX,3
1599 MOV    AL,AH      ; GET PARMs BACK
1600 AND    AL,01FH    ; STRIP OFF THE BAUD BITS
1601 OUT   DX,AL
1602 DEC    DX
1603 DEC    DX
1604 MOV    AL,0
1605 OUT   DX,AL
1606 JMP    SHORT A1B  ; INTERRUPT ENABLES ALL OFF
1607
1608 ;----- SEND CHARACTER IN (AL) OVER COMMO LINE
1609
1610 A5:           PUSH   AX      ; SAVE CHAR TO SEND
1611 ADD    DX,4      ; MODEM CONTROL REGISTER
1612 MOV    AL,3      ; DTR AND RTS
1613
1614 OUT   DX,AL      ; DATA TERMINAL READY, REQUEST TO SEND
1615 INC    DX
1616 INC    DX
1617 MOV    BH,30H    ; DATA SET READY & CLEAR TO SEND
1618 CALL   WAIT_FOR_STATUS ; ARE BOTH TRUE
1619 JE    A9         ; YES, READY TO TRANSMIT CHAR
1620
1621 POP    CX
1622 MOV    AL,CL      ; RELOAD DATA BYTE
1623
1624 OR     AH,80H    ; INDICATE TIME OUT
1625 JMP    A3         ; RETURN
1626
1627 A9:           DEC    DX      ; CLEAR_TO_SEND
1628 A10:          DEC    DX      ; LINE STATUS REGISTER
1629 MOV    BH,20H    ; WAIT_SEND
1630 CALL   WAIT_FOR_STATUS ; TEST FOR TRANSMITTER READY
1631 JNZ   A7         ; RETURN WITH TIME OUT SET
1632
1633 SUB    DX,5      ; OUT_CHAR
1634 POP    CX
1635 MOV    AL,CL      ; DATA PORT
1636 OUT   DX,AL      ; RECOVER IN CX TEMPORARILY
1637 JMP    A3         ; MOVE CHAR TO AL FOR OUT, STATUS IN AH
1638
1639 ;----- RECEIVE CHARACTER FROM COMMO LINE
1640
1641 A12:          ADD    DX,4      ; OUTPUT CHARACTER
1642 MOV    AL,4      ; MODEM CONTROL REGISTER
1643
1644 OUT   DX,AL      ; DATA TERMINAL READY
1645 INC    DX
1646 INC    DX
1647
1648 MOV    BH,20H    ; MODEM STATUS REGISTER
1649 CALL   WAIT_FOR_STATUS ; WAIT_DSR
1650 JNZ   A8         ; DATA_SET READY
1651
1652 DEC    DX      ; TEST FOR DSR
1653
1654 MOV    BH,1      ; RETURN WITH ERROR
1655 CALL   WAIT_FOR_STATUS ; WAIT_DSR_END
1656 JNZ   A8         ; LINE_STATUS_REGISTER
1657
1658 AND    AH,00001110B ; WAIT_RECV
1659 MOV    DX,RS232_BASE[SI] ; GET_RECV_BUFFER_FULL
1660 IN    AL,DX      ; TEST FOR REC'D. FULL
1661
1662 JMP    A3         ; SET TIME OUT_ERROR
1663
1664 A18:          MOV    DX,RS232_BASE[SI] ; GET_CHAR
1665 ADD    DX,5      ; TEST FOR ERR CONDITIONS ON RECV CHAR
1666 IN    AL,DX      ; DATA_PORT
1667 ADD    DX,5      ; GET CHARACTER FROM LINE
1668 IN    AL,DX
1669 MOV    AH,AL
1670 INC    DX
1671 IN    AL,DX
1672 JMP    A3         ; RETURN

```

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 ;-----  

E7F2 E7F2 8A5D7C 1684 WAIT_FOR_STATUS PROC NEAR  

E7F5 2BC9 1685 MOV BL,RS232_TIM_OUT[DI] ; LOAD OUTER LOOP COUNT  

E7F7 EC 1686 WFS0: SUB CX,CX  

E7F8 8AE0 1687 WFS1: IN AL,DX ; GET STATUS  

E7FA 22C7 1688 MOV AH,AL ; MOVE TO AH  

E7FC 3AC1 1689 AND AL,BH ; ISOLATE BITS TO TEST  

E7FD 0000 1690 CMP AL,BH ; EXACTLY = TO MASK  

E800 E2F5 1691 JE WFS-END ; RETURN WITH ZERO FLAG ON  

E802 FECB 1692 LOOP WFS1 ; TRY AGAIN  

E804 75EF 1693 DEC BL  

1694 JNZ WFS0  

1695 1696  

1697 1698 OR BH,BH ; SET ZERO FLAG OFF  

E808 C3 1699 WFS-END: RET  

1700 1701 WAIT_FOR_STATUS ENDP  

1702 RS232_10 ENDP  

1703 1704 F3D DB 'ERROR. (RESUME = F1 KEY)',13,10 ; ERROR PROMPT
E809 4552524F522E20
25524552554045
203D202353122
204B455929
E823 0D
E824 0A
1705

```

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

```

I706 :---- INT 16 -----
I707 : KEYBOARD I/O
I708 : THESE ROUTINES PROVIDE KEYBOARD SUPPORT
I709 : INPUT
I710 : (AH)=0 READ THE NEXT ASCII CHARACTER STRUCK FROM THE KEYBOARD
I711 : RETURN THE RESULT IN AL, SCAN CODE IN [AH]
I712 : SET THE Z FLAG TO INDICATE IF AN ASCII CHARACTER IS
I713 : AVAILABLE TO BE READ.
I714 : (ZF)=1 -- NO CODE AVAILABLE
I715 : (ZF)=0 -- CODE IS AVAILABLE
I716 : IF IN AX=0, THE NEW CHARACTER IN THE BUFFER TO BE READ
I717 : IS IN AX. IF THE NEW CHARACTER IS NOT AVAILABLE, THE
I718 : (AH)=2 RETURN THE CURRENT SHIFT STATUS IN AL REGISTER
I719 : THE BIT SETTINGS FOR THIS CODE ARE INDICATED IN THE
I720 : THE EQUATES FOR KB_FLAG
I721 : OUTPUT
I722 : AS NOTED ABOVE, ONLY AX AND FLAGS CHANGED
I723 : ALL REGISTERS PRESERVED
I724 :----- ASSUME CS:CODE,DS:DATA
I725 E82E ORG 0EB2EH
I726 E82E FB STI ; INTERRUPTS BACK ON
I727 E82E FE PUSH DS ; SAVE CURRENT DS
I728 E82E 53 PUSH BX ; SAVE BX TEMPORARILY
I729 E831 E82512 CALL DDS
I730 E834 0AE4 OR AH,AH ; AH=0
I731 E836 T40A JZ K1 ; ASCII_READ
I732 E838 FECC DEC AH ; AH=1
I733 E83A 011E JZ K2 ; ASCII_STATUS
I734 E83C FECD DEC AH ; AH=2
I735 E83E T42B JZ K3 ; SHIFT_STATUS
I736 E840 EB2C JMP SHORT INT10_END ; EXIT
I739 ;----- READ THE KEY TO FIGURE OUT WHAT TO DO
I740 K1: ; ASCII READ
I741 E842 FB STI ; INTERRUPTS BACK ON DURING LOOP
I742 E843 90 NOP ; ALLOW AN INTERRUPT TO OCCUR
I743 E844 FA CLI ; INTERRUPTS BACK OFF
I744 E845 BB1E1A00 MOV BX,BUFFER_HEAD ; GET POINTER TO HEAD OF BUFFER
I745 E849 3B1E1C00 CMP BX,BUFFER_TAIL ; TEST END OF BUFFER
I746 E84F 74F3 JZ K1 ; LOOP BACK TO SUCCEED IN BUFFER
I747 E84F BB07 MOV AX,[BX] ; GET SCAN CODE AND ASCII CODE
I748 E851 E81D00 CALL K4 ; MOVE POINTER TO NEXT POSITION
I749 E854 891E1A00 MOV BUFFER_HEAD,BX ; STORE VALUE IN VARIABLE
I750 E858 EB14 JMP SHORT INT10_END ; RETURN
I751 ;----- ASCII STATUS
I752 E854 K2: ; ASCII STATUS
I753 E855 BB1E1A00 CLI ; INTERRUPTS OFF
I754 E856 3B1E1C00 MOV BX,BUFFER_HEAD ; GET HEAD POINTER
I755 E859 00 CMP BX,BUFFER_TAIL ; IF EQUAL (Z=1) THEN NOTHING THERE
I756 E860 90 MOV AX,[BX]
I757 E865 FB STI ; INTERRUPTS BACK ON
I758 E866 5B POP BX ; RECOVER REGISTER
I759 E867 1F POP DS ; RECOVER SEGMENT
I760 E868 CA0200 RET 2 ; THROW AWAY FLAGS
I765 ;----- SHIFT STATUS
I766 E86B B010END: K3: ; GET THE SHIFT STATUS FLAGS
I767 E86B A01700 MOV AL,KB_FLAG
I768 E86E INT10_END: ; RECOVER REGISTER
I769 E86C 5B POP BX
I770 E86F 1F POP DS
I771 E870 CF IRET ; RECOVER REGISTERS
I772 E874 KEYBOARD_10 ENDP ; RETURN TO CALLER
I775 ;----- INCREMENT A BUFFER POINTER
I776 E871 K4: ; INCREMENT A BUFFER POINTER
I777 E871 43 PROC NEAR ; MOVE TO NEXT WORD IN LIST
I778 E872 43 INC BX
I779 E873 3B1E8200 INC BX
I780 E877 7504 CMP BX,BUFFER_END ; AT END OF BUFFER?
I781 E879 BB1E8000 JNE K5 ; NO, CONTINUE
I782 E880 44 MOV BX,BUFFER_START ; YES, RESET TO BUFFER BEGINNING
I783 E881 C3 K5: ; NO, CONTINUE
I784 E882 46 RET ; YES, RESET TO BUFFER BEGINNING
I785 E883 46 ENDP ; NO, CONTINUE
I786 E884 46
I787 E885 46
I788 E886 46 ;----- TABLE OF SHIFT KEYS AND MASK VALUES
I789 E887 52 K6 LABEL BYTE
I790 E887 53 DB INS_KEY ; INSERT KEY
I791 E887 3A DB CAPS_KEY,NUM_KEY,SCROLL_KEY,ALT_KEY,CTL_KEY
I792 E888 45
I793 E889 46
I794 E88A 46
I795 E88B 46
I796 E88C 46 ;----- SHIFT_MASK_TABLE
I797 E886 80 K7 LABEL BYTE
I798 E887 40 DB INS_SHIFT ; INSERT MODE SHIFT
I799 E888 20 DB CAPS_SHIFT,NUM_SHIFT,SCROLL_SHIFT,ALT_SHIFT,CTL_SHIFT
I800 E889 10
I801 E88A 08
I802 E88B 04 ;----- SCAN CODE TABLES
I803 E88C 02 DB LEFT_SHIFT,RIGHT_SHIFT
I804 E88D 01
I805 E88E 1B K8 DB 27,-1,0,-1,-1,-1,30,-1
E88F FF
E890 00
E891 FF
E892 FF
E893 FF
E894 1E

```

LOC	OBJECT	LINE	SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82
E895 FF		1806	DB -1,-1,-1,31,-1,127,-1,17
E896 FF		1807	DB 23,5,18,20,25,21,9,15
E897 FF		1808	DB 16,27,29,10,-1,1,19
E898 FF		1809	DB 4,6,7,8,10,11,12,-1,-1
E899 FF		1810	DB -1,-1,28,26,24,3,22,2
E8A0 FF		1811	DB 14,13,-1,-1,-1,-1,-1,-1
E8A1 05		1812	DB * *, -1
E8A2 12		1813	I----- CTL_TABLE_SCAN
E8A3 14		1814	K9 LABEL BYTE
E8A4 19		1815	DB 94,95,96,97,98,99,100,101
E8A5 15			
E8A6 09			
E8A7 0F			
E8A8 10			
E8A9 1B			
E8A9 1D			
E8A9 0A			
E8AA FF			
E8AB 01			
E8AC 13			
E8AD 04			
E8AE 06			
E8AF 07			
E8B0 08			
E8B1 0A			
E8B2 0B			
E8B3 0C			
E8B4 FF			
E8B5 FF			
E8B6 FF			
E8B7 FF			
E8B8 1C			
E8B9 1A			
E8BA 18			
E8BB 03			
E8BC 16			
E8BD 02			
E8BE 0E			
E8BF 00			
E8C0 FF			
E8C1 FF			
E8C2 FF			
E8C3 FF			
E8C4 FF			
E8C5 FF			
E8C6 20			
E8C7 FF			
E8C8 5E		1816	DB 102,103,-1,-1,119,-1,132,-1
E8C9 5F		1817	DB 115,-1,116,-1,117,-1,118,-1
E8CA 00		1818	DB -1
E8CB 61		1819	I----- LC_TABLE DB
E8CC 62		1820	K10 LABEL BYTE
E8CD 63		1821	DB 01BH,'1234567890-*',08H,09H
E8CE 64			
E8CF 65			
E8D0 66			
E8D1 17			
E8D2 FF			
E8D3 FF			
E8D4 77			
E8D5 FF			
E8D6 44			
E8D7 FF			
E8D8 73			
E8D9 FF			
E8DA 74			
E8DB FF			
E8DC 55			
E8DD FF			
E8DE 76			
E8DF FF			
E8E0 FF			
E8E1 1B		1822	DB 'qwertyuiop[],0DH,-1,'asdfghjkl;',027H 696FT05BSD
E8E2 3132334353637 3839302D3D			
E8EE 08			
E8EF 09			
E8F0 776572747975			
E8F1 0D			
E8F2 FF			
E8F3 6173646667686A 666C3B			
E8F8 27			
E8F9 10		1823	DB 60H,-1,5CH,'zxcvbnm,-/,,-1,*-*,-1,*'
E8FA FF			
E8FB 5C			
E8OC TAT8637662E6D 2C2E2F			
E916 FF			
E917 1A			
E918 FF			
E919 20			
E91A FF		1824	DB -1
E91B 1B		1825	I----- UC_TABLE DB
E91C 21402324		1826	K11 LABEL BYTE
E920 25		1827	DB 27,'!*#*&,37,05EH,'*&()_-*,08H,0
E921 5E			
E922 262A28295F2B			
E928 08			
E929 10			
E92A 51574552545955 494F507B7D		1828	DB 'QWERTYUIOP{},0DH,-1,'ASDFGHJKL:**'
E936 0D			
E937 FF			
E938 4153444647484A			

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

```

4B4C3A22          1829      DB      07EH,-1,|ZXCVBNM<?,-1,0,-1,* *,-
E943 7E
E944 FF
E945 TC5A584356424E
E946 3E3C3E3F
E950 FF
E951 00
E952 FF
E953 20
E954 FF
E955 1830 ;---- UC TABLE SCAN
E956 54          1831 K12   LABEL  BYTE
E957 55          1832           DB      84,85,86,87,88,89,90
E958 56
E959 57
E95A 58
E95B 59
E95C 5A
E95D 5B
E95E 5C
E95F 5D
E95F 1833      DB      91,92,93
E95G 64
E95H 65
E95I 66
E95J 67
E95K 68
E95L 69
E95M 6A
E95N 6B
E95O 6C
E95P 6D
E95Q 6E
E95R 6F
E95S 6G
E95T 70
E95U 71
E95V 1834 ;---- ALT TABLE SCAN
E95W 64          1835 K13   LABEL  BYTE
E95X 65          1836           DB      104,105,106,107,108
E95Y 66
E95Z 67
E960 68
E961 69
E962 6A
E963 6B
E964 6C
E965 6D
E966 6E
E967 6F
E968 6G
E969 1838 ;---- NUM STATE TABLE
E96A 3738392D343536 1839 K14   LABEL  BYTE
E96B 2B313233302E 1840           DB      *789-456+1230.+
E96C 1841 ;---- BASE CASE TABLE
E96D 47          1842 K15   LABEL  BYTE
E96E 48          1843           DB      71,72,73,-1,75,-1,77
E96F 49
E970 4A
E971 4B
E972 4C
E973 4D
E974 4E
E975 4F
E976 4G
E977 4H
E978 4I
E979 4J
E97A 4B
E97B 4C
E97C 4D
E97D 4E
E97E 4F
E97F 4G
E980 51
E981 52
E982 53
E983 1845 ;---- KEYBOARD INTERRUPT ROUTINE
E984 1846
E985 1847 ORG    0E987H
E986 1848 KB_INT PROC   FAR
E987 1849 STI
E988 1850 ORG    0E987H
E989 1851 PUSH   AX
E98A 1852 PUSH   BX
E98B 1853 PUSH   CX
E98C 1854 PUSH   DX
E98D 1855 PUSH   SI
E98E 1856 PUSH   DI
E98F 1857 PUSH   DS
E990 1858 PUSH   ES
E991 1859 CALL   DDS
E992 1860 IN     AL,_KB_DATA
E993 1861 PUSH   AX
E994 1862 MOV    AH,AL
E995 1863 IN     AL,_KB_CTL
E996 1864 MOV    AH,AL
E997 1865 OR    AL,0DH
E998 1866 OUT   KB_CTL,AL
E999 1867 XCHG   AH,AL
E99A 1868 OUT   KB_CTL,AL
E99B 1869 POP    AX_
E99C 1870 MOV    AH,AL
E99D 1871
E99E 1872 ;---- TEST FOR OVERRUN SCAN CODE FROM KEYBOARD
E99F 1873
E9A0 1874 CMP    AL,0FFH
E9A1 1875 JNZ    K16
E9A2 1876 JMP    K62
E9A3 1877
E9A4 1878 ;---- TEST FOR SHIFT KEYS
E9A5 1879
E9A6 1880 K16:
E9A7 1881 AND    AL,07FH
E9A8 1882 PUSH   CS
E9A9 1883 POP    ES
E9A0 1884 MOV    DI,OFFSET K6
E9A1 1885 MOV    CX,K6L
E9A2 1886 REPNE SCASB
E9A3 1887 MOV    AL,AH
E9A4 1888 JE    K17
E9A5 1889 JMP    K25
E9A6 1890
E9A7 1891 ;---- SHIFT KEY FOUND
E9A8 1892
E9A9 1893 K17: SUB    DI,OFFSET K6+1
E9A0 1894 MOV    AH,SKT[DI]
E9A1 1895 TEST   AL,0BH
E9A2 1896 JNZ    K23
E9A3 1897
E9A4 1898 ;---- SHIFT MAKE FOUND, DETERMINE SET OR TOGGLE
E9A5 1899
E9A6 1900 CMP    AH,SCROLL_SHIFT
E9A7 1901 JAE    K18
E9A8 1902
E9A9 1903 ;---- PLAIN SHIFT KEY, SET SHIFT ON
E9A0 1904
E9A1 1905 OR    KB_FLAG,AH
E9A2 1906 JMP    K26

```

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

```

1907 ;----- TOGGLED SHIFT KEY, TEST FOR 1ST MAKE OR NOT
1908
1909 K18: 1910 TEST KB_FLAG, CTL_SHIFT : SHIFT-TOGGLE
E9D9 1911 JNZ K25 : CHECK CTL SHIFT STATE
E9D9 F606170004 1912 CMP AL, INS_KEY : JUMP IF CTL STATE
E9D9 E5C5 1913 JNZ K22 : CHECK FOR INSERT KEY
E9D9 E5C2 1914 CMP AL, NUM_KEY : JUMP IF NOT INSERT KEY
E9D9 F606170008 1915 TEST KB_FLAG, ALT_SHIFT : CHECK FOR ALTERNATE SHIFT
E9D9 E55A 1916 JNZ K26 : JUMP IF ALTERNATE SHIFT
E9D9 F606170020 1917 K19: TEST KB_FLAG, NUM_STATE : CHECK FOR BASE STATE
E9F0 75D0 1918 JNZ K27 : JUMP IF NUM LOCK IS ON
E9F2 F606170003 1919 TEST KB_FLAG, LEFT_SHIFT+RIGHT_SHIFT : LEFT SHIFT
E9F7 74D0 1920 JZ K22 : JUMP IF BASE STATE
1921
1922 K20: 1923 MOV AX, 5230H : NUMERIC ZERO, NOT INSERT KEY
E9F9 B83052 1924 JMP K57 : PUT OUT AN ASCII ZERO
E9FC E9D601 1925 K21: 1926 TEST KB_FLAG, LEFT_SHIFT+RIGHT_SHIFT : BUFFER FILL
E9FF E9D601 1927 JZ K20 : MIGHT BE NUMERIC
E9FF F606170003 1928 1929 ;----- TEST FOR 1ST MAKE OF INSERT KEY
EA04 74F3 1930 TEST AH,KB_FLAG_I : SHIFT TOGGLE KEY HIT; PROCESS IT
EA06 84261800 1931 JNZ K26 : IS KEY ALREADY DEPRESSED
EA06 75D4 1932 OR KB_FLAG_I, AH : JUMP IF KEY ALREADY DEPRESSED
EA0C 08261800 1933 XOR KB_FLAG, AH : INDICATE THAT THE KEY IS DEPRESSED
EA10 30017000 1934 CMP AL,INS_KEY : DO THIS IN THE SHIFT STATE
EA14 E5C5 1935 JNE K24 : TEST FOR 1ST MAKE OF INSERT KEY
EA16 7541 1936 MOV AX,INS_KEY*256 : JUMP IF NOT INSERT KEY
EA1B B80052 1937 JMP K57 : SET SCAN CODE INTO AH, 0 INTO AL
EA1B E9B701 1938 1939 ;----- BREAK SHIFT FOUND
1940
1941 K23: 1942 CMP AH,SCROLL_SHIFT : BREAK-SHIFT-FOUND
EA1E 80FC10 1943 JAE K24 : IS THIS A TOGGLE KEY
EA21 731A 1944 NOT AH : YES, HANDLE BREAK TOGGLE
EA23 F6D4 1945 AND KB_FLAG,AH : INVERT MASK
EA25 20261700 1946 CMP AL,ALT_KEY+80H : TURN OFF SHIFT BIT
EA29 3C8B 1947 JNE K26 : IS THIS ALTERNATE SHIFT RELEASE
EA2B 752C 1948 1949 ;----- ALTERNATE SHIFT KEY RELEASED, GET THE VALUE INTO BUFFER
1950
EA2D A01900 1951 MOV AL,ALT_INPUT : NO-SHIFT-FOUND
EA30 B400 1952 MOV AH,0 : TEST FOR BREAK KEY
EA32 88261900 1953 MOV ALT_INPUT,AH : NOTHING FOR BREAK CHARS FROM HERE ON
EA36 3C00 1954 CMP AL,0 : ARE WE IN HOLD STATE
EA38 741F 1955 JE K26 : WAS THE INPUT=0
EA3A E9A101 1956 JMP K58 : INVERT MASK
EA3D 1957 K24: 1958 NOT AH : IT WASN'T, SO PUT IN BUFFER
EA3D F6D4 1959 AND KB_FLAG_I,AH : INVERT MASK
EA3F 20261800 1960 JMP SHORT K26 : INDICATE NO LONGER DEPRESSED
EA43 EB14 1961
1962 ;----- TEST FOR HOLD STATE
1963
EA45 1964 K25: 1965 CMP AL,80H : CANT END HOLD ON NUM LOCK
EA47 T310 1966 JAE K26 : TURN OFF THE HOLD STATE BIT
EA47 082618008 1967 TEST KB_FLAG_I,HOLD_STATE : INTERRUPT-RETURN
EA4E 741F 1968 JZ K28 : TURN OFF INTERRUPTS
EA50 3C45 1969 CMP AL,NUM_KEY : END OF INTERRUPT COMMAND
EA52 T405 1970 JE K26 : SEND COMMAND TO INT CONTROL PORT
EA54 80261800F7 1971 AND KB_FLAG_I,NOT HOLD_STATE : INTERRUPT-RETURN-NO-E0I
EA55 1972 K26: 1973 CLI : INTERRUPT-RETURN
EA55 FA 1974 MOV AL,E0I : TURN OFF INTERRUPTS
EA55 B020 1975 OUT 020H,AL : SEND COMMAND TO INT CONTROL PORT
EA55 E620 1976 K27: 1977 POP ES : INTERRUPT-RETURN-NO-E0I
EA55 07 1978 POP DS : END OF INTERRUPT
EA55 1F 1979 POP DI : TURN OFF INTERRUPT
EA55 5F 1980 POP SI : SEND COMMAND TO INT CONTROL PORT
EA55 SE 1981 POP DX : INTERRUPT-RETURN-NO-E0I
EA62 5A 1982 POP CX : WITH FLAG CHANGE
EA63 59 1983 POP BX
EA65 5B 1984 POP AX : RESTORE STATE
EA66 CF 1985 IRET : RETURN, INTERRUPTS BACK ON
1986
1987 ;----- NOT IN HOLD STATE, TEST FOR SPECIAL CHARS
1988
1989 K28: 1990 TEST KB_FLAG,ALT_SHIFT : NO-HOLD-STATE
EA67 F606170008 1991 JNZ K29 : ARE WE IN ALTERNATE SHIFT
EA6C 7503 1992 JMP K38 : JUMP IF ALTERNATE SHIFT
EA6E E99100 1993 1994 ;----- TEST FOR RESET KEY SEQUENCE (CTL ALT DEL)
1995
EA71 1996 K29: 1997 TEST KB_FLAG,CTL_SHIFT : TEST-RESET
EA71 F606170004 1998 JZ K37 : ARE WE IN CONTROL SHIFT ALSO
EA76 T433 1999 CMP AL,DEL_KEY : NO RESET
EA78 3C53 2000 JNE K31 : SHIFT STATE IS THERE, TEST KEY
EA7A 752F 2001 2002 ;----- CTL-ALT-DEL HAS BEEN FOUND, DO I/O CLEANUP
2003
EA7C C70672003412 2004 MOV RESET_FLAG, 1234H : SET FLAG FOR RESET FUNCTION
EA82 EA5BE000F0 2005 JMP RESET- : JUMP TO POWER ON DIAGNOSTICS
2006
2007
EA87 2008 ;----- ALT-INPUT-TABLE
EA87 52 2009 K30 LABEL BYTE : 10 NUMBERS ON KEYPAD
EA88 4F 2010 DB 82,79,80,81,75,76,77
EA89 50
EA8B 51
EA8B 4B
EA8C 4C
EA8D 4D
EA8E 47
EA8F 48
EA90 49
EA91 10 2011 DB 71,72,73 : 10 NUMBERS ON KEYPAD
EA92 11 2012 ;----- SUPER-SHIFT-TABLE
2013 DB 16,17,18,19,20,21,22,23 : A-Z TYPEWRITER CHARS

```

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

EA93 I2
EA94 I3
EA95 I4
EA96 I5
EA97 I6
EA98 I7
EA99 I8 2014 DB 24,25,30,31,32,33,34,35
EA9A I9
EA9B IE
EA9C IF
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
EAA9 31 2016 DB 49,50
EAA0 32
2017
2018 ;----- IN ALTERNATE SHIFT, RESET NOT FOUND
2019
EAB0 3C39 2020 K31: ; NO-RESET
EAB1 7655 2021 CMP AL,57 ; TEST FOR SPACE KEY
EAB2 7655 2022 JNE K32 ; NOT THERE
EAB3 B020 2023 MOV AL,' ' ; SET SPACE CHAR
EAB1 E92101 2024 JMP K57 ; BUFFER_FILL
2025
2026 ;----- LOOK FOR KEY PAD ENTRY
2027
2028 K32: ; ALT-KEY-PAD
EAB4 BF87EA 2029 MOV DI,OFFSET K30 ; ALT-INPUT_TABLE
EAB7 B90A00 2030 MOV CX,10 ; LOOK FOR ENTRY USING KEYPAD
EABA F2 2031 REPNE SCASB ; LOOK FOR MATCH
EABB AE
EABC 7512
EABD 1EF88EA 2032 JNE K33 ; NO_ALT_KEYPAD
EAC2 A01900 2033 MOV DI,OFFSET K30+1 ; DI NOW HAS ENTRY VALUE
EAC5 B40A 2034 MOV AL,ALT_INPUT ; GET THE CURRENT BYTE
EACT F6E4 2035 MOH AH,10 ; MULTIPLY BY 10
EAC9 03C7 2036 MUL AH
EACB A21900 2037 ADD AX,DI ; ADD IN THE LATEST ENTRY
EACE EB89 2038 MOV ALT_INPUT,AL ; STORE IT AWAY
2039 JMP K26 ; THROW AWAY THAT KEYSTROKE
2040
2041 ;----- LOOK FOR SUPERSHIFT ENTRY
2042
EAD0 C606190000 2043 K33: ; NO-ALT-KEYPAD
EAD0 B90A00 2044 MOV ALT_INPUT,0 ; ZERO ANY PREVIOUS ENTRY INTO INPUT
EAD0 B90A00 2045 MOV CX,26 ; DI_ES ALREADY POINTING
EAD9 AE 2046 REPNE SCASB ; 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
2052 ;----- LOOK FOR TOP ROW OF ALTERNATE SHIFT
2053 K34: ; ALT-TOP-ROW
EAE1 3C02 2054 CMP AL,2 ; KEY WITH '1' ON IT
EAE3 720C 2055 JB K35 ; NOT ONE OF INTERESTING KEYS
EAE5 3C01 2056 CMP AL,14 ; IS IT IN THE REGION
EAE6 3C08 2057 JAE K35 ; ALT-FUNCTION
EAE9 80C476 2058 ADD AL,118 ; CONVERT PSEUDO SCAN CODE TO RANGE
EAC2 B000 2059 MOV AL,0 ; INDICATE AS SUCH
EAEF E9E400 2060 JMP K57 ; BUFFER_FILL
2061
2062 ;----- TRANSLATE ALTERNATE SHIFT PSEUDO SCAN CODES
2063
EAF1 3C3B 2064 K35: ; ALT-FUNCTION
EAF3 7303 2065 CMP AL,59 ; TEST FOR IN TABLE
EAF5 3C47 2066 JAE K37 ; ALT-CONTINUE
EAF5 E961FF 2067 K36: ; CLOSE-RETURN
EAF6 73F9 2068 JMP K26 ; IGNORE THE KEY
EAF8 3C47 2069 K37: ; ALT-CONTINUE
EAF9 73F9 2070 CMP AL,T1 ; IN THE SAME REGION
EAFB BB5FEE 2071 JAE K36 ; IF SO, IGNORE
EAFF E91B01 2072 MOV BX,OFFSET K13 ; ALT SHIFT PSEUDO SCAN TABLE
2073 JMP K63 ; TRANSLATE THAT
2074
2075 ;----- NOT IN ALTERNATE SHIFT
2076
EB02 7303 2077 K38: ; NOT-ALT-SHIFT
EB02 F606170004 2078 TEST KB_FLAG,CTL_SHIFT ; ARE WE IN CONTROL SHIFT
EB07 7458 2079 JZ K44 ; NOT-CTL-SHIFT
2080
2082 ;----- CONTROL SHIFT, TEST SPECIAL CHARACTERS
2082 ;----- 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 8B1E0000 2086 MOV BX,BUFFER_START ; RESET BUFFER TO EMPTY
EB0E 891E1A00 2087 INT BX ; BREAK_INTERRUPT_VECTOR
EB15 891E1C00 2088 MOV BUFFER_HEAD,BX
EB19 C606710080 2089 MOV BIOS_BREAK,80H ; TURN ON BIOS BREAK BIT
EB1E CD1B 2090 INT 1BH ; BREAK_INTERRUPT_VECTOR
EB20 28C0 2091 SUB AX,AX ; PUT OUT DUMMY CHARACTER
EB22 E9B000 2092 JMP K57 ; BUFFER_FILL
EB23 3C45 2093 CMP AL,PAUSE ; NO-BREAK
EB25 3C45 2094 CMP AL,NUM_KEY ; LONG FOR PAUSE KEY
EB27 7521 2095 JNE K41 ; NO-PAUSE
EB29 800E180008 2096 OR KB_FLAG,1,HOLD_STATE ; TURN ON THE HOLD FLAG
EB2E B020 2097 MOV AL,E01 ; END OF INTERRUPT TO CONTROL PORT
EB30 E620 2098 OUT 020H,AL ; ALLOW FURTHER KEYSTROKE INTS
2099
2100 ;----- 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 B0D803 2104 MOV DX,03DBH ; PORT FOR COLOR CARD
EB3D 803E490500 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_!,HOLD_STATE ; LOOP UNTIL FLAG TURNED OFF
EB45 75F9      2109 JNZ K40
EB47 E914FF      2110 JMP K27 ; INTERRUPT_RETURN_NO_EOI
EB4A          2111 K41:           ; NO-PAUSE
2112
2113 ;----- TEST SPECIAL CASE KEY 55
2114
EB4A 3C37      2115 CMP AL,,55 ; NOT-KEY-55
EB4C T506      2116 JNE K42 ; START/STOP PRINTING SWITCH
EB4E BB0072      2117 MOV AX,114*256 ; BUFFER_FILL
EB51 E98100      2118 JMP K57
2119
2120 ;----- SET UP TO TRANSLATE CONTROL SHIFT
2121
EB54          2122 K42:           ; NOT-KEY-55
EB54 BB8EE8      2123 MOV BX,OFFSET K8 ; SET UP TO TRANSLATE CTL
EB57 3C3B      2124 CMP AL,,59 ; IS IT IN TABLE?
2125
EB59 7276      2126 JB K56 ; GO-TRANSLATE
EB5B          2127 K43:           ; CTL-TRANSLATE
EB5B BBC8E8      2128 MOV BX,OFFSET K9 ; CTL_TABLE-TRANSLATE
EB5E E9BC00      2129 JMP K63 ; CTL_TABLE_SCAN
2130
2131 ;----- NOT IN CONTROL SHIFT
2132
EB61          2133 K44:           ; NOT-CTL-SHIFT
EB61 3C47      2134 CMP AL,,71 ; TEST FOR KEYPAD REGION
EB63 T32C      2135 JAE K48 ; HANDLE KEYPAD REGION
EB65 F606170003 2136 TEST KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT ; TEST FOR SHIFT STATE
EB6A 745A      2137 JZ K54
2138
2139 ;----- UPPER CASE, HANDLE SPECIAL CASES
2140
EB6C 3C0F      2141 CMP AL,,15 ; BACK TAB KEY
EB6E T505      2142 JNE K45 ; NOT-BACK-TAB
EB71 BB000F      2143 MOV AX,115*256 ; SCAN CODE
EB73 E860      2144 JMP SHORT K57 ; BUFFER_FILL
EB75          2145 K45:           ; NOT-BACK-TAB
EB75 3C37      2146 CMP AL,,55 ; PRINT SCREEN KEY
EB77 T509      2147 JNE K46 ; NOT-PRINT-SCREEN
2148
2149 ;----- ISSUE INTERRUPT TO INDICATE PRINT SCREEN FUNCTION
2150
EB79 B020      2151 MOV AL,E01 ; END OF CURRENT INTERRUPT
EB7B E620      2152 OUT 020H,AL ; SO FURTHER THINGS CAN HAPPEN
EB7D CD05      2153 INT 5H ; ISSUE PRINT SCREEN INTERRUPT
EB7F E9DCFE      2154 JMP K27 ; GO BACK WITHOUT EOI OCCURRING
EB82          2155 K46:           ; NOT-UPPER-SCREEN
EB82 3C3B      2156 CMP AL,,59 ; FUNCTION KEY
EB84 T206      2157 JB K47 ; NOT-UPPER-FUNCTION
EB86 BB5E59      2158 MOV BX,OFFSET K12 ; UPPER CASE PSEUDO SCAN CODES
EB89 E99100      2159 JMP K63 ; TRANSLATE SCAN
EB8C          2160 K47:           ; NOT-UPPER-FUNCTION
EB8C BB1BE9      2161 MOV BX,OFFSET K11 ; POINT TO UPPER CASE TABLE
EB8F EB40      2162 JMP SHORT K56 ; OK, TRANSLATE THE CHAR
2163
2164 ;----- 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 T520      2168 JNZ K52 ; TEST FOR SURE
EB98 F606170003 2169 TEST KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT ; ARE WE IN SHIFT STATE
EB9D T520      2170 JNZ K53 ; IF SHIFTED, REALLY NUM STATE
2171
2172 ;----- BASE CASE FOR KEYPAD
2173
EB9F          2174 K49:           ; BASE-CASE
EB9F 3C4A      2175 CMP AL,,74 ; SPECIAL CASE FOR A COUPLE OF KEYS
EBA1 T40B      2176 JE K50 ; MINUS
EBA3 3C4E      2177 CMP AL,,78
EBA5 T40C      2178 JE K51
EBAT 2C47      2179 SUB AL,,71 ; CONVERT ORIGIN
EBA9 BB76E9      2180 MOV BX,OFFSET K15 ; BASE CASE TABLE
EBB1 EB71      2181 JMP SHORT K64 ; CONVERT TO PSEUDO SCAN
EBAE          2182 K50:           ; MINUS
EBAE BB2D4A      2183 MOV AX,114*256+'-' ; BUFFER_FILL
EBB1 EB22      2184 JMP SHORT K57
EBB3          2185 K51:           ; PLUS
EBB3 BB2B4E      2186 MOV AX,118*256+'+' ; BUFFER_FILL
EBB6 EB1D      2187 JMP SHORT K57
2188
2189 ;----- MIGHT BE NUM LOCK, TEST SHIFT STATUS
2190
EBB8          2191 K52:           ; ALMOST-NUM-STATE
EBB8 F606170003 2192 TEST KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT ; ADDED TEMP OUT OF NUM STATE
EBB9 T5E0      2193 JNZ K49 ; REALLY NUM STATE
EBBF 2C46      2194 K53:           ; CONVERT ORIGIN
EBBC BB69E9      2195 SUB AL,,70 ; NUM STATE TABLE
EBC4 EB0B      2196 MOV BX,OFFSET K14 ; TRANSLATE_CHAR
2197 JMP SHORT K56
2198
2199 ;----- PLAIN OLD LOWER CASE
2200
EBCE 3C3B      2201 K54:           ; NOT-SHIFT
EBCE 7204      2202 CMP AL,,59 ; TEST FOR FUNCTION KEYS
EBCA B000      2203 JB K55 ; NOT-LOWER-FUNCTION
EBCC EB07      2204 MOV AL,,0 ; SCAN CODE IN AH ALREADY
EBCE EB08      2205 JMP SHORT K57 ; BUFFER_FILL
EBCE BBE1E8      2206 K55:           ; NOT-LOWER-FUNCTION
2207 MOV BX,OFFSET K10 ; LC TABLE
2208
2209 ;----- TRANSLATE THE CHARACTER
2210
EBD1          2211 K56:           ; TRANSLATE-CHAR
EBD1 FECB      2212 DEC AL ; CONVERT ORIGIN
EBD3 2E01      2213 XLAT CS:K11 ; CONVERT THE SCAN CODE TO ASCII
2214
2215 ;----- PUT CHARACTER INTO BUFFER
2216
EBD5 3CFF      2217 K57:           ; BUFFER-FILL
EBD7 T41F      2218 CMP AL,-1 ; IS THIS AN IGNORE CHAR
EBD9 80FCFF      2219 JE K59 ; YES, DO NOTHING WITH IT
EBDC T41A      2220 CMP AH,-1 ; LOOK FOR -1 PSEUDO SCAN
EBDC T41A      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
2224 K58: TEST KB_FLAG,CAPS_STATE ; ARE WE IN CAPS LOCK STATE
2225 JZ K61 ; SKIP IF NOT
2226 ;----- IN CAPS LOCK STATE
2227 TEST KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT ; TEST FOR SHIFT STATE
2228 JZ K60 ; IF NOT SHIFT, CONVERT LOWER TO UPPER
2229 ;----- CONVERT ANY UPPER CASE TO LOWER CASE
2230 TEST KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT ; TEST FOR SHIFT STATE
2231 JZ K60 ; IF NOT SHIFT, CONVERT LOWER TO UPPER
2232 ADD AL,'A'-'a' ; NEAR-INTERRUPT-RETURN
2233 JMP SHORT K61 ; INTERRUPT_RETURN
2234 ;----- CONVERT ANY LOWER CASE TO UPPER CASE
2235 CMP AL,'A' ; FIND OUT IF ALPHABETIC
2236 JB K61 ; NOT_CAPS_STATE
2237 CMP AL,'Z' ; NOT_CAPS_STATE
2238 JA K61 ; NOT_CAPS_STATE
2239 ADD AL,'a'-'A' ; CONVERT TO LOWER CASE
2240 JMP SHORT K61 ; NOT_CAPS_STATE
2241 ;----- NEAR-INTERRUPT-RETURN
2242 K59: JMP K26 ; INTERRUPT_RETURN
2243 ;----- CONVERT ANY LOWER CASE TO UPPER CASE
2244 ADD AL,'A'-'a' ; NEAR-INTERRUPT-RETURN
2245 JMP SHORT K61 ; INTERRUPT_RETURN
2246 ;----- LOWER-TO-UPPER
2247 K60: CMP AL,'a' ; FIND OUT IF ALPHABETIC
2248 JB K61 ; NOT_CAPS_STATE
2249 CMP AL,'z' ; NOT_CAPS_STATE
2250 JA K61 ; NOT_CAPS_STATE
2251 ADD AL,'A'-'a' ; CONVERT TO UPPER CASE
2252 SUB AL,'a'-'A' ; NOT_CAPS_STATE
2253 K61: MOV BX,BUFFER_TAIL ; GET THE END POINTER TO THE BUFFER
2254 MOV SI,BX ; SEND THE VALUE
2255 CALL K49 ; ADVANCE THE TAIL
2256 ;----- HAS THE BUFFER WRAPPED AROUND
2257 CMP BX,BUFFER_HEAD ; BUFFER FULL_BEEP
2258 JE K62 ; STORE THE VALUE
2259 MOV [SI],AX ; MOVE THE POINTER UP
2260 MOV BUFFER_TAIL,BX ; INTERRUPT_RETURN
2261 JMP K26
2262 ;----- TRANSLATE SCAN FOR PSEUDO SCAN CODES
2263 ;----- TRANSLATE-SCAN
2264 K63: SUB AL,59 ; CONVERT ORIGIN TO FUNCTION KEYS
2265 ;----- TRANSLATE-SCAN-ORGD
2266 K64: SUB AL,59 ; DO THE SAME
2267 ;----- PUT VALUE INTO AH
2268 XLAT CS:K9 ; ADVANCE THE TAIL
2269 MOV AH,AL ; HAS THE BUFFER WRAPPED AROUND
2270 MOV AL,0 ; BUFFER_FULL_BEEP
2271 JMP K57 ; STORE THE VALUE
2272 ;----- PUT IT INTO THE BUFFER
2273 KB_INT ENDP
2274 ;----- BUFFER IS FULL, SOUND THE BEEPER
2275 ;----- BUFFER-FULL-BEEP
2276 K62: MOV AL,E01 ; END OF INTERRUPT COMMAND
2277 OUT 20H,AL ; SEND COMMAND TO INT CONTROL PORT
2278 MOV BX,080H ; NUMBER OF CYCLES FOR 1/12 SECOND TONE
2279 IN AL,KB_CTL ; GET CONTROL INFORMATION
2280 PUSH AX ; SAVE
2281 ;----- BEEP-CYCLE
2282 XOR AL,0FCH ; TURN OFF TIMER GATE AND SPEAKER DATA
2283 AND AL,0FCH ; OUTPUT TO CONTROL
2284 OUT KB_CTL,AL ; HALF CYCLE TIME FOR TONE
2285 MOV CX,4BH ;----- SPEAKER OFF
2286 K65: LOOP K66 ; TURN ON SPEAKER BIT
2287 OR AL,2 ; OUTPUT TO CONTROL
2288 OUT KB_CTL,AL ; SET UP COUNT
2289 MOV CX,4BH ;----- ANOTHER HALF CYCLE
2290 K66: LOOP K67 ; TOTAL TIME COUNT
2291 OUT KB_CTL,AL ; DO ANOTHER CYCLE
2292 K67: OUT KB_CTL,AL ; RECOVER CONTROL
2293 JMP K27 ; OUTPUT THE CONTROL
2294 ;----- DO ANOTHER CYCLE
2295 DEC BX ; RECOVER CONTROL
2296 JNZ K65 ; OUTPUT THE CONTROL
2297 POP AX ;----- KEYBOARD ERROR
2298 OUT KB_CTL,AL ; DISKETTE ERROR
2299 JMP K27
2300 F1 DB '301',13,10 ; KEYBOARD ERROR
2301 F3 DB '601',13,10 ; DISKETTE ERROR

```

```

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 :          0=NOT USED FOR FORMAT)
2319 : (AL) - NUMBER OF SECTORS (1 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 FORMAT THE DESIRED TRACK
2327 : FOR THE FORMAT OPERATION, THE BUFFER POINTER (ES,BX)
2328 : MUST POINT TO THE COLLECTION OF DESIRED ADDRESS FIELDS
2329 : FOR THE TRACK. EACH FIELD IS COMPOSED OF 4 BYTES,
2330 : (LH,HL,HM,ML). L = LENGTH OF SECTOR, H = SECTOR NUMBER,
2331 : R = SECTOR NUMBER, N = NUMBER OF BYTES PER SECTOR
2332 : (100=128, 01=256, 02=512, 03=1024). THERE MUST BE ONE
2333 : ENTRY FOR EVERY SECTOR ON THE TRACK. THIS INFORMATION
2334 : IS USED TO FIND THE REQUESTED SECTOR DURING READ/WRITE
2335 : ACCESS.
2336 :
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, THEN DISKETTE CODE WILL
2351 : APPROPRIATE ACTION 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 :-----ASSUME DS:CODE,DS:DATA,ES:DATA
2357 :-----ORG 0ECS59H
EC59
EC59 FB
2359 DISKETTE_10 PROC FAR
2360 ST1 ; INTERRUPTS BACK ON
EC5A 53
2361 PUSH BX ; SAVE ADDRESS
EC5B 51
2362 PUSH CX
EC5C 5E
2363 PUSH DS ; SAVE SEGMENT REGISTER VALUE
EC5D 56
2364 PUSH SI ; SAVE ALL REGISTERS DURING OPERATION
EC5E 57
2365 PUSH DI
EC5F 55
2366 PUSH BP
EC60 52
2367 PUSH DX
EC61 8BE8
2368 MOV BP,SP ; SET UP POINTER TO HEAD PARM
EC62 89E8
2369 CALL DS ; CALL THE REST TO ENSURE DS RESTORED
EC63 5000
2370 CALL J1 ; GET THE MOTOR_WAIT PARAMETER
EC64 E81C00
2371 MOV BX,4 ; CALL THE TIMER COUNT FOR THE MOTOR
EC65 BB0400
2372 CALL GET_PARM ; GET STATUS OF OPERATION
EC66 E8FD01
2373 MOV MOTOR_COUNT,AH ; SET THE TIMER COUNT FOR THE MOTOR
EC67 88264000
2374 MOV AH,DISKETTE_STATUS ; GET STATUS OF OPERATION
EC73 BA264100
2375 CMP AH,1 ; SET THE READY TO ACT TO INDICATE
EC77 00FC01
2376 CMC ; SUCCESS OR FAILURE
EC7A F5
2377 POP DX ; RESTORE ALL REGISTERS
EC7B 5A
2378 POP BP
EC7C 5D
2379 POP DI
EC7E 5E
2380 POP SI
EC7F 5F
2381 POP DS
EC80 59
2382 POP CX
EC81 5B
2383 POP BX ; RECOVER ADDRESS
EC82 CA0200
2384 RET 2 ; THROW AWAY SAVED FLAGS
2385 DISKETTE_10 ENDP

EC85
2386 J1 PROC NEAR
EC86 8AF0
2387 J1 PROC NEAR
2388 MOV DH,AL ; SAVE # SECTORS IN DH
EC87 80263F007F
2389 AND MOTOR_STATUS,07FH ; INDICATE A READ OPERATION
EC8C 0AE4
2390 OR AH,AH ; AH=0
EC8E T427
2391 JZ DISK_RESET ; AH=1
EC90 FECC
2392 DEC AH ; TEST FOR DRIVE IN 0-3 RANGE
EC92 73
2393 JZ DISK_STATUS ; AH=2
EC94 C606410000
2394 MOV DISKETTE_STATUS,0 ; RESET THE STATUS INDICATOR
EC99 80FA04
2395 CMP DL,4 ; TEST FOR DRIVE IN 0-3 RANGE
EC9C T313
2396 JAE J3 ; AH=2
EC9E FECC
2397 DEC AH ; AH=3
EC9A T469
2398 JZ DISK_READ ; TEST_DISK_VRF
EC9C FECC
2399 DEC AH ; AH=4
EC94 7503
2400 JNZ J2 ; AH=2
EC96 E99500
2401 JMP DISK_WRITE ; AH=3
EC99 2402 JZ: ; TEST_DISK_VRF
EC9A FECC
2403 DEC AH ; AH=4
ECAB T467
2404 JZ DISK_VRF ; AH=5
ECAB FECC
2405 DEC AH ; AH=5
EC9F T467
2406 JZ DISK_FORMAT ; BAD_COMMAND
ECB1
2407 J3: ; ECB1 C606410001
2408 MOV DISKETTE_STATUS,BAD_CMD ; ERROR_CODE, NO SECTORS TRANSFERRED
ECB6 C3
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
2414 DISK_RESET PROC NEAR
2415 MOV DX,03F2H ; ADAPTER CONTROL PORT
2416 CLI ; NO INTERRUPTS
2417 MOV AL,MOTOR_STATUS ; WHICH MOTOR IS ON
2418 MOV CL,01H ; SHIFT COMMAND
2419 SAL AL,CL ; MOVE MOTOR VALUE TO HIGH NYBBLE
2420 TEST AL,20H ; SELECT CORRESPONDING DRIVE
2421 JNZ JS ; JUMP IF MOTOR ONE IS ON
2422 TEST AL,40H ; JUMP IF MOTOR TWO IS ON
2423 JNZ J4 ; JUMP IF MOTOR ZERO IS ON
2424 TEST AL,80H
2425 JZ J6 ; JUMP IF MOTOR ONE IS ON
2426 INC AL
2427 J4:
2428 INC AL ; TURN ON INTERRUPT ENABLE
2429 J5: INC AL ; RESET THE ADAPTER
2430 INC AL ; SET RECAL REQUIRED ON ALL DRIVES
2431 J6: INC AL ; SET OK STATUS FOR DISKETTE
2432 OR AL,8 ; TURN OFF RESET
2433 OUT DX,AL ; TURN OFF THE RESET
2434 MOV SEEK_STATUS,0 ; REENABLE THE INTERRUPTS
2435 MOV DISKETTE_STATUS,0 ; DO SENSE INTERRUPT STATUS
2436 OR AL,4 ; FOLLOWING RESET
2437 OUT DX,AL ; IGNORE ERROR RETURN AND DO OWN TEST
2438 STI ; TEST FOR DRIVE READY TRANSITION
2439 CALL CHK_STAT_2 ; EVERYTHING OK
2440
2441 MOV AL,NEC_STATUS ; SET ERROR CODE
2442 CMP AL,0COH ; SET OK STATUS FOR DISKETTE
2443 JZ J7 ; SET ERROR CODE
2444 OR DISSKETTE_STATUS,BAD_NEC ; SET ERROR CODE
2445 RET
2446
2447 ;----- SEND SPECIFY COMMAND TO NEC
2448
2449 J7: ; DRIVE_READY
2450 MOV AH,03H ; SPECIFY_COMMAND
2451 CALL NEC_OUTPUT ; OUTPUT THE COMMAND
2452 MOV BX,T ; FIRST_BYTE_PARM IN BLOCK
2453 CALL GET_PARM ; TO THE NEC CONTROLLER
2454 MOV BX,3 ; SECOND_BYTE_PARM IN BLOCK
2455 CALL GET_PARM ; TO THE NEC CONTROLLER
2456 J8: ; RESET_RET
2457 RET ; RETURN TO CALLER
2458 DISK_RESET ENDP
2459
2460 ;----- DISKETTE STATUS ROUTINE
2461
2462 DISK_STATUS PROC NEAR
2463 MOV AL,DISKETTE_STATUS ; READ COMMAND FOR DMA
2464 RET ; DISK_READ_CONT
2465 DISK_STATUS ENDP ; SET UP THE DMA
2466
2467 ;----- DISKETTE READ
2468
2469 DISK_READ PROC NEAR ; SET UP RD COMMAND FOR NEC CONTROLLER
2470 MOV AL,046H ; GO DO THE OPERATION
2471 J9: ; READ COMMAND FOR DMA
2472 CALL DMA_SETUP ; DISK_READ_CONT
2473 MOV AH,0E6H ; SET UP THE DMA
2474 JMP SHORT RW_OPEN ; SET UP RD COMMAND FOR NEC CONTROLLER
2475
2476 DISK_READ ENDP ; GO DO THE OPERATION
2477
2478 ;----- DISKETTE VERIFY
2479 DISK_VRF PROC NEAR ; VERIFY COMMAND FOR DMA
2480 MOV AL,042H ; DO AS IF DISK READ
2481 J0: ; VERIFY COMMAND FOR DMA
2482 DISK_VRF ENDP ; DO AS IF DISK READ
2483
2484 ;----- DISKETTE FORMAT
2485
2486 DISK_FORMAT PROC NEAR ; INDICATE WRITE OPERATION
2487 MOV MOTOR_STATUS,80H ; WILL WRITE TO THE DISKETTE
2488 MOV AL,04AH ; SET UP THE DMA
2489 CALL DMA_SETUP ; ESTABLISH THE FORMAT COMMAND
2490 MOV AH,04DH ; DO THE OPERATION
2491 JMP SHORT RW_OPEN ; CONTINUATION OF RW_OPEN FOR FMT
2492 J10: ; GET THE
2493 MOV BX,T ; BYTES/SECTOR VALUE TO NEC
2494 CALL GET_PARM ; GET THE
2495 MOV BX,S ; SECTORS/TRACK VALUE TO NEC
2496 CALL GET_PARM ; GET THE
2497 MOV BX,T5 ; GAP LENGTH VALUE TO NEC
2498 CALL GET_PARM ; GET THE FILLER BYTE
2499 MOV BX,17 ; GET THE
2500 JMP J16 ; TO THE CONTROLLER
2501 DISK_FORMAT ENDP
2502
2503 ;----- DISKETTE WRITE ROUTINE
2504
2505 DISK_WRITE PROC NEAR ; INDICATE WRITE OPERATION
2506 OR MOTOR_STATUS,80H ; DMA WRITE COMMAND
2507 MOV AL,04AH ; NEC COMMAND TO WRITE TO DISKETTE
2508 CALL DMA_SETUP
2509 MOV AH,0C5H
2510 DISK_WRITE ENDP
2511
2512 ;----- ALLOW WRITE ROUTINE TO FALL INTO RW_OPEN
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 7308 2518 RW_OPN PROC NEAR
ED4C C606410009 2519 JNC J11 ; TEST FOR DMA ERROR
ED51 B000 2520 MOV DISKETTE_STATUS,DMA_BOUNDARY ; SET ERROR
ED53 C3 2521 MOV AL,0 ; NO SECTORS TRANSFERRED
ED54 50 2522 RET ; RETURN TO MAIN ROUTINE
ED54 50 2523 J11 ; DO RW_OPN
ED54 50 2524 PUSH AX ; SAVE THE COMMAND
ED54 50 2525
2526 ;----- TURN ON THE MOTOR AND SELECT THE DRIVE
ED55 51 2527 PUSH CX ; SAVE THE T/S PARSMS
ED56 8ACA 2528 MOV CL,DL ; GET DRIVE NUMBER AS SHIFT COUNT
ED58 B001 2529 MOV AL,1 ; MASK FOR DETERMINING MOTOR BIT
ED5A D2E0 2530 SAL AL,CL ; SHIFT THE MASK BIT
ED5C FA 2531 CL1 ; NO INTERRUPTS WHILE DETERMINING
2532
2533 ;-----+
ED5D C6064000FF 2534 MOV MOTOR_COUNT,0FFH ; MOTOR STATUS
ED62 84063F00 2535 TEST AL,MOTOR_STATUS ; SET SHIFT COUNT DURING OPERATION
ED66 7531 2536 JNZ J14 ; TEST THAT MOTOR FOR OPERATING
ED68 80263F00F0 2537 AND MOTOR_STATUS,0F0H ; IF RUNNING, SKIP THE WAIT
ED6D 08063F00 2538 OR MOTOR_STATUS,AL ; TURN OFF ALL MOTOR BITS
ED71 FB 2539 STI ; TURN ON THE CURRENT MOTOR
ED72 B0 2540 MOV AL,10H ; INTERRUPTS BACK ON
ED73 E20 2541 SAL AL,CL ; MASK
ED76 0AC2 2542 OR AL,DL ; DEVELOP BIT MASK FOR MOTOR ENABLE
ED78 0C0C 2543 OR AL,0CH ; GET DRIVE SELECT BITS IN
ED7A 52 2544 PUSH DX ; NO RESET, ENABLE DMA/INT
ED7B BAF203 2545 MOV DX,03F2H ; SAVE REG
ED7E EE 2546 OUT DX,AL ; CONTROL PORT ADDRESS
ED7F 5A 2547 POP DX ; RECOVER REGISTERS
2548
2549 ;----- WAIT FOR MOTOR IF WRITE OPERATION
2550
ED80 F6063F0080 2551 TEST MOTOR_STATUS,80H ; IS THIS A WRITE
ED85 7412 2552 JZ J14 ; NO, CONTINUE WITHOUT WAIT
ED86 B0 400 2553 MOV BX,20 ; GET THE MOTOR WAIT
ED8A B8DF00 2554 CALL GET_PARM ; PARAMETER
ED8D 0AE4 2555 OR AH,AH ; TEST FOR NO WAIT
2556 J12: ;-----+
2557 JZ J14 ; TEST_WAIT_TIME
ED91 2BC9 2558 SUB CX,CX ; EXIT WITH TIME EXPIRED
2559 J13: ;-----+
2560 LOOP J13 ; SET UP 1/8 SECOND LOOP TIME
ED93 E2FE 2561 DEC AH ; WAIT FOR THE REQUIRED TIME
ED94 FCC 2562 JMP J12 ; DECREMENT TIME VALUE
ED97 EB76 2563 J14: ; ARE WE DONE YET
ED99 59 2564 STI ; MOTOR RUNNING
ED99 FB 2565 POP CX ; INTERRUPTS BACK ON FOR BYPASS WAIT
ED9A 59 2566
2567 ;-----+
2568 ;----- DO THE SEEK OPERATION
ED9B E8DF00 2569 CALL SEEK ; MOVE TO CORRECT TRACK
ED9E 58 2570 POP AX ; RECOVER COMMAND
ED9F 8AFC 2571 MOV BH,AH ; SAVE COMMAND IN BH
EDAA B600 2572 MOV DH,0 ; SEE NO SECTORS READ IN CASE OF ERROR
EDCB 724B 2573 JC J17 ; IF ERROR, THEN EXIT AFTER MOTOR OFF
EDA5 BEF0ED90 2574 MOV SI,[OFFSET J17] ; DUMMY RETURN ON STACK FOR NEC OUTPUT
EDA9 56 2575 PUSH SI ; SO THAT IT WILL RETURN TO MOTOR OFF
2576
2577 ;-----+
2578 ;----- SEND OUT THE PARAMETERS TO THE CONTROLLER
2579
EDAA E89400 2580 CALL NEC_OUTPUT ; OUTPUT THE OPERATION COMMAND
EDAD B66001 2581 MOV AH,[BP+1] ; GET THE CURRENT HEAD NUMBER
EDB0 D0E4 2582 SAL AH,1 ; MOVE IT TO BIT 2
EDB2 D0E4 2583 SAL AH,1 ; ISOLATE THAT BIT
EDB4 B0E404 2584 AND AH,4 ; OR IN THE DRIVE NUMBER
EDB7 DAE2 2585 OR AH,0D ; LOCATION
EDB9 E88500 2586 CALL NEC_OUTPUT
2587
2588 ;----- TEST FOR FORMAT COMMAND
2589
EDBC B0FF4D 2590 CMP BH,04DH ; IS THIS A FORMAT OPERATION
EDBB 7503 2591 JNE J15 ; NO, CONTINUE WITH R/W/V
EDC1 E962FF 2592 JMP J10 ; IF SO, HANDLE SPECIAL
EDC4 2593 J15: ;-----+
2594 MOV AH,CH ; CYLINDER NUMBER
EDC6 E87800 2595 CALL NEC_OUTPUT ; HEAD NUMBER FROM STACK
EDC9 B66001 2596 MOV AH,[BP+1] ; NO SECTORS
EDC0 E82000 2597 CALL NEC_OUTPUT ; SECTOR NUMBER
EDCF E1 2598 MOV AH,CL ; BYTES/SECTOR PARM FROM BLOCK
EDDI E6A000 2599 CALL NEC_OUTPUT ; TO THE NEC
EDD4 BB0700 2600 MOV BX,T ; EOT PARM FROM BLOCK
EDD7 E89200 2601 CALL GET_PARM ; TO THE NEC
EDDA BB0900 2602 MOV BX,9 ; GAP LENGTH PARM FROM BLOCK
EDDD E89000 2603 CALL GET_PARM ; TO THE NEC
EDDE BB0800 2604 MOV BX,T ; DTL PARM FROM BLOCK
EDE3 E8A600 2605 CALL GET_PARM ; RW_OPN FINISH
EDE6 BB0600 2606 MOV BX,T3 ; TO THE NEC
EDE9 2607 J16: ; CAN NOW DISCARD THAT DUMMY
EDE9 E88000 2608 CALL GET_PARM ; RETURN ADDRESS
EDEC 5E 2609 POP SI
2610
2611 ;-----+
2612 ;----- LET THE OPERATION HAPPEN
2613
EDED E84301 2614 CALL WAIT_INT ; WAIT FOR THE INTERRUPT
EDF0 7245 2615 J17: JC J21 ; MOTOR OFF
EDF2 E74701 2616 CALL RESULTS ; LOOK FOR ERROR
EDF5 723F 2617 JC J20 ; GET THE NEC STATUS
2618 JC J20 ; LOOK FOR ERROR
2619
2620 ;----- CHECK THE RESULTS RETURNED BY THE CONTROLLER
2621
EDF7 FC 2622 CLD ; SET THE CORRECT DIRECTION
EDF8 BE4200 2623 MOV SI,[OFFSET NEC_STATUS] ; POINT TO STATUS FIELD
EDFB AC 2624 LODS NEC_STATUS ; GET STO
EDFC 24C0 2625 AND AL,0C0H ; TEST FOR NORMAL TERMINATION
EDEF 743B 2626 JZ J22 ; NOT ABNORMAL, BAD NEC
EDF0 7240 2627 CMP AL,040H ; TEST FOR ABNORMAL TERMINATION
EDF2 7529 2628 JNZ J18 ; NOT ABNORMAL, BAD NEC
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 LODS NEC_STATUS ; GET STI
EE05 D0E0 2633 SAL AL,T ; TEST FOR EOT FOUND
EE07 B404 2634 MOV AH,RECORD_NOT_FND
EE08 A4 2635 JC J19 ; RW_FAIL
EE0B D0E0 2636 SAL AL,1 ; TEST FOR CRC ERROR
EE0D D0E0 2637 SAL AL,1 ; RW_FAIL
EE0F B410 2638 MOV AH,BAD_CRC ; TEST FOR DMA OVERRUN
EE11 721C 2639 JC J19 ; RW_FAIL
EE12 D0E0 2640 SAL AL,1 ; TEST FOR DMA_OVERRUN
EE13 D0E0 2641 MOV AH,BAD_DMA ; RW_FAIL
EE17 7216 2642 JC J19 ; TEST FOR RECORD NOT FOUND
EE19 D0E0 2643 SAL AL,1 ; RW_FAIL
EE1B D0E0 2644 SAL AL,1 ; TEST FOR WRITE PROTECT
EE1D B404 2645 MOV AH,RECORD_NOT_FND ; RW_FAIL
EE1F 720E 2646 JC J19 ; TEST MISSING ADDRESS MARK
EE21 D0E0 2647 SAL AL,1 ; RW_FAIL
EE23 B403 2648 MOV AH,WRITE_PROTECT ; TEST FOR WRITE_PROTECT
EE25 7208 2649 JC J19 ; RW_FAIL
EE27 D0E0 2650 SAL AL,1 ; TEST MISSING ADDRESS MARK
EE29 B402 2651 MOV AH,BAD_ADDR_MARK ; RW_FAIL
EE2B T202 2652 JC J19 ; RW_FAIL
2653
2654 ;----- NEC MUST HAVE FAILED
2655
EE2D 2656 J18: MOV AH,BAD_NECK ; RW-NEC-FAIL
EE2D B420 2657
EE2F 2658 J19: OR DISKETTE_STATUS,AH ; RW-FAIL
EE2F 08264100 2659 CALL NUM_TRANS ; HOW MANY WERE REALLY TRANSFERRED
EE33 E67801 2660
EE36 C3 2661 J20: RET ; RW_ERR
EE37 2662 RET ; RETURN TO CALLER
EE37 E62F01 2663 J21: CALL RESULTS ; RW_ERR_RES
EE3A C3 2664 RET ; FLUSH THE RESULTS BUFFER
2665
2666
2667 ;----- OPERATION WAS SUCCESSFUL
2668
EE3B 2669 J22: CALL NUM_TRANS ; OPN_OK
EE3B E67001 2670 XOR AH,AH ; HOW MANY GOT MOVED
EE3E 32E4 2671 RET ; NO ERRORS
EE40 C3 2672 RW_OPN ENDP
2673
2674 ;----- NEC_OUTPUT
2675
EE41 2676 : THIS ROUTINE SENDS A BYTE TO THE NEC CONTROLLER AFTER TESTING
EE42 51 2677 : THE SELECTED DIRECTION AND COUNT LENGTH. THIS ROUTINE WILL
EE43 BAF403 2678 : TIME OUT IF THE BYTE IS NOT ACCEPTED WITHIN A REASONABLE
EE44 33C9 2679 : AMOUNT OF TIME, SETTING THE DISKETTE STATUS ON COMPLETION.
2680 : INPUT (AH) BYTE TO BE OUTPUT
2681 :----- OUTPUT
2682 :----- CY = 0 SUCCESS
2683 :----- CY = 1 FAILURE -- DISKETTE STATUS UPDATED
2684 :----- IF A FAILURE HAS OCCURRED, THE RETURN IS MADE ONE LEVEL
2685 :----- HIGHER THAN THE CALLER OF NEC_OUTPUT.
2686 :----- THIS REMOVES THE REQUIREMENT OF TESTING AFTER EVERY
2687 :----- CALL OF NEC_OUTPUT.
2688 :----- (AL) DESTROYED
2689
2690 :----- NEC_OUTPUT PROC NEAR
EE41 2691 NEC_OUTPUT PROC NEAR ; SAVE REGISTERS
EE42 51 2692 PUSH DX ; STATUS PORT
EE43 BAF403 2693 PUSH CX ; COUNT FOR TIME OUT
EE44 33C9 2694 MOV DX,03F4H ; TEST_DIRECTION BIT
EE45 2695 XOR CX,CX ; DIRECTION OK
2696 J23: LOOP J23 ; TIME_ERROR
EE46 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 ;----- NEC_OUTPUT
EE4F 2701 J24: OR DISKETTE_STATUS,TIME_OUT ;----- NEC_OUTPUT
EE4F 800E410080 2702 POP CX ;----- NEC_OUTPUT
EE54 59 2703 POP DX ;----- NEC_OUTPUT
EE55 5A 2704 POP AX ;----- NEC_OUTPUT
EE56 5A 2705 POP AX ;----- NEC_OUTPUT
EE57 F9 2706 STC ;----- NEC_OUTPUT
EE58 C3 2707 RET ;----- NEC_OUTPUT
EE59 33C9 2708 J25: XOR CX,CX ;----- NEC_OUTPUT
EE5B 2709 J26: XOR CX,CX ;----- NEC_OUTPUT
EE5B EC 2710 J26: XOR CX,CX ;----- NEC_OUTPUT
EE5C A880 2711 IN AL,DX ;----- NEC_OUTPUT
EE5E 7504 2712 TEST AL,080H ;----- NEC_OUTPUT
EE60 E2F9 2713 JNZ J27 ;----- NEC_OUTPUT
EE62 EBEB 2714 LOOP J26 ;----- NEC_OUTPUT
EE64 2715 JMP J24 ;----- NEC_OUTPUT
EE64 BAC4 2716 J27: ;----- NEC_OUTPUT
EE66 B2F5 2717 MOV AL,AH ;----- NEC_OUTPUT
EE68 EE 2718 MOV DL,0F5H ;----- NEC_OUTPUT
EE69 59 2719 OUT DX,AL ;----- NEC_OUTPUT
EE6A 5A 2720 POP CX ;----- NEC_OUTPUT
EE6B C3 2721 POP DX ;----- NEC_OUTPUT
2722 RET ;----- NEC_OUTPUT
2723 NEC_OUTPUT ENDP ;----- NEC_OUTPUT

```

```

2724 ;-----+
2725 ; GET_PARM
2726 ; THIS ROUTINE FETCHES THE INDEXED POINTER FROM THE DISK BASE
2727 ; BLOCK POINTED AT BY THE DATA VARIABLE DISK_POINTER, A BYTE FROM
2728 ; THAT TABLE IS THEN MOVED INTO AH, THE INDEX OF THAT BYTE BEING
2729 ; THE PARM IN BX
2730 ; ENTRY --
2731 ; BX = INDEX OF BYTE TO BE FETCHED *
2732 ; IF THE LOW BIT OF BX IS ON, THE BYTE IS IMMEDIATELY OUTPUT
2733 ; TO THE NEC CONTROLLER
2734 ; EXIT --
2735 ; AH = THAT BYTE FROM BLOCK
2736 ;-----+
2737 GET_PARM PROC NEAR
2738 PUSH DS,AX ; SAVE SEGMENT
2739 SUB AX,AX ; ZERO TO AX
2740 MOV DS,AX
2741 ASSUME DS:AB50
2742 LDS SI,DISK_POINTER ; POINT TO BLOCK
2743 SHR BX,1 ; DIVIDE BX BY 2, AND SET FLAG
2744 TEST BX,BX ; TEST FOR ONE
2745 MOV AH,[SI+BX] ; GET THE WORD
2746 POP DS ; RESTORE SEGMENT
2747 ASSUME DS:DATA
2748 JC NEC_OUTPUT ; IF FLAG SET, OUTPUT TO CONTROLLER
2749 RET ; RETURN TO CALLER
2750 GET_PARM ENDP
2751 ;-----+
2752 ; SEEK
2753 ; THIS ROUTINE WILL MOVE THE HEAD ON THE NAMED DRIVE TO THE
2754 ; NAMED TRACK. IF THE DRIVE HAS NOT BEEN ACCESSED SINCE THE
2755 ; DRIVE RESET COMMAND WAS ISSUED, THE DRIVE WILL BE RECALIBRATED.
2756 ; INPUT
2757 ; (DL) = DRIVE TO SEEK ON
2758 ; (CH) = TRACK TO SEEK TO
2759 ; OUTPUT
2760 ; CY = 0 SUCCESS
2761 ; CY = 1 FAILURE. -- DISKETTE_STATUS SET ACCORDINGLY
2762 ; (AX) DESTROYED
2763 ;-----+
2764 SEEK PROC NEAR
2765 MOV AL,1 ; ESTABLISH MASK FOR RECAL TEST
2766 PUSH CX ; SAVE INPUT VALUE INTO CL
2767 MOV CL,DL ; GET DRIVE VALUE
2768 ROL AL,CL ; SHIFT IT BY THE DRIVE VALUE
2769 POP CX ; RECOVER TRACK VALUE
2770 TEST AL,SEEK_STATUS ; TEST FOR RECAL REQUIRED
2771 JNZ J28 ; NO RECAL
2772 OR SEEK_STATUS,AL ; TURN ON THE NO RECAL BIT IN FLAG
2773 MOV AX,7 ; RECALIBRATE COMMAND
2774 CALL NEC_OUTPUT
2775 MOV AH,DL ;-----+
2776 CALL NEC_OUTPUT ; OUTPUT THE DRIVE NUMBER
2777 CALL CHK_STAT_2 ; GET THE INTERRUPT AND SENSE INT STATUS
2778 JC J32 ; SEEK_ERROR
2779
2780 ;-----+ DRIVE IS IN SYNC WITH CONTROLLER, SEEK TO TRACK
2781
2782 J28:
2783 MOV AH,0FH ; SEEK COMMAND TO NEC
2784 CALL NEC_OUTPUT
2785 MOV AH,0E ; DRIVE NUMBER
2786 CALL NEC_OUTPUT
2787 MOV AH,EH ; TRACK NUMBER
2788 CALL NEC_OUTPUT
2789 CALL CHK_STAT_2 ; GET ENDING INTERRUPT AND
2790 CALL CHK_STAT_2 ; SENSE STATUS
2791
2792 ;-----+ WAIT FOR HEAD SETTLE
2793
2794 PUSHF ; SAVE STATUS FLAGS
2795 MOV BX,I8 ; GET HEAD SETTLE PARAMETER
2796 CALL GET_PARM
2797 PUSH CX
2798 J29: PUSH CX ; SAVE REGISTER
2799 MOV CX,550 ; HEAD SETTLE
2800 OR AH,AH ; 1 MS LOOP
2801 JZ J31 ; TEST FOR TIME EXPIRED
2802 J30: DEC AH ; DELAY FOR 1 MS
2803 LOOP J30 ; DECREMENT THE COUNT
2804 DEC AH ; DO IT SOME MORE
2805 JMP J29
2806 J31: POP CX ; RECOVER STATE
2807 POPF CX ; SEEK_ERROR
2808 POPF CX ; RETURN TO CALLER
2809 J32: RET
2810 SEEK ENDP
2811

```

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

```

2812 :----- DMA_SETUP
2813 : THIS ROUTINE SETS UP THE DMA FOR READ/WRITE/VERIFY OPERATIONS. :
2815 : INPUT (AL) = MODE BYTE FOR THE DMA
2816 : (ES:BX) - ADDRESS TO READ/WRITE THE DATA
2817 : OUTPUT (AX) DESTROYED
2819 :----- DMA_SETUP PROC NEAR
2820 :----- DMA_SETUP PROC NEAR
2821 EEC8 51 2822 PUSH CX : SAVE THE REGISTER
2822 EEC9 FA 2823 CLI : NO MORE INTERRUPTS
2823 EEC A60C 2824 OUT DMA+12, AL : SET THE FIRST/LAST F/F
2825 EEEC 50 2825 PUSH AX
2826 EECF 59 2826 POP AX
2827 EECF 60B 2827 OUT AX, 11, AL : OUTPUT THE MODE BYTE
2828 EED0 8CC0 2828 MOV AX, ES : GET THE ES VALUE
2829 EED2 B104 2829 MOV CL, 4 : SHIFT COUNT
2830 EED4 D3C0 2830 ROL AX, CL : ROTATE LEFT
2831 EED6 8AE8 2831 MOV CH, AL : GET HIGHEST NYBLE OF ES TO CH
2832 EED8 24F0 2832 AND AL, 0FH : ZERO THE LOW NYBLE FROM SEGMENT
2833 EEDC 7302 2833 ADD AX, BX : TEST FOR CARRY FROM ADDITION
2834 EEDF FEC5 2834 JNC J33
2835 EEE0 2363 INC CH : CARRY MEANS HIGH 4 BITS MUST BE INC
2836 J33: EEE0 50 2837 PUSH AX : SAVE START ADDRESS
2837 EEE1 5A4 2838 SUB AL, AL : OUTPUT LOW ADDRESS
2838 EEE3 5AC4 2839 MOV AL, AH : OUTPUT HIGH ADDRESS
2839 EEE5 E604 2840 OUT DMA+4, AL : GET HIGH 4 BITS
2840 EEE7 8AC5 2841 MOV AL, CH : GET HIGH 4 BITS
2841 EEE9 240F 2842 AND AL, 0FH
2842 EEEB E681 2843 OUT 081H, AL : OUTPUT THE HIGH 4 BITS TO
2843 2844 :----- THE PAGE REGISTER
2844 2845 :
2845 2846 :----- DETERMINE COUNT
2846 EEEF 8AE6 2847 MOV AH,DH : NUMBER OF SECTORS
2847 EEEF 2AC0 2848 SUB AL, AL : TIMES 256 INTO AX
2848 EEF0 D1E8 2849 MOV AL, AH : SECTORS * 128 INTO AX
2849 EEF2 5A00 2850 SHR AX, 1
2850 EEF4 B06000 2851 PUSH AX
2851 EEF7 E812FF 2852 MOV BH, 6 : GET THE BYTES/SECTOR PARM
2852 EEEF 8ACC 2853 CALL GET_PARM : USE AS SHIFT COUNT (0=128, 1=256 ETC)
2853 EECF 58 2854 MOV CL, AH
2854 EEEF D3E0 2855 POP AX : MULTIPLY BY CORRECT AMOUNT
2855 EEF0 5A 2856 SHL AX, CL : FOR COUNT VALUE
2856 EEF0 50 2857 DEC AX : SAVE COUNT VALUE
2857 EEF0 E605 2858 PUSH AX : LOW BYTE OF COUNT
2858 EEF3 8AC4 2859 OUT DMA+5, AL : NUMBER OF SECTORS
2860 EEF5 E605 2861 MOV AL, AH : TIMES 256 INTO AX
2861 EEF7 FB 2862 OUT DMA+5, AL : SECTORS * 128 INTO AX
2862 EEF8 59 2863 ST1 : HIGH BYTE OF COUNT
2863 EEF9 58 2864 POP CX : INTERRUPTS BACK ON
2864 EFOA 03C1 2865 ADD AX, CX : RECOVER COUNT VALUE
2865 EFOC 59 2866 POP CX : ADD TEST FOR 64K OVERFLOW
2866 EF0D B002 2867 MOV AL, 2 : RECOVER REGISTER
2867 EF0F E60A 2868 OUT DMA+10, AL : MODE FOR 8237
2868 EF11 C3 2869 RET : INITIALIZE THE DISKETTE CHANNEL
2869 2870 :----- RETURN TO CALLER,
2870 :----- ; CFL SET BY ABOVE IF ERROR
2871 DMA_SETUP ENDP
2872 :----- CHK_STAT_2
2873 : THIS ROUTINE HANDLES THE INTERRUPT RECEIVED AFTER A
2874 : RECALIBRATE, SEEK, OR RESET TO THE ADAPTER.
2875 : THE INTERRUPT IS WAITED FOR, THE INTERRUPT STATUS SENSED,
2876 : AND THE RESULT RETURNED TO THE CALLER.
2877 :
2878 :----- INPUT
2879 :----- NONE
2880 :----- OUTPUT
2881 :----- CY = 0 SUCCESS
2882 :----- CY = 1 FAILURE -- ERROR IS IN DISKETTE_STATUS
2883 :----- (AX) DESTROYED
2884 :----- CHK_STAT_2 PROC NEAR
2885 CHK_STAT_2 :----- WAIT FOR THE INTERRUPT
2886 CALL WAIT_INT :----- IF ERROR, RETURN IT
2887 JC J34 :----- SENSE INTERRUPT STATUS COMMAND
2888 JC J34 :----- READ IN THE RESULTS
2889 MOV AH, 0BH :----- CHK2 RETURN
2890 CALL NEW_OUTPUT :----- GET THE FIRST STATUS BYTE
2891 JC J34 :----- ISOLATE THE BITS
2892 MOV AL, NEC_STATUS :----- TEST FOR CORRECT VALUE
2893 AND AL, 060H :----- IF ERROR, GO MARK IT
2894 CMP AL, 060H :----- GOOD RETURN
2895 JZ J35
2896 CLC :----- RETURN TO CALLER
2897 J34: EF2B C3 2898 RET :----- CHK2_ERROR
2898 EF2C 800E410040 2899 J35: OR DISKETTE_STATUS,BAD_SEEK :----- ERROR RETURN CODE
2899 EF31 F9 2900 STC
2901 EF32 C3 2902 RET
2903 CHK_STAT_2 ENDP

```

```

2904 ;----- WAIT_INT
2905 ; THIS ROUTINE WAITS FOR AN INTERRUPT TO OCCUR. A TIME OUT
2906 ; ROUTINE TAKES PLACE DURING THE WAIT, SO THAT AN ERROR MAY BE
2907 ; RETURNED IF THE DRIVE IS NOT READY.
2908 ;----- INPUT
2910 ; NONE
2911 ;----- OUTPUT
2912 ; CY = 0 SUCCESS
2913 ; CY = 1 FAILURE -- DISKETTE_STATUS IS SET ACCORDINGLY
2914 ; (AH) DESTROYED
2915 ;----- WAIT_INT PROC NEAR
2916 WAIT_INT PROC NEAR
2917 STI ; TURN ON INTERRUPTS, JUST IN CASE
2918 PUSH BX
2919 PUSH CX ; SAVE REGISTERS
2920 MOV BL,2 ; CLEAR THE COUNTERS
2921 XOR CX,CX ; FOR 2 SECOND WAIT
2922 J36: ;----- INPUT
2923 TEST SEEK_STATUS,INT_FLAG ; TEST FOR INTERRUPT OCCURRING
2924 JNZ J37 ;----- OUTPUT
2925 LOOP J36 ; COUNT DOWN WHILE WAITING
2926 DEC BL ; SECOND LEVEL COUNTER
2927 JNZ J36 ;----- INPUT
2928 OR DISKETTE_STATUS,TIME_OUT ; NOTHING HAPPENED
2929 STC ;----- OUTPUT
2930 RET ;----- INPUT
2931 PUSHF SEEK_STATUS,NOT INT_FLAG ; SAVE CURRENT CARRY
2932 ANDF POPF ;----- OUTPUT
2933 POPF ; RECOVER CARRY
2934 POP CX ;----- INPUT
2935 POP BX ; RECOVER REGISTERS
2936 RET ;----- OUTPUT
2937 ; GOOD RETURN CODE COMES
2938 ; FROM TEST INST
2939 ;----- INPUT
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 ;----- DISK_INT PROC FAR
2947 ORG 0E57H
2948 DISK_INT PROC FAR
2949 STI ; RE ENABLE INTERRUPTS
2950 PUSH DS
2951 PUSH AX
2952 CALL DDS
2953 OR SEEK_STATUS,INT_FLAG ;----- INPUT
2954 MOV AL,20H ; END OF INTERRUPT MARKER
2955 OUT 20H,AL ; INTERRUPT CONTROL PORT
2956 POP AX ;----- INPUT
2957 POP DS ; RECOVER SYSTEM
2958 IRET ;----- OUTPUT
2959 ;----- DISK_INT ENDP
2960 ;----- INPUT
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 ;----- RESULTS PROC NEAR
2972 RESULTS PROC NEAR
2973 CLD
2974 MOV DI,[OFFSET NEC_STATUS] ; POINTER TO DATA AREA
2975 PUSH CX ;----- INPUT
2976 PUSH DX ;----- INPUT
2977 PUSH BX ;----- INPUT
2978 MOV BL,7 ; MAX STATUS BYTES
2979 ;----- WAIT FOR REQUEST FOR MASTER
2980 ;----- INPUT
2981 J38: ;----- INPUT
2982 XOR CX,CX ;----- INPUT
2983 MOV DX,03F4H ;----- INPUT
2984 J39: ;----- INPUT
2985 MOV CX,DX ;----- INPUT
2986 IN AL,DX ;----- INPUT
2987 TEST AL,000H ;----- INPUT
2988 J40A ;----- INPUT
2989 LOOP J39 ;----- INPUT
2990 OR DISKETTE_STATUS,TIME_OUT ;----- INPUT
2991 J40: ;----- INPUT
2992 STC ;----- INPUT
2993 POP BX ;----- INPUT
2994 POP DX ;----- INPUT
2995 POP CX ;----- INPUT
2996 RET ;----- INPUT
2997 ;----- TEST THE DIRECTION BIT
2998 ;----- INPUT
2999 J40A: ;----- INPUT
3000 IN AL,DX ;----- INPUT
3001 TEST AL,040H ;----- INPUT
3002 J42 ;----- INPUT
3003 J40A ;----- INPUT
3004 J41: ;----- INPUT
3005 OR DISKETTE_STATUS,BAD_NEC ;----- INPUT
3006 JMP J40 ;----- INPUT
3007 ;----- INPUT
3008 ;----- READ IN THE STATUS
3009 J42: ;----- INPUT
3010 IN AL,DX ;----- INPUT
3011 INC DX ;----- INPUT
3012 IN AL,DX ;----- INPUT
3013 MOV [DI],AL ;----- INPUT
3014 INC DX ;----- INPUT
3015 MOV CX,10 ;----- INPUT
3016 LOOP J43 ;----- INPUT
3017 DEC DX ;----- INPUT
3018 IN AL,DX ;----- INPUT
3019 TEST AL,010H ;----- INPUT
EF94 ;----- INPUT
EF94 42 ;----- INPUT
EF95 EC ;----- INPUT
EF95 0005 ;----- INPUT
EF98 87 ;----- INPUT
EF99 B90A00 ;----- INPUT
EF9C E2FE ;----- INPUT
EF9E 4A ;----- INPUT
EF9F EC ;----- INPUT
EF9D AB10 ;----- INPUT

```

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
3027 J44:
3028     POP BX
3029     POP DX
3030     POP CX
3031     RET      ; RECOVER REGISTERS
3032     ; GOOD RETURN CODE FROM TEST INST
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 ;----- PROC NEAR
3043 NUM_TRANS 3044 MOV AL,NEC_STATUS+3 ; GET CYLINDER ENDED UP ON
3044          3045 CMP AL,CH      ; SAME AS WE STARTED
3045          3046 MOV AL,NEC_STATUS+5 ; GET ENDING SECTOR
3046          3047 JZ J45      ; IF ON SAME CYL, THEN NO ADJUST
3047          3048 MOV BX,8
3048          3049 CALL GET_PARM ; GET EOT VALUE
3049          3050 MOV AL,AH      ; INTO AL
3050          3051 INC AL      ; USE EOT+1 FOR CALCULATION
3051          3052 J45: SUB AL,CL      ; SUBTRACT START FROM END
3052          3053 RET
3053          3054
3054          3055 NUM_TRANS 3056 RESULTS ENDP
3056          3057 ;----- DISK_BASE
3057          3058 ;----- THIS IS THE SET OF PARAMETERS REQUIRED FOR DISKETTE OPERATION.
3058          3059 ;----- THEY ARE POINTED AT BY THE DATA VARIABLE DISK_POINTER. TO
3059          3060 ;----- MODIFY THE PARAMETERS, BUILD ANOTHER PARAMETER BLOCK AND POINT
3060          3061 ;----- DISK_POINTER TO IT.
3061          3062 ;----- DISK_POINTER TO IT.
3062          3063 ;----- ORG 0EFC7H
3063          3064 DISK_BASE 3064 LABEL BYTE
3064          3065 DB 11001111B ; SRT=C, HD UNLOAD=0F - 1ST SPECIFY BYTE
3065          3066 DB 2           ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
3066          3067 DB
3067          3068 DB MOTOR_WAIT ; WAIT AFTER OPN TIL MOTOR OFF
3068          3069 DB 2           ; 512 BYTES/SECTOR
3069          3070 DB 8           ; EDTR ( LAST SECTOR ON TRACK)
3070          3071 DB 02AH        ; GAP LENGTH
3071          3072 DB 0FFH        ; DTL
3072          3073 DB 050H        ; GAP LENGTH FOR FORMAT
3073          3074 DB 0F6H        ; FILL BYTE FOR FORMAT
3074          3075 DB 25          ; HEAD SETTLE TIME (MILLISECONDS)
3075          3076 DB 4           ; MOTOR START TIME (1/8 SECONDS)
3076          3077
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 :      T   6   5   4   3   2   1   0
3089 :      |   |   |   |   |   |   |   |
3090 :      |   |   |   |   |   |   |   | TIME OUT
3091 :      |   |   |   |   |   |   |   | = UNUSED
3092 :      |   |   |   |   |   |   |   | = I/O ERROR
3093 :      |   |   |   |   |   |   |   | = SELECTED
3094 :      |   |   |   |   |   |   |   | = ACKNOWLEDGE
3095 :      |   |   |   |   |   |   |   | = OUT OF PAPER
3096 :      |   |   |   |   |   |   |   | = I = NOT BUSY
3097 : (DX) = PRINTER TO BE USED (0,1,2) CORRESPONDING TO ACTUAL
3098 : VALUES IN PRINTER_BASE AREA
3099 :
3100 : DATA AREA PRINTER_BASE CONTAINS THE BASE ADDRESS OF THE PRINTER
3101 : CARD(S) AVAILABLE (LOCATED AT BEGINNING OF DATA SEGMENT,
3102 : 408H ABSOLUTE, 3 WORDS)
3103 :
3104 : DATA AREA PRINT_TIM_OUT (BYTE) MAY BE CHANGED TO CAUSE DIFFERENT
3105 : TIME-OUT WAITS. DEFAULT=20
3106 :
3107 : REGISTERS AH IS MODIFIED
3108 : ALL OTHERS UNCHANGED
3109 :
3110 : ASSUME CS:CODE,DS:DATA
3111 : ORG 0EFD2H
3112 : PRINTER PROC FAR
3113 : STI ; INTERRUPTS BACK ON
3114 : PUSH DS ; SAVE SEGMENT
3115 : PUSH DX
3116 : PUSH SI
3117 : PUSH CX
3118 : PUSH BX
3119 : CALL DOS
3120 : MOV SI,DX ; GET PRINTER PARM
3121 : MOV BL,PRINT_TIM_OUT[SI] ; LOAD TIME-OUT PARM
3122 : SHL SI,1 ; WORD OFFSET INTO TABLE
3123 : MOV DX,PRINTER_BASE[SI] ; GET BASE ADDRESS FOR PRINTER CARD
3124 : OR DX,DX ; TEST DX FOR ZERO,
3125 : ; INDICATING NO PRINTER
3126 : JZ B1 ; RETURN
3127 : OR AH,AH ; TEST FOR (AH)=0
3128 : JZ B2 ; PRINT AL
3129 : DEC AH ; TEST FOR (AH)=1
3130 : JZ B3 ; INITIATE
3131 : DEC AH ; TEST FOR (AH)=2
3132 : JZ B5 ; PRINTER STATUS
3133 : B1: ; RETURN
3134 : POP BX
3135 : POP CX
3136 : POP SI ; RECOVER REGISTERS
3137 : POP DX ; RECOVER REGISTERS
3138 : POP DS
3139 : IRET
3140 :
3141 : ----- PRINT THE CHARACTER IN (AL)
3142 :
3143 : B2: ; SAVE VALUE TO PRINT
3144 : PUSH AX
3145 : OUT DX,AL ; OUTPUT CHAR TO PORT
3146 : INC DX ; POINT TO STATUS PORT
3147 : B3: ; WAIT_BUSY
3148 : SUB CX,CX ; GET STATUS
3149 : B3_1: ; STATUS TO AH ALSO
3150 : IN AL,DX ; IS THE PRINTER CURRENTLY BUSY
3151 : MOV AH,AL
3152 : TEST AL,80H
3153 : JNZ B4 ; OUT_STROBE
3154 : LOOP B3_1 ; TRY AGAIN
3155 : DEC DX ; DROP COUNT
3156 : JNZ B3 ; GO TILL TIMEOUT ENDS
3157 : OR AH,I ; SET ERROR FLAG
3158 : AND AH,0F9H ; TURN OFF THE OTHER BITS
3159 : JMP SHORT B7 ; RETURN WITH ERROR FLAG SET
3160 : B4: ; OUT_STROBE
3161 : MOV AL,0DH ; SET THE STROBE HIGH
3162 : INC DX ; STROBE IS BIT 0 OF PORT C OF 8255
3163 : OUT DX,AL
3164 : MOV AL,0CH ; SET THE STROBE LOW
3165 : OUT DX,AL
3166 : POP AX ; RECOVER THE OUTPUT CHAR
3167 :
3168 : ----- PRINTER STATUS
3169 :
3170 : B5: ; SAVE AL REG
3171 : PUSH AX
3172 : B6: ; GET PRINTER STATUS
3173 : MOV DX,PRINTER_BASE[SI]
3174 : INC DX
3175 : IN AL,DX ; GET PRINTER STATUS
3176 : MOV AH,AL
3177 : AND AH,0FH ; TURN OFF UNUSED BITS
3178 : B7: ; STATUS SET
3179 : POP DX ; RECOVER AL REG
3180 : MOV AL,DL ; GET CHARACTER INTO AL
3181 : XOR AH,48H ; FLIP A COUPLE OF BITS
3182 : JMP BI ; RETURN FROM ROUTINE

```

LOC	OBJECT	LINE	SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82
		3183	
		3184	;----- INITIALIZE THE PRINTER PORT
F030		3185	
F030 50		3186	B8:
F031 42		3187	PUSH AX
F032 C		3188	; SAVE AL
F033 B008		3189	INC DX
F035 EE		3190	INC DX
F036 BBEB03		3191	MOV AL,B
F039		3192	; POINT TO OUTPUT PORT
F039 48		3193	OUT DX,AL
F03A 75FD		3194	; SET INIT LINE LOW
F03C B00C		3195	DEC AX
		3196	JNZ B9
		3197	MOV AL,0CH
		3198	; NO INTERRUPTS, NON AUTO LF,
F03E EE		3199	OUT DX,AL
F03F EB00		3200	; INIT HIGH
		3201	JMP B6
			ENDP
			; PRT_STATUS_!

```

3202 :---- INT 10 -----
3204 : VIDEO IO
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= 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 BLINN
3222 : ** SETTING IT 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 : (DH,DL) = 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 0-4, 0-3 FOR MODES 2-3)
3241 : (AH)=6 SCROLL 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 : SCROLL 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) = COLOR OF CHARACTER
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 : HIGH ADDRESS (PAGE 0000H) AND SET IT 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 = RED(1)/GREEN(2)/BLUE(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

```

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

```

3314 ; ASCII TELETYPE ROUTINE FOR OUTPUT
3315 ; (AH) = 14 WRITE TELETYPE TO ACTIVE PAGE
3316 ; (AL) = CHAR TO WRITE
3317 ; (BL) = FOREGROUND COLOR IN GRAPHICS MODE
3318 ; NOTE -- SCREEN WIDTH IS CONTROLLED BY PREVIOUS MODE SET
3319 ;
3320 ; (AH) = 15 CURRENT VIDEO STATE
3321 ; RETURNS THE CURRENT VIDEO STATE
3322 ; (AL) = MODE CURRENTLY SET I SEE AH=0 FOR EXPLANATION
3323 ; (AH) = NUMBER OF CHARACTER COLUMNS ON SCREEN
3324 ; (BH) = CURRENT ACTIVE DISPLAY PAGE
3325 ;
3326 ; CS,SS,DS,ES,BX,CX,DX PRESERVED DURING CALL
3327 ;
3328 ; ALL OTHERS DESTROYED
3329 ;
3330 ;-----ASSUME CS:CODE,DS:DATA,ES:VIDEO_RAM
3331 ;-----ORG 0F045H
3332 ;-----M1 LABEL WORD ; TABLE OF ROUTINES WITHIN VIDEO I/O
3333 M1
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
0020
3350 MIL EQU $-M1
3351
3352 ;-----ORG 0F065H
3353 VIDEO_IO PROC NEAR
3354 STI ; INTERRUPTS BACK ON
3355 CLD ; SET DIRECTION FORWARD
3356 PUSH DS ; SAVE SEGMENT REGISTERS
3357 PUSH DS
3358 PUSH DX
3359 PUSH CX
3360 PUSH BX
3361 PUSH SI
3362 PUSH DI
3363 PUSH AX ; SAVE AX VALUE
3364 MOV AL,AH ; GET INTO LOW BYTE
3365 XOR AH,AH ; ZERO TO HIGH BYTE
3366 SAL AX,1 ; *2 FOR TABLE LOOKUP
3367 MOV SI,AX ; PUT INTO SI FOR BRANCH
3368 JNP AX,MIL ; TEST FOR < THAN RANGE
3369 JB M2 ; BRANCH AROUND BRANCH
3370 POP AX ; THROW AWAY THE PARAMETER
3371 JMP VIDEO_RETURN ; DO NOTHING IF NOT IN RANGE
3372 M2:
3373 CALL DDS ; SEGMENT FOR COLOR CARD
3374 MOV AX,0B800H ; GET EQUIP_FLAG
3375 MOV DI,EQUIP_FLAG ; GET EQUIPMENT SETTING
3376 AND DI,1E0H ; ISOLATE CRT SWITCHES
3377 CMP DI,30H ; IS SETTING FOR BW CARD?
3378 JNE M3 ; SEGMENT FOR BW CARD
3379 MOV AH,0B0H
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] ; ENDP
3385 VIDEO_IO

```

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 ;----- TABLES FOR USE IN SETTING OF MODE  

3397  

F0A4                    3398        ORG        0F0A4H  

F0A4                    3399        VIDEO_PARMS    LABEL     BYTE  

F0A4                    3400 ;---- INIT_TABLE  

F0A4                    3401        DB         38H,28H,2DH,0AH,1FH,6,19H      ; SET UP FOR 40X25  

F0A4 38  

F0A5 28  

F0A6 2D  

F0A7 0A  

F0A8 F  

F0A9 06  

F0AA 19  

F0AB 1C  

F0AC 02  

F0AD 07  

F0AE 06  

F0AF 07  

F0B0 00  

F0B1 00  

F0B2 00  

F0B3 00  

0010                    3402        DB         1CH,2,7,6,7  

F0B4 71  

F0B5 50  

F0B6 5A  

F0B7 0A  

F0B8 1F  

F0B9 06  

F0BA 19  

F0BB 1C  

F0BC 02  

F0BD 07  

F0BE 06  

F0BF 07  

F0C0 00  

F0C1 00  

F0C2 00  

F0C3 00  

F0C4 38  

F0C5 28  

F0C6 2D  

F0C7 0A  

F0C8 1F  

F0C9 06  

F0CA 64  

F0CB 70  

F0CD 02  

F0CD 01  

F0CE 06  

F0CF 07  

F0D0 00  

F0D1 00  

F0D2 00  

F0D3 00  

F0D4 61  

F0D5 50  

F0D6 52  

F0D7 0F  

F0D8 19  

F0D9 06  

F0DA 19  

F0DB 19  

F0DC 02  

F0DD 00  

F0DE 00  

F0DF 0C  

F0E0 00  

F0E1 00  

F0E2 00  

F0E3 00  

F0E4 0008  

F0E4 0010  

F0E8 0040  

F0EA 0040  

F0EC 28  

F0ED 28  

F0EF 55  

F0F0 28  

F0F1 28  

F0F2 50  

F0F3 50  

F0F4 2C  

F0F5 28  

F0F7 29  

F0F8 2A  

F0F9 2E  

F0FA 1E  

F0FB 29  

3404        M4        EQU        $-VIDEO_PARMS  

3405  

3406        DB         7IH,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,2DH,0AH,7FH,6,64H      ; SET UP FOR GRAPHICS  

3411        DB         10H,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  

3429 ;---- C_REG_TAB  

3430  

3431        M7        LABEL     BYTE                        ; TABLE OF MODE SETS  

3432        DB         2CH,28H,2DH,29H,2AH,2EH,1EH,29H  


```

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

```

F0FC F0FC 3433 SET_MODE PROC NEAR
F0FD BAD403 3434 MOV DX,03D4H ; ADDRESS OF COLOR CARD
F0FF B500 3435 MOV BL,0 ; MODE SET FOR COLOR CARD
F101 83F300 3436 CMP DI,30H ; IF COLOR CARD INSTALLED
F104 7506 3437 JNE M8 ; OK WITH COLOR
F106 B007 3438 MOV AL,7 ; INDICATE BW CARD MODE
F108 B2B4 3439 MOV DL,04BH ; ADDRESS OF BW CARD
F10A FEC0 3440 MOV BL,04BH ; MODE SET FOR BW CARD
F10C 8AE0 3441 INC M8: ; MODE SET FOR BW CARD
F10E A24900 3442 MOV AH,AL ; SAVE MODE IN AH
F111 B9166300 3443 MOV CRT_MODE,AL ; SAVE IN GLOBAL VARIABLE
F115 1E 3444 MOV ADDR_6845,DX ; SAVE ADDRESS OF BASE
F116 50 3445 PUSH DS ; SAVE POINTER TO DATA SEGMENT
F118 52 3446 PUSH AX ; SAVE MODE
F11B 8C204 3447 ADD DX,4 ; SAVE OUTPUT PORT VALUE
F11B 8AC3 3448 MOV AL,BL ; POINT TO CONTROL REGISTER
F11D EE 3449 OUT DX,AL ; GET MODE SET FOR CARD
F11E 5A 3450 MOV AL,BL ; RESET VIDEO
F11F 2BC0 3451 OUT DX,AL ; BACK TO BASE REGISTER
F121 8ED0 3452 POP DX ; SET UP FOR ABS0 SEGMENT
F123 C51E7400 3453 SUB AX,AX ; ESTABLISH VECTOR TABLE ADDRESSING
F127 58 3454 MOV DS,AX
F128 B91000 3455 ASSUME DS:ABSO
F129 80FC02 3456 LDSD,DX,[PARM_PTR]
F12E 80 3457 POP AX ; GET POINTER TO VIDEO PARM
F130 0209 3458 ASSUME DS:CODE ; RECOVER PARM
F132 80FC04 3459 MOV CX,M4 ; LENGTH OF EACH ROW OF TABLE
F135 T209 3460 CMP AH,2 ; DETERMINE WHICH ONE TO USE
F137 03D9 3461 JC M9 ; MODE IS 0 OR 1
F139 80FC07 3462 ADD BX,CX ; MOVE TO NEXT ROW OF INIT_TABLE
F13C T202 3463 CMP AH,4 ; MODE IS 2 OR 3
F13E 03D9 3464 JC M9 ; MODE TO GRAPHICS ROW OF INIT_TABLE
3465 ADD BX,CX ; MODE IS 4,5, OR 6
3466 CMP AH,7 ; MODE IS 6,7, OR 8
3467 JC M9 ; MODE IS 8,9, OR 10
3468 ADD BX,CX ; MODE TO BW CARD ROW OF INIT_TABLE
3469
3470 ;----- BX POINTS TO CORRECT ROW OF INITIALIZATION TABLE
3471
F140 3472 M9: ; OUT INIT
F140 50 3473 PUSH AX ; SAVE MODE IN AH
F141 32E4 3474 XOR AH,AH ; AH WILL SERVE AS REGISTER
3475 ; NUMBER DURING LOOP
3476
3477 ;----- LOOP THROUGH TABLE, OUTPUTTING REG ADDRESS, THEN VALUE FROM TABLE
3478 M10: ; INIT LOOP
F143 8AC4 3479 MOV AL,AH ; GET 6845 REGISTER NUMBER
F145 EE 3480 OUT DX,AL
F146 42 3481 INC DX
F147 FEC4 3482 INC AH
F149 8A07 3483 MOV AL,[BX]
F14A 8000 3484 INC BX
F14C 43 3485 OUT DX,AL
F14D 4A 3486 INC BX
F14E E2F3 3487 DEC DX
F150 58 3488 LOOP M10 ; NEXT TO REGISTER
F151 1F 3489 POP AX ; DO THE WHOLE TABLE
3490 POP DS ; GET MODE BACK
3491 ASSUME DS:DATA ; RECOVER SEGMENT VALUE
3492
3493 ;----- FILL REGEN AREA WITH BLANK
3494
F152 33FF 3495 XOR DI,DI ; SET UP POINTER FOR REGEN
F158 B93E4E00 3496 MOV CRT_START,DI ; START ADDRESS SAVED IN GLOBAL
F159 2E8A4E00 3497 MOV ACTIVE_PAGE,0 ; SET PAGE VALUE
F15D B90020 3498 MOV CX,8192 ; NUMBER OF WORDS IN COLOR CARD
F160 80FC04 3499 CMP AH,4 ; TEST FOR GRAPHICS
F163 720B 3500 JC M12 ; NO GRAPHICS INIT
F165 80FC07 3501 CMP AH,7 ; TEST FOR BW-CARD
F168 T404 3502 JE M11 ; BY CARD INIT
F16A 33C0 3503 XOR AX,AX ; FILL FOR GRAPHICS MODE
F16C EB05 3504 JMP SHORT M13 ; CLEAR BUFFER
F16E B508 3505 MII: ; NO GRAPHICS INIT
F170 B2007 3506 MOV CH,08H ; BUFFER SIZE ON BW CARD
F170 B2007 3507 M12: ; NO GRAPHICS INIT
F170 B2007 3508 MOV AX,'+'*7*256 ; FILL CHAR FOR ALPHA
F173 3509 M13: ; CLEAR BUFFER
F173 F3 3510 REP STOSW ; FILL THE REGEN BUFFER WITH BLANKS
F174 AB
3511
3512 ;----- ENABLE VIDEO AND CORRECT PORT SETTING
3513
F175 C70660000706 3514 MOV CURSOR_MODE,607H ; SET CURRENT CURSOR MODE
F17B A04900 3515 MOV AL,CRT_MODE ; GET THE MODE
F17E 32E4 3516 XOR AH,AL ; INTO AX REGISTER
F180 8BF0 3517 MOV SI,AX ; TABLE POINTER, INDEXED BY MODE
F182 B9166300 3518 MOV DX,ADDR_6845 ; PREPARE TO OUTPUT TO
3519 ; VIDEO ENABLE PORT
3520 ADD DX,4
3521 MOV AL,CS:[SI+OFFSET M7]
3522 OUT DX,AL ; SET VIDEO ENABLE PORT
3523 MOV CRT_MODE_SET,AL ; SAVE THAT VALUE
3524
3525 ;----- DETERMINE NUMBER OF COLUMNS, BOTH FOR ENTIRE DISPLAY
3526 ;----- AND THE NUMBER TO BE USED FOR TTY INTERFACE
3527
F192 2E8A4EC0F0 3528 MOV AL,CS:[SI+OFFSET M6] ; NUMBER OF COLUMNS IN THIS SCREEN
F197 32E4 3529 XOR AH,AL
F199 A34A00 3530 MOV CRT_COLS,AX
3531
3532 ;----- SET CURSOR POSITIONS
3533
F19C 81E60E00 3534 AND SI,0EH ; WORD OFFSET INTO CLEAR LENGTH TABLE
F1A0 2E888CE4F0 3535 MOV CX,CS:[SI+OFFSET M5] ; LENGTH TO CLEAR
F1A5 B90E4C00 3536 MOV CRT_LEN,CX ; SAVE LENGTH OF CRT -- NOT USED FOR BW
F1A9 B90800 3537 MOV CX,8 ; CLEAR ALL CURSOR POSITIONS
F1AB 81E5000 3538 MOV DI,OFFSET_CURSOR_POSN
F1AF 1E 3539 PUSH DS ; ESTABLISH SEGMENT
F1B0 0T 3540 POP ES ; ADDRESSING
F1B1 33C0 3541 XOR AX,AX
F1B3 F3 3542 REP STOSW ; FILL WITH ZEROES
F1B4 AB

```

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

```

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    XOR    AL,AL                         ; EXCEPT 640X200
F1B8 803E490006    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    MI4:                                     ; OUTPUT THE CORRECT VALUE TO 3D9 PORT
F1C2 A26600        3553    OUT    DX,AL                         ; SAVE THE VALUE FOR FUTURE USE
3554                3554    MOV    CRT_PALETTE,AL             ; -----
3555                3555    RET                                    ; -----
3556 ;----- NORMAL RETURN FROM ALL VIDEO RETURNS
3557
F1C5 5F            3558    VIDEO_RETURN:
3559                3559    POP    DI                             ; -----
F1C5 5E            3560    POP    SI                             ; -----
F1C7 5B            3561    POP    BX                             ; -----
F1CA                3562    MI51                                 ; VIDEO_RETURN_C
F1C8 59            3563    POP    CX                             ; -----
F1C9 5A            3564    POP    DX                             ; -----
F1CA 1F            3565    POP    DS                             ; RECOVER SEGMENTS
F1CB 07            3566    POP    ES                             ; -----
F1CC CF            3567    POP    IRET                         ; ALL DONE
3568                3568    SET_MODE                             ; -----
3569 ;----- -----
3570 ; SET_CTYPE
3571 ; THIS ROUTINE SETS THE CURSOR VALUE
3572 ; INPUT
3573 ; (CX) HAS CURSOR VALUE CH-START LINE, CL-STOP LINE
3574 ; OUTPUT
3575 ; NONE
3576 ; -----
3577 SET_CTYPE        PROC    NEAR
3578                3578    MOV    AH,10                         ; 6845 REGISTER FOR CURSOR SET
3579                3579    MOV    CURSOR_MODE,CX             ; -----
3580                3580    CALL   M16                             ; -----
3581                3581    JMP    VIDEO_RETURN             ; -----
3582                3582    RET                                    ; -----
3583 ;----- THIS ROUTINE OUTPUTS THE CX REGISTER TO THE 6845 REGS NAMED IN AH
3584 M16:
3585
F1D0 8B166300    3586    MOV    DX,ADDR_6845             ; ADDRESS REGISTER
F1DC 8AC4           3587    MOV    AL,AH                     ; GET VALUE
3588                3588    OUT    DX,AL                         ; REGISTER SET
F1DE EE            3589    INC    DX                             ; DATA REGISTER
F1E0 8AC5           3590    MOV    AL,CH                     ; -----
F1E2 EE            3591    OUT    DX,AL                         ; DATA
F1E3 FA            3592    DEC    DX                             ; -----
F1E4 8AC4           3593    MOV    AL,AH                     ; -----
F1E6 FEC0           3594    INC    AL                             ; POINT TO OTHER DATA REGISTER
F1E8 EE            3595    OUT    DX,AL                         ; SET FOR SECOND REGISTER
F1E9 42            3596    INC    DX                             ; -----
F1EA 8AC1           3597    MOV    AL,CL                     ; SECOND DATA VALUE
F1EC EE            3598    OUT    DX,AL                         ; -----
F1ED C3            3599    RET                                    ; ALL DONE
3600                3600    SET_CTYPE                             ; -----
3601 ;----- -----
3602 ; SET_CPOS
3603 ; THIS ROUTINE SETS THE CURRENT CURSOR
3604 ; POSITION TO THE NEW X-Y VALUES PASSED
3605 ; INPUT
3606 ; DX - ROW,COLUMN OF NEW CURSOR
3607 ; BH - DISPLAY PAGE OF CURSOR
3608 ; OUTPUT
3609 ; CURSOR IS SET AT 6845 IF DISPLAY PAGE
3610 ; IS CURRENT DISPLAY
3611 ; -----
3612 SET_CPOS        PROC    NEAR
3613                3613    MOV    CL,BH                         ; ESTABLISH LOOP COUNT
3614                3614    XOR    CH,CH                         ; -----
3615                3615    SAL    CX,1                             ; WORD OFFSET
3616                3616    MOV    SI,CX                         ; USE INDEX REGISTER
3617                3617    MOV    [SI+OFFSET_CURSOR_POSN],DX ; SAVE THE POINTER
3618                3618    CMP    ACTIVE_PAGE,BH             ; -----
3619                3619    JNZ    M17                             ; SET_CPOS_RETURN
3620                3620    MOV    AX,DX                         ; GET_ROW/COLUMN TO AX
3621                3621    CALL   M18                             ; CURSOR_SET
3622 M17:
3623                3623    JMP    VIDEO_RETURN             ; SET_CPOS_RETURN
3624 SET_CPOS        ENDP
3625                3625    RET                                    ; -----
3626 ;----- SET CURSOR POSITION, AX HAS ROW/COLUMN FOR CURSOR
3627
3628 M18            PROC    NEAR
3629                3629    CALL   POSITION                     ; DETERMINE LOCATION IN REGEN BUFFER
3630                3630    MOV    CX,AX                         ; -----
3631                3631    ADD    CX,CRT_START             ; ADD IN THE START ADDR FOR THIS PAGE
3632                3632    SAR    CX,1                             ; DIVIDE BY 2 FOR CHAR ONLY COUNT
3633                3633    MOV    AH,14                         ; REGISTER NUMBER FOR CURSOR
3634                3634    CALL   M16                             ; OUTPUT THE VALUE TO THE 6845
3635                3635    RET                                    ; -----
3636 M18            3636    ENDP

```

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

```

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 8BC8 3646 ACT_DISP_PAGE PROC NEAR
F217 A26200 3647 MOV ACTIVE_PAGE,AL ; SAVE ACTIVE PAGE VALUE
F21A BB0E4C00 3648 MOV CX,CRT_LEN ; GET SAVED LENGTH OF REGEN BUFFER
F21E 8000 3649 CDE ; CONVERT WORD TO WORD
F21F 50 3650 PUSH AX ; SAVE PAGE VALUE
F220 FTE1 3651 MUL CX ; DISPLAY PAGE TIMES REGEN LENGTH
F222 A34E00 3652 MOV CRT_START,AX ; SAVE START ADDRESS FOR
3653 ; LATER REQUIREMENTS
3654 MOV CX,AX ; START ADDRESS TO CX
3655 SAR CX,1 ; DIVIDE BY 2 FOR 6845 HANDLING
3656 ADD AX,12 ; 16 BYTES = 12
3657 CALL M16 ; 6845 REGISTER FOR START ADDRESS
F225 8BC8 3658 POP BX ; RECOVER PAGE VALUE
F227 8AF9 3659 SAL BX,1 ; *2 FOR WORD OFFSET
F229 B40C 3660 MOV AX,[BX + OFFSET_CURSOR_POSN] ; GET CURSOR FOR THIS PAGE
F22B E6AAFF 3661 CALL M16 ; SET THE CURSOR POSITION
F22E 5B 3662 JMP SHORT_VIDEO_RETURN
F231 D1E3 3663 ACT_DISP_PAGE ENDP
F231 BB4750 3664 ;-----+
F234 E6CFFF 3665 ; READ_CURSOR
F237 EB8C 3666 ;-----+
F239 8ADF 3667 ; THIS ROUTINE READS THE CURRENT CURSOR VALUE FROM THE
F239 32FF 3668 ; 6845, FORMATS IT, AND SENDS IT BACK TO THE CALLER
F23D DIE3 3669 ;-----+
F23E BB5750 3670 ;-----+
F242 BB0E6000 3671 ;-----+
F246 5F 3672 ;-----+
F247 5E 3673 ;-----+
F248 5B 3674 ;-----+
F249 58 3675 ;-----+
F24A 58 3676 ;-----+
F24B 1F 3677 ;-----+
F24C 07 3678 ;-----+
F24D CF 3679 ;-----+
F24E 8B166300 3679 READ_CURSOR PROC NEAR
F252 B3C205 3680 MOV BL,BH ;-----+
F255 A06600 3681 XOR BH,BH ;-----+
F258 0AFF 3682 POP BX ;-----+
F25A 750E 3683 POP AX ;-----+
3684 POP AX ;-----+
3685 POP DS ;-----+
3686 POP ES ;-----+
3687 INT3 ;-----+
3688 READ_CURSOR ENDP
3689 ;-----+
3690 ; SET COLOR
3691 ;-----+
3692 ; THIS ROUTINE WILL ESTABLISH THE BACKGROUND COLOR, THE OVERSCAN
3693 ; COLOR, AND THE FOREGROUND COLOR SET FOR MEDIUM RESOLUTION
3694 ;-----+
3695 ;-----+
3696 ;-----+
3697 ;-----+
3698 ;-----+
3699 ;-----+
3700 ;-----+
3701 ;-----+
3702 ;-----+
3703 ;-----+
3704 ;-----+
F25C 24E0 3705 ;-----+
F25E 80E31F 3706 SET_COLOR PROC NEAR
F261 0AC3 3707 MOV DX,ADDR_6845 ; I/O PORT FOR PALETTE
F263 EE 3708 ADD DX,5 ; OVERSCAN PORT
F264 A26600 3709 MOV AL,CRT_PALETTE ; GET THE CURRENT PALETTE VALUE
F267 E95BFF 3710 OR BH,BH ;-----+
3711 JNZ M20 ;-----+
3712 ;-----+
3713 ;-----+ HANDLE COLOR 0 BY SETTING THE BACKGROUND COLOR
3714 ;-----+
3715 AND AL,0E0H ;-----+
3716 AND BL,01FH ;-----+
3717 OR AL,BL ;-----+
3718 M19: OUT DX,AL ;-----+
3719 MOV CRT_PALETTE,AL ;-----+
3720 JMP VIDEO_RETURN ;-----+
3721 ;-----+
3722 ;-----+ HANDLE COLOR 1 BY SELECTING THE PALETTE TO BE USED
3723 ;-----+
M20: 3724 ;-----+
3725 AND AL,0DFH ;-----+
3726 SHR BL,1 ;-----+
3727 JNC M19 ;-----+
3728 OR AL,20H ;-----+
3729 MOV M19 ;-----+
3730 JMP M20 ;-----+
3731 ;-----+
SET_COLOR ENDP
3732 ;-----+
3733 ;-----+ VIDEO STATE
3734 ;-----+
3735 ;-----+ RETURNS THE CURRENT VIDEO STATE IN AX
3736 ;-----+
3737 ;-----+ AH = NUMBER OF COLUMNS ON THE SCREEN
3738 ;-----+
3739 ;-----+ BH = CURRENT ACTIVE PAGE
3740 ;-----+
3741 ;-----+ VIDEO_STATE PROC NEAR
3742 ;-----+
3743 ;-----+ AH,BYTE PTR CRT_COLS ; GET NUMBER OF COLUMNS
3744 ;-----+ AL,CURRENT_MODE ; CURRENT MODE
3745 ;-----+ BH,ACTIVE_PAGE ; GET CURRENT ACTIVE PAGE
3746 ;-----+ POP DI ;-----+
3747 ;-----+ POP SI ;-----+
3748 ;-----+ POP CX ;-----+
3749 ;-----+ POP M15 ;-----+
3750 ;-----+ JMP VIDEO_RETURN ;-----+
3751 ;-----+
VIDEO_STATE ENDP

```

LOC OBJECT

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

```

3748 ;-----  

3749 ; POSITION  

3750 ; THIS SERVICE ROUTINE CALCULATES THE REGEN  

3751 ; BUFFER ADDRESS OF A CHARACTER IN THE ALPHA MODE  

3752 ; INPUT  

3753 ; AX = ROW, COLUMN POSITION  

3754 ; OUTPUT  

3755 ; AX = OFFSET OF CHAR POSITION IN REGEN BUFFER  

3756 ;-----  

F285 53 3757 POSITION PROC NEAR  

F286 BBD8 3758 PUSH BX ; SAVE REGISTER  

F287 84A 3759 MOV BX,AX  

F28A F6264A00 3760 MOV AL,AH ; ROWS TO AL  

3761 MUL BX,CH PTR CRT_COLS ; DETERMINE BYTES TO ROW  

F28E 32FF 3762 XOR BH,BH  

F290 03C3 3763 ADD AX,BX ; ADD IN COLUMN VALUE  

F292 D1E0 3764 SAL AX,1 ; * 2 FOR ATTRIBUTE BYTES  

F294 5B 3765 POP BX  

F295 C3 3766 RET  

F2A0 54 3767 POSITION ENDP  

F2A1 55 3768 ;-----  

3769 ; SCROLL_UP  

3770 ; THIS ROUTINE MOVES A BLOCK OF CHARACTERS UP  

3771 ; ON THE SCREEN  

3772 ; INPUT  

3773 ; (AH) = CURRENT CRT MODE  

3774 ; (AL) = NUMBER OF ROWS TO SCROLL  

3775 ; (CX) = ROW/COLUMN OF UPPER LEFT CORNER  

3776 ; (DX) = ROW/COLUMN OF LOWER RIGHT CORNER  

3777 ; (BH) = ATTRIBUTE TO BE USED ON BLANKED LINE  

3778 ; (DS) = DATA SEGMENT  

3779 ; (ES) = REGEN BUFFER SEGMENT  

3780 ; OUTPUT  

3781 ; NONE -- THE REGEN BUFFER IS MODIFIED  

3782 ;-----  

F296 8AD8 3783 ASSUME CS:CODE,DS:DATA,ES:DATA  

F298 80FC04 3784 SCROLL_UP PROC NEAR  

3785 MOV BL,AL ; SAVE LINE COUNT IN BL  

3786 CMP AH,4 ; TEST FOR GRAPHICS MODE  

3787 JC NI ; HANDLE SEPARATELY  

F29D 80FC07 3788 CMP AH,7 ; TEST FOR BW CARD  

F2A0 T403 3789 JE NI  

F2A2 E9F001 3790 JMP GRAPHICS_UP  

F2A3 54 3791 NI: ;-----  

3792 PUSH BX ; UP_CONTINUE  

3793 MOV AX,CX ; SAVE_FILL_ATTRIBUTE IN BH  

3794 CALL SCROLL_POSITION ; UPPER LEFT POSITION  

3795 JZ NT ; DO SETUP FOR SCROLL  

F2A8 D3F0 3796 ADD SI,AX ; BLANK_FIELD  

F2A9 8AE6 3797 MOV AH,DH ; FROM ADDRESS  

3798 SUB AI,BL ; # ROWS IN BLOCK  

F2B1 2A3 3799 JE NI ; # ROWS TO BE MOVED  

F2B3 3800 SUB AL,BL ; ROW_LOOP  

F2B3 E67200 3801 CALL N10 ; MOVE ONE ROW  

F2B6 03F5 3802 ADD SI,BP ; POINT TO NEXT LINE IN BLOCK  

F2B8 03FD 3803 ADD DI,BP ; COUNT OF LINES TO MOVE  

F2B9 FEC0 3804 DEC AH ; ROW_LOOP  

F2B9 75F5 3805 JNZ N2 ; CLEAR_ENTRY  

F2B8 3806 N3: POP AX ; RECOVER ATTRIBUTE IN AH  

F2BF B020 3807 MOV AL,1 ; FILL WITH BLANKS  

F2C1 1E6D00 3808 N4: ;-----  

3809 CALL N11 ; CLEAR_FIELD  

F2C4 03FD 3810 ADD DI,BP ; CLEAR THIS ROW  

F2C6 FECB 3811 DEC BL ; POINT TO NEXT LINE  

F2C8 75F7 3812 JNZ N4 ; COUNTER OF LINES TO SCROLL  

F2CA 3813 N5: ;-----  

3814 CALL DDS ; CLEAR_LOOP  

F2D0 A03E490007 3815 CMP CRT_MODE,7 ; SCROLL_END  

3816 JE N6 ; IS THIS THE BLACK AND WHITE CARD  

F2D4 A06500 3817 MOV AL,CRT_MODE_SET ; IF SO, SKIP THE MODE RESET  

F2D7 BAD803 3818 MOV DX,03DBH ; GET THE VALUE OF THE MODE SET  

F2DA EE 3819 OUT DX,AL ; ALWAYS SET COLOR CARD PORT  

F2DB 3820 N6: ;-----  

3821 PUSH DS ; VIDEO_RET_HERE  

3822 N7: ;-----  

3823 JMP VIDEO_RETURN ; BLANK_FIELD  

F2DE 8ADE 3824 MOV BL,DH ; GET ROW COUNT  

F2E0 EBDC 3825 JMP N3 ; GO CLEAR THAT AREA  

3826 SCROLL_UP ENDP  

3827 ;-----  

3828 HANDLE COMMON SCROLL SET UP HERE  

3829 SCROLL_POSITION PROC NEAR  

3830 CMP CRT_MODE,2 ; TEST FOR SPECIAL CASE HERE  

3831 JB N9 ; HAVE TO HANDLE 80X25 SEPARATELY  

3832 CMP CRT_MODE,3  

3833 JA N9  

3834 ;-----  

3835 80X25 COLOR CARD SCROLL  

3836 ;-----  

F2F0 52 3837 PUSH DX ; GUARANTEED TO BE COLOR CARD HERE  

F2F4 BADA03 3838 MOV DX,3DAH  

3839 PUSH AX  

F2F5 50 3840 N8: ;-----  

3841 IN AL,DX ; WAIT_DISP_ENABLE  

F2F6 A808 3842 TEST AL,8 ; GET_PORT  

F2F8 74FB 3843 JZ N8 ; WAIT FOR VERTICAL RETRACE  

F2FA B025 3844 MOV AL,25H ; WAIT_DISP_ENABLE  

F2FB 03D8 3845 MOV DL,0DBH ; DX=2D8  

F2FE EC 3846 OUT DL,DL ; TURN OFF VIDEO  

F2F8 58 3847 POP AX ; DURING VERTICAL RETRACE  

F300 5A 3848 POP DX  

F301 ;-----  

3849 N9: ;-----  

3850 CALL POSITION ; CONVERT TO REGEN POINTER  

3851 ADD AX,CRT_START ; OFFSET OF ACTIVE PAGE  

F308 8BF8 3852 MOV DL,AX ; TO ADDRESS FOR SCROLL  

F30A 8BF0 3853 MOV SI,AL ; FROM ADDRESS FOR SCROLL  

F30C 2BD1 3854 SUB DX,CX ; DX = # ROWS, #COLS IN BLOCK  

F30E FEC6 3855 INC DH  

F310 F5C2 3856 INC DL ; INCREMENT FOR 0 ORIGIN  

F311 80ED 3857 XOR CH,CH ; SET HIGH BYTE OF COUNT TO ZERO  

F314 B82E4A00 3858 MOV BP,CRT_COLS ; GET NUMBER OF COLUMNS IN DISPLAY  

F318 03ED 3859 ADD BP,BP ; TIMES 2 FOR ATTRIBUTE BYTE  

F31A 8AC3 3860 MOV AL,BL ; GET LINE COUNT  

F31C F6264A00 3861 MUL BYTE PTR CRT_COLS ; DETERMINE OFFSET TO FROM ADDRESS  

F320 03C0 3862 ADD AX,AX ; * 2 FOR ATTRIBUTE BYTE  

F322 06 3863 PUSH ES ; ESTABLISH ADDRESSING TO REGEN BUFFER

```

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

```

F323 IF      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 8ACA   3871  N10   PROC   NEAR
3872     MOV   CL,DL
3873     PUSH  SI
3874     PUSH  DI
3875     REP   MOVSW
3876     POP   DI
3877     POP   SI
3878     RET
3879 N10   ENDP
3880
3881 ;----- CLEAR_ROW
3882
F329 8ACA   3883  N11   PROC   NEAR
3884     MOV   CL,DL
3885     PUSH  DI
3886     REP   STOSW
3887     POP   DI
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
3906 SCROLL_DOWN PROC  NEAR
3907 STD
3908 MOV BL,AL
3909 CMP AH,?
3910 JC N12
3911 CMP AH,7
3912 JE N12
3913 JMP GRAPHICS_DOWN
3914 N12:
3915 PUSH BX
3916 MOV AX,DX
3917 CALL SCROLL_POSITION
3918 JZ N16
3919 SUB SI,AX
3920 MOV AH,DH
3921 SUB AH,BL
3922 N13:
3923 CALL N10
3924 SUB SI,BP
3925 SUB DI,BP
3926 DEC AH
3927 JNZ N13
3928 N14:
3929 POP AX
3930 MOV AL,' '
3931 N15:
3932 CALL N11
3933 SD DI,BP
3934 DEC BL
3935 JNZ N15
3936 JMP N5
3937 N16:
3938 MOV BL,DH
3939 JMP N14
3940 SCROLL_DOWN ENDP

```

```

3941 ;----- READ_AC_CURRENT
3942 ;----- THIS ROUTINE READS THE ATTRIBUTE AND CHARACTER
3943 ;----- AT THE CURRENT CURSOR POSITION AND RETURNS THEM
3944 ;----- TO THE CALLER
3945 ;----- INPUT
3946 ;----- (AH) = CURRENT CRT MODE
3947 ;----- (BH) = DISPLAY PAGE ( ALPHA MODES ONLY )
3948 ;----- (DS) = DATA SEGMENT
3949 ;----- (ES) = REGEN SEGMENT
3950 ;----- OUTPUT
3951 ;----- (AL) = CHAR READ
3952 ;----- (AH) = ATTRIBUTE READ
3953 ;----- :
3954 ;----- :
3955 ASSUME CS:CODE,DS:DATA,ES:DATA
F374 80FC04 READ_AC_CURRENT PROC NEAR
3956 CMP AH,4 ; IS THIS GRAPHICS
F377 T208 JC P1
F379 80FC07 3958 CMP AH,7 ; IS THIS BW CARD
F37C T403 3960 JE P1
F37E E9A802 3961 JMP GRAPHICS_READ
F381 3962 PI:
F383 E81A00 3963 CALL FIND_POSITION ; READ_AC_CONTINUE
F384 8BF3 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
F388 83C206 3969 ADD DX,6 ; POINT AT STATUS PORT
F38D 06 3970 PUSH ES
F38E IF 3971 POP DS ; GET SEGMENT FOR QUICK ACCESS
F38F EC 3972 P2: ; WAIT FOR RETRACE LOW
F390 A801 3973 IN AL,DX ; GET STATUS
F392 75FB 3974 TEST AL,1 ; IS IT LOW
3975 JNZ P2 ; WAIT UNTIL IT IS
F394 FA 3976 CLI ; NO MORE INTERRUPTS
F395 3977 P3: ; WAIT FOR RETRACE HIGH
F395 EC 3978 IN AL,DX ; GET STATUS
F396 A801 3979 TEST AL,1 ; IS IT HIGH
F398 74FB 3980 JZ P3 ; WAIT UNTIL IT IS
F399 AD 3981 LODSW ; GET THE CHAR/ATTR
F39B E927FE 3982 JMP VIDEO_RETURN
3983 READ_AC_CURRENT ENDP
3984
F39E 8ACF 3985 FIND_POSITION PROC NEAR
3986 MOV CL,BH ; DISPLAY PAGE TO CX
F3A0 32ED 3987 XOR CH,CH
F3A1 8001 3988 MOV SI,CX ; MOVE TO SI FOR INDEX
F3A4 D1E6 3989 SAL P1,1 ; *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: ; PAGE_LOOP
3994 ADD BX,CRT_LEN ; LENGTH OF BUFFER
F3B1 E2FA 3995 LOOP P4 ; NO_PAGE
F3B3 E8CFE 3996 P5: ; DETERMINE LOCATION IN REGEN
F3B6 03D8 3997 CALL POSITION
F3B8 C3 3998 ADD BX,AX ; ADD TO START OF REGEN
3999 RET
4000 FIND_POSITION ENDP
4001 ;----- WRITE_AC_CURRENT
4002 ;----- THIS ROUTINE WRITES THE ATTRIBUTE
4003 ;----- AND CHARACTER AT THE CURRENT CURSOR
4004 ;----- POSITION
4005 ;----- INPUT
4006 ;----- (AH) = CURRENT CRT MODE
4007 ;----- (BH) = DISPLAY PAGE
4008 ;----- (CX) = COUNT OF CHARACTERS TO WRITE
4009 ;----- (AL) = CHAR TO WRITE
4010 ;----- (BL) = ATTRIBUTE OF CHAR TO WRITE
4011 ;----- (DS) = DATA SEGMENT
4012 ;----- (ES) = REGEN SEGMENT
4013 ;----- OUTPUT
4014 ;----- NONE
4015 ;----- :
4016 ;----- :
F3B9 80FC04 4017 WRITE_AC_CURRENT PROC NEAR
4018 CMP AH,4 ; IS THIS GRAPHICS
F3BC T208 4019 JC P6
F3BE 80FC07 4020 CMP AH,7 ; IS THIS BW CARD
F3C1 T403 4021 JE P6
F3C3 E9B201 4022 JMP GRAPHICS_WRITE
F3C6 3923 P6: ; WRITE_AC_CONTINUE
4024 MOV AH,BL ; GET ATTRIBUTE TO AH
F3C8 50 4025 PUSH AX ; SAVE ON STACK
F3C9 51 4026 PUSH CX ; SAVE WRITE COUNT
F3CA E8D1FF 4027 CALL FIND_POSITION
F3CD 88FB 4028 MOV DI,BX ; ADDRESS TO DI REGISTER
F3CF 59 4029 POP CX ; WRITE COUNT
F3D0 5B 4030 POP BX ; CHARACTER IN BX REG
F3D1 P7: 4031 ; WRITE_LOOP
4032 CLI ; NO MORE INTERRUPTS
4033 ;----- WAIT FOR HORIZONTAL RETRACE
4034 F3D1 8B166300 4035 MOV DX,ADDR_6845 ; GET BASE ADDRESS
F3D3 83C206 4036 ADD DX,6 ; POINT AT STATUS PORT
F3D8 3937 P8: ; GET STATUS
4038 IN AL,DX ; IS IT LOW
F3D9 A801 4039 TEST AL,1 ; WAIT UNTIL IT IS
F3DB 75FB 4040 JNZ P8 ; NO MORE INTERRUPTS
F3DD FA 4041 CLI ; GET STATUS
F3DE 4042 P9: ; IS IT HIGH
F3DF A801 4043 IN AL,DX ; RECOVER THE CHAR/ATTR
F3E1 74FB 4044 TEST AL,1 ; PUT THE CHAR/ATTR
F3E3 8BC3 4045 JZ P9 ; INTERRUPTS BACK ON
F3E4 FB 4046 MOV AX,BX ; AS MANY TIMES AS REQUESTED
F3E5 AB 4047 STOSW
F3E6 FB 4048 STI
F3E7 E2E8 4049 LOOP P7
F3E9 E9D9FD 4050 JMP VIDEO_RETURN
4051 WRITE_AC_CURRENT ENDP

```

LOC

OBJECT

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

```

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 ; (DI) = CHAR TO WRITE
4062 ; (DS) = DATA SEGMENT
4063 ; (ES) = REGEN SEGMENT
4064 ; OUTPUT
4065 ; NONE
4066 ;-----+
4067 WRITE_C_CURRENT PROC NEAR
4068 CMP AH,4 : IS THIS GRAPHICS
4069 JC P10
4070 CMP AH,7 : IS THIS BW CARD
4071 JE P10
4072 JMP GRAPHICS_WRITE
4073 P10:
4074 PUSH AX : SAVE ON STACK
4075 PUSH CX : SAVE WRITE COUNT
4076 CALL FIND_POSITION
4077 MOV DI,BX : ADDRESS TO DI
4078 POP CX : WRITE COUNT
4079 POP BX : BE HAS CHAR TO WRITE
4080 P11:
4081 :----- WAIT FOR HORIZONTAL RETRACE
4082
4083
4084 MOV DX,ADDR_6845 : GET BASE ADDRESS
4085 ADD DX,6 : POINT AT STATUS PORT
4086 P12:
4087 IN AL,DX : GET STATUS
4088 TEST AL,1 : IS IT LOW
4089 JNZ P12 : WAIT UNTIL IT IS
4090 CL1 : NO MORE INTERRUPTS
4091 P13:
4092 IN AL,DX : GET STATUS
4093 TEST AL,1 : IS IT HIGH
4094 JZ P13 : WAIT UNTIL IT IS
4095 MOV AL,BL : RECOVER CHAR
4096 STOSB : PUT THE CHAR/ATTR
4097 STI : INTERRUPTS BACK ON
4098 INC DI : BUMP POINTER PAST ATTRIBUTE
4099 LOOP P11 : AS MANY TIMES AS REQUESTED
4100 JNC VIDEO_RETURN
4101 WRITE_C_CURRENT ENDP
4102 ;-----+
4103 ; READ DOT -- WRITE DOT
4104 ; THESE ROUTINES WILL WRITE A DOT, OR READ THE DOT AT
4105 ; THE INDICATED LOCATION
4106 ; ENTRY POINTS
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 ; EXIT
4115 ; EXIT
4116 ; AL = DOT VALUE READ, RIGHT JUSTIFIED, READ ONLY
4117 ;-----+
4118 ASSUME DS:[CODE],DS:DATA,ES:DATA
4119 READ_DOT PRGC NEAR
4120 CALL R3 : DETERMINE BYTE POSITION OF DOT
4121 MOV AL,ES:[SI] : GET THE BYTE
4122 AND AL,AH : MASK OFF THE OTHER BITS IN THE BYTE
4123 SHL AL,CL : LEFT JUSTIFY THE VALUE
4124 MOV CL,DH : GET NUMBER OF BITS IN RESULT
4125 ROR AL,CL : RIGHT JUSTIFY THE RESULT
4126 JMP VIDEO_RETURN : RETURN FROM VIDEO IO
4127 READ_DOT ENDP
4128 ;-----+
4129 WRITE_DOT PRGC 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 : RECEIVE XOR FLAG
4137 TEST BL,80H : YES, XOR THE DOT
4138 JNZ R2 : SET THE MASK TO REMOVE THE
4139 NOT AH : INDICATED BITS
4140 AND CL,AH : OR IN THE NEW VALUE OF THOSE BITS
4141 OR AL,CL : FINISH DOT
4142 R1: RESTORE THE BYTE IN MEMORY
4143 MOV ES:[SI],AL : RETURN FROM VIDEO IO
4144 POP AX : XOR DOT
4145 JMP VIDEO_RETURN : EXCLUSIVE OR THE DOTS
4146 R2: : FINISH UP THE WRITING
4147 XOR AL,CL
4148 JMP R1
4149 WRITE_DOT ENDP

```

```

4150 ;----- DETERMINE ROW COLUMN VALUE IN GRAPHICS MODE
4151 ; THIS SUBROUTINE DETERMINES THE REGEN BYTE LOCATION
4152 ; OF THE INDICATED ROW COLUMN VALUE IN GRAPHICS MODE.
4153 ; ENTRY : DX = ROW VALUE (0-199)
4154 ; CX = COLUMN VALUE (0-639)
4155 ; EXIT ---
4156 ; SI = OFFSET INTO REGEN BUFFER FOR BYTE OF INTEREST
4157 ; AH = MASK TO STRIP OFF THE BITS OF INTEREST
4158 ; CL = BITS TO SHIFT TO RIGHT JUSTIFY THE MASK IN AH
4159 ; DH = BITS IN RESULT
4160 ;----- DETERMINE 1ST BYTE IN INDICATED ROW BY MULTIPLYING ROW VALUE BY 40
4161 ;----- (LOW BIT OF ROW DETERMINES EVEN/ODD, 80 BYTES/ROW)
4162 R3 PROC NEAR
4163 PUSH BX ; SAVE BX DURING OPERATION
4164 PUSH AX ; WILL SAVE AL DURING OPERATION
4165 ;----- DETERMINE 1ST BYTE IN INDICATED ROW BY MULTIPLYING ROW VALUE BY 40
4166 ;----- (LOW BIT OF ROW DETERMINES EVEN/ODD, 80 BYTES/ROW)
4167 ;----- DETERMINE 1ST BYTE IN INDICATED ROW BY MULTIPLYING ROW VALUE BY 40
4168 ;----- (LOW BIT OF ROW DETERMINES EVEN/ODD, 80 BYTES/ROW)
4169 MOV AL,40
4170 PUSH DX ; SAVE ROW VALUE
4171 AND DL,0FEH ; STRIP OFF ODD/EVEN BIT
4172 MUL DL ; AX HAS ADDRESS OF 1ST BYTE
4173 ;----- OF INDICATED ROW
4174 POP DX ; RECOVER IT
4175 TEST DL,1 ; TEST FOR EVEN/ODD
4176 JZ R4 ; JUMP IF EVEN ROW
4177 ADD AX,2000H ; OFFSET TO LOCATION OF ODD ROWS
4178 ;----- EVEN ROW
4179 MOV SI,AX ; MOVE POINTER TO SI
4180 POP AX ; RECOVER AL VALUE
4181 MOV DX,CX ; COLUMN VALUE TO DX
4182 ;----- DETERMINE GRAPHICS MODE CURRENTLY IN EFFECT
4183 ;----- DETERMINE GRAPHICS MODE CURRENTLY IN EFFECT
4184 ;----- DETERMINE GRAPHICS MODE CURRENTLY IN EFFECT
4185 ;----- SET UP THE REGISTERS ACCORDING TO THE MODE
4186 ; CH = MASK FOR LOW OF COLUMN ADDRESS ( 7/3 FOR HIGH/MED RES )
4187 ; CL = # OF ADDRESS BITS IN COLUMN VALUE ( 3/2 FOR H/M )
4188 ; BL = MASK TO SELECT BITS FROM POINTEED BYTE ( 80H/COH FOR H/M )
4189 ; BH = NUMBER OF SIGNAL BITS IN POINTEED BYTE ( 1/2 FOR H/M )
4190 ;----- DETERMINE GRAPHICS MODE CURRENTLY IN EFFECT
4191 ;----- DETERMINE GRAPHICS MODE CURRENTLY IN EFFECT
4192 ;----- DETERMINE GRAPHICS MODE CURRENTLY IN EFFECT
4193 MOV BX,200H
4194 MOV DX,1302H ; SET PARM FOR MED RES
4195 CMP CX,MODE,6
4196 JC R5 ; HANDLE IF MED ARES
4197 MOV BX,180H
4198 MOV CX,703H ; SET PARM FOR HIGH RES
4199 ;----- DETERMINE BIT OFFSET IN BYTE FROM COLUMN MASK
4200 ;----- DETERMINE BIT OFFSET IN BYTE FROM COLUMN MASK
4201 R5: AND CH,DL ; ADDRESS OF PEL WITHIN BYTE TO CH
4202 ;----- DETERMINE BIT OFFSET FOR THIS LOCATION IN COLUMN
4203 ;----- DETERMINE BIT OFFSET FOR THIS LOCATION IN COLUMN
4204 ;----- DETERMINE BIT OFFSET FOR THIS LOCATION IN COLUMN
4205 ;----- DETERMINE BIT OFFSET FOR THIS LOCATION IN COLUMN
4206 ;----- DETERMINE BIT OFFSET FOR THIS LOCATION IN COLUMN
4207 SHR DX,CL ; SHIFT BY CORRECT AMOUNT
4208 ADD SI,DX ; INCREMENT THE POINTER
4209 MOV DH,BH ; GET THE # OF BITS IN RESULT TO DH
4210 ;----- MULTIPLY BX (VALID BITS IN BYTE) BY CH (BIT OFFSET)
4211 ;----- MULTIPLY BX (VALID BITS IN BYTE) BY CH (BIT OFFSET)
4212 ;----- MULTIPLY BX (VALID BITS IN BYTE) BY CH (BIT OFFSET)
4213 SUB CL,CL ; ZERO INTO STORAGE LOCATION
4214 R6: ROR AL,1 ; LEFT JUSTIFY THE VALUE
4215 ;----- IN AL (FOR WRITE)
4216 ;----- ADD IN THE BIT OFFSET VALUE
4217 ADD CL,CH ; LOW CONTROL
4218 DEC BH ; ONE EXIT - CL HAS SHIFT COUNT
4219 JNZ R6 ; TO RESTORE BITS
4220 ;----- GET MASK TO AH
4221 MOV AH,BL ; GET MASK TO AH
4222 SHR AH,CL ; MOVE THE MASK TO CORRECT LOCATION
4223 POP BX ; RECOVER REG
4224 RET ; RETURN WITH EVERYTHING SET UP
4225 R3 ENDP ;----- SCROLL UP
4226 ;----- THIS ROUTINE SCROLLS UP THE INFORMATION ON THE CRT
4227 ;----- ENTRY : CH,CL = UPPER LEFT CORNER OF REGION TO SCROLL
4228 ;----- DH,DL = LOWER RIGHT CORNER OF REGION TO SCROLL
4229 ;----- BOTH OF THE ABOVE ARE IN CHARACTER POSITIONS
4230 ;----- BH = FILL VALUE FOR BLANKED LINES
4231 ;----- AL = # LINES TO SCROLL (AL=0 MEANS BLANK THE ENTIRE
4232 ;----- FIELD)
4233 ;----- DS = DATA SEGMENT
4234 ;----- ES = REGEN SEGMENT
4235 ;----- EXIT
4236 ;----- NOTHING, THE SCREEN IS SCROLLED
4240 ;----- GRAPHICS_UP PROC NEAR
4241 ;----- ENTRY : BL,AL = SAVE LINE COUNT IN BL
4242 ;----- MOV AX,CX ; GET UPPER LEFT POSITION INTO AX REG
4243 ;----- EXIT
4244 ;----- USE CHARACTER SUBROUTINE FOR POSITIONING
4245 ;----- ADDRESS RETURNED IS Multiplied BY 2 FROM CORRECT VALUE
4247 ;----- DETERMINE SIZE OF WINDOW
4248 CALL GRAPH_POSN ;----- SAVE RESULT AS DESTINATION ADDRESS
4249 MOV DI,AX
4250 ;----- DETERMINE SIZE OF WINDOW
4251 ;----- DETERMINE CRT MODE
4252 ;----- DETERMINE CRT MODE
4253 SUB DX,CX ;----- ADJUST VALUES
4254 ADD DX,101H ;----- MULTIPLY # ROWS BY 4
4255 SAL DH,1 ;----- SINCE 8 VERT DOTS/CHAR
4256 ;----- AND EVEN/ODD ROWS
4257 SAL DH,1
4258 ;----- DETERMINE CRT MODE
4259 ;----- DETERMINE CRT MODE
4260 ;----- DETERMINE CRT MODE
4261 CMP CRT_MODE,6 ;----- TEST FOR MEDIUM RES
4262 JNC R7 ;----- FIND_SOURCE

```

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

```

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 R7: ; FIND_SOURCE
4272 PUSH ES ; GET SEGMENTS BOTH POINTING TO REGEN
F4B4 4273 POP DS
F4B5 4274 SUB CH,CH ; ZERO TO HIGH OF COUNT REG
F4B6 4275 SAL BL,I ; MULTIPLY NUMBER OF LINES BY 4
F4B9 D0E3 4276 SAL BL,I
4277 JZ R11 ; IF ZERO, THEN BLANK ENTIRE FIELD
4278 MOV AL,BL ; GET NUMBER OF LINES IN AL
F4BD BAC3 4279 MOV AH,80 ; 80 BYTES/ROW
F4C0 B0F0 4280 MUL AH ; DETERMINE OFFSET TO SOURCE
F4C1 B4E4 4281 ADD SI,DI ; SET UP SOURCE
F4C3 BAE7 4282 ADD SI,AX ; ADD IN OFFSET TO IT
F4C5 03F0 4283 MOV AH,DH ; NUMBER OF ROWS IN FIELD
F4C7 8AE6 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
F4D2 81EFB01F 4291 SUB DI,2000H-80
F4D6 FEC3 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
F4D8 8AC7 4298 MOV AL,BH ; ATTRIBUTE TO FILL WITH
F4DC 4299
R10: ; CLEAR THAT ROW
4300 CALL R18 ; POINT TO NEXT LINE
4301 SUB DI,2000H-80 ; NUMBER OF LINES TO FILL
F4DF 81EFB01F 4302 DEC BL ; CLEAR_LOOP
F4E3 FECB 4303 JNZ R10 ; EVERYTHING DONE
F4E5 75F5 4304 JMP VIDEO_RETURN
F4E7 E9DFBC
4305 R11: ; BLANK FIELD
4306 MOV BL,DH ; SET BLANK COUNT TO
4307 ADD DI,DX ; EVERYTHING IN FIELD
F4EC EBEC 4308 JMP R9 ; CLEAR THE FIELD
4309 GRAPHICS_UP ENDP
4310
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
GRAPHICS_DOWN PROC NEAR
4325 STD ; SET DIRECTION
4326 MOV BL,AL ; SAVE LINE COUNT IN BL
4327 MOV AX,DX ; GET LOWER RIGHT POSITION INTO AX REG
4328
4329 ;----- USE CHARACTER SUBROUTINE FOR POSITIONING
4330 ;----- ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE
4331
F4F3 E80F02 4332 CALL GRAPH_POSN ; SAVE RESULT AS DESTINATION ADDRESS
F4F6 8BF8 4334 MOV DI,AX
4335
4336 ;----- DETERMINE SIZE OF WINDOW
4337
F4F8 2BD1 4338 SUB DX,CX ; ADJUST VALUES
F4FA 81C20101 4339 ADD DX,10H ; MULTIPLY # ROWS BY 4
F4FE D0E6 4340 SAL DH,I ; SINCE 8 VERT DOTS/CHAR
F500 D0E6 4341 SAL DH,I ; AND EVEN/ODD ROWS
4342
4343 ;----- DETERMINE CRT MODE
4344
F502 803E490006 4345 CMP CRT_MODE,6 ; TEST FOR MEDIUM RES
F507 7305 4346 JNC R12_ ; FIND_SOURCE_DOWN
4347
4348 ;----- MEDIUM RES DOWN
4349
F509 D0E2 4350 SAL DL,I ; # COLUMNS * 2, SINCE
4351 ; 2 BYTES/CHAR (OFFSET OK)
F50B D1E7 4352 SAL DI,I ; OFFSET * 2 SINCE 2 BYTES/CHAR
4353 INC DI ; POINT TO LAST BYTE
4354
4355 ;----- DETERMINE THE SOURCE ADDRESS IN THE BUFFER
4356
F50E 06 4357 SAL AH,80 ; FIND_SOURCE_DOWN
F50F 1F 4358 R12: ; BOTH SEGMENTS TO REGEN
F510 2AED 4359 PUSH ES
F512 81CF000 4360 POP DS
4361 SUB CH,CH ; ZERO TO HIGH OF COUNT REG
F514 D0E3 4362 ADD DI,240 ; POINT TO LAST ROW OF PIXELS
F516 D0E3 4363 SAL BL,I ; MULTIPLY NUMBER OF LINES BY 4
F51A T42E 4364 SAL BL,I
4365 JZ R16 ; IF ZERO, THEN BLANK ENTIRE FIELD
F51C D0E3 4366 MOV AL,BL ; GET NUMBER OF LINES IN AL
F51E B450 4367 MOV AH,80 ; DETERMINE OFFSET TO SOURCE
F520 FB4E 4368 MUL AH ; SET UP SOURCE
F522 8BF7 4369 MOV SI,DI ; SUBTRACT THE OFFSET
F524 2BF0 4370 SUB SI,AX ; NUMBER OF ROWS IN FIELD
F526 8AE6 4371 MOV AH,DH ; DETERMINE NUMBER TO MOVE
F528 2AE3 4372 SUB AH,BL

```

```

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

4373
4374 ;---- LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD FIELDS
4375
F52A
4376 R13: CALL R17 ; ROW_LOOP_DOWN
4377 SUB SI,2000H+80 ; MOVE ONE ROW
4378 SUB DI,2000H+80 ; MOVE TO NEXT ROW
4379 DEC AH ; NUMBER OF ROWS TO MOVE
4380 JNZ R13 ; CONTINUE TILL ALL MOVED
4381
4382 ;---- FILL IN THE VACATED LINE(S)
4383
4384 R14: MOV AL,BH ; CLEAR_ENTRY_DOWN
4385 ; ATTRIBUTE TO FILL WITH
4386 MOV AL,BH ; CLEAR_LOOP_DOWN
4387 R15: CALL R18 ; CLEAR_A_ROW
4388 SUB DI,2000H+80 ; CLEAR_VACANT_LINE
4389 DEC BL ; NUMBER OF LINES TO FILL
4390 JNZ R15 ; CLEAR_LOOP_DOWN
4391 CLD ; RESET THE DIRECTION FLAG
4392 JMP VIDEO_RETURN ; EVERYTHING DONE
4393
F53B E82100
4394 R16: MOV BL,DH ; BLANK_FIELD_DOWN
4395 ADD DI,BL ; SET_BLANK_COUNT TO EVERYTHING
4396
F53B E82900
4397 JMP R14 ; CLEAR_FIELD
4398 GRAPHICS_DOWN ENDP ; CLEAR THE FIELD
4399
4400 ;---- ROUTINE TO MOVE ONE ROW OF INFORMATION
4401
F54E
4402 R17 PROC NEAR
4403 MOV CL,DL ; NUMBER OF BYTES IN THE ROW
4404 PUSH SI
4405 PUSH DI ; SAVE POINTERS
4406 REP MOVSB ; MOVE THE EVEN FIELD
4407
F54E 8ACA
4408 POP DI
4409 POP SI
4410 ADD SI,2000H ; POINT TO THE ODD FIELD
4411 PUSH SI
4412 PUSH DI ; SAVE THE POINTERS
4413 MOV CL,DL ; COUNT BACK
4414 REP MOVSB ; MOVE THE ODD FIELD
4415
F54E 5F
4416 POP DI ; POINTERS BACK
4417 RET ; RETURN TO CALLER
4418 R17 ENDP ; RETURN TO CALLER
4419
4420 ;---- CLEAR A SINGLE ROW
4421
F566
4422 R18 PROC NEAR
4423 MOV CL,DL ; NUMBER OF BYTES IN FIELD
4424 PUSH DI ; SAVE POINTER
4425 REP STOSB ; STORE THE NEW VALUE
4426
F566 81C60020
4427 ADD DI,2000H ; POINT TO ODD FIELD
4428 PUSH DI
4429 MOV CL,DL
4430 REP STOSB ; FILL THE ODD FILED
4431
F566 5E
4432 POP DI
4433 R18 ENDP ; RETURN TO CALLER
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 BL = 0, 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 ROM 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 ; ZERO TO HIGH OF CODE POINT
4469 PUSH AX ; SAVE CODE POINT VALUE
4470
4471 ;---- DETERMINE POSITION IN REGEN BUFFER TO PUT CODE POINTS
4472
F578
4473 CALL S26 ; FIND LOCATION IN REGEN BUFFER
4474 MOV DI,AX ; REGEN POINTER IN DI
4475
F578 B400
4476 ;---- DETERMINE REGION TO GET CODE POINTS FROM
4477
F578 50
4478 POP AX ; RECOVER CODE POINT
4479 CMP AL,80H ; IS IT IN SECOND HALF
4480 JAE S1 ; YES
4481
F580 58
4482
F581 3C80
4483
F583 7306

```

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

```

4481
4482 ;----- IMAGE IS IN FIRST HALF, CONTAINED IN ROM
4483
F585 BE6FA 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 IS IN SECOND HALF, IN USER RAM
4489
F58B 4490 SI: 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
F590 8E0E 4494 ASSUME DS:ABS0 ESTABLISH VECTOR ADDRESSING
F592 C5367C00 4495 LDSD PTR SI,CRT_PTR : GET THE OFFSET OF THE TABLE
F596 8CDA 4496 MOV DI,DS : GET THE SEGMENT OF THE TABLE
4497 ASSUME DS:DATA
F598 1F 4498 POP DS : RECOVER DATA SEGMENT
F599 52 4500 PUSH DX : SAVE TABLE SEGMENT ON STACK
4501
4502 ;----- DETERMINE GRAPHICS MODE IN OPERATION
4503
F59A 4504 S2: SAL AX,1 : DETERMINE_MODE
F59A DIE0 4505 SAL AX,1 : MULTIPLY CODE POINT
F59C D1E0 4506 SAL AX,1 : VALUE BY 8
F59D E1E0 4507 SAL AX,1
F5A0 03F0 4508 ADD SI,AX : SI HAS OFFSET OF DESIRED CODES
F5A2 803E490006 4509 CMP CRT_MODE,6
F5A7 1F 4510 POP DS : RECOVER TABLE POINTER SEGMENT
F5AB 722C 4511 JC S7 : TEST FOR MEDIUM RESOLUTION MODE
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 ADD DS,SI : NUMBER OF TIMES THROUGH LOOP
4519
F5AE AC 4520 LODSB : GET BYTE FROM CODE POINTS
F5AF F6C380 4521 TEST BL,80H : SHOULD WE USE THE FUNCTION
F5B2 7516 4522 JNZ S6 : TO PUT CHAR IN
4523
F5B4 AA 4523 STOSB : STORE IN REGEN BUFFER
F5B5 AC 4524 LODSB
F5B6 4525 S5: MOV ES:[DI+2000H-1],AL : STORE IN SECOND HALF
4526 ADD DI,79 : MOVE TO NEXT ROW IN REGEN
F5B7 83C74F 4527 DEC DH : DONE WITH LOOP
F5C0 FECE 4528 CALL S1
F5C0 2EC 4529 XOR SI,S4
F5C2 5E 4530 POP SI
F5C3 5F 4531 POP DI : RECOVER REGEN POINTER
F5C4 47 4532 INC DI : POINT TO NEXT CHAR POSITION
F5C5 E2E3 4533 LOOP S3 : MORE CHARS TO WRITE
F5C6 E9FBFB 4534 JMP VIDEO_RETURN
F5CA 4535 S6: XOR AL,ES:[DI] : EXCLUSIVE OR WITH CURRENT
F5CD AA 4536 STOSB : STORE THE CODE POINT
F5CE AC 4537 LODSB : AGAIN FOR ODD FIELD
F5CF 263285FF1F 4538 XOR AL,ES:[DI+2000H-1]
F5D4 EBE0 4539 JMP SS : BACK TO MAINSTREAM
4540
4541
4542 ;----- MEDIUM RESOLUTION WRITE
4543
F5D6 4544 S7: MOV DL,BL : MED_RES_WRITE
F5D6 8AD3 4545 MOV SAL DI,1 : SAVE HIGH COLOR BIT
F5D8 D1E7 4546 CALL S19 : OFFSET*2 SINCE 2 BYTES/CHAR
F5D9 E8D100 4547 ADD DS,SI : EXPAND_BL TO FULL WORD OF COLOR
F5D9 4548 S8: CALL S21 : TYPE COLOR
4549 PUSH DI : SAVE_REGEN_POINTER
F5DE 56 4550 PUSH SI : SAVE THE CODE_POINTER
F5DF B604 4551 MOV DH,4 : NUMBER OF LOOPS
4552
F5E1 4552 S9: LODSB : GET_CODE_POINT
F5E2 E8D000 4553 CALL S21 : LOAD UP ALL THE BITS
F5E5 23C3 4554 XOR AH,ES:[DI] : CONVERT THEM TO FOREGROUND
4555 AND AX,BX : COLOR (0 BACK)
4556 XOR AL,ES:[DI+1] : DO FUNCTION WITH HALF
4557 S10: XOR AL,ES:[DI+1] : AND WITH OTHER HALF
4558 TEST DL,80H : STORE FIRST BYTE
4559 JZ S10 : STORE SECOND BYTE
4560 XOR AH,ES:[DI] : GET_CODE_POINT
4561 XOR AL,ES:[DI+1]
4562 MOV ES:[DI],AH : CONVER TO COLOR
F5F6 26884501 4563 MOV ES:[DI+1],AL : AGAIN, IS THIS XOR FUNCTION
F5FA A1 4564 LODSB : NO, JUST STORE THE VALUES
F5FB 26C500 4565 CALL S21 : FUNCTION WITH FIRST HALF
F5FE 23C3 4566 AND AX,BX : AND WITH SECOND HALF
F600 F6C280 4567 TEST DI,80H
F603 T40A 4568 XOR S11
F605 2632A50020 4569 XOR AH,ES:[DI+2000H]
F60A 2632B50120 4570 XOR AL,ES:[DI+2001H]
4571 S11: XOR ES:[DI+2000H],AH : STORE IN SECOND PORTION OF BUFFER
4572 MOV ES:[DI+2000H+1],AH : POINT TO NEXT LOCATION
4573 MOV ES:[DI+2000H+1],AL
4574 ADD DI,80
F61C FECE 4575 DEC DH
F61E 75C1 4576 JNZ S9 : KEEP GOING
F620 80 4577 POP SI : RECOVER_CODE_POINTER
F621 8F 4578 POP DI : RECOVER_REGEN_POINTER
F622 47 4579 INC DI : POINT TO NEXT CHAR POSITION
F623 47 4580 INC DI
F624 E2B7 4581 LOOP S8 : MORE TO WRITE
F626 E99CFB 4582 JMP VIDEO_RETURN
4583 GRAPHICS_WRITE ENDP

```

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

```

4584 ;-----+
4585 : GRAPHICS READ :-----+
4586 :-----+
F629 4587 GRAPHICS_READ PROC NEAR
F629 E8D600 4588 CALL S26 : CONVERTED TO OFFSET IN REGEN
F62C 8BF0 : 4589 MOV SI,AX : SAVE IN SI
F62E 83EC08 4590 SUB SP,8 : ALLOCATE SPACE TO SAVE THE
4591 :-----+
F631 8BEC 4592 MOV BP,SP : READ CODE POINT
4593 :-----+
4594 :----- DETERMINE GRAPHICS MODES
4595 :-----+
F633 803E490006 4596 CMP CRT_MODE,6
F638 06 4597 PUSH ES : POINT TO REGEN SEGMENT
F639 1F 4598 POP DS : MEDIUM RESOLUTION
F63A 721A 4599 JC S13
4600 :-----+
4601 :----- HIGH RESOLUTION READ
4602 :-----+
F63C B604 4603 GET VALUES FROM REGEN BUFFER AND CONVERT TO CODE POINT
4604 :-----+
F63C B604 4605 MOV DH,4 : NUMBER OF PASSES
F63E 4606 S12: MOV AL,[SI] : GET FIRST BYTE
F640 :-----+
F640 884600 4608 MOV [BP],AL : SAVE IN STORAGE AREA
F643 45 4609 INC BP : NEXT LOCATION
F644 8A840020 4610 MOV AL,[SI+200H] : GET LOWER REGION BYTE
F648 884600 4611 MOV [BP],AL : ADJUST AND STORE
F64B 45 4612 INC BP :-----+
F64C 81C60050 4613 ADD SI,80 : POINTER INTO REGEN
F64F FECF 4614 DEC DI : LOOP CONTROL
F651 75EB 4615 JNZ S12 : DO IT SOME MORE
F653 EB1790 4616 JMP S15 : GO MATCH THE SAVED CODE POINTS
4617 :-----+
4618 :----- MEDIUM RESOLUTION READ
4619 :-----+
F656 4620 S13: :-----+
F656 DIE6 4621 SAL SI,1 : MED RES READ
F658 B604 4622 MOV DH,4 : OFFSET*2 SINCE 2 BYTES/CHAR
F65A 4623 S14: :-----+
F65A E88800 4624 CALL S23 : NUMBER OF PASSES
4625 :-----+
F65D 81C60020 4626 ADD SI,200H : GET PAIR BYTES FROM REGEN
F661 E8B100 4627 CALL S23 : INTO SINGLE SAVE
F664 81EEB01F 4628 SUB SI,200H-80 :-----+
F668 FECF 4629 DEC DH : ADJUST POINTER BACK INTO UPPER
F66A 75EE 4630 JNZ S14 :-----+
4631 :-----+
4632 :----- SAVE AREA HAS CHARACTER IN IT, MATCH IT
4633 :-----+
F66C 4634 S15: :-----+
F66C BF6EFA90 4635 MOV DI,OFFSET CRT_CHAR_GEN : FIND_CHAR
F670 08 :-----+
F671 07 4636 PUSH CS : ESTABLISH ADDRESSING
F672 83ED08 4637 POP ES :-----+
4638 SUB BP,8 : CODE POINTS IN CS
4639 :-----+
F675 8BF5 4640 MOV SI,BP : ADJUST POINTER TO BEGINNING
F676 FC :-----+
F678 B000 4641 CLD : OF SAVE AREA
F67A :-----+
F67A 16 4642 MOV AL,0 : ENSURE DIRECTION
F67B IF :-----+
F67C BA5000 4643 S16: :-----+
F67D 128 :-----+
4644 PUSH SS : CURRENT CODE POINT BEING MATCHED
F67E 128 :-----+
4645 POP DS :-----+
F67F BA5000 4646 MOV DI,128 : ESTABLISH ADDRESSING TO STACK
F680 56 :-----+
4647 S17: :-----+
4648 PUSH SI : FOR THE STRING COMPARE
F680 57 4649 PUSH DI : NUMBER TO TEST AGAINST
F681 B90800 4650 MOV CX,8 :-----+
F684 F3 4651 REPE CMPSB : NUMBER OF BYTES TO MATCH
F685 A6 :-----+
F686 5F :-----+
F687 5E 4652 POP DI : COMPARE THE 8 BYTES
F688 :-----+
4653 POP SI :-----+
F688 T4IE 4654 JZ S18 : RECOVER THE POINTERS
F68A FEC0 4655 INC AL :-----+
F68C 83C708 4656 ADD DI,8 : IF ZERO FLAG SET, THEN MATCH OCCURRED
F68F 4A 4657 DEC DX : NO MATCH, MOVE ON TO NEXT
F690 75ED 4658 JNZ S17 :-----+
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 EBD0 4665 MOV DS,AX :-----+
4666 ASSUME DS:DB50 : ESTABLISH ADDRESSING TO VECTOR
F69A C43E7C00 4667 LES DI,EXT_PTR :-----+
F69E BC00 4668 MOV AX,ES : GET POINTER
F6A0 0BC7 4669 OR AX,DI : SEE IF THE POINTER REALLY EXISTS
F6A2 7404 4670 JZ S18 : IF ALL 0, THEN DOESN'T EXIST
F6A4 B080 4671 MOV AL,128 :-----+
F6A6 EBD2 4672 JMP S18 : NO SENSE LOOKING
4673 ASSUME DS:DATA : ORIGIN FOR SECOND HALF
4674 :-----+
4675 :----- CHARACTER IS FOUND ( AL=0 IF NOT FOUND )
4676 :-----+
F6AB :-----+
4677 S18: :-----+
4678 ADD SP,8 :-----+
F6AB 83C40B 4679 JMP VIDEO_RETURN :-----+
F6AB E917FB 4680 GRAPHICS_READ ENDP :-----+

```

LOC OBJECT LINE SOURCE IBIOS FOR THE IBM PERSONAL COMPUTER XT 11/08/82

```

4681 : EXPAND_MED_COLOR
4682 : THIS ROUTINE EXPANDS THE LOW 2 BITS IN BL TO
4683 : FILL THE ENTIRE BX REGISTER
4684 : ENTRY
4685 : EXIT
4686 : BL = COLOR TO BE USED ( LOW 2 BITS )
4687 : BX = COLOR TO BE USED ( 8 REPLICATIONS OF THE
4688 : 2 COLOR BITS )
4689 :
4690 S19 PROC NEAR
4691 AND BL,3 ; ISOLATE THE COLOR BITS
4692 MOV AL,BL ; COPY TO AL
4693 PUSH CX ; SAVE REGISTER
4694 POP CX ; NUMBER OF TIMES TO DO THIS
4695 MOV CX,3
4696 S20: -----
4697 SAL AL,1 ; LEFT SHIFT BY 2
4698 SAL AL,1 ; ANOTHER COLOR VERSION INTO BL
4699 OR BL,AL ; ADDITION OF BL
4700 LOOP S20 ; FILL UPPER PORTION
4701 MOV BH,BL ; REGISTER BACK
4702 POP CX ; ALL DONE
4703 RET
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 :
4711 S21 PROC NEAR
4712 PUSH DX ; SAVE REGISTERS
4713 PUSH CX
4714 PUSH BX
4715 SUB DX,DX ; RESULT REGISTER
4716 MOV CX,I ; MASK REGISTER
4717 S22: -----
4718 MOV BX,AX ; BASE INTO TEMP
4719 AND BX,AX ; USE MASK TO EXTRACT A BIT
4720 OR DX,BX ; PUT INTO RESULT REGISTER
4721 SHL AX,I ; SHIFT BASE AND MASK BY 1
4722 SHL CX,I ; SHIFT BASE AND MASK BY 1
4723 MOV BX,AX ; BASE TO TEMP
4724 AND BX,CX ; EXTRACT THE SAME BIT
4725 OR DX,BX ; PUT INTO RESULT
4726 SHL CX,I ; SHIFT ONLY MASK NOW,
4727 JNC S22 ; MOVING TO NEXT BASE
4728 MOV AX,DX ; USE MASK BIT COMING OUT TO TERMINATE
4729 MOV AX,DX ; RESULT TO PARM REGISTER
4730 POP BX
4731 POP CX ; RECOVER REGISTERS
4732 POP DX
4733 RET
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 : S1,DS = POINTER TO REGEN AREA OF INTEREST
4744 : BX = EXPANDED FOREGROUND COLOR
4745 : BP = POINTER TO SAVE AREA
4746 : EXIT
4747 : BP IS INCREMENT AFTER SAVE
4748 :
4749 S23 PROC NEAR
4750 MOV AH,[S1] ; GET FIRST BYTE
4751 MOV AL,[S1+1] ; GET SECOND BYTE
4752 MOV CX,0C00H ; 2 BIT MASK TO TEST THE ENTRIES
4753 MOV DL,0 ; RESULT REGISTER
4754 S24: -----
4755 TEST AX,CX ; IS THIS SECTION BACKGROUND?
4756 CLC ; CLEAR CARRY IN HOPES THAT IT IS
4757 JC S25 ; IF ZERO, IT IS BACKGROUND
4758 STC ; WASN'T, SO SET CARRY
4759 S25: RCL DL,1 ; MOVE THAT BIT INTO THE RESULT
4760 SHR CX,I
4761 SHR CX,I ; MOVE THE MASK TO THE RIGHT BY 2 BITS
4762 JNC S24 ; DO IT AGAIN IF MASK DIDN'T FALL OUT
4763 MOV [BP],DL ; STORE RESULT IN SAVE AREA
4764 INC BP ; ADJUST POINTER
4765 RET ; ALL DONE
4766 ENDP
4767 -----
4768 : V4_POSITION
4769 : THIS ROUTINE TAKES THE CURSOR POSITION
4770 : CONTAINED IN 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 :
4781 S26 PROC NEAR
4782 MOV AX,CURSOR_POSN ; GET CURRENT CURSOR
4783 GRAPH_POSN LABEL NEAR
4784 PUSH BX ; SAVE REGISTER
4785 MOV BX,AX ; SAVE A COPY OF CURRENT CURSOR
4786 MOV AL,AH ; GET ROWS TO AL
4787 MUL BYTE PTR CRT_COLS ; MULTIPLY BY BYTES/COLUMN
4788 SHL AX,1 ; MULTIPLY * 4 SINCE 4 ROWS/BYTE
4789 SHL AX,1 ; ISOLATE COLUMN VALUE
4790 SUB BH,BH ; DETERMINE OFFSET
4791 ADD AX,BX ; RECOVER POINTER
4792 POP BX ; ALL DONE
4793 RET
4794 S26 ENDP

```

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

```

4795 :----- WRITE_TTY
4796 :----- THIS INTERFACE PROVIDES A TELETYPE LIKE INTERFACE TO THE VIDEO
4797 :----- CARD. THE INPUT CHARACTER IS WRITTEN TO THE CURRENT CURSOR
4798 :----- POSITION, AND THE CURSOR IS MOVED TO THE NEXT POSITION. IF THE
4799 :----- CURSOR LEAVES THE LAST COLUMN OF THE FIELD, THE COLUMN IS SET
4800 :----- TO ZERO, AND THE ROW VALUE IS INCREMENTED. IF THE ROW VALUE
4801 :----- LEAVES THE FIELD, THE CURSOR IS PLACED ON THE LAST ROW, FIRST
4802 :----- COLUMN, AND THE ENTIRE SCREEN IS SCROLLED ONE LINE WHEN
4803 :----- THE SCREEN IS FULL. THE TOP LINE IS BLANKED OUT. THE NEWLY
4804 :----- BLANCHED LINE IS READ FROM THE CURSOR POSITION ON THE PREVIOUS
4805 :----- LINE BEFORE THE SCROLL. IN CHARACTER MODE, IN GRAPHICS MODE,
4806 :----- THE 0 COLOR IS USED.
4807 :----- ENTRY
4808 :----- [AH] = CURRENT CRT MODE
4809 :----- [AL] = CHARACTER TO BE WRITTEN
4810 :----- NOTE THAT BACK SPACE, CAR RET, BELL AND LINE FEED ARE HANDLED
4811 :----- AS COMMANDS RATHER THAN AS DISPLAYABLE GRAPHICS
4812 :----- [BL] = FOREGROUND COLOR FOR CHAR WRITE IF CURRENTLY IN A
4813 :----- GRAPHICS MODE
4814 :----- EXIT
4815 :----- ALL REGISTERS SAVED
4816 :----- ALL REGISTERS SAVED
4817 :----- ASSUME CS:CODE,DS:DATA
4818 F718 4795 WRITE_TTY PROC NEAR
4819 F718 50 PUSH AX ; SAVE REGISTERS
4820 F719 50 PUSH AX ; SAVE CHAR TO WRITE
4821 F71C 4A03 MOV AH,3 ; IS IT A BACKSPACE
4822 F720 CD10 MOV BH,ACTIVE_PAGE ; GET THE CURRENT ACTIVE PAGE
4823 F722 58 INT 10H ; READ THE CURRENT CURSOR POSITION
4824 F723 3C08 POP AX ; RECOVER CHAR
4825 F724 7452
4826 F727 3C0D
4827 F729 7457
4828 F72B 3C0A
4829 F72F 7457
4830 F731 745A
4831 F733 B40A
4832 F735 B90100
4833 F738 CD10
4834 F73A FEC2
4835 F73C 3A164A00
4836 F740 7533
4837 F742 B200
4838 F744 80FE18
4839 F747 752A
4840 F748 4A04
4841 F749 4A05
4842 F74B CD10
4843 F74E 4A06
4844 F74F 4A07
4845 F750 3C04
4846 F752 7206
4847 F754 3C07
4848 F756 B700
4849 F758 7506
4850 F75A B40B
4851 F75C CD10
4852 F75E 8AFc
4853 F760 B80106
4854 F762 2BC9
4855 F764 B618
4856 F766 B164A00
4857 F768 FECA
4858 F76A B408
4859 F76C CD10
4860 F76E 8ABC
4861 F770 E952FA
4862 F772 4A08
4863 F774 4A09
4864 F776 4A0A
4865 F778 4A0B
4866 F77A 4A0C
4867 F77C 4A0D
4868 F77E 4A0E
4869 F77F 4A0F
4870 F780 EBF3
4871 F782 B200
4872 F784 EBEF
4873 F786 80FE18
4874 F788 EBBC
4875 F789 80FA00
4876 F7TC 74F7
4877 F7TE FECA
4878 F780 EBF3
4879 F782 B200
4880 F784 EBEF
4881 F786 80FE18
4882 F788 EBBC
4883 F789 80FA00
4884 F78A 74F7
4885 F78C FECA
4886 F78E B200
4887 F790 EBF3
4888 F792 B200
4889 F794 EBEF
4890 F796 80FE18
4891 F798 EBBC
4892 F799 80FA00
4893 F79A 74F7
4894 F79C FECA
4895 F79E B200
4896 F79F EBF3
4897 F7A0 80FE18
4898 F7A1 74F7
4899 F7A2 FECA
4900 F7A3 B200
4901 F7A4 EBF3
4902 F7A5 80FE18
4903 F7A6 74F7
4904 F7A7 FECA
4905 F7A8 B200
4906 F7A9 EBF3
4907 F7A9 80FE18
4908 F7AA EBBC
4909 F7AB 74F7
4910 F7AC FECA
4911 F7AD B200
4912 F7AE EBF3
4913 F7B0 80FE18
4914 F7B1 74F7
4915 F7B2 FECA
4916 F7B3 B200
4917 F7B4 EBF3
4918 F7B5 80FE18
4919 F7B6 74F7
4920 F7B7 FECA
4921 F7B8 B200
4922 F7B9 EBF3
4923 F7B9 80FA00
4924 F7B9 74F7
4925 F7B9 FECA
4926 F7B9 B200
4927 F7B9 EBF3
4928 F7B9 80FE18
4929 F7B9 74F7
4930 F7B9 FECA
4931 F7B9 B200
4932 F7B9 EBF3
4933 F7B9 80FA00
4934 F7B9 74F7
4935 F7B9 FECA
4936 F7B9 B200
4937 F7B9 EBF3
4938 F7B9 80FE18
4939 F7B9 74F7
4940 F7B9 FECA
4941 F7B9 B200
4942 F7B9 EBF3
4943 F7B9 80FA00
4944 F7B9 74F7
4945 F7B9 FECA
4946 F7B9 B200
4947 F7B9 EBF3
4948 F7B9 80FE18
4949 F7B9 74F7
4950 F7B9 FECA
4951 F7B9 B200
4952 F7B9 EBF3
4953 F7B9 80FA00
4954 F7B9 74F7
4955 F7B9 FECA
4956 F7B9 B200
4957 F7B9 EBF3
4958 F7B9 80FE18
4959 F7B9 74F7
4960 F7B9 FECA
4961 F7B9 B200
4962 F7B9 EBF3
4963 F7B9 80FA00
4964 F7B9 74F7
4965 F7B9 FECA
4966 F7B9 B200
4967 F7B9 EBF3
4968 F7B9 80FE18
4969 F7B9 74F7
4970 F7B9 FECA
4971 F7B9 B200
4972 F7B9 EBF3
4973 F7B9 80FA00
4974 F7B9 74F7
4975 F7B9 FECA
4976 F7B9 B200
4977 F7B9 EBF3
4978 F7B9 80FE18
4979 F7B9 74F7
4980 F7B9 FECA
4981 F7B9 B200
4982 F7B9 EBF3
4983 F7B9 80FA00
4984 F7B9 74F7
4985 F7B9 FECA
4986 F7B9 B200
4987 F7B9 EBF3
4988 F7B9 80FE18
4989 F7B9 74F7
4990 F7B9 FECA
4991 F7B9 B200
4992 F7B9 EBF3
4993 F7B9 80FA00
4994 F7B9 74F7
4995 F7B9 FECA
4996 F7B9 B200
4997 F7B9 EBF3
4998 F7B9 80FE18
4999 F7B9 74F7
5000 F7B9 FECA
5001 F7B9 B200
5002 F7B9 EBF3
5003 F7B9 80FA00
5004 F7B9 74F7
5005 F7B9 FECA
5006 F7B9 B200
5007 F7B9 EBF3
5008 F7B9 80FE18
5009 F7B9 74F7
5010 F7B9 FECA
5011 F7B9 B200
5012 F7B9 EBF3
5013 F7B9 80FA00
5014 F7B9 74F7
5015 F7B9 FECA
5016 F7B9 B200
5017 F7B9 EBF3
5018 F7B9 80FE18
5019 F7B9 74F7
5020 F7B9 FECA
5021 F7B9 B200
5022 F7B9 EBF3
5023 F7B9 80FA00
5024 F7B9 74F7
5025 F7B9 FECA
5026 F7B9 B200
5027 F7B9 EBF3
5028 F7B9 80FE18
5029 F7B9 74F7
5030 F7B9 FECA
5031 F7B9 B200
5032 F7B9 EBF3
5033 F7B9 80FA00
5034 F7B9 74F7
5035 F7B9 FECA
5036 F7B9 B200
5037 F7B9 EBF3
5038 F7B9 80FE18
5039 F7B9 74F7
5040 F7B9 FECA
5041 F7B9 B200
5042 F7B9 EBF3
5043 F7B9 80FA00
5044 F7B9 74F7
5045 F7B9 FECA
5046 F7B9 B200
5047 F7B9 EBF3
5048 F7B9 80FE18
5049 F7B9 74F7
5050 F7B9 FECA
5051 F7B9 B200
5052 F7B9 EBF3
5053 F7B9 80FA00
5054 F7B9 74F7
5055 F7B9 FECA
5056 F7B9 B200
5057 F7B9 EBF3
5058 F7B9 80FE18
5059 F7B9 74F7
5060 F7B9 FECA
5061 F7B9 B200
5062 F7B9 EBF3
5063 F7B9 80FA00
5064 F7B9 74F7
5065 F7B9 FECA
5066 F7B9 B200
5067 F7B9 EBF3
5068 F7B9 80FE18
5069 F7B9 74F7
5070 F7B9 FECA
5071 F7B9 B200
5072 F7B9 EBF3
5073 F7B9 80FA00
5074 F7B9 74F7
5075 F7B9 FECA
5076 F7B9 B200
5077 F7B9 EBF3
5078 F7B9 80FE18
5079 F7B9 74F7
5080 F7B9 FECA
5081 F7B9 B200
5082 F7B9 EBF3
5083 F7B9 80FA00
5084 F7B9 74F7
5085 F7B9 FECA
5086 F7B9 B200
5087 F7B9 EBF3
5088 F7B9 80FE18
5089 F7B9 74F7
5090 F7B9 FECA
5091 F7B9 B200
5092 F7B9 EBF3
5093 F7B9 80FA00
5094 F7B9 74F7
5095 F7B9 FECA
5096 F7B9 B200
5097 F7B9 EBF3
5098 F7B9 80FE18
5099 F7B9 74F7
5100 F7B9 FECA
5101 F7B9 B200
5102 F7B9 EBF3
5103 F7B9 80FA00
5104 F7B9 74F7
5105 F7B9 FECA
5106 F7B9 B200
5107 F7B9 EBF3
5108 F7B9 80FE18
5109 F7B9 74F7
5110 F7B9 FECA
5111 F7B9 B200
5112 F7B9 EBF3
5113 F7B9 80FA00
5114 F7B9 74F7
5115 F7B9 FECA
5116 F7B9 B200
5117 F7B9 EBF3
5118 F7B9 80FE18
5119 F7B9 74F7
5120 F7B9 FECA
5121 F7B9 B200
5122 F7B9 EBF3
5123 F7B9 80FA00
5124 F7B9 74F7
5125 F7B9 FECA
5126 F7B9 B200
5127 F7B9 EBF3
5128 F7B9 80FE18
5129 F7B9 74F7
5130 F7B9 FECA
5131 F7B9 B200
5132 F7B9 EBF3
5133 F7B9 80FA00
5134 F7B9 74F7
5135 F7B9 FECA
5136 F7B9 B200
5137 F7B9 EBF3
5138 F7B9 80FE18
5139 F7B9 74F7
5140 F7B9 FECA
5141 F7B9 B200
5142 F7B9 EBF3
5143 F7B9 80FA00
5144 F7B9 74F7
5145 F7B9 FECA
5146 F7B9 B200
5147 F7B9 EBF3
5148 F7B9 80FE18
5149 F7B9 74F7
5150 F7B9 FECA
5151 F7B9 B200
5152 F7B9 EBF3
5153 F7B9 80FA00
5154 F7B9 74F7
5155 F7B9 FECA
5156 F7B9 B200
5157 F7B9 EBF3
5158 F7B9 80FE18
5159 F7B9 74F7
5160 F7B9 FECA
5161 F7B9 B200
5162 F7B9 EBF3
5163 F7B9 80FA00
5164 F7B9 74F7
5165 F7B9 FECA
5166 F7B9 B200
5167 F7B9 EBF3
5168 F7B9 80FE18
5169 F7B9 74F7
5170 F7B9 FECA
5171 F7B9 B200
5172 F7B9 EBF3
5173 F7B9 80FA00
5174 F7B9 74F7
5175 F7B9 FECA
5176 F7B9 B200
5177 F7B9 EBF3
5178 F7B9 80FE18
5179 F7B9 74F7
5180 F7B9 FECA
5181 F7B9 B200
5182 F7B9 EBF3
5183 F7B9 80FA00
5184 F7B9 74F7
5185 F7B9 FECA
5186 F7B9 B200
5187 F7B9 EBF3
5188 F7B9 80FE18
5189 F7B9 74F7
5190 F7B9 FECA
5191 F7B9 B200
5192 F7B9 EBF3
5193 F7B9 80FA00
5194 F7B9 74F7
5195 F7B9 FECA
5196 F7B9 B200
5197 F7B9 EBF3
5198 F7B9 80FE18
5199 F7B9 74F7
5200 F7B9 FECA
5201 F7B9 B200
5202 F7B9 EBF3
5203 F7B9 80FA00
5204 F7B9 74F7
5205 F7B9 FECA
5206 F7B9 B200
5207 F7B9 EBF3
5208 F7B9 80FE18
5209 F7B9 74F7
5210 F7B9 FECA
5211 F7B9 B200
5212 F7B9 EBF3
5213 F7B9 80FA00
5214 F7B9 74F7
5215 F7B9 FECA
5216 F7B9 B200
5217 F7B9 EBF3
5218 F7B9 80FE18
5219 F7B9 74F7
5220 F7B9 FECA
5221 F7B9 B200
5222 F7B9 EBF3
5223 F7B9 80FA00
5224 F7B9 74F7
5225 F7B9 FECA
5226 F7B9 B200
5227 F7B9 EBF3
5228 F7B9 80FE18
5229 F7B9 74F7
5230 F7B9 FECA
5231 F7B9 B200
5232 F7B9 EBF3
5233 F7B9 80FA00
5234 F7B9 74F7
5235 F7B9 FECA
5236 F7B9 B200
5237 F7B9 EBF3
5238 F7B9 80FE18
5239 F7B9 74F7
5240 F7B9 FECA
5241 F7B9 B200
5242 F7B9 EBF3
5243 F7B9 80FA00
5244 F7B9 74F7
5245 F7B9 FECA
5246 F7B9 B200
5247 F7B9 EBF3
5248 F7B9 80FE18
5249 F7B9 74F7
5250 F7B9 FECA
5251 F7B9 B200
5252 F7B9 EBF3
5253 F7B9 80FA00
5254 F7B9 74F7
5255 F7B9 FECA
5256 F7B9 B200
5257 F7B9 EBF3
5258 F7B9 80FE18
5259 F7B9 74F7
5260 F7B9 FECA
5261 F7B9 B200
5262 F7B9 EBF3
5263 F7B9 80FA00
5264 F7B9 74F7
5265 F7B9 FECA
5266 F7B9 B200
5267 F7B9 EBF3
5268 F7B9 80FE18
5269 F7B9 74F7
5270 F7B9 FECA
5271 F7B9 B200
5272 F7B9 EBF3
5273 F7B9 80FA00
5274 F7B9 74F7
5275 F7B9 FECA
5276 F7B9 B200
5277 F7B9 EBF3
5278 F7B9 80FE18
5279 F7B9 74F7
5280 F7B9 FECA
5281 F7B9 B200
5282 F7B9 EBF3
5283 F7B9 80FA00
5284 F7B9 74F7
5285 F7B9 FECA
5286 F7B9 B200
5287 F7B9 EBF3
5288 F7B9 80FE18
5289 F7B9 74F7
5290 F7B9 FECA
5291 F7B9 B200
5292 F7B9 EBF3
5293 F7B9 80FA00
5294 F7B9 74F7
5295 F7B9 FECA
5296 F7B9 B200
5297 F7B9 EBF3
5298 F7B9 80FE18
5299 F7B9 74F7
5300 F7B9 FECA
5301 F7B9 B200
5302 F7B9 EBF3
5303 F7B9 80FA00
5304 F7B9 74F7
5305 F7B9 FECA
5306 F7B9 B200
5307 F7B9 EBF3
5308 F7B9 80FE18
5309 F7B9 74F7
5310 F7B9 FECA
5311 F7B9 B200
5312 F7B9 EBF3
5313 F7B9 80FA00
5314 F7B9 74F7
5315 F7B9 FECA
5316 F7B9 B200
5317 F7B9 EBF3
5318 F7B9 80FE18
5319 F7B9 74F7
5320 F7B9 FECA
5321 F7B9 B200
5322 F7B9 EBF3
5323 F7B9 80FA00
5324 F7B9 74F7
5325 F7B9 FECA
5326 F7B9 B200
5327 F7B9 EBF3
5328 F7B9 80FE18
5329 F7B9 74F7
5330 F7B9 FECA
5331 F7B9 B200
5332 F7B9 EBF3
5333 F7B9 80FA00
5334 F7B9 74F7
5335 F7B9 FECA
5336 F7B9 B200
5337 F7B9 EBF3
5338 F7B9 80FE18
5339 F7B9 74F7
5340 F7B9 FECA
5341 F7B9 B200
5342 F7B9 EBF3
5343 F7B9 80FA00
5344 F7B9 74F7
5345 F7B9 FECA
5346 F7B9 B200
5347 F7B9 EBF3
5348 F7B9 80FE18
5349 F7B9 74F7
5350 F7B9 FECA
5351 F7B9 B200
5352 F7B9 EBF3
5353 F7B9 80FA00
5354 F7B9 74F7
5355 F7B9 FECA
5356 F7B9 B200
5357 F7B9 EBF3
5358 F7B9 80FE18
5359 F7B9 74F7
5360 F7B9 FECA
5361 F7B9 B200
5362 F7B9 EBF3
5363 F7B9 80FA00
5364 F7B9 74F7
5365 F7B9 FECA
5366 F7B9 B200
5367 F7B9 EBF3
5368 F7B9 80FE18
5369 F7B9 74F7
5370 F7B9 FECA
5371 F7B9 B200
5372 F7B9 EBF3
5373 F7B9 80FA00
5374 F7B9 74F7
5375 F7B9 FECA
5376 F7B9 B200
5377 F7B9 EBF3
5378 F7B9 80FE18
5379 F7B9 74F7
5380 F7B9 FECA
5381 F7B9 B200
5382 F7B9 EBF3
5383 F7B9 80FA00
5384 F7B9 74F7
5385 F7B9 FECA
5386 F7B9 B200
5387 F7B9 EBF3
5388 F7B9 80FE18
5389 F7B9 74F7
5390 F7B9 FECA
5391 F7B9 B200
5392 F7B9 EBF3
5393 F7B9 80FA00
5394 F7B9 74F7
5395 F7B9 FECA
5396 F7B9 B200
5397 F7B9 EBF3
5398 F7B9 80FE18
5399 F7B9 74F7
5400 F7B9 FECA
5401 F7B9 B200
5402 F7B9 EBF3
5403 F7B9 80FA00
5404 F7B9 74F7
5405 F7B9 FECA
5406 F7B9 B200
5407 F7B9 EBF3
5408 F7B9 80FE18
5409 F7B9 74F7
5410 F7B9 FECA
5411 F7B9 B200
5412 F7B9 EBF3
5413 F7B9 80FA00
5414 F7B9 74F7
5415 F7B9 FECA
5416 F7B9 B200
5417 F7B9 EBF3
5418 F7B9 80FE18
5419 F7B9 74F7
5420 F7B9 FECA
5421 F7B9 B200
5422 F7B9 EBF3
5423 F7B9 80FA00
5424 F7B9 74F7
5425 F7B9 FECA
5426 F7B9 B200
5427 F7B9 EBF3
5428 F7B9 80FE18
5429 F7B9 74F7
5430 F7B9 FECA
5431 F7B9 B200
5432 F7B9 EBF3
5433 F7B9 80FA00
5434 F7B9 74F7
5435 F7B9 FECA
5436 F7B9 B200
5437 F7B9 EBF3
5438 F7B9 80FE18
5439 F7B9 74F7
5440 F7B9 FECA
5441 F7B9 B200
5442 F7B9 EBF3
5443 F7B9 80FA00
5444 F7B9 74F7
5445 F7B9 FECA
5446 F7B9 B200
5447 F7B9 EBF3
5448 F7B9 80FA00
5449 F7B9 74F7
5450 F7B9 FECA
5451 F7B9 B200
5452 F7B9 EBF3
5453 F7B9 80FA00
5454 F7B9 74F7
5455 F7B9 FECA
5456 F7B9 B200
5457 F7B9 EBF3
5458 F7B9 80FA00
5459 F7B9 74F7
5460 F7B9 FECA
5461 F7B9 B200
5462 F7B9 EBF3
5463 F7B9 80FA00
5464 F7B9 74F7
5465 F7B9 FECA
5466 F7B9 B200
5467 F7B9 EBF3
5468 F7B9 80FA00
5469 F7B9 74F7
5470 F7B9 FECA
5471 F7B9 B200
5472 F7B9 EBF3
5473 F7B9 80FA00
5474 F7B9 74F7
5475 F7B9 FECA
5476 F7B9 B200
5477 F7B9 EBF3
5478 F7B9 80FA00
5479 F7B9 74F7
5480 F7B9 FECA
5481 F7B9 B200
5482 F7B9 EBF3
5483 F7B9 80FA00
5484 F7B9 74F7
5485 F7B9 FECA
5486 F7B9 B200
5487 F7B9 EBF3
5488 F7B9 80FA00
5489 F7B9 74F7
5490 F7B9 FECA
5491 F7B9 B200
5492 F7B9 EBF3
5493 F7B9 80FA00
5494 F7B9 74F7
5495 F7B9 FECA
5496 F7B9 B200
5497 F7B9 EBF3
5498 F7B9 80FA00
5499 F7B9 74F7
5500 F7B9 FECA
5501 F7B9 B200
5502 F7B9 EBF3
5503 F7B9 80FA00
5504 F7B9 74F7
5505 F7B9 FECA
5506 F7B9 B200
5507 F7B9 EBF3
5508 F7B9 80FA00
5509 F7B9 74F7
5510 F7B9 FECA
5511 F7B9 B200
5512 F7B9 EBF3
5513 F7B9 80FA00
5514 F7B9 74F7
5515 F7B9 FECA
5516 F7B9 B200
5517 F7B9 EBF3
5518 F7B9 80FA00
5519 F7B9 74F7
5520 F7B9 FECA
5521 F7B9 B200
5522 F7B9 EBF3
5523 F7B9 80FA00
5524 F7B9 74F7
5525 F7B9 FECA
5526 F7B9 B200
5527 F7B9 EBF3
5528 F7B9 80FA00
5529 F7B9 74F7
5530 F7B9 FECA
5531 F7B9 B200
5532 F7B9 EBF3
5533 F7B9 80FA00
5534 F7B9 74F7
5535 F7B9 FECA
5536 F7B9 B200
5537 F7B9 EBF3
5538 F7B9 80FA00
5539 F7B9 74F7
5540 F7B9 FECA
5541 F7B9 B200
5542 F7B9 EBF3
5543 F7B9 80FA00
5544 F7B9 74F7
5545 F7B9 FECA
5546 F7B9 B200
5547 F7B9 EBF3
5548 F7B9 80FA00
5549 F7B9 74F7
5550 F7B9 FECA
5551 F7B9 B200
5552 F7B9 EBF3
5553 F7B9 80FA00
5554 F7B9 74F7
5555 F7B9 FECA
5556 F7B9 B200
5557 F7B9 EBF3
5558 F7B9 80FA00
5559 F7B9 74F7
5560 F7B9 FECA
5561 F7B9 B200
5562 F7B9 EBF3
5563 F7B9 80FA00
5564 F7B9 74F7
5565 F7B9 FECA
5566 F7B9 B200
5567 F7B9 EBF3
5568 F7B9 80FA00
5569 F7B9 74F7
5570 F7B9 FECA
5571 F7B9 B200
5572 F7B9 EBF3
5573 F7B9 80FA00
5574 F7B9 74F7
5575 F7B9 FECA
5576 F7B9 B200
5577 F7B9 EBF3
5578 F7B9 80FA00
5579 F7B9 74F7
5580 F7B9 FECA
5581 F7B9 B200
5582 F7B9 EBF3
5583 F7B9 80FA00
5584 F7B9 74F7
5585 F7B9 FECA
5586 F7B9 B200
5587 F7B9 EBF3
5588 F7B9 80FA00
5589 F7B9 74F7
5590 F7B9 FECA
5591 F7B9 B200
5592 F7B9 EBF3
5593 F7B9 80FA00
5594 F7B9 74F7
5595 F7B9 FECA
5596 F7B9 B200
5597 F7B9 EBF3
5598 F7B9 80FA00
5599 F7B9 74F7
5600 F7B9 FECA
5601 F7B9 B200
5602 F7B9 EBF3
5603 F7B9 80FA00
5604 F7B9 74F7
5605 F7B9 FECA
5606 F7B9 B200
5607 F7B9 EBF3
5608 F7B9 80FA00
5609 F7B9 74F7
5610 F7B9 FECA
5611 F7B9 B200
5612 F7B9 EBF3
5613 F7B9 80FA00
5614 F7B9 74F7
5615 F7B9 FECA
5616 F7B9 B200
5617 F7B9 EBF3
5618 F7B9 80FA00
5619 F7B9 74F7
5620 F7B9 FECA
5621 F7B9 B200
5622 F7B9 EBF3
5623 F7B9 80FA00
5624 F7B9 74F7
5625 F7B9 FECA
5626 F7B9 B200
5627 F7B9 EBF3
5628 F7B9 80FA00
5629 F7B9 74F7
5630 F7B9 FECA
5631 F7B9 B200
5632 F7B9 EBF3
5633 F7B9 80FA00
5634 F7B9 74F7
5635 F7B9 FECA
5636 F7B9 B200
5637 F7B9 EBF3
5638 F7B9 80FA00
5639 F7B9 74F7
5640 F7B9 FECA
5641 F7B9 B200
5642 F7B9 EBF3
5643 F7B9 80FA00
5644 F7B9 74F7
5645 F7B9 FECA
5646 F7B9 B200
5647 F7B9 EBF3
5648 F7B9 80FA00
5649 F7B9 74F7
5650 F7B9 FECA
5651 F7B9 B200
5652 F7B9 EBF3
5653 F7B9 80FA00
5654 F7B9 74F7
5655 F7B9 FECA
5656 F7B9 B200
5657 F7B9 EBF3
5658 F7B9 80FA00
5659 F7B9 74F7
5660 F7B9 FECA
5661 F7B9 B200
5662 F7B9 EBF3
5663 F7B9 80FA00
5664 F7B9 74F7
5665 F7B9 FECA
5666 F7B9 B200
5667 F7B9 EBF3
5668 F7B9 80FA00
5669 F7B9 74F7
5670 F7B9 FECA
5671 F7B9 B200
5672 F7B9 EBF3
5673 F7B9 80FA00
5674 F7B9 74F7
5675 F7B9 FECA
5676 F7B9 B200
5677 F7B9 EBF3
5678 F7B9 80FA00
5679 F7B9 74F7
5680 F7B9 FECA
5681 F7B9 B200
5682 F7B9 EBF3
5683 F7B9 80FA00
5684 F7B9 74F7
5685 F7B9 FECA
5686 F7B9 B200
5687 F7B9 EBF3
5688 F7B9 80FA00
5689 F7B9 74F7
5690 F7B9 FECA
5691 F7B9 B200
5692 F7B9 EBF3
5693 F7B9 80FA00
5694 F7B9 74F7
5695 F7B9 FECA
5696 F7B9 B200
5697 F7B9 EBF3
5698 F7B9 80FA00
5699 F7B9 74F7
5700 F7B9 FECA
5701 F7B9 B200
5702 F7B9 EBF3
5703 F7B9 80FA00
5704 F7B9 74F7
5705 F7B9 FECA
5706 F7B9 B200
5707 F7B9 EBF3
5708 F7B9 80FA00
5709 F7B9 74F7
5710 F7B9 FECA
5711 F7B9 B200
5712 F7B9 EBF3
5713 F7B9 80FA00
5714 F7B9 74F7
5715 F7B9 FECA
5716 F7B9 B200
5717 F7B9 EBF3
5718 F7B9 80FA00
5719 F7B9 74F7
5720 F7B9 FECA
5721 F7B9 B200
5722 F7B9 EBF3
5723 F7B9 80FA00
5724 F7B9 74F7
5725 F7B9 FECA
5726 F7B9 B200
5727 F7B9 EBF3
5728 F7B9 80FA00

```

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

```

4909
4910 ;----- BELL FOUND
4911
4912 UII:
F78D
4913 MOV BL,2 ; SET UP COUNT FOR BEEP
F78D B302
4914 CALL BEEP ; SOUND THE POD BELL
F78F E87602
4915 JMP U5 ; TTY_RETURN
F792 EBD0
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 ; (BH) = BEST GUESS AT PIXEL HORIZONTAL POSITION :
4931 ;-
4932 ; ASSUME CS:CODE,DS:DATA
4933 ;----- SUBTRACT_TABLE
4934 V1 LABEL BYTE
4935 DB 3,3,5,5,3,3,4 ;
```

F794

F794 03

F795 03

F795 06

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 BB166300
4941 MOV DX,ADDR_6845 ; GET BASE ADDRESS OF 6845
F7A2 C2C06
4942 ADD DL,4 ; POINT TO STATUS REGISTER
F7A5 EC
4943 IN AL,DX ; GET STATUS REGISTER
F7A6 A804
4944 TEST AL,4 ; TEST LIGHT PEN SWITCH
F7A8 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 JNZ V7A ; RETURN WITHOUT RESETTING TRIGGER
F7AE E98100
4951 JMP V7
4952
4953 ;----- TRIGGER HAS BEEN SET, READ THE VALUE IN
4954
F7B1 V7A:
4955 MOV AH,16 ; LIGHT PEN REGISTERS ON 6845
4956
F7B1 B410
4957 ;----- INPUT REGS POINTED TO BY AH, AND CONVERT TO ROW COLUMN IN DX
4958
F7B3 BB166300
4959 MOV DX,ADDR_6845 ; ADDRESS REGISTER FOR 6845
F7B7 8AC4
4960 MOV AL,AH ; REGISTER TO READ
F7B9 EE
4961 OUT DX,AL ; SET IT UP
F7BA 42
4962 INC DX ; DATA REGISTER
F7BB EC
4963 IN AL,DX ; GET THE VALUE
F7BC 8AE8
4964 MOV CH,AL ; SAVE IN CX
F7BD 8AE8
4965 DEC DX ; ADDRESS REGISTER
F7BF FEC4
4966 INC AH ; SECOND DATA REGISTER
F7C1 8AC4
4967 MOV AL,AH ; ADDRESS REGISTER FOR 6845
F7C3 EE
4968 OUT DX,AL ; SET IT UP
F7C4 42
4969 INC DX ; DATA REGISTER
F7C5 EC
4970 IN AL,DX ; GET THE VALUE
F7C6 8AE5
4971 MOV AH,CH ; AX HAS INPUT VALUE
4972
4973 ;----- AX HAS THE VALUE READ IN FROM THE 6845
4974
F7C8 8A1E4900
4975 MOV BL,CRT_MODE ; MODE VALUE TO BX
F7CC 2AFF
4976 SUB BH,BH ; DETERMINE AMOUNT TO SUBTRACT
F7CD 2E8A9F94F7
4977 SUB AX,BX ; TAKE IT AWAY
F7D3 803E490004
4978 CMP AX,CS:V1[BX]
F7D5 BB1E4E00
4979 SUB AX,BX ; TAKE IT AWAY
F7D9 D1EB
4980 MOV BX,CRT_START ; DETERMINE MODE
F7DB 2BC3
4981 SHR BX,1 ; DETERMINE MODE
F7DD 7902
4982 SUB AX,BX ; DETERMINE MODE
F7DF 7902
4983 JNS V2 ; IF POSITIVE, DETERMINE MODE
F7E0 2BC0
4984 SUB AX,AX ; <0 PLAYS AS 0
4985
4986 ;----- DETERMINE MODE OF OPERATION
4987
F7E1 V2:
4988 MOV CL,3 ; DETERMINE MODE
F7E1 B103
4989 MOV CL,4 ; SET *8 SHIFT COUNT
F7E3 803E490004
4990 CMP CRT_MODE,4 ; DETERMINE IF GRAPHICS OR ALPHA
F7E8 7200
4991 JB V4 ; ALPHA_PEN
F7E8 803E490007
4992 CMP CRT_MODE,7 ; ALPHA_PEN
F7EF 7423
4993 JE V4
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
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 MOY 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 MOY V3 ; LOW HIGH RES
F803 B104
5009 MOY 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	
F80A	8AD4	5013		
F80C	8AF0	5014	;----- DETERMINE ALPHA CHAR POSITION	
F80E	D0EE	5015		
F810	D0EE	5016	MOV DL,AH	; COLUMN VALUE FOR RETURN
F812	EB12	5017	MOV DH,AL	; ROW VALUE
		5018	SHR DH,1	; DIVIDE BY 4
		5019	SHR DH,1	; FOR VALUE IN 0-24 RANGE
		5020	JMP SHORT V5	; LIGHT_PEN_RETURN_SET
		5021		
		5022	;----- ALPHA MODE ON LIGHT PEN	
		5023		
F814	F6364A00	5024	V4:	
F818	8AF0	5025	DIV BYTE PTR CRT_COLS	; ALPHA PEN
F81A	8AD4	5026	MOV DH,AL	; DETERMINE ROW,COLUMN VALUE
F81C	D2E0	5027	MOV DL,AH	; ROWS TO DH
F81D	8B8	5028	SAL AL,CL	; COLS TO DL
F820	8ADC	5029	MOV CH,AL	; MULTIPLY ROWS * 8
F822	32FF	5030	MOV BH,AH	; GET RASTER VALUE TO RETURN REG
F824	D3E3	5031	XOR BH,BH	; COLUMN VALUE
F826		5032	SAL BX,CL	; TO BX
F826	B401	5033	V5:	; LIGHT PEN RETURN SET
F828	52	5034	MOV AH,I	; INDICATE EVERTHIN' SET
F829	8B166300	5035	V6:	; LIGHT_PEN_RETURN
F82D	83C207	5036	PUSH DX	; SAVE RETURN VALUE (IN CASE)
F830	EE	5037	MOV DX,ADDR_6845	; SET BASE ADDRESS
F831	5A	5038	ADD DX,7	; POINT TO RESET PARM
F832	5F	5039	OUT DX,AL	; ADDRESS, NOT DATA, IS IMPORTANT
F833	5E	5040	POP DX	; RECOVER VALUE
F834	1F	5041	VT:	; RETURN_NO_RESET
F835	1F	5042	POP DI	
F836	1F	5043	POP SI	
F838	07	5044	POP DS	
F839	CF	5045	POP DS	
		5046	POP DS	
		5047	POP DS	
		5048	POP ES	
		5049	IRET	
		5050	READ_LPEN	ENDP

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

```

5051      ;-- INT 12 --
5052      ;   MEMORY_SIZE_DET
5053      ;     THIS ROUTINE DETERMINES THE AMOUNT OF MEMORY IN THE SYSTEM
5054      ;     AS REPRESENTED BY THE SWITCHES ON THE PLANAR. NOTE THAT THE
5055      ;     SYSTEM MAY NOT BE ABLE TO USE I/O MEMORY UNLESS THERE IS A FULL
5056      ;     COMPLEMENT OF 64K BYTES ON THE PLANAR.
5057      ; INPUT
5058      ;     NO REGISTERS
5059      ;     THE MEMORY_SIZE VARIABLE IS SET DURING POWER ON DIAGNOSTICS
5060      ;     ACCORDING TO THE FOLLOWING HARDWARE ASSUMPTIONS:
5061      ;     PORT 60 BITS 3,2 = 00 - 16K BASE RAM
5062      ;           01 - 32K BASE RAM
5063      ;           10 - 48K BASE RAM
5064      ;           11 - 64K BASE RAM
5065      ;     PORT 62 BITS 3-0 INDICATE MODE OF 16K I/O RAM IN 32K INCREMENTS
5066      ;     E.G., 0000 - NO RAM IN I/O CHANNEL
5067      ;           0010 - 64K RAM IN I/O CHANNEL, ETC.
5068      ; OUTPUT
5069      ;     (AX) = NUMBER OF CONTIGUOUS 1K BLOCKS OF MEMORY
5070      ;-----+
5071
5072      ; ASSUME CS:CODE,DS:DATA
5073      ORG 0F841H
5074      MEMORY_SIZE_DET PROC FAR
5075      STI          ; INTERRUPTS BACK ON
5076      PUSH DS       ; SAVE SEGMENT
5077      CALL DDS
5078      MOV AX,MEMORY_SIZE ; GET VALUE
5079      POP DS       ; RECOVER SEGMENT
5080      RET          ; 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 37A = 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 0F840H
5119      EQUIPMENT PROC FAR
5120      STI          ; INTERRUPTS BACK ON
5121      PUSH DS       ; SAVE SEGMENT REGISTER
5122      CALL DDS
5123      MOV AX,EQUIP_FLAG ; GET THE CURRENT SETTINGS
5124      POP DS       ; RECOVER SEGMENT
5125      RET          ; RETURN TO CALLER
5126      EQUIPMENT ENDP
5127
5128      ;-- INT 15 --
5129      ; DUMMY CASSETTE I/O ROUTINE--RETURNS 'INVALID CMD' IF THE ROUTINE IS
5130      ; IS EVER CALLED BY ACCIDENT (AH=86H, CARRY FLAG=1)
5131      ;-----+
5132      ORG 0F859H
5133      CASSETTE_10 PROC FAR
5134      STC          ; CARRY INDICATOR=1
5135      MOV AH,86H
5136      RET          2
5137      CASSETTE_10 ENDP

```

```

5138 ;-----+
5140 ; NON-MASKABLE INTERRUPT ROUTINE:
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 (INTERMITTENT :
5145 ; READ PROBLEM) ?????<-WILL BE PRINTED WHERE THE ADDRESS :
5146 ; WOULD NORMALLY GO.
5147 ; IF ADDRESS OF 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,0COH ; PARITY CHECK?
5156 JNZ NMI_1
5157 JMP DI4+ ; NO, EXIT FROM ROUTINE
5158 NMI_1:
5159 MOV DX,DATA
5160 MOV DS,DX
5161 MOV SI,OFFSET D1 ; ADDR OF ERROR MSG
5162 TEST AL,40H ; I/O PARITY CHECK
5163 JNZ D13 ; DISPLAY ERROR MSG
5164 MOV SI,OFFSET D2 ; MUST BE PLANAR
5165 D13: ; INIT AND SET MODE FOR VIDEO
5166 MOV AH,0
5167 MOV AL,CRT_MODE
5168 INT 10H ; CALL VIDEO IO PROCEDURE
5169 CALL P_MSG ; PRINT ERROR MSG
5170
5171 ;----- SEE IF LOCATION THAT CAUSED PARITY CHECK CAN BE FOUND
5172
5173 MOV AL,00H ; DISABLE TRAP
5174 OUT 0OH,AL
5175 IN AL,PORT_B
5176 OR AL,0010000B ; TOGGLE PARITY CHECK ENABLES
5177 OUT PORT_B,AL
5178 AND AL,0001111B
5179 OUT PORT_B,AL
5180 MOV BX,MEMORY_SIZE ; GET MEMORY SIZE WORD
5181 CLD ; SET DIR FLAG TO INCREMENT
5182 SUB DX,DX ; POINT DX AT START OF MEM
5183 NMI_LOOP:
5184 MOV DS,DX
5185 MOV ECX
5186 MOV CX,4000H ; SET FOR 16KB SCAN
5187 SUB SI,SI ; SET SI TO BE RELATIVE TO
5188 ; START OF ES
5189 REP LODSB ; READ 16KB OF MEMORY
5190 IN AL,PORT_C ; SEE IF PARITY CHECK HAPPENED
5191 AND AL,1000000B
5192 JNZ PRT_NMI ; GO PRINT ADDRESS IF IT DID
5193 ADD DX,6400H ; POINT TO NEXT 16K BLOCK
5194 SUB BX,16D
5195 JNZ NMI_LOOP
5196 MOV BX,16D ; PRINT ROW OF ????? IF PARITY
5197 CALL P_MSG ; CHECK COULD NOT BE RE-CREATED
5198 CLI
5199 HLT ; HALT SYSTEM
5200 PRT_NMI:
5201 MOV DX,DS ; PRINT SEGMENT VALUE
5202 CALL PRT_SEG
5203 MOV DX,5213H
5204 MOV AL,00
5205 OUT DX,AL ; DISABLE EXPANSION BOX
5206 MOV AL,'*' ; (CAN'T WRITE TO MEM)
5207 CALL PRT_HEX
5208 MOV AL,055AH
5209 MOV CX,AH
5210 SUB BX,BX
5211 MOV [BX],AX ; WRITE A WORD TO SEGMENT THAT
5212 NOP
5213 NOP
5214 MOV AX,[BX] ; HAD THE ERROR
5215 CMP AX,CX ; IS IT THERE?
5216 JE SYS_BOX_ERR ; YES- MUST BE SYS UNIT
5217 MOV AL,'*' ; NO- MUST BE IN EXP. BOX
5218 CALL PRT_HEX
5219 SHORT HLT_NMI
5220 HLT_NMI:
5221 MOV AL,'$' ; PRINTER
5222 CALL PRT_HEX
5223 HLT_NMI:
5224 MOV AL,'*' ; HALT SYSTEM
5225 CALL PRT_HEX
5226 CLI
5227 HLT
5228 DI4: ; RESTORE ORIG CONTENTS OF AX
5229 POP AX
5230 IRET
5231 NMI_INT ENDP
5232
5233 ;-----+
5234 ; ROS CHECKSUM SUBROUTINE
5235 ;-----+
F8F2
5236 ROS_CHECKSUM PROC NEAR ; NEXT ROS MODULE
5237 MOV CX,8192 ; NUMBER OF BYTES TO ADD
5238 ROS_CHECKSUM_CNT: ; ENTRY FOR OPTIONAL ROS TEST
5239 XOR AL,AL
5240 C26: ;-----+
5241 ADD AL,DS:[BX]
5242 INC BX ; POINT TO NEXT BYTE
5243 LOOP C26 ; ADD ALL BYTES IN ROS MODULE
5244 OR AL,AL ; SUM = 0?
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 0A
F904 20323031 5251 E1 DB '201',13,10 ; MEMORY ERROR
F908 0D
F909 0A
F90A 524F4D
F90D 0D
F90E 0A
F90F 1383031 5253 F3C DB '1801',13,10 ; EXPANSION IO BOX ERROR
F913 0D
F914 0A
F915 50415249545920 5254 D1 DB 'PARITY CHECK 2',13,10
43484543B2032
F923 0D
F924 0A
F925 50415249545920 5255 D2 DB 'PARITY CHECK 1',13,10
43484543B2031
F933 0D
F934 0A
F935 3F3F3F3F3F 5256 D2A DB '?????',13,10
F93A 0D
F93B 0A
5257
5258 ;-----+
5259 ; BLINK LED PROCEDURE FOR MFG RUN-IN TESTS
5260 ; IF LED IS ON, TURN IT OFF. IF OFF, TURN ON.
5261 ;-----+
5262 ASSUME DS:DATA
5263 BLINK_INT PROC NEAR
F93C FB
5264 STI
F93D 50 5265 PUSH AX ; SAVE AX REG CONTENTS
F93E E4E1 5266 IN AL,PORT_B ; READ CURRENT VAL OF PORT B
F940 FAE0 5267 MOV AH,AL
F942 F6D0 5268 NOT AL ; FLIP ALL BITS
F944 2440 5269 AND AL,0000000B ; ISOLATE CONTROL BIT
F946 80E4BF 5270 AND AH,1011111B ; MASK OUT OF ORIGINAL VAL
F949 0AC4 5271 OR AL,AH ; OR NEW CONTROL BIT IN
F94B E661 5272 OUT PORT_B,AL
F94C B020 5273 MOV AL,ED1
F94F E4D0 5274 OUT AL,ED1
F951 58 5275 POP INTA00,AL
F952 CF 5276 IRET ; RESTORE AX REG
5277 BLINK_INT ENDP
5278
5279
5280 ; THIS ROUTINE CHECKSUMS OPTIONAL ROM MODULES AND
5281 ; IF CHECKSUM IS OK, CALLS INIT/TEST CODE IN MODULE :
5282 ;-----+
5283 ROM_CHECK PROC NEAR
F953 B64000 5284 MOV AX,DATA ; POINT ES TO DATA AREA
F954 BE00 5285 MOV ES,AX
F958 2AE4 5286 SUB AL,AH ; ZERO OUT AH
F95A BA702 5287 MOV AL,[BX+2] ; GET LENGTH INDICATOR
F95D B109 5288 MOV CL,09H ; MULTIPLY BY 512
F95F D3E0 5289 SHL AX,CL
F961 8BC8 5290 MOV CX,AX ; SET COUNT
F962 5A 5291 PUSH CX ; SAVE COUNT
F964 904000 5292 MOV CL,4 ; ADJUST
F967 D3E8 5293 SHR AX,CL
F969 0300 5294 ADD DX,AX ; SET POINTER TO NEXT MODULE
F96B 59 5295 POP CX ; RETRIEVE COUNT
F96C E886FF 5296 CALL ROS_CHECKSUM_CNT ; DO CHECKSUM
F96F 4000 5297 JZ ROM_CHECK_I
F971 ED7ED 5298 CALL ROM_END ; POST CHECKSUM ERROR
F974 EB1490 5299 JMP ROM_CHECK_END ; AND EXIT
F977 52 5300 ROM_CHECK_I: ;-----+
5301 PUSH DX ; SAVE POINTER
F978 26C7067000300 5302 MOV ES:IO_ROM_INIT,0003H ; LOAD OFFSET
F979 268C1E900 5303 MOV ES:IO_ROM_SEG,05 ; LOAD SEGMENT
F984 26F1E6700 5304 CALL DWORD_PTR ES:IO_ROM_INIT ; CALL INIT./TEST ROUTINE
F989 5A 5305 POP DX
F98A C3 5306 ROM_CHECK_END: ;-----+
5307 RET ; RETURN TO CALLER
F98A C3 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_BYTc PROC NEAR
F98B 50 5316 PUSH AX ; SAVE FOR LOW NIBBLE DISPLAY
F98C B104 5317 MOV CL,4 ; SHIFT COUNT
F98E D2E8 5318 SHR AL,CL ; NYBBLE SWAP
F990 E80300 5319 CALL XLAT_PR ; DO THE HIGH NIBBLE DISPLAY
F993 58 5320 POP AX ; RECOVER THE NIBBLE
F994 240F 5321 AND AL,0FH ; ISOLATE TO LOW NIBBLE
5322 DAA ; FALL INTO LOW NIBBLE CONVERSION
F996 0490 5323 XLAT_PR PROC NEAR ; CONVERT 0-0F TO ASCII CHARACTER
5324 ADD AL,090H ; ADD FIRST CONVERSION FACTOR
F998 27 5325 DAA ; ADJUST FOR NUMERIC AND ALPHA RANGE
F999 1440 5326 ADC AL,040H ; ADD CONVERSION AND ADJUST LOW NIBBLE
F99B 27 5327 INT 10H ; ADJUST HIGH NIBBLE TO ASCII RANGE
F99D 00 PRT_HEX ENDP ;-----+
F99C B40E 5328 XLAT_PR ENDP ; DISPLAY CHARACTER IN AL
F99E B700 5329 MOV AH,14
F99F B700 5330 MOV BH,0 ; CALL VIDEO_10
F9A0 CD10 5331 INT 10H
F9A2 C3 5332 RET
5333 PRT_HEX ENDP ;-----+
5334 XLAT_PR ENDP ;-----+
5335 XPC_BYTc ENDP ;-----+
5336
F9A3 F4 LABEL WORD ; PRINTER SOURCE TABLE
5337 F4 LABEL WORD
F9A3 BC03 5338 DW 3BCH
F9A5 7803 5339 DW 378H
F9A7 7802 5340 DW 218H
F9A9 F4E LABEL WORD
5342

```

```

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

5343 ; THIS SUBROUTINE WILL PRINT A MESSAGE ON THE DISPLAY :
5344 ; : : : : :
5345 ; ENTRY REQUIREMENTS:
5346 ; SI = OFFSET(ADDRESS) OF MESSAGE BUFFER
5347 ; CX = MESSAGE BYTE COUNT
5348 ; MAXIMUM MESSAGE LENGTH IS 36 CHARACTERS
5349 ; : : : : :

F9A9 E_ MSG PROC NEAR
5350 ; : : : : :
5351 ; E_ MSG PROC NEAR
5352 MOV BP,SI ; SET BP NON-ZERO TO FLAG ERR
5353 CALL P_MSG ; PRINT MESSAGE
5354 PUSH DS
5355 CALL DDS
5356 MOV AL,BYTE PTR EQUIP_FLAG ; LOOP/HALT ON ERROR
5357 AND AL,01H ; SWITCH ON?
5358 JNZ G12 ; NO - RETURN
5359 MFG_HALT2:
5360 CLI ; YES - HALT SYSTEM
5361 MOV AL,89H
5362 OUT DX,PORT_AL
5363 MOV AL,1000010B
5364 OUT PORT_B_AL ; DISABLE KB
5365 MOV AL,MFG_ERR_FLAG ; RECOVER ERROR INDICATOR
5366 OUT PORT_A_AL ; SET INTO B255 REG
5367 HLT ; HALT SYS
5368 G12:
5369 POP DS ; WRITE_MSG:
5370 RET
5371 E_ MSG ENDP
5372 : : : : :

F9CA F9CA 2E8A04
5373 P_MSG PROC NEAR
5374 GT2A: ; : : : :
5375 MOV AL,CS:[SI] ; PUT CHAR IN AL
5376 INC SI ; POINT TO NEXT CHAR
5377 PUSH AX ; SAVE PRINT CHAR
5378 CALL PRT_HEX ; CALL VIDEO_ID
5379 POP AX ; RECOVER PRINT CHAR
5380 CMP AL,10 ; WAS IT LINE FEED?
5381 JNE G12A ; : NO,KEEP PRINTING STRING
5382 RET
5383 P_MSG ENDP
5384 : : : : :

5385 ; INITIAL RELIABILITY TEST -- SUBROUTINES :
5386 ; : : : : :
5387 ; ASSUME CS:CODE,DS:DATA
5388 ; : : : : :
5389 ; SUBROUTINES FOR POWER ON DIAGNOSTICS :
5390 ; : : : : :
5391 ; THIS PROCEDURE WILL ISSUE ONE LONG TONE (3 SECs) AND ONE OR
5392 ; MORE SHORT TONES (1 SEC) TO INDICATE A FAILURE ON THE PLANAR
5393 ; BOARD, A BAD RAM MODULE, OR A PROBLEM WITH THE CRT.
5394 ; : : : : :
5395 ; ENTRY PARAMETERS:
5396 ; DH = NUMBER OF LONG TONES TO BEEP
5397 ; DL = NUMBER OF SHORT TONES TO BEEP.
5398 ; : : : : :

F9D8 F9D8 9C
5399 ERR_BEEP PROC NEAR
5400 PUSHF ; SAVE FLAGS
5401 CLI ; DISABLE SYSTEM INTERRUPTS
5402 PUSH DS ; SAVE DS REG CONTENTS
5403 CALL DDS ; : : : : :
5404 OR DH,DH ; ANY LONG ONES TO BEEP
5405 JZ G3 ; NO, DO THE SHORT ONES
5406 G1: ; : : : :
5407 MOV BL,6 ; LONG BEEP!
5408 CALL BEEP ; COUNTER FOR BEEPS
5409 G2: ; : : : :
5410 LOOP G2 ; DELAY BETWEEN BEEPS
5411 DEC DH ; ANY MORE TO DO
5412 JNZ G1 ; DO IT
5413 CMP MFG_TST,1 ; MFG TEST MODE?
5414 JNE G3 ; YES - CONTINUE BEEPING SPEAKER
5415 JMP MFG_HALT ; STOP BLINKING LED
5416 SJMP SHORT_BEEP ; SHORT BEEP
5417 MOV BL,1 ; COUNTER FOR A SHORT BEEP
5418 CALL BEEP ; DO THE SOUND
5419 G4: ; : : : :
5420 LOOP G4 ; DELAY BETWEEN BEEPS
5421 DEC DL ; DONE WITH SHORTS
5422 JNZ G3 ; DO SOME MORE
5423 G5: ; : : : :
5424 LOOP G5 ; LONG DELAY BEFORE RETURN
5425 G6: ; : : : :
5426 LOOP G6 ; : : : : :
5427 POP DS ; RESTORE ORIG CONTENTS OF DS
5428 POPF DS ; RESTORE FLAGS TO ORIG SETTINGS
5429 RET ; RETURN TO CALLER
5430 ERR_BEEP ENDP
5431 :---- ROUTINE TO SOUND BEEPER
5432 ; : : : : :
5433 BEEP PROC NEAR
5434 ; : : : : :
5435 MOV AL,1011010B ; SEL TIM 2,LSB,MSB,BINARY
5436 OUT TIMER+3,AL ; WRITE THE TIMER MODE REG
5437 MOV AX,533H ; DIVISOR FOR 1000 Hz
5438 OUT TIMER+2,AL ; WRITE TIMER 2 CNT - LSB
5439 MOV AX,1000H ; : : : : :
5440 OUT TIMER+2,AL ; WRITE TIMER 2 CNT - MSB
5441 IN AL,PORT_B ; GET CURRENT SETTING OF PORT
5442 MOV AH,AL ; SAVE THAT SETTINGH
5443 OR AL,03 ; TURN SPEAKER ON
5444 OUT PORT_B_AL ; : : : : :
5445 SUB CX,CX ; SET CNT TO WAIT 500 MS
5446 G7: ; : : : :
5447 LOOP G7 ; DELAY BEFORE TURNING OFF
5448 DEC BL ; DELAY CNT EXPRIED?
5449 JNZ G7 ; NO - CONTINUE SPEECH SPK
5450 MOV AL,AH ; RESTORE VALUE OF PORT
5451 OUT PORT_B_AL ; : : : : :
5452 RET ; RETURN TO CALLER
5453 BEEP ENDP ; : : : : :

```

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

```

5454
5455 ;----- THIS PROCEDURE WILL SEND A SOFTWARE RESET TO THE KEYBOARD. :
5456 ;----- SCAN CODE 'AA' SHOULD BE RETURNED TO THE CPU. :-----:
5457
5458 ;-----:
5459 KBD_RESET    PROC NEAR
5460 ASSUME DS:AB50
5461 MOV AL,08H ; SET KBD CLK LINE LOW
5462 OUT PORT_B,AL ; WRITE 8255 PORT B
5463 MOV CX,10582 ; HOLD KBD CLK LOW FOR 20 MS
5464
5465 GB: LOOP G8 ; LOOP FOR 20 MS
5466 MOV AL,0C8H ; SET CLK, ENABLE LINES HIGH
5467 OUT PORT_B,AL
5468 SP_TEST: ;-----:
5469 MOV AL,48H ; ENTRY FOR MANUFACTURING TEST 2
5470 OUT PORT_B,AL ; SET KBD CLK HIGH, ENABLE LOW
5471 MOV AL,0F0H ;-----:
5472 OUT INTA0,AL ; WRITE B259 IMR
5473 MOV DATA_AREA[OFFSET INTR_FLAG] ;-----: RESET INTERRUPT INDICATOR
5474 STI ;-----: ENABLE INTERRUPTS
5475 SUB CX,CX ;-----: SETUP INTERRUPT TIMEOUT CNT
5476
5477 TEST DATA_AREA[OFFSET INT_R_FLAG],02H ;-----: DID A KEYBOARD INTR OCCUR?
5478 JNZ G10 ;-----: YES - READ SCAN CODE RETURNED
5479 LOOP G9 ;-----: NO - LOOP TILL TIMEOUT
5480 G10: ;-----:
5481 IN AL,PORT_A ;-----: READ KEYBOARD SCAN CODE
5482 MOV BL,AL ;-----: SAVE SCAN CODE JUST READ
5483 MOV AL,0C8H ;-----: CLEAR KEYBOARD
5484 OUT PORT_B,AL
5485 RET ;-----: RETURN TO CALLER
5486 KBD_RESET ENDP
5487
5488 DDS PROC NEAR
5489 PUSH AX ;-----: SAVE AX
5490 MOV AX,DATA ;-----:
5491 MOV DS,AX ;-----: SET SEGMENT
5492 POP AX ;-----: RESTORE AX
5493 RET
5494 DDS ENDP
5495
5496 ;-----: CHARACTER GENERATOR GRAPHICS FOR 320X200 AND 640X200 GRAPHICS :-----:
5497
5498 ;-----:
5499 OGG 0F46EH
5500 CRT_CHANGE GEN LINE BYTE
5501 DB 0004,0004,000H,000H,000H,000H,000H,000H ; D_00
5502 DB 0TEH,0A1H,0A5H,0A1H,0RDH,099H,0A1H,0TEH ; D_01
5503 DB 0TEH,0FFH,0D9H,0FFH,0C3H,0E7H,0FFH,0TEH ; D_02
5504 DB 0E6H,0FEH,0FEH,0FEH,07CH,0E7H,038H,010H,000H ; D_03
5505 DB 010H,038H,010H,07CH,0E7H,038H,010H,000H ; D_04
5506 DB 028H,0E6H,028H,0E6H,03CH,0E7H,038H,010H,000H ; D_05
5507 DB 010H,010H,038H,07CH,0E7H,038H,010H,000H ; D_06
5508 DB 000H,000H,010H,03CH,03CH,018H,000H,000H ; D_07
5509 DB 0FFH,0FFH,0E7H,03CH,0E7H,0FFH,0FFH,0FFH ; D_08
5510 DB 000H,03CH,06H,042H,042H,06H,03CH,000H ; D_09
5511 DB 0FFH,0C3H,099H,0B0H,0B0H,099H,0C3H,0FFH ; D_0A
5512 DB 0FFH,0C3H,099H,0B0H,0B0H,099H,0C3H,0FFH ; D_0B
5513 DB 03CH,06H,06H,06H,06H,03CH,018H,0TEH,018H,000H ; D_0C
5514 DB 03FH,033H,03FH,030H,030H,070H,0FOH,0EOH,0EOH ; D_0D
5515 DB 07FH,063H,07FH,063H,063H,063H,067H,0E6H,0DC0H ; D_0E
5516 DB 099H,05AH,093H,0E7H,0E7H,03CH,05AH,099H,099H ; D_0F
5517 DB 080H,000H,0B0H,0E6H,0E6H,0E6H,0E6H,0E0H,000H ; D_10
5518 DB 010H,038H,010H,038H,010H,038H,010H,038H ; D_11
5519 DB 018H,03CH,07EH,018H,018H,0TEH,03CH,018H,000H ; D_12
5520 DB 066H,066H,066H,066H,066H,066H,000H,064H,000H ; D_13
5521 DB 07FH,0D9H,0D9H,078H,01BH,01BH,01BH,000H,000H ; D_14
5522 DB 03EH,06H,03H,06H,06H,06H,03CH,018H,0TEH,018H,000H ; D_15
5523 DB 000H,000H,000H,000H,000H,000H,000H,000H,000H ; D_16
5524 DB 018H,018H,018H,018H,018H,018H,018H,018H,018H ; D_17
5525 DB 018H,03CH,07EH,018H,018H,018H,018H,018H,018H ; D_18
5526 DB 018H,018H,018H,018H,018H,018H,018H,018H,018H ; D_19
5527 DB 000H,010H,00CH,0FEH,00CH,018H,010H,000H,000H ; D_1A
5528 DB 000H,030H,006H,0FEH,006H,030H,000H,000H,000H ; D_1B
5529 DB 000H,000H,0C0,0C0H,0C0H,0C0H,0C0H,0C0H,0C0H ; D_1C
5530 DB 000H,024H,06H,06H,06H,06H,06H,024H,000H,000H ; D_1D
5531 DB 000H,000H,000H,000H,000H,000H,000H,000H,000H ; D_1E
5532 DB 000H,0FFH,0FFH,07EH,03CH,018H,000H,000H,000H ; D_1F
5533 DB 000H,000H,000H,000H,000H,000H,000H,000H,000H ; SF_D_20
5534 DB 030H,07H,07H,030H,030H,030H,030H,030H,030H ; D_21
5535 DB 06CH,06CH,06CH,06CH,06CH,06CH,06CH,06CH,06CH ; D_22
5536 DB 0CCH,0CCH,0CCH,0CCH,0CCH,0CCH,0CCH,0CCH,0CCH ; D_23
5537 DB 030H,0CCH,0CCH,078H,078H,078H,078H,078H,078H ; D_24
5538 DB 000H,0C6H,0C6H,0C6H,0C6H,0C6H,0C6H,0C6H,0C6H ; PER-CENT_D_25
5539 DB 038H,06H,038H,076H,0DCH,0CCH,0CCH,076H,000H ; D_26
5540 DB 06H,06H,06H,0C0H,0C0H,0C0H,0C0H,0C0H,0C0H ; D_27
5541 DB 018H,030H,06H,06H,06H,06H,030H,018H,000H,000H ; D_28
5542 DB 000H,030H,030H,030H,030H,030H,030H,030H,030H ; D_29
5543 DB 000H,064H,064H,064H,064H,064H,064H,064H,064H ; D_2A
5544 DB 000H,030H,030H,030H,030H,030H,030H,030H,030H ; D_2B
5545 DB 000H,000H,000H,000H,000H,000H,000H,000H,000H ; D_2C
5546 DB 000H,000H,000H,000H,000H,000H,000H,000H,000H ; D_2D
5547 DB 000H,000H,000H,000H,000H,000H,000H,000H,000H ; D_2E
5548 DB 004H,004H,004H,004H,004H,004H,004H,004H,004H ; D_2F
5549 DB 07C4,06CDEF,E647C0 ;-----:
5550 DB 020H,010H,020H,030H,020H,020H,05CH,000H,000H ; D_30
5551 DB 078H,0C0CH,0C0CH,038H,060H,0CCH,0FCH,000H ; D_31
5552 DB 078H,0C0CH,0C0CH,038H,060H,0CCH,078H,000H ; D_32
5553 DB 01CH,03CH,06CH,0CCH,0CCH,0CCH,01EH,000H ; D_34
5554 DB 0CCH,0CCH,0CCH,0CCH,0CCH,0CCH,0CCH,0CCH,0CCH ; D_35
5555 DB 038H,060H,0CCH,0FCH,0CCH,0CCH,078H,000H ; D_36
5556 DB 0FCFH,0C0CH,0C0CH,018H,030H,030H,030H,000H ; D_37
5557 DB 078H,0C0CH,0C0CH,078H,0CCH,0CCH,078H,000H ; D_38
5558 DB 078H,0C0CH,0C0CH,07CH,0CCH,018H,070H,000H ; D_39
5559 DB 000H,030H,030H,000H,000H,000H,030H,030H,000H ; D_3A
5560 DB 000H,030H,030H,000H,000H,000H,030H,030H,000H ; D_3B
5561 DB 018H,030H,060H,0C0H,060H,030H,018H,000H,000H ; D_3C
5562 DB 000H,000H,0FCFH,000H,000H,0FCFH,000H,000H,000H ; D_3D
5563 DB 060H,030H,018H,00CH,018H,030H,060H,000H,000H ; D_3E
5564 DB 078H,0C0CH,0C0CH,018H,030H,000H,030H,000H,000H ; D_3F

```

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

FC6E 7CC6DEDEDE0C7800 5565 DB 07CH,0C6H,0DEH,0DEH,0DEH,0C0H,07BH,00H; I D_40
 FC7E 3078C0C76666FC00 5566 DB 030H,07H,0CCH,0CCH,0FCH,0CCH,0CCH,00H; I D_41
 FC7E F66666C76666FC00 5567 DB 0FCH,066H,066H,07CH,066H,066H,0FCH,00H; I D_42
 FC86 3C6660C0063C600 5568 DB 0CCH,066H,0C0H,0CCH,0CCH,0CCH,0CCH,00H; I C_D_43
 FC86 F66660C0063C600 5569 DB 0CCH,066H,0C0H,0CCH,0CCH,0CCH,0CCH,00H; I C_D_44
 FC96 FE626878686F2E00 5570 DB 0FCH,062H,068H,07BH,068H,062H,0FCH,00H; I E_45
 FC98 FE626878686F0E00 5571 DB 0FCH,062H,068H,07BH,068H,060H,0FCH,00H; I F_46
 FCA6 3C6660C0CE63E00 5572 DB 03CH,066H,0C0H,0CCH,0CCH,0CCH,0CCH,00H; I G_47
 FCB6 7B30303030307800 5574 DB 07BH,030H,030H,030H,030H,030H,07BH,00H; I I_D_49
 FCB6 F666667C6666E600 5575 DB 0CCH,066H,0C0H,0CCH,0CCH,0CCH,0CCH,00H; I J_A_49
 FCB6 E6666C786C64E600 5576 DB 0FCH,066H,0C0H,0CCH,0CCH,0CCH,0CCH,00H; I K_A_49
 FCB6 F06060606266F2E00 5577 DB 0FCH,060H,060H,060H,062H,066H,0FCH,00H; I L_D_4C
 FCD6 C6EEFFEDC6C600 5578 DB 0C6H,066H,0E0H,0FCH,0FCH,066H,0C6H,00H; I D_4D
 FCF6 C6E6F6DECEC6C600 5579 DB 0C6H,066H,0F6H,0C0H,0E0H,0C6H,0C6H,00H; N_D_4E
 FEC6 386CC6C6C63C800 5580 DB 0C6H,066H,0C0H,0C6H,0C6H,0C6H,03BH,00H; O_D_4F
 FEC6 F66667C0666C0600 5581 DB 0C6H,066H,0C0H,0C6H,0C6H,0C6H,03BH,00H; P_D_50
 FEC6 7866667C6666C000 5582 DB 0TCH,066H,0C0H,0CCH,0CCH,0CCH,0CCH,00H; Q_D_51
 FCF6 F666667C6666E600 5583 DB 0FCH,066H,066H,07CH,06CH,066H,0E64H,00H; R_D_52
 FD06 78CC0701CCTC7800 5584 DB 07BH,0CCH,0E0H,07H,01CH,0CCH,07BH,00H; I S_D_53
 FD06 FBC4B030303078000 5585 DB 0FCH,048H,030H,030H,030H,030H,07BH,00H; T_D_54
 FD16 CCCCCCCCCCCCFC000 5586 DB 0CCH,0CCH,0CCH,0CCH,0CCH,0CCH,0CCH,00H; U_D_55
 FD16 CCCCCCCCCCCCFC000 5587 DB 0CCH,0CCH,0CCH,0CCH,0CCH,0CCH,0CCH,00H; V_D_56
 FD26 666666D9FEE6C00 5588 DB 0C6H,066H,0C0H,0CCH,0CCH,0CCH,0CCH,00H; W_D_57
 FD26 C6C6C638386C6C600 5589 DB 0C6H,066H,0C6H,02BH,03BH,06CH,0C6H,00H; X_D_58
 FD36 CCCCCC7830307800 5590 DB 0CCH,0CCH,0CCH,0CCH,0CCH,0CCH,0CCH,00H; Y_D_59
 FEC6 F6C8C1826F6C00 5591 DB 0FCH,066H,0C0H,01BH,032H,066H,0FCH,00H; Z_D_5A
 FD46 7B60606060607800 5592 DB 07BH,060H,060H,060H,060H,060H,07BH,00H; I_D_5B
 FD46 C0630180C0620000 5593 DB 0C0H,060H,030H,01BH,00CH,00H,002H,00H; BACKSLASH_D_5C
 FD50 7866667C6666C000 5594 DB 0C6H,066H,0C0H,0C6H,0C6H,0C6H,0C6H,00H; CIRCUMFLX_D_5E
 FD56 10386CC0000000000 5595 DB 010H,038H,0C0H,0C6H,00H,00H,00H,00H; D_5F
 FD66 00000000000000FF 5596 DB 000H,000H,000H,000H,000H,000H,000H,00H; E_5F
 FD66 3030180000000000 5597 DB 030H,030H,01BH,00H,00H,00H,00H,00H; F_5F
 FD76 000078C077C7C600 5598 DB 000H,000H,07BH,0C0H,07CH,0CCH,07BH,00H; LOWER CASE_A_D_61
 FD76 E60607C66666C000 5599 DB 0E0H,060H,0C0H,0C6H,0C6H,0C6H,0C6H,00H; L_C_B_D_62
 FD76 F66666667C66666C00 5600 DB 01CH,066H,0C0H,0CCH,0CCH,0CCH,0CCH,00H; L_C_D_D_63
 FD96 000078CCFC078000 5601 DB 000H,000H,07BH,0C0H,07CH,0CCH,07CH,00H; L_C_E_D_64
 FD96 386C60F06060F000 5603 DB 03BH,06CH,060H,0F0H,060H,060H,0F0H,00H; L_C_F_D_66
 FDAE 000076CCCC7C0C8F 5604 DB 000H,000H,07BH,0C0H,07CH,0CCH,07CH,00H; L_C_G_D_67
 FDAE E606067C66666E600 5605 DB 0E0H,060H,0C0H,0C6H,0C6H,0C6H,0C6H,00H; L_C_H_D_68
 FDAE F66666667C66666C00 5606 DB 01CH,066H,0C0H,0C6H,0C6H,0C6H,0C6H,00H; L_C_I_D_69
 FB9E 0000000000CCCCC78 5607 DB 000H,000H,00CH,0C0H,0CCH,0CCH,0CCH,00H; L_C_J_D_6A
 FDC6 E606666C786C600 5608 DB 0E0H,060H,046H,0C0H,047H,0C6H,06EH,00H; L_C_K_D_6B
 FDCE 7030303030307800 5609 DB 070H,030H,030H,030H,030H,030H,07BH,00H; L_C_L_D_6C
 FDDE 0000CCFFEF6C600 5610 DB 000H,000H,00CH,0C0H,0E0H,0FCH,06DH,0C6H,00H; L_C_M_D_6D
 FDDE 0000F8C6666666C00 5611 DB 000H,000H,00CH,0C0H,0CCH,0CCH,0CCH,00H; L_C_N_D_6E
 FDE6 000078CCCCC7C800 5612 DB 000H,000H,07BH,0C0H,0CCH,0CCH,0CCH,00H; L_C_O_D_6F
 FDE6 000078CCCCC7C800 5613 DB 000H,000H,07BH,0C0H,0CCH,0CCH,0CCH,00H; L_C_P_D_70
 FDF6 000076CCCC7C0C8E 5614 DB 000H,000H,07BH,0C0H,0CCH,0CCH,07CH,00H; L_C_Q_D_71
 FDFE 0000DC76666F0P00 5615 DB 000H,000H,00CH,0C0H,07H,06H,06H,0F0H,00H; L_C_R_D_72
 FE00 00007C0780C8F800 5616 DB 000H,000H,07CH,0C0H,07BH,06CH,00H,0FCH,00H; L_C_S_D_73
 FE00 10307C30303141800 5617 DB 010H,030H,07CH,0C0H,030H,030H,01BH,00H; L_C_T_D_74
 FE16 0000000000CCCC1760 5618 DB 000H,000H,00CH,0C0H,0CCH,0CCH,0CCH,00H; L_C_U_D_75
 FE16 0000000000CCCC1760 5619 DB 000H,000H,00CH,0C0H,0CCH,0CCH,0CCH,00H; L_C_V_D_76
 FE26 0000C6d6FEE6C00 5620 DB 000H,000H,00CH,0C0H,0E0H,0FCH,06CH,0C6H,00H; L_C_W_D_77
 FE26 0000C66C386C600 5621 DB 000H,000H,00CH,0C0H,0CCH,0CCH,0CCH,00H; L_C_X_D_78
 FE33 0000CCCCC7C0C8F 5622 DB 000H,000H,00CH,0C0H,0CCH,0CCH,07CH,00H; L_C_Y_D_79
 FE3E 0000FC983064FC00 5623 DB 000H,000H,00CH,0C0H,09BH,030H,064H,0FCH,00H; L_C_Z_D_7A
 FE46 1C3030E030301C00 5624 DB 01CH,030H,030H,0E0H,030H,030H,01CH,00H; I_D_7B
 FE46 1C3030E030301C00 5625 DB 01CH,030H,030H,0E0H,030H,030H,01CH,00H; I_D_7C
 FE56 E030301C3030E000 5626 DB 0E0H,030H,030H,030H,030H,030H,0E0H,00H; J_D_7D
 FE56 76D00000000000000 5627 DB 07BH,0DCH,00H,00H,00H,00H,00H,00H; TILDE_D_TE
 FE66 0010386CC6C6FEE00 5628 DB 000H,010H,038H,0C6H,0C6H,0C6H,0FCH,00H; DELTA_D_TE
 5629

```

  ;-- 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 :             I NOTE: COUNTS OCCUR AT THE RATE OF
  5645 :             1193180/65536 COUNTS/SEC
  5646 :             (OR ABOUT 18.2 PER SECOND -- SEE EQUATES BELOW)
  5647 :-----ASSUME CS:CODE,DS:D1DATA
  5648 :-----ORG 0FE0EH
  5649 :-----TIME_OF_DAY PROC FAR
  5650 :-----STI
  5651 :-----CALL DDS
  5652 :-----PUSH DS
  5653 :-----CALL DDS
  5654 :-----OR AH,AH
  5655 :-----JZ T2
  5656 :-----DEC AH
  5657 :-----JZ T3
  5658 :-----T1:
  5659 :-----STI
  5660 :-----POP DS
  5661 :-----IRET
  5662 :-----T2:
  5663 :-----CLI
  5664 :-----MOV AL,TIMER_OFL
  5665 :-----MOV TIMER_OFL,0
  5666 :-----MOV CX,TIMER_HIGH
  5667 :-----MOV DX,TIMER_LOW
  5668 :-----JMP T1
  5669 :-----T3:
  5670 :-----CLI
  5671 :-----MOV TIMER_LOW,DX
  5672 :-----MOV TIMER_HIGH,CX
  5673 :-----MOV TIMER_OFL,0
  5674 :-----JMP TOD_RETURN
  5675 :-----ENDP
  
```

F64E F64E FA 5650 : INTERRUPTS BACK ON
 F64E F64E FB 5651 : SAVE SEGMENT
 F64E F64E FB 5652 : AH=0
 F64E F64E FB 5653 : READ_TIME
 F64E F64E FB 5654 : AH=AH_
 F64E F64E FB 5655 : SET_TIME
 F64E F64E FB 5656 : TOD_RETURN
 F64E F64E FB 5657 : INTERRUPTS BACK ON
 F64E F64E FB 5658 : RECOVER_SEGMENT
 F64E F64E FB 5659 : RETURN_TO_CALLER
 F64E F64E FB 5660 : READ_TIME
 F64E F64E FB 5661 : NO_TIMER_INTERRUPTS WHILE READING
 F64E F64E FB 5662 : SET_TIME
 F64E F64E FB 5663 : NO_INTERRUPTS WHILE WRITING
 F64E F64E FB 5664 : GET_OVERFLOW, AND RESET THE FLAG
 F64E F64E FB 5665 : SET_TIME
 F64E F64E FB 5666 : RESET_OVERFLOW
 F64E F64E FB 5667 : TOD_RETURN
 F64E F64E FB 5668 : SET_TIME
 F64E F64E FB 5669 : NO_INTERRUPTS WHILE WRITING
 F64E F64E FB 5670 : SET_TIME
 F64E F64E FB 5671 : NO_INTERRUPTS WHILE WRITING
 F64E F64E FB 5672 : SET_TIME
 F64E F64E FB 5673 : RESET_OVERFLOW
 F64E F64E FB 5674 : TOD_RETURN
 F64E F64E FB 5675 : ENDP

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

```

5676
5677 ; THIS ROUTINE HANDLES THE TIMER INTERRUPT FROM
5678 ; CHANNEL 0 OF THE 8254 TIMER.  INPUT FREQUENCY
5679 ; IS 1.19318 MHZ AND THE DIVISOR IS 65536, RESULTING
5680 ; IN APPROX. 16.2 INTERRUPTS EVERY SECOND.
5681 ;
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 0FEA5H
5696     TIMER_INT PROC FAR
5697     STI             ; : INTERRUPTS BACK ON
5698     PUSH DS          ; : SAVE MACHINE STATE
5699     PUSH AX
5700     PUSH DX
5701     CALL DDS
5702     INC  TIMER_LOW    ; : INCREMENT TIME
5703     JNZ  T4           ; : TEST DAY
5704     INC  TIMER_HIGH   ; : INCREMENT HIGH WORD OF TIME
5705     T4:              ; : TEST FOR COUNT EQUALING 24 HOURS
5706     CMP  TIMER_HIGH,01BH ; : TEST_DAY
5707     JNZ  T5           ; : DISKETTE_CTL
5708     CMP  TIMER_LOW,0B0H
5709     JNZ  T5           ; : DISKETTE_CTL
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_OF,L1
5717 ;
5718 ;----- TEST FOR DISKETTE TIME OUT
5719
5720     T5:              ; : DISKETTE_CTL
5721     DEC  MOTOR_COUNT
5722     JNZ  T5           ; : RETURN IF COUNT NOT OUT
5723     AND  MOTOR_STATUS,0F0H ; : TURN OFF MOTOR RUNNING BITS
5724     MOV  AL,OCH
5725     MOV  DX,03F2H
5726     OUT  DX,AL
5727     T6:              ; : FDC CTL PORT
5728     INT  ICH          ; : TURN OFF THE MOTOR
5729     MOV  AL,E01        ; : TIMER_RET;
5730     OUT  020H,AL       ; : TRANSFER CONTROL TO A USER ROUTINE
5731     POP  DX
5732     POP  AX
5733     POP  DS
5734     RET
5735     TIMER_INT ENDP
5736

```

```

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 0FFE3H
5746 VECTOR TABLE LABEL WORD ; VECTOR TABLE FOR MOVE TO INTERRUPTS
5747 DW OFFSET TIMER_INT ; INTERRUPT 8
5748 DW OFFSET KB_INT ; INTERRUPT 9
5749 DW OFFSET DI ; INTERRUPT A
5750 DW OFFSET D11 ; INTERRUPT B
5751 DW OFFSET D11 ; INTERRUPT C
5752 DW OFFSET D11 ; INTERRUPT D
5753 DW OFFSET DISK_INT ; INTERRUPT E
5754 DW OFFSET D11 ; INTERRUPT F
5755 DW OFFSET EQUIPMENT ; INTERRUPT 11H
5756 DW OFFSET MEMORY_SIZE_DET ; INTERRUPT 12H
5757 DW OFFSET DISKETTE_10_ ; INTERRUPT 13H
5758 DW OFFSET RS232_10_ ; INTERRUPT 14H
5759 DW CASSETTE_10 ; INTERRUPT 15H(FORMER CASSETTE 10)
5760 DW OFFSET KEYBOARD_10 ; INTERRUPT 16H
5761 DW OFFSET PRINTER_10 ; INTERRUPT 17H
5762 DW OFFSET PRINTER_10 ; INTERRUPT 17H
5763
FF13 0000
5764 DW 00000H ; INTERRUPT 18H
5765 ; DW 0F600H ; MUST BE INSERTED INTO TABLE LATER
5766
FF15 F2E6
5767 DW OFFSET BOOT_STRAP ; INTERRUPT 19H
FF17 6EFE
5768 DW TIME_OF_DAY ; INTERRUPT 1AH -- TIME OF DAY
FF19 4BFF
5769 DW DUMMY_RETURN ; INTERRUPT 1BH -- KEYBOARD BREAK ADDR
FF1B 4BF0
5770 DW DUMMY_RETURN ; INTERRUPT 1C -- TIMER BREAK ADDR
FF1D AAF0
5771 DW VIDEO_PARMS ; INTERRUPT 1D -- VIDEO PARAMETERS
FF1F CTEF
5772 DW OFFSET DISK_BASE ; INTERRUPT 1E -- DISK PARMS
FF21 0000
5773 DW 0 ; INTERRUPT 1F -- POINTER TO VIDEO EXT
5774
5775 ;-----+
5776 ; TEMPORARY INTERRUPT SERVICE ROUTINE
5777 ; 1. THIS ROUTINE IS ALSO LEFT IN PLACE AFTER THE :
5778 ; POWER ON DIAGNOSTICS TO SERVICE UNUSED :
5779 ; INTERRUPT VECTORS. INITIATION 'INTR_FLAG' WILL :
5780 ; CONTAIN EITHER 1. LEVEL OF HARDWARE INT. THAT :
5781 ; CAUSED CODE TO BE EXECUTED. :
5782 ; 2. 'FF' FOR NON-HARDWARE INTERRUPTS THAT WAS :
5783 ; EXECUTED ACCIDENTLY. :
5784
5785 D11
5786 PUSH DS;CLEAR
5787 ASSUME DS:DATA
5788 PUSH DS
5789 PUSH AX
5790 CALL DDS ; SAVE REG AX CONTENTS
5791 MOV AL,0BH ; READ IN-SERVICE REG
FF23 1E
5792 OUT INTA00,AL ; FIND OUT WHAT LEVEL BEING
5793 NOP ; SERVICED)
FF25 50
5794 IN AL,INTA00 ; GET LEVEL
FF26 E830FB
5795 MOV AH,AL ; SAVE IT
FF29 B00B
5796 OR AL,AH ; 00? (NO HARDWARE ISR ACTIVE)
FF31 E620
5797 JNZ HW_INT
FF32 B04C
5798 MOV AH,0FFH
FF33 B4FF
5799 JMP SHORT_SET_INTR_FLAG ; SET FLAG TO FF IF NON-HWADRE
FF34 E80A
5800 HW_INT: ; MASK OFF LVL BEING SERVICED
FF3A E421
5801 IN AL,INTA01 ; GET MASK VALUE
FF3C 0AC4
5802 OR AL,AH
FF3E E621
5803 OUT INTA01,AL
FF3F B020
5804 MOV AL,EDH
FF42 E620
5805 OUT INTA00,AL
FF44
5806 SET_INTR_FLAG: ; SET FLAG
FF44 88266B00
5807 MOV INTR_FLAG,AH ; RESTORE REG AX CONTENTS
FF48 58
5808 POP AX
FF49 5A
5809 POP DX
FF4B 1F
5810 POP DS
FF4B CF
5811 DUMMY_RETURN: ; NEED IRET FOR VECTOR TABLE
5812 IRET
5813 D11 ENDP
5814
5815 ;-----+
5816 ; DUMMY RETURN FOR ADDRESS COMPATIBILITY :
5817 ;-----+
FF53
5818 ORG 0FF53H
FF53 CF
5819 IRET
5820

```

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

```

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 DERESSED 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 RETURN FROM 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
5840 PUSH DS
5841 PUSH AX
5842 PUSH BX
5843 PUSH CX
5844 PUSH DX
5845 MOV AX,XXDATA
5846 MOV DS,AX
5847 CMP STATUS_BYTE,I
5848 JZ EXIT
5849 MOV STATUS_BYTE,I
5850 MOV AH,15
5851 INT 10H
5852 INT 10H
5853 INT 10H
5854 ;-----[AT THIS POINT WE KNOW THE COLUMNS/LINE ARE IN :]
5855 ; [AX] AND THE PAGE IF APPLICABLE ARE IN[BH]. THE STACK :]
5856 ; HAS DS,AX,BX,CX,DX PUSHED. [A] HAS VIDEO MODE :]
5857 ; [AH]=NUMBER COLUMNS/LINE :]
5858 ; [BH]=VISUAL PAGE :]

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 ; SET ROW NEAR 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 ;-----[THE LOOP FROM PRI10 TO THE INSTRUCTION PRI120 :]
5869 ; IS THE LOOP TO READ EACH CURSOR POSITION FROM THE :]
5870 ; SCREEN AND PRINT. :]
5871 ;-----[SCROLLING :]
5872 ;-----[PRI10: :]

5873 PRI10:
5874 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 PRI15:
5882 PUSH DX ; SAVE CURSOR POSITION
5883 XOR DX,DX ; INDICATE PRINTER !
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 PRINT ERROR
5888 JZ END10 ; JUMP IF ERROR DETECTED
5889 INC DL ; ADVANCE TO NEXT COLUMN
5890 CMP CL,DL ; SEE IF AT END OF LINE
5891 JNZ PRI10 ; IF NOT PROCEEDED
5892 XOR DL,DL ; BACK TO COLUMN 0
5893 MOV AH,DL ; [DL]=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 PRI120:
5901 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 ;-----[ERR10: :]

5907 POP DX ; GET CURSOR POSITION
5908 MOV AH,2 ; TO REQUEST CURSOR SET
5909 INT 10H ; CURSOR POSITION RESTORED
5910 ERR20:
5911 MOV STATUS_BYTE,0FFH ; INDICATE ERROR
5912 EXIT: RET ;-----[RESTORE ALL THE REGISTERS USED
5913 POP DX
5914 POP CX
5915 POP BX
5916 POP AX
5917 POP DS
5918 INT 10H ;-----[PRINT_SCREEN ENDP
5919 PRINT_SCREEN ENDP
5920 ;-----[CARRIAGE RETURN, LINE FEED SUBROUTINE
5921 ;
5922 ;
5923 CRLF PROC NEAR
5924 XOR DX,DX ;-----[PRINTER 0
5925 XOR AH,AH ;-----[WILL NOW SEND INITIAL LF,CR
5926 ;
5927 MOV AL,12Q ;-----[LF
5928 INT 17H ;-----[SEND THE LINE FEED
5929 MOV AH,AH ;-----[NOW FOR THE CR
5930 MOV AL,15Q ;-----[CR
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 0AC6 5939 PRT_SEG PROC NEAR  
FFDC E8ACF9 5940 CALL AL,DX ;GET MSB  
FFDF 8AC2 5941 XPC_BYTE  
FFE1 E8A7F9 5942 MOV AL,DL ;LSB  
FFE4 B030 5943 CALL XPC_BYTE  
FFE6 E8B3F9 5944 MOV AL,T0' ; PRINT A '0'  
FFE9 B020 5945 CALL PRT_HEX  
FFEB E8AEF9 5946 MOV AL,T' ;SPACE  
FFEE C9 5947 CALL PRT_HEX  
5948 RET  
5949 PRT_SEG ENDP  
5950 CODE ENDS  
5952 -----  
5953 -----  
5954 ;----- POWER ON RESET VECTOR :  
5955 ;-----  
5956 VECTOR SEGMENT AT 0FFFFH  
5957 -----  
5958 ;----- POWER ON RESET  
5959 -----  
0000 EA5BE000F0 5960 JMP RESET  
0005 31312F30382F38 5961  
32 5962 DB '11/08/82' ; RELEASE MARKER  
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

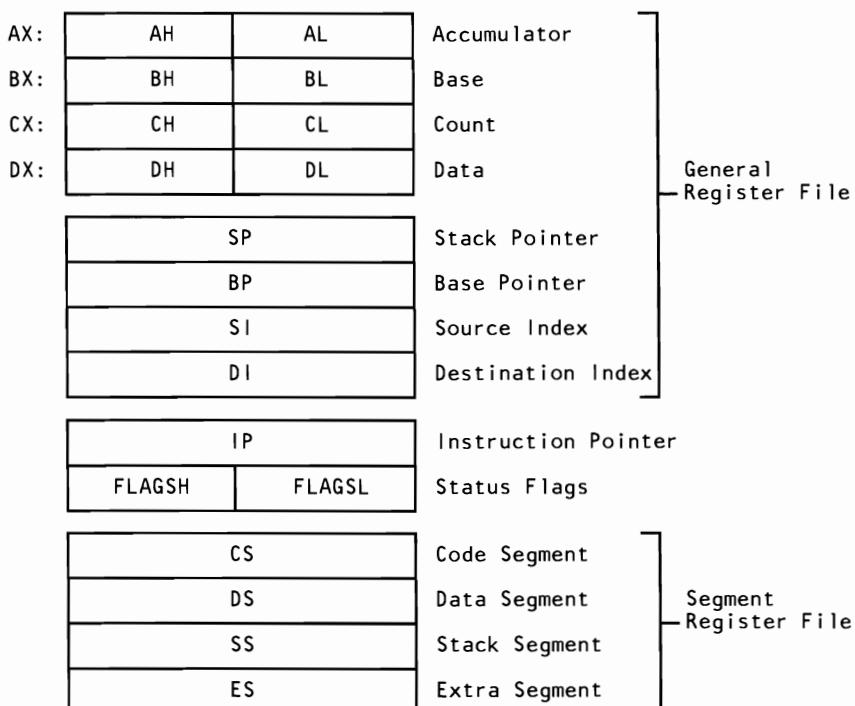
DS = Data segment

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

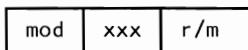
AF: Auxiliary Carry - BCD	8080 Flags
CF: Carry Flag	
PF: Parity Flag	
SF: Sign Flag	
ZF: Zero Flag	8088 Flags
DF: Direction Flag	
IF: Interrupt Enable Flag	
OF: Overflow Flag (CF + SF)	
TF: Trap-Single Step Flag	

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	

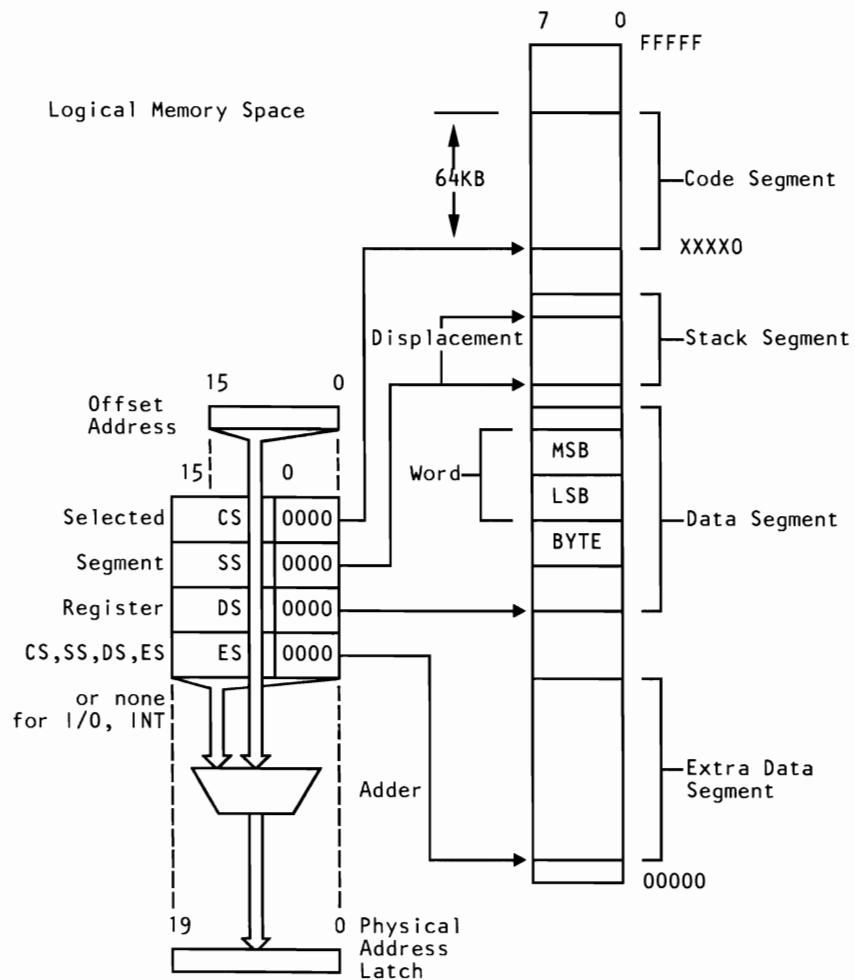
Second Instruction Byte Summary



mod	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

LODS = Load String

1010110w

STOS = Store String

1010101w

Control Transfer

CALL = Call

Direct within Segment

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

Indirect within Segment

11111111	mod 010 r/m
----------	-------------

Direct Intersegment

10011010	offset-low	offset-high
----------	------------	-------------

seg-low	seg-high
---------	----------

Indirect Intersegment

11111111	mod 011 r/m
----------	-------------

JMP = Unconditional Jump

Direct within Segment-Short

11101011	disp
----------	------

Indirect within Segment

11111111	mod 100 r/m
----------	-------------

Direct Intersegment

11101010	offset-low	offset-high
----------	------------	-------------

seg-low	seg-high
---------	----------

Indirect Intersegment

11111111	mod 101 r/m
----------	-------------

RET = Return from Call

Within Segment

11000011

Within Segment Adding Immediate to SP

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

Intersegment

11001011

Intersegment Adding Immediate to SP

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

JE/JZ = Jump on Equal/Zero

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

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

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

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

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

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

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

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

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

JP/JPE = Jump on Parity/Parity Even

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

JO = Jump on Overflow

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

JS = Jump on Sign

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

JNE/JNZ = Jump on Not Equal/Not Zero

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

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

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

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

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

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

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

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

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

JNP/JPO = Jump on Not Parity/Parity Odd

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

JNO = Jump on Not Overflow

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

JNS = Jump on Not Sign

01111001	disp
----------	------

LOOP = Loop CX Times

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

LOOPZ/LOOPE = Loop while Zero/Equal

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

LOOPNZ/LOOPNE = Loop while Not Zero/Not Equal

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

JCXZ = Jump on CX Zero

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

8088 Instruction Set Matrix

L0	0	1	2	3	4	5	6	7
H1 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
H1 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 1,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 1,(1+SP)	RET 1	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 1,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 1,id	JMP id	JMP 1,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

MF	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
------------	----------

FCHS = Change Sign of ST(0)

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

Transcendental

FPTAN = Partial Tangent of ST(0)

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

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

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

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

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

FYL2X = ST(1) x Log₂ [ST(0)]

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

FYL2XP1 = ST(1) x Log₂ [ST(0) + 1]

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

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 Log₂ 10 into ST(0)

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

FLDLG2 = Load Log₁₀ 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) = ith register below Stack top

d = Destination

0—Destination is ST(0)

1—Destination is ST(i)

P = POP

0—No Pop

1—Pop ST(0)

R = Reverse

0—Destination (op) Source

1—Source (op) Destination

For **FSQRT**: $-0 \leq ST(0) \leq +\infty$

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

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

For **FYL2X**: $0 < ST(0) < \infty - \infty < ST(1) < +\infty$

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

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

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

Notes:

SECTION 7. CHARACTERS, KEYSTROKES, AND COLORS

Character Codes	7-3
Quick Reference	7-14

Notes:

Character Codes

Value		As Characters			As Text Attributes		
					Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground	
00	0	Blank (Null)	Ctrl 2		Black	Black	Non-Display
01	1	☺	Ctrl A		Black	Blue	Underline
02	2	☻	Ctrl B		Black	Green	Normal
03	3	♥	Ctrl C		Black	Cyan	Normal
04	4	♦	Ctrl D		Black	Red	Normal
05	5	♣	Ctrl E		Black	Magenta	Normal
06	6	♠	Ctrl F		Black	Brown	Normal
07	7	●	Ctrl G		Black	Light Grey	Normal
08	8	•	Ctrl H, Backspace, Shift Backspace		Black	Dark Grey	Non-Display
09	9	○	Ctrl I		Black	Light Blue	High Intensity Underline
0A	10	○	Ctrl J, Ctrl ←		Black	Light Green	High Intensity
0B	11	♂	Ctrl K		Black	Light Cyan	High Intensity
0C	12	♀	Ctrl L		Black	Light Red	High Intensity
0D	13	♪	Ctrl M, ↓ Shift ←		Black	Light Magenta	High Intensity
0E	14	♫	Ctrl N		Black	Yellow	High Intensity
0F	15	☼	Ctrl O		Black	White	High Intensity
10	16	►	Ctrl P		Blue	Black	Normal
11	17	◀	Ctrl Q		Blue	Blue	Underline
12	18	↑	Ctrl R		Blue	Green	Normal
13	19	!!	Ctrl S		Blue	Cyan	Normal
14	20	¶	Ctrl T		Blue	Red	Normal
15	21	§	Ctrl U		Blue	Magenta	Normal
16	22	▬	Ctrl V		Blue	Brown	Normal
17	23	▬	Ctrl W		Blue	Light Grey	Normal

Value		As Characters				As Text Attributes		
						Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground		
18	24	↑	Ctrl X		Blue	Dark Grey	High Intensity	
19	25	↓	Ctrl Y		Blue	Light Blue	High Intensity Underline	
1A	26	→	Ctrl Z		Blue	Light Green	High Intensity	
1B	27	←	Ctrl [, Esc, Shift Esc, 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	Underline
72	114	r	r	Note 5	Light Grey	Green	Normal
73	115	s	s	Note 5	Light Grey	Cyan	Normal
74	116	t	t	Note 5	Light Grey	Red	Normal
75	117	u	u	Note 5	Light Grey	Magenta	Normal
76	118	v	v	Note 5	Light Grey	Brown	Normal
77	119	w	w	Note 5	Light Grey	Light Grey	Normal
78	120	x	x	Note 5	Light Grey	Dark Grey	Reverse Video
79	121	y	y	Note 5	Light Grey	Light Blue	High Intensity Underline
7A	122	z	z	Note 5	Light Grey	Light Green	High Intensity
7B	123	{	{	Shift	Light Grey	Light Cyan	High Intensity
7C	124			Shift	Light Grey	Light Red	High Intensity
7D	125	}	}	Shift	Light Grey	Light Magenta	High Intensity
7E	126	~	~	Shift	Light Grey	Yellow	High Intensity
7F	127	△	Ctrl ←		Light Grey	White	High Intensity

Value		As Characters			As Text Attributes		
					Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground	
* * * * 80 to FF Hex are Flashing In both Color & IBM Monochrome * * * *							
80	128	Ç	Alt 128	Note 6	Black	Black	Non-Display
81	129	ü	Alt 129	Note 6	Black	Blue	Underline
82	130	é	Alt 130	Note 6	Black	Green	Normal
83	131	â	Alt 131	Note 6	Black	Cyan	Normal
84	132	ä	Alt 132	Note 6	Black	Red	Normal
85	133	à	Alt 133	Note 6	Black	Magenta	Normal
86	134	å	Alt 134	Note 6	Black	Brown	Normal
87	135	ç	Alt 135	Note 6	Black	Light Grey	Normal
88	136	è	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		
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	
CO	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	
DO	208	[[]]	Alt 208	Note 6	Magenta	Black	Normal	

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

Value		As Characters			As Text Attributes			
					Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter	
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground		
EC	236	∞	Alt 236	Note 6	Brown	Light Red	High Intensity	
ED	237	ϕ	Alt 237	Note 6	Brown	Light Magenta	High Intensity	
EE	238	ϵ	Alt 238	Note 6	Brown	Yellow	High Intensity	
EF	239	\cap	Alt 239	Note 6	Brown	White	High Intensity	
F0	240	\equiv	Alt 240	Note 6	Light Grey	Black	Reverse Video	
F1	241	\pm	Alt 241	Note 6	Light Grey	Blue	Underline	
F2	242	\geq	Alt 242	Note 6	Light Grey	Green	Normal	
F3	243	\leq	Alt 243	Note 6	Light Grey	Cyan	Normal	
F4	244	\cup	Alt 244	Note 6	Light Grey	Red	Normal	
F5	245	\cup	Alt 245	Note 6	Light Grey	Magenta	Normal	
F6	246	\div	Alt 246	Note 6	Light Grey	Brown	Normal	
F7	247	\approx	Alt 247	Note 6	Light Grey	Light Grey	Normal	
F8	248	\circ	Alt 248	Note 6	Light Grey	Dark Grey	Reverse Video	
F9	249	\bullet	Alt 249	Note 6	Light Grey	Light Blue	High Intensity Underline	
FA	250	\bullet	Alt 250	Note 6	Light Grey	Light Green	High Intensity	
FB	251	$\sqrt{-}$	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	△

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. ($1,073,741,824 = 2^{30}$ 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. ($1,048,576 = 2^{20}$ to the 20th power.)

mA. Milliampere; 0.001 ampere.

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

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

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

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

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

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

masked. Synonym for disabled.

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

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

MB. 1,048,576 bytes.

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

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

memory. Term for main storage.

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

MFM. Modified frequency modulation.

MHz. Megahertz; 1,000,000 hertz.

micro (μ). Prefix 0.000,001.

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

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

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

microsecond (μs). 0.000,001 second.

milli (m). Prefix 0.001.

milliampere (mA). 0.001 ampere.

millisecond (ms). 0.001 second.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

ms. Millisecond; 0.001 second.

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

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

n. Prefix nano; 0.000,000,001.

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

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

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

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

negative true. Synonym for active low.

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

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

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

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

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

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

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

ns. Nanosecond; 0.000,000,001 second.

NUL. The null character.

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

odd-even check. Synonym for parity check.

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

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

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

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

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

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

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

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

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

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

overcurrent. A current of higher than specified strength.

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

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

overvoltage. A voltage of higher than specified value.

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

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

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

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

PEL. Picture element.

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

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

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

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

port. An access point for data entry or exit.

positive true. Synonym for active high.

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

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

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

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

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

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

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

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

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

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

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

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

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

PROM. Programmable read-only memory.

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

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

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

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

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

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

RAM. Random access memory. Read/write memory.

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

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

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

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

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

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

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

red-green-blue-intensity (RGBI). The description of a direct-drive color monitor that accepts input signals of red, green, blue, and intensity.

redundancy check. A check that depends on extra characters attached to data for the detection of errors. See cyclic redundancy check.

register. (1) A storage device, having a specified storage capacity such as a bit, a byte, or a computer word, and usually intended for a special purpose. (2) A storage device in which specific data is stored.

retry. To resend the current block of data (from the last EOB or ETB) a prescribed number of times, or until it is entered correctly or accepted.

reverse video. A form of highlighting a character, field, or cursor by reversing the color of the character, field, or cursor with its background; for example, changing a red character on a black background to a black character on a red background.

RF. Radio frequency.

RF modulator. The device used to convert the composite video signal to the antenna level input of a home TV.

RGBI. Red-green-blue-intensity.

rising edge. Synonym for positive-going edge.

ROM. Read-only memory.

ROM/BIOS. The ROM resident basic input/output system, which provides the level control of the major I/O devices in the computer system.

row address strobe (RAS). A signal that latches the row address in a memory chip.

RS-232C. A standard by the EIA for communication between computers and external equipment.

RTS. Request to send. Associated with modem control.

run. A single continuous performance of a computer program or routine.

saturation. In computer graphics, the purity of a particular hue. A color is said to be saturated when at least one primary color (red, blue, or green) is completely absent.

scaling. In computer graphics, enlarging or reducing all or part of a display image by multiplying the coordinates of the image by a constant value.

schematic. The representation, usually in a drawing or diagram form, of a logical or physical structure.

Schottky TTL. A version (S series) of TTL with faster switching speed, but requiring more power. See also transistor-transistor logic and low power Schottky TTL.

SDL. Shielded Data Link

SDLC. Synchronous Data Link Control.

sector. That part of a track or band on a magnetic drum, a magnetic disk, or a disk pack that can be accessed by the magnetic heads in the course of a predetermined rotational displacement of the particular device.

SERDES. Serializer/deserializer.

serial. (1) Pertaining to the sequential performance of two or more activities in a single device. In English, the modifiers serial and parallel usually refer to devices, as opposed to sequential and consecutive, which refer to processes. (2) Pertaining to the sequential or consecutive occurrence of two or more related activities in a single device or channel. (3) Pertaining to the sequential processing of the individual parts of a whole, such as the bits of a character or the characters of a word, using the same facilities for successive parts. (4) Contrast with parallel.

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-(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
 FDISH 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
 FSSCALE 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

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

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

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
Overrun command 4-27

P

parameter passing (ROM
BIOS) 5-4
software interrupt
 listing 5-5
pause 4-11

pause key 4-42
POP 6-8
POPF 6-9
POR (power-on reset) 4-25
power good signal 3-5, 3-8
power-on reset 4-25
power-on routine 4-25
 basic assurance test 4-25
 BAT (basic assurance test) 4-25
 POR (power-on reset) 4-25
 power-on reset 4-25
power-on self test 4-4
power requirements 4-51
power supply
 connectors 1-32
power supply (system) 3-3
 connectors 3-6, 3-9
 input requirements 3-4,
 3-7
 outputs 3-4, 3-8
 overvoltage/overcurrent protection 3-5
pin assignments 3-6, 3-9
power good signal 3-5,
 3-8
PPI 1-26
print screen key 4-11, 4-42
priorities, shift key 4-41
processor control, 8087 6-29
Programmable Peripheral Interface 1-26
protocol 4-25
PUSH 6-7
PUSHF 6-9

Q

QS0 (coprocessor) 2-4
QS1 (coprocessor) 2-4
quick reference charts 7-14
quick reference, character set 7-14

R

RAM modules 1-12
RAM subsystem 1-12
rate, typematic 4-24
RCL 6-14
RCR 6-14
read command I/O
 channel 1-22
read memory command (-MEMR) 1-23
ready (RDY), I/O
 channel 1-22
real numbers
 (coprocessor) 2-3, 2-4
refresh request
 generator 1-10
REP 6-15
request interrupt 2 to 7
 (IRQ2-IRQ7) 1-22
reserved interrupts
 BASIC and DOS 5-7
Reset command 4-26
RESET DRV, I/O
 channel 1-23
reset, power-on 4-25
reset, system 4-42
RET 6-17
ROL 6-13
ROM scan codes 4-33
ROM subsystem 1-13

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

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

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

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

U

UK English keyboard 4-17,
4-49
US English keyboard 4-18,
4-50

)

)

)



The Personal Computer
Hardware Reference Library

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

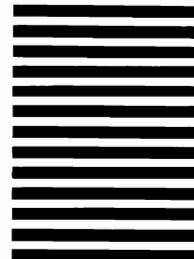
FIRST CLASS

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