

IBM

*Personal Computer
Hardware Reference
Library*

Technical Reference

6139821



IBM

*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 1

Section 2. Coprocessor

SECTION 2

Section 3. Power Supplies

SECTION 3

Section 4. Keyboards

SECTION 4

Section 5. System BIOS

SECTION 5

Section 6. Instruction Set

SECTION 6

Notes:



Section 7. Characters, Keystrokes, and Colors

SECTION 7

Glossary

GLOSSARY

Bibliography

BIBLIOGRAPHY

Index

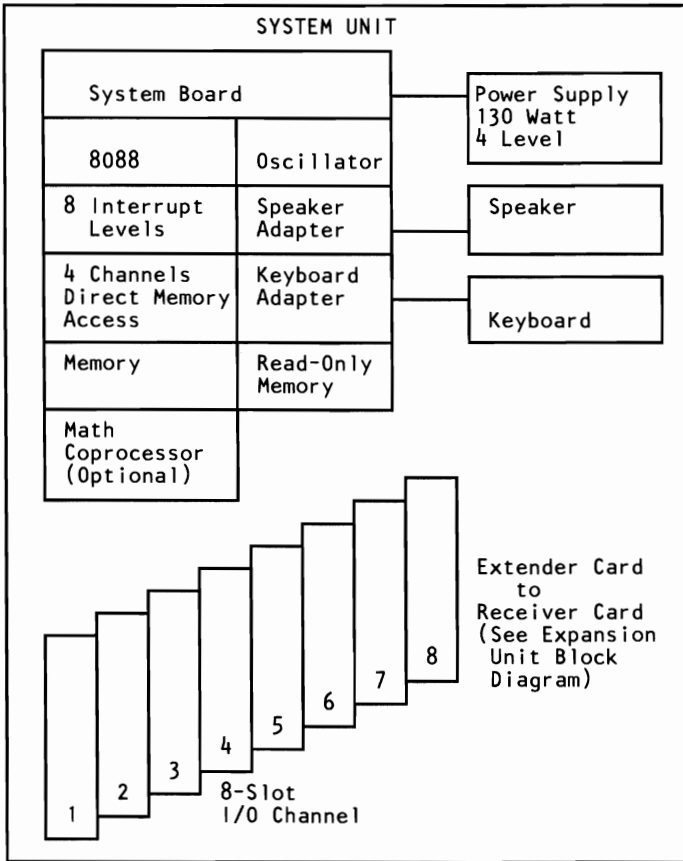
INDEX

Notes:



System Block Diagram (XT)

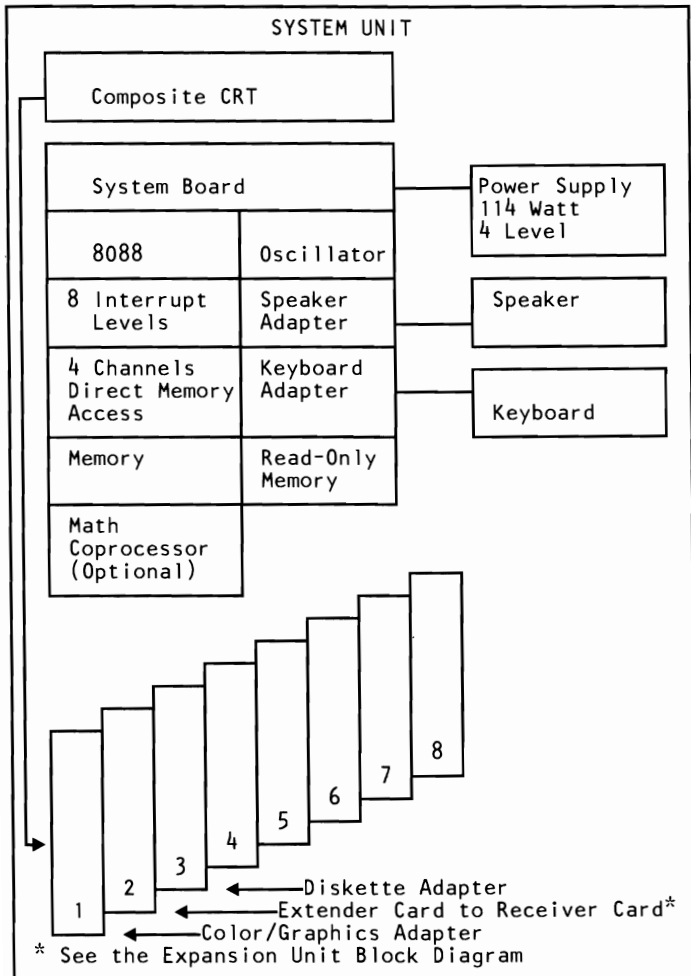
The following is a system block diagram of the IBM Personal Computer XT.



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

System Block Diagram (Portable)

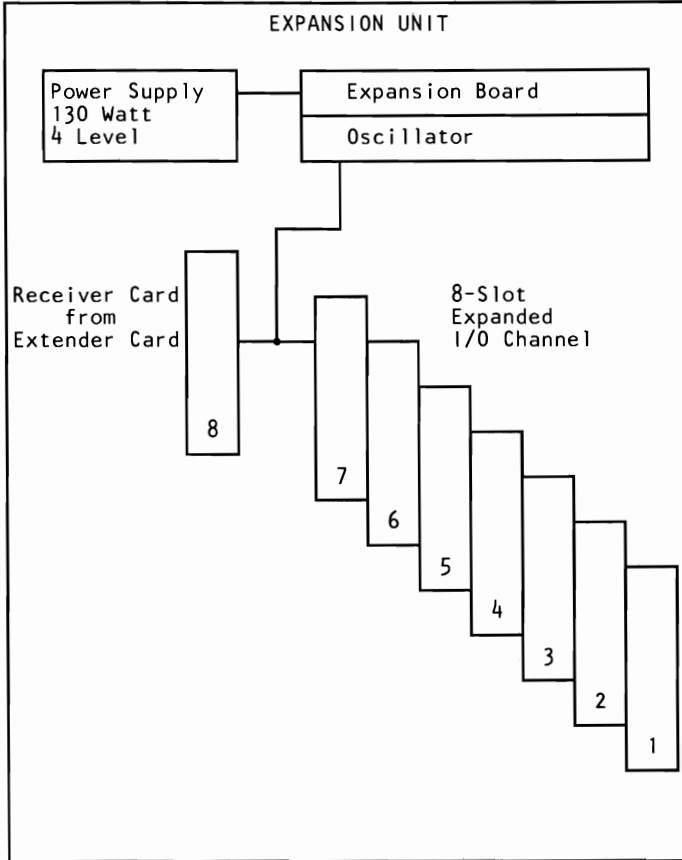
The following is a system block diagram of the IBM Portable Personal Computer.



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

Expansion Unit Block Diagram

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



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

Notes:



SECTION 1. SYSTEM BOARD

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

Notes:



Description

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

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

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

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

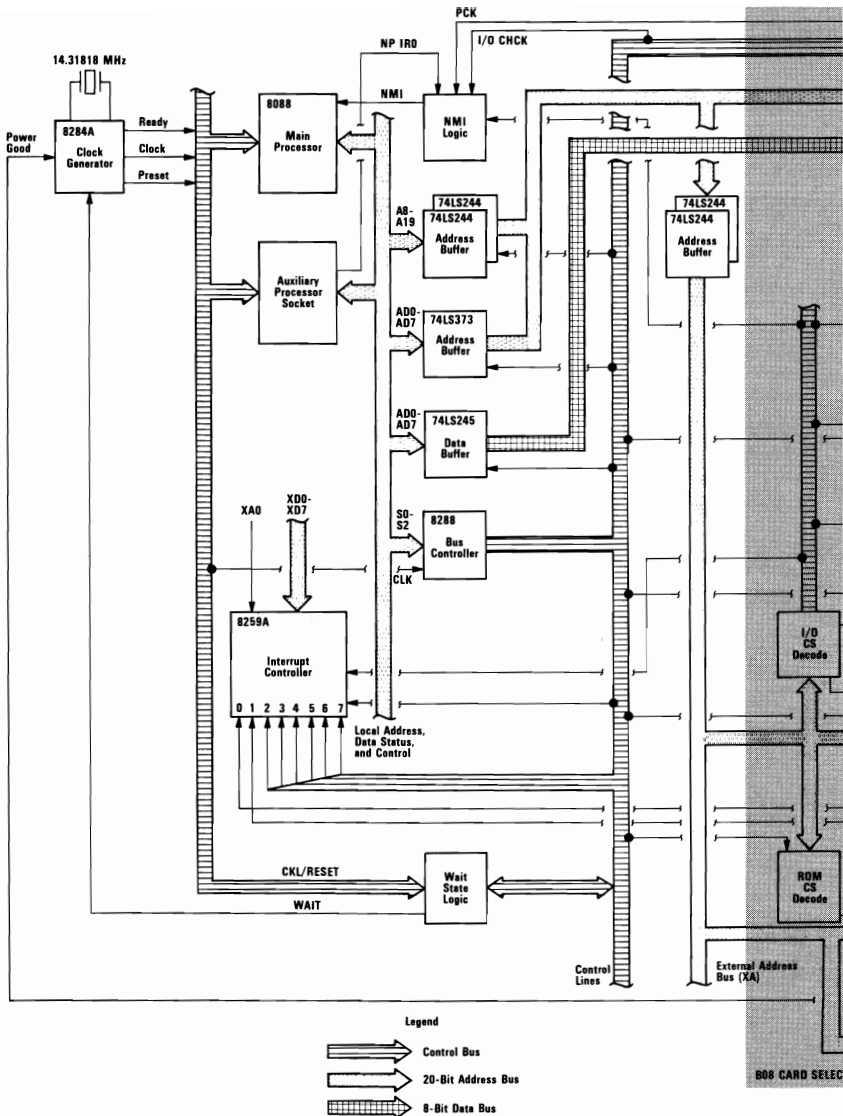
Microprocessor

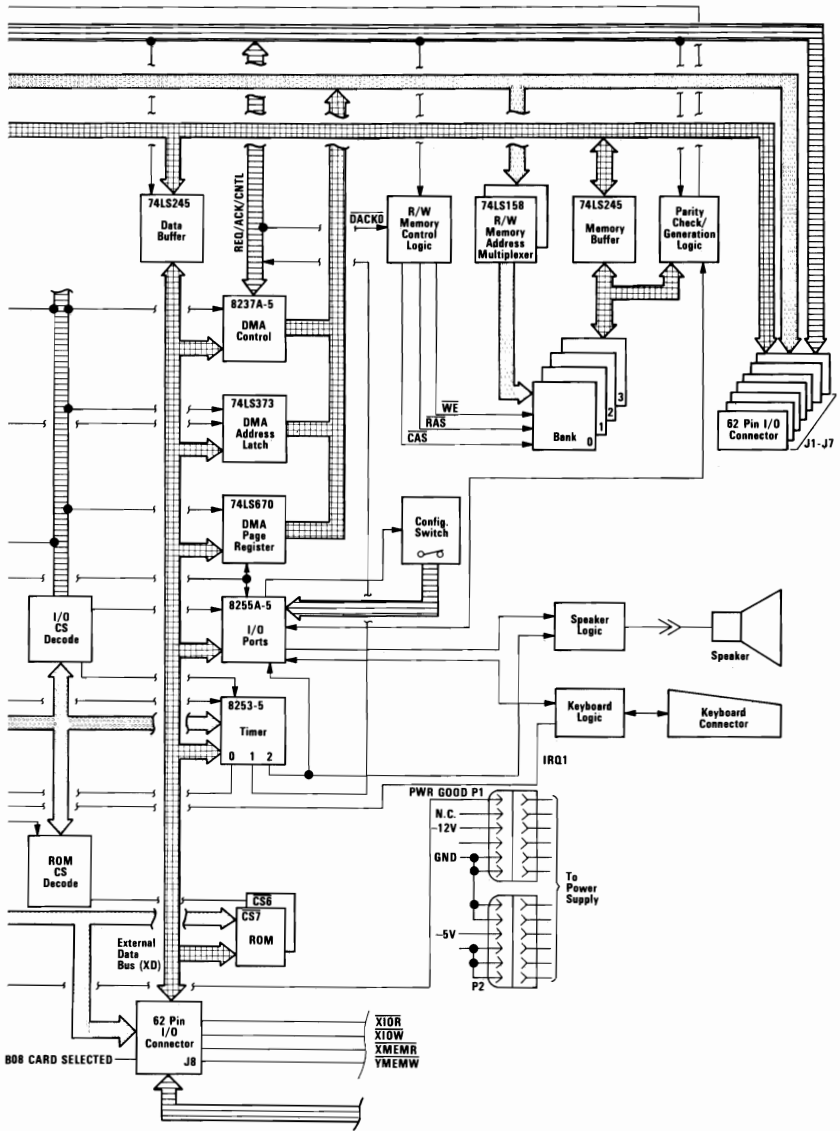
The heart of the system board is the Intel 8088 Microprocessor. This is an 8-bit external-bus version of Intel's 16-bit 8086 Microprocessor, and is software-compatible with the 8086. Thus, the 8088 supports 16-bit operations, including multiply and divide, and 20 bits of addressing (1M byte of storage).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

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

Data Flow Diagrams

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





System Memory Map

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

System Memory Map (Part 1 of 2)

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

System Memory Map (Part 2 of 2)

System Timers

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

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

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

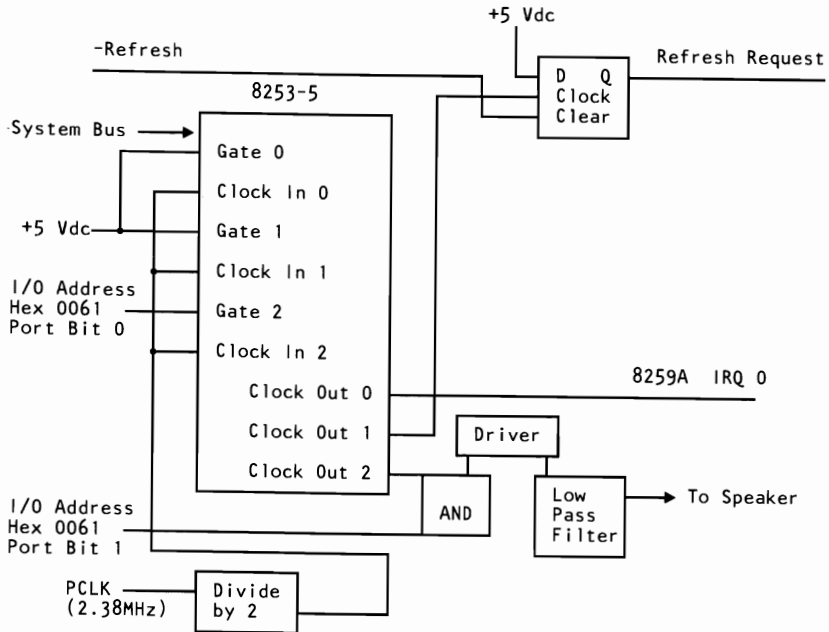
Channel 1	Refresh Request Generator
GATE 1	Tied on
CLK IN 1	1.193182 MHz OSC
CLK OUT 1	Request refresh cycle

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

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

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

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



System-Timer Block Diagram

System Interrupts

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

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

The following diagram contains the System Interrupt Listing.

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

* Jumper selectable to 2, 3, 4, 7.

8088 Hardware Interrupt Listing

System Boards

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

RAM

64/256K System Board

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

256/640K System Board

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

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

ROM

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

DMA

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

Three of the four DMA channels are available on the I/O bus and support high-speed data transfers between I/O devices and memory without microprocessor intervention. The fourth DMA channel is programmed to refresh the system's dynamic memory. This is done by programming a channel of the timer/counter device to periodically request a dummy DMA transfer. This action creates a memory-read cycle, which is available to refresh dynamic memory both on the system board and in the system expansion slots. DMA data transfers take five clock cycles of 210ns, or 1.05 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 X 1

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

256K X 1

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

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.

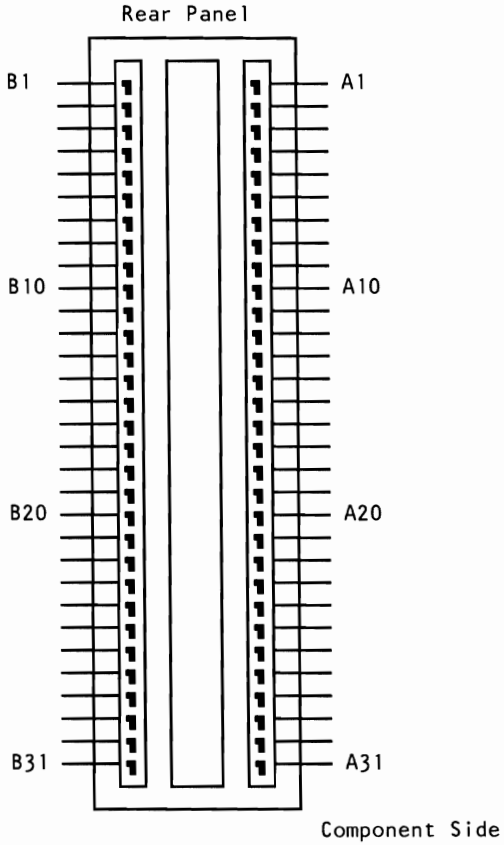
A 'channel check' line exists for reporting error conditions to the microprocessor. Activating this line results in a non-maskable interrupt (NMI) to the 8088 microprocessor. Memory expansion options use this line to report parity errors.

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

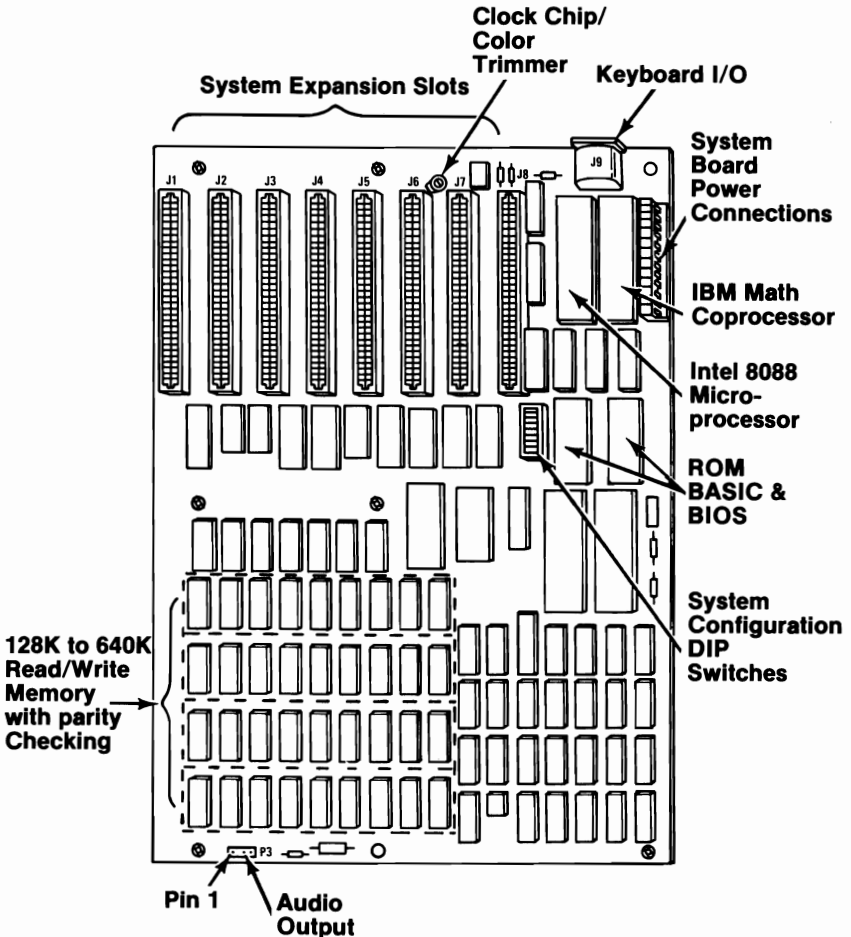
I/O Channel (A-Side, J1 through J8)

I/O Pin	Signal Name	I/O
B1	GND	Ground
B2	RESET DRV	0
B3	+5 Vdc	Power
B4	IRQ 2	I
B5	-5 Vdc	Power
B6	DRQ2	I
B7	-12 Vdc	Power
B8	-CARD SLCTD	I
B9	+12 Vdc	Power
B10	GND	Ground
B11	-MEMW	0
B12	-MEMR	0
B13	-IOW	I/O
B14	-IOR	I/O
B15	-DACK3	0
B16	DRQ3	I
B17	-DACK1	0
B18	DRQ1	I
B19	-DACK0	I/O
B20	CLK	0
B21	IRQ7	I
B22	IRQ6	I
B23	IRQ5	I
B24	IRQ4	I
B25	IRQ3	I
B26	-DACK2	0
B27	T/C	0
B28	ALE	0
B29	+5Vdc	Power
B30	OSC	0
B31	GND	Ground

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

System Board Diagram

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



System Board Component Diagram

I/O Channel Description

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

A0–A19 (O)

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

AEN (O)

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

ALE (O)

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

-CARD SLCTD (I)

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

CLK (O)

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

D0—D7 I/O

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

-DACK0 to -DACK3 (O)

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

DRQ1—DRQ3 (I)

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

-I/O CH CK (I)

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

I/O CH RDY (I)

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

-IOR (O)

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

-IOW (O)

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

IRQ2—IRQ7 (I)

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

-MEMR (O)

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

-MEMW (O)

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

OSC (O)

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

RESET DRV (O)

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

T/C (O)

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

I/O Address Map

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

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

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

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

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

Planar I/O Address Map

Hex Range*	Device
200-20F	Game Control
201	Game I/O
20C-20D	Reserved
210-217	Expansion Unit
21F	Reserved
278-27F	Parallel printer port 2
280-2DF	Alternate Enhanced Graphics Adapter
2E1	GPiB (Adapter 0)
2E2 & 2E3	Data Acquisition (Adapter 0)
2F8-2FF	Serial port 2
300-31F	Prototype card
320-32F	Fixed Disk
348-357	DCA 3278
360-367	PC Network (low address)
368-36F	PC Network (high address)
378-37F	Parallel printer port 1
380-38F***	SDLC, bisynchronous 2
390-393	Cluster
3A0-3AF	Bisynchronous 1
3B0-3BF	Monochrome Display and Printer Adapter
3C0-3CF	Enhanced Graphics Adapter
3D0-3DF	Color/Graphics Monitor Adapter
3F0-3F7	Diskette controller
3F8-3FF	Serial port 1
6E2 & 6E3	Data Acquisition (Adapter 1)
790-793	Cluster (Adapter 1)
AE2 & AE3	Data Acquisition (Adapter 2)
B90-B93	Cluster (Adapter 2)
EE2 & EE3	Data Acquisition (Adapter 3)
1390-1393	Cluster (Adapter 3)
22E1	GPiB (Adapter 1)
2390-2393	Cluster (Adapter 4)
42E1	GPiB (Adapter 2)
62E1	GPiB (Adapter 3)
82E1	GPiB (Adapter 4)
A2E1	GPiB (Adapter 5)
C2E1	GPiB (Adapter 6)
E2E1	GPiB (Adapter 7)

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

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

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

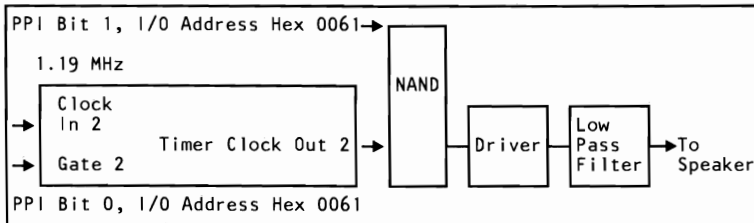
Channel I/O Address Map

Other Circuits

Speaker Circuit

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

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



Speaker Drive System Block Diagram

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

Speaker Tone Generation

The speaker connection is a 4-pin Berg connector.

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

Speaker Connector (P3)

8255A I/O Bit Map

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

Hex Port Number 0060	I N P U T	PA0	+ Keyboard Scan Code	0 1 2 3 4 5 6 7	Or	Diagnostic Outputs	0 1 2 3 4 5 6 7		
		O U T P U T	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)		1 2 3 4 5 6 7	Sw-1 Sw-2 Sw-3 Sw-4	Display 0 Display 1 # Drives # Drives 1	**Sw-5 **Sw-6 ***Sw-7 ***Sw-8
			I N P U T	PC0		Loop on POST + CoProcessor Installed + Planar RAM Size 0 + Planar RAM Size 1 Spare + Timer Channel 2 Out + I/O Channel Check + RAM Parity Check			1 2 3 4 5 6 7

0063 Command/Mode Register Hex 99

Mode Register Value

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

* Sw-4	Sw-3	Amount of Memory on System Board - 64/256K
0	0	64K
0	1	128K
1	0	192K
1	1	256K
* Sw-4	Sw-3	Amount of Memory on System Board - 256/640K
0	0	256K
0	1	512K
1	0	576K
1	1	640K
** Sw-6	Sw-5	Display at Power-Up Mode
0	0	Reserved
0	1	Color 40 X 25 (BW Mode)
1	0	Color 80 X 25 (BW Mode)
1	1	IBM Monochrome 80 X 25
*** Sw-8	Sw-7	Number of Diskette Drives in System
0	0	1
0	1	2
1	0	3
1	1	4

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

8255A I/O Bit Map

Specifications

System Unit

Size

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

Weight

- 14.2 kilograms (31.6 pounds)

Power Cable

- Length: 1.8 meters (6 feet)

Environment

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

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

Heat Output

- 1229 British Thermal Units (BTU) per hour

Noise Level

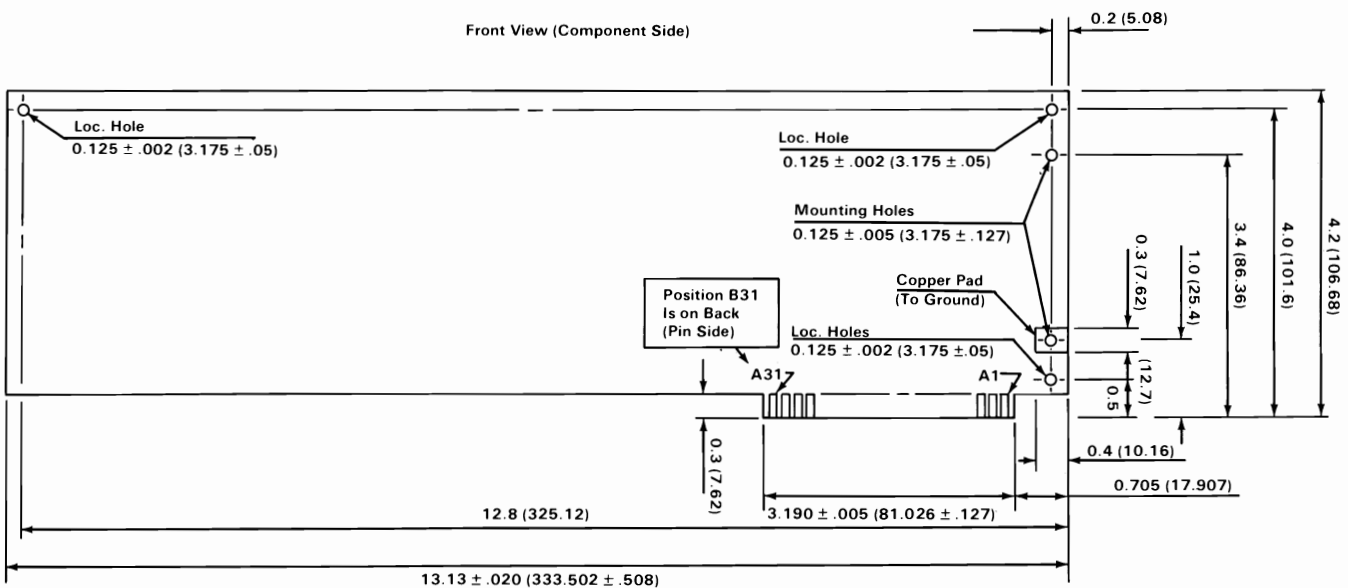
- 43 decibels average-noise rating (without printer)

Electrical

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

Card Specifications

The specifications for option cards follow.



Notes:

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

Connectors

The system board has the following additional connectors:

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

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

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

Power Supply Connectors (P1, P2)

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

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

Speaker Connector (J19)

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

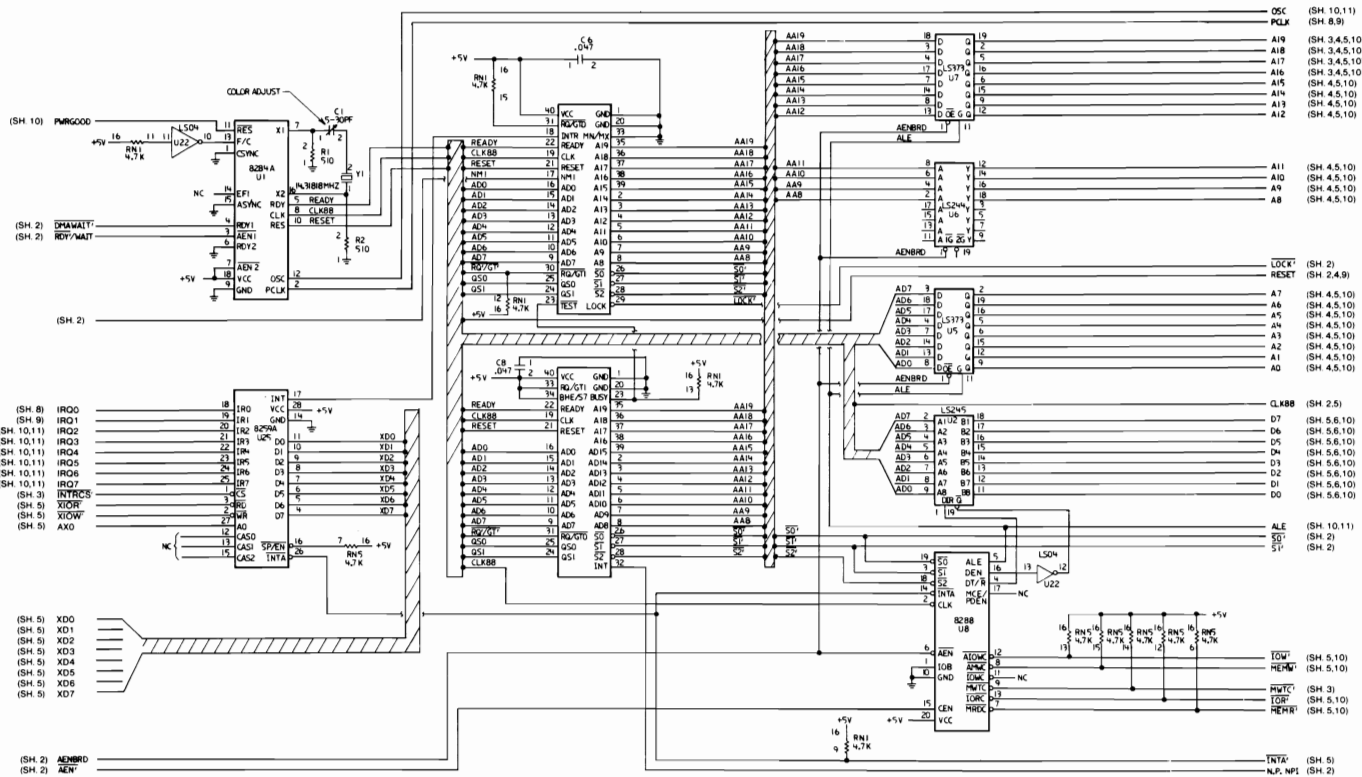
Connector	Pin	Assignments
J22	1	Keyboard Clock
	2	Keyboard Data
	3	Reserved
	4	Ground
	5	+5 Vdc

Keyboard Connector (J22)

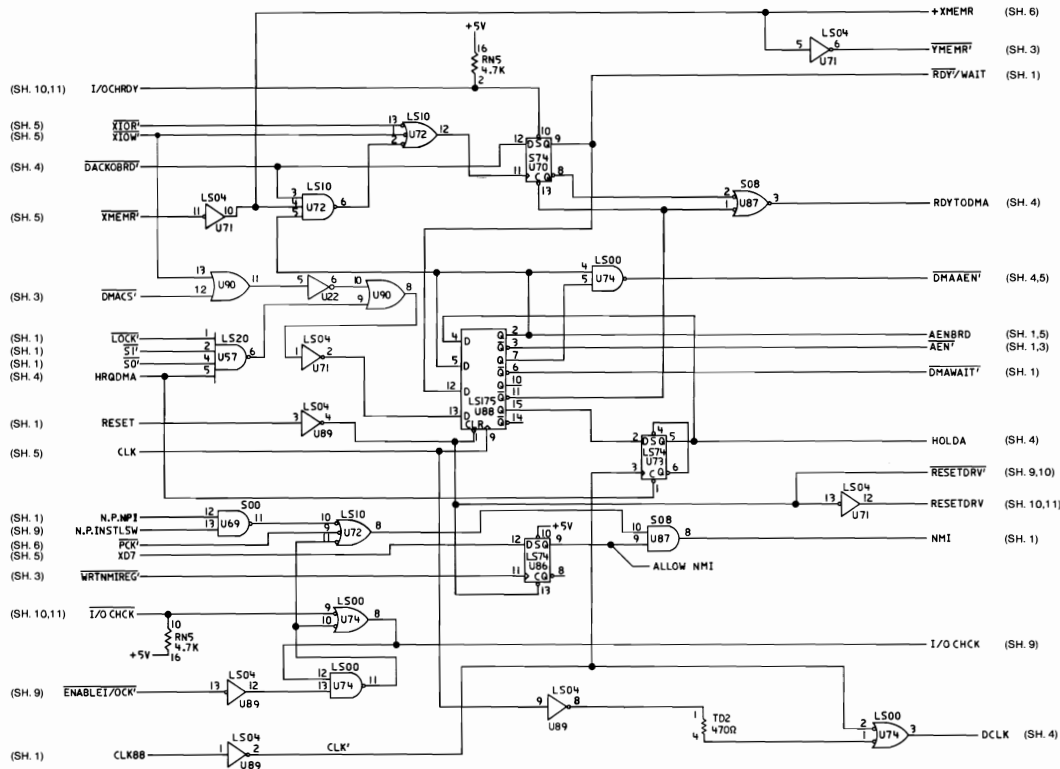
Logic Diagrams - 64/256K

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

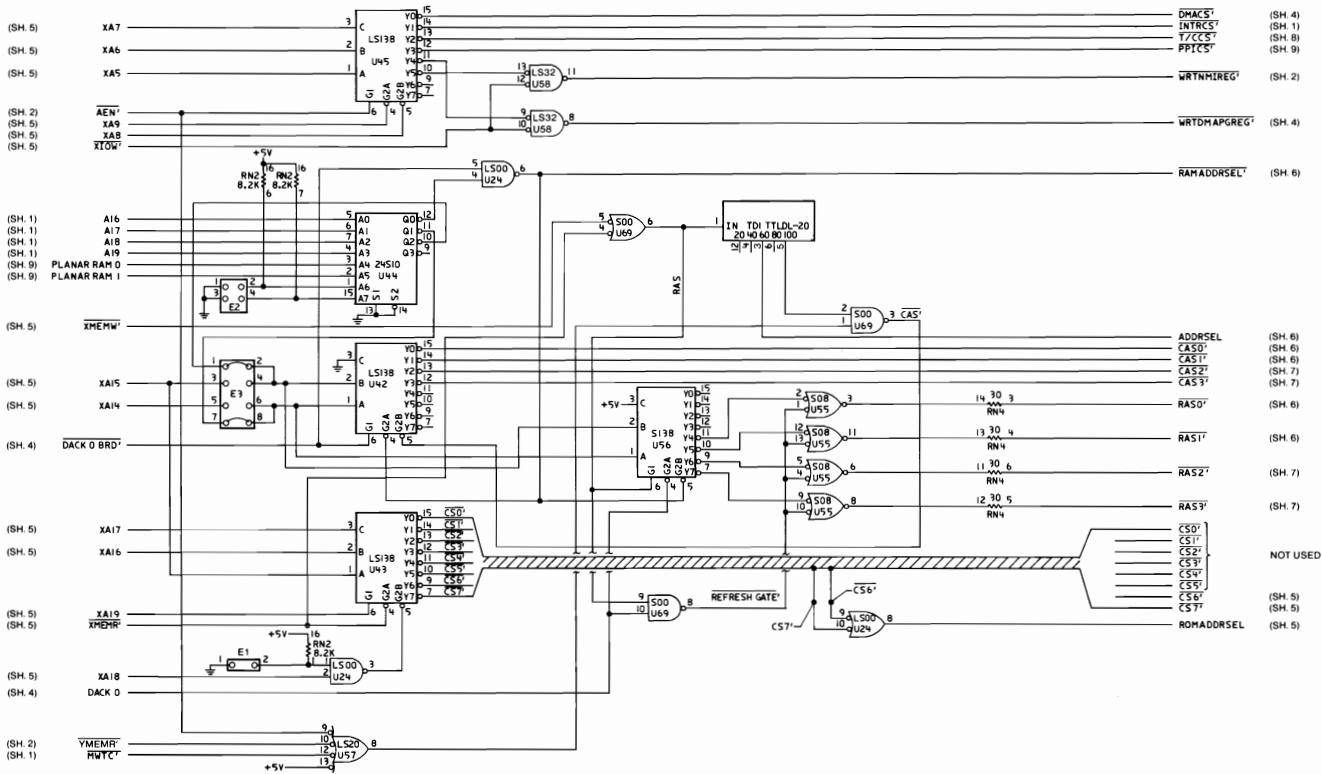




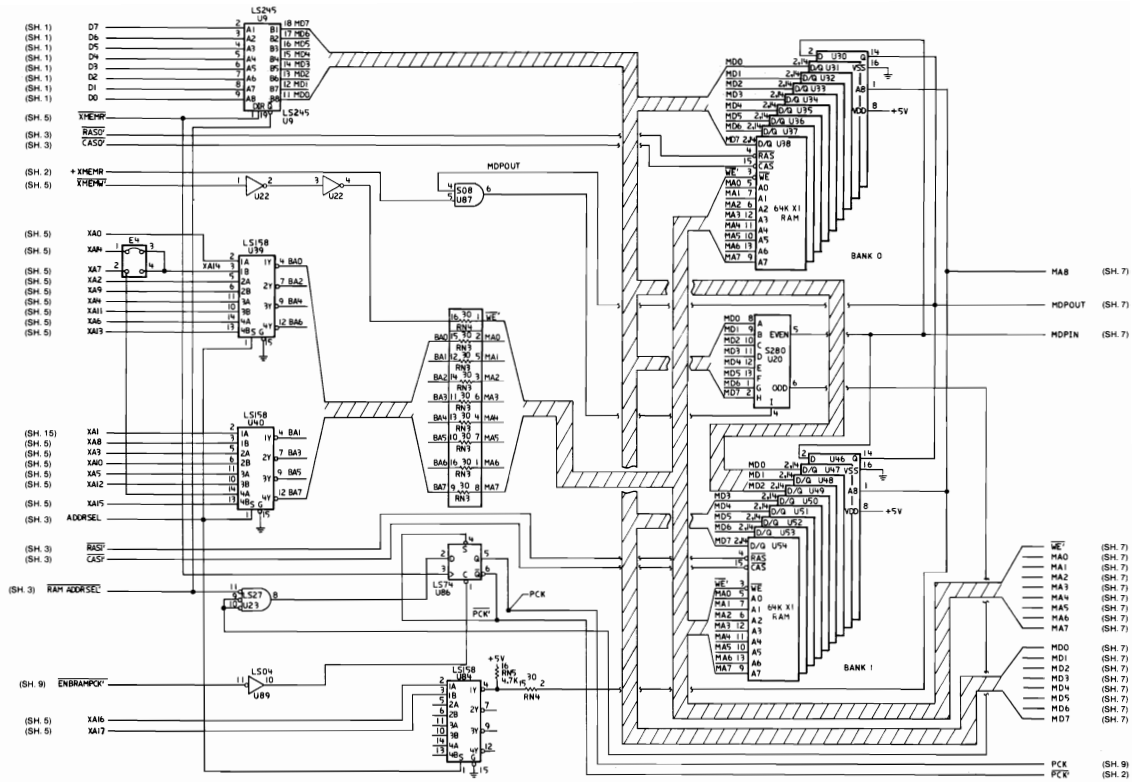
64/256K System Board (Sheet 1 of 11)



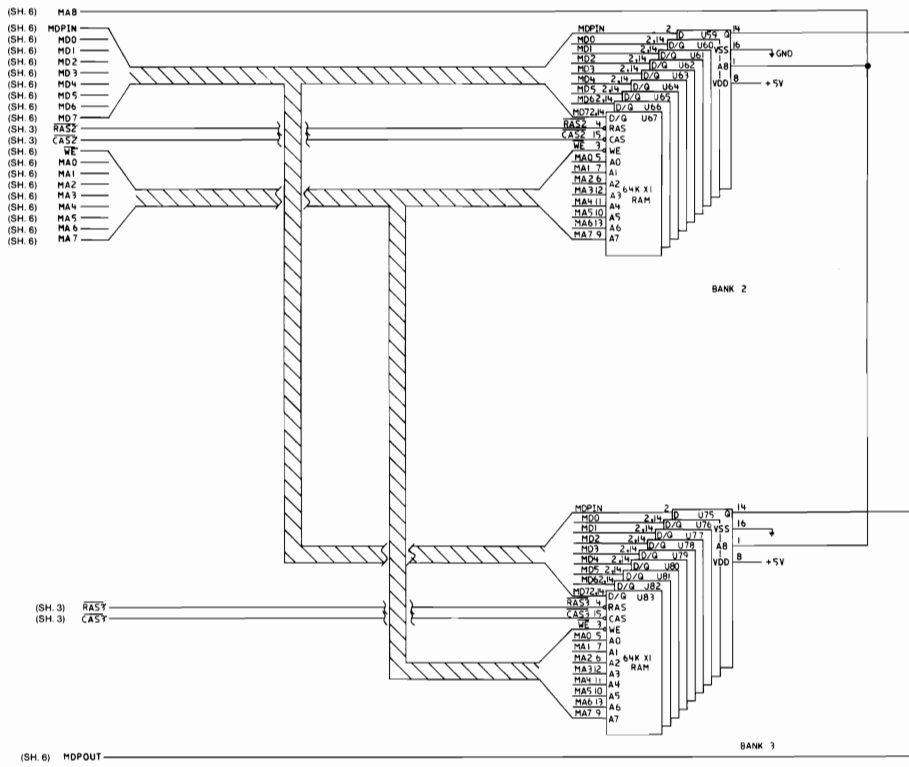
64/256K System Board (Sheet 2 of 11)

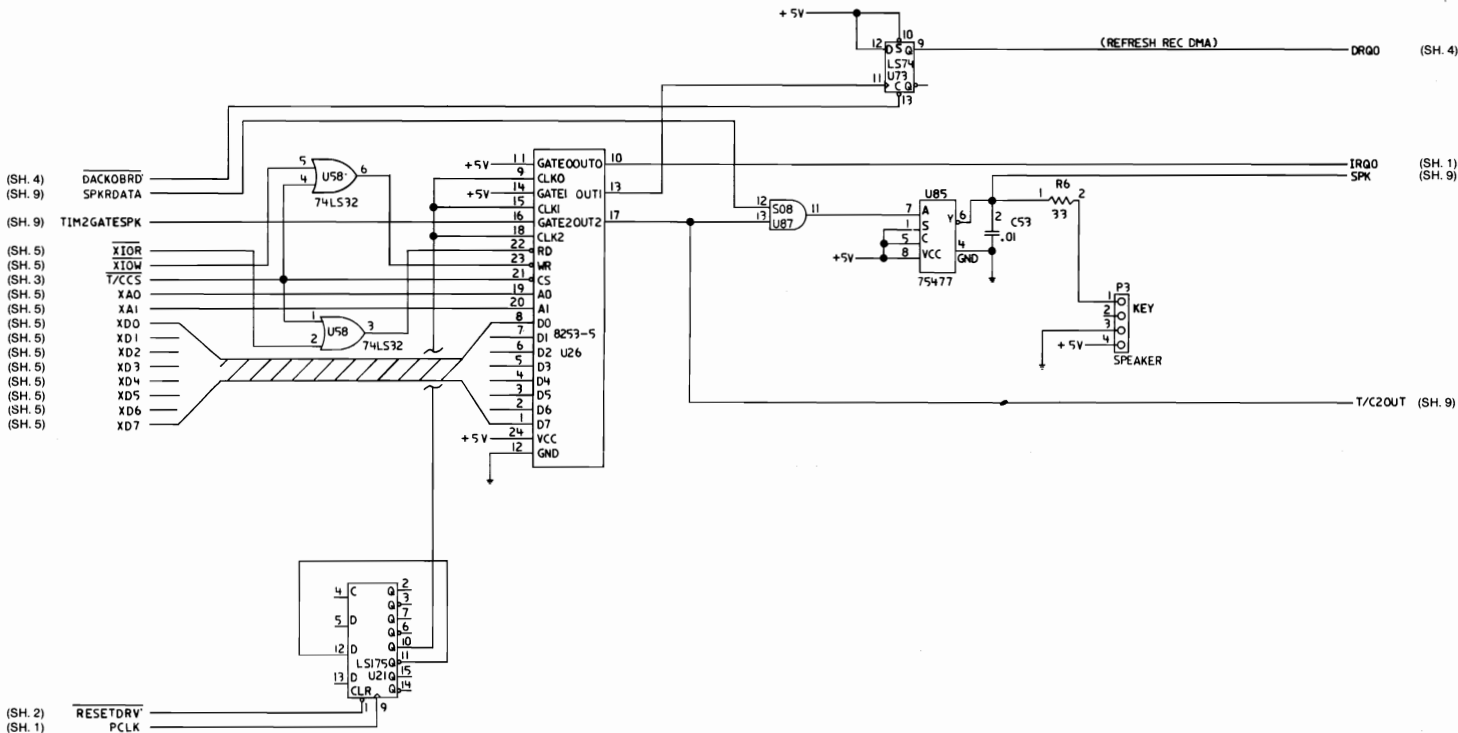


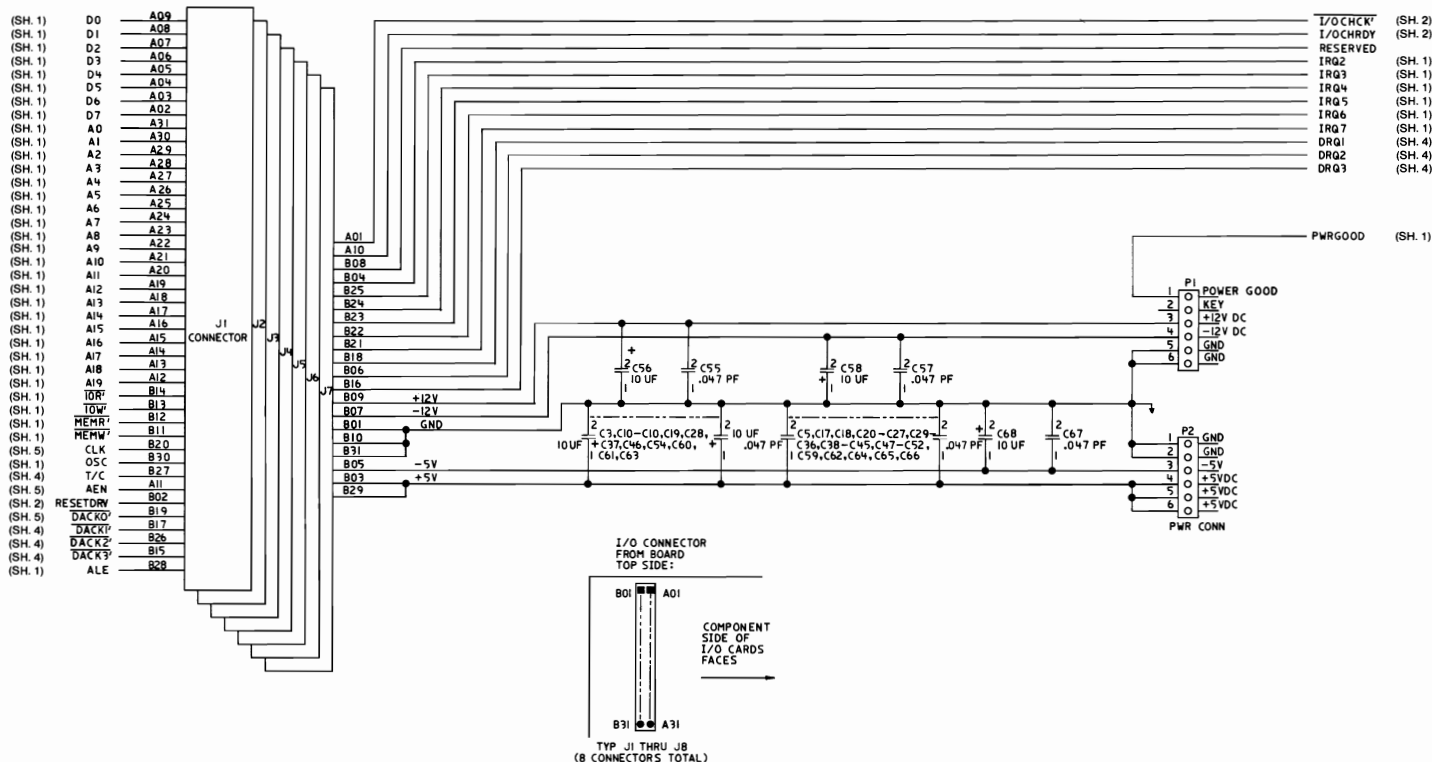
64/256K System Board (Sheet 3 of 11)

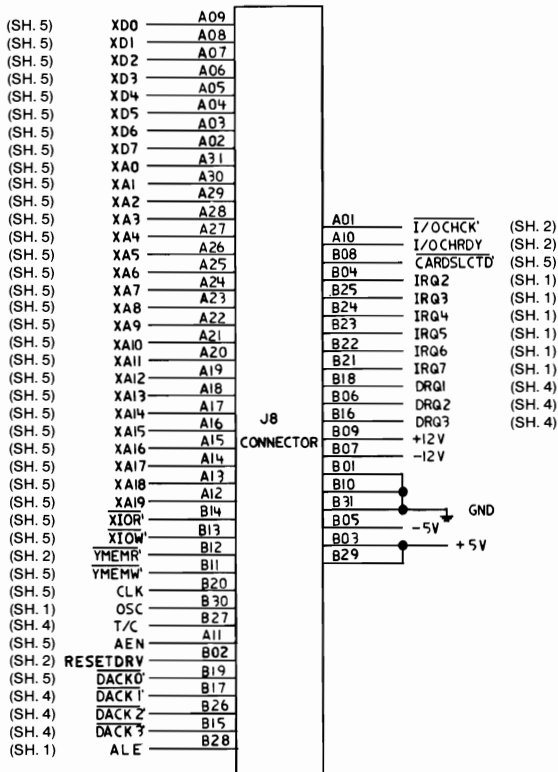


64/256K System Board (Sheet 6 of 11)







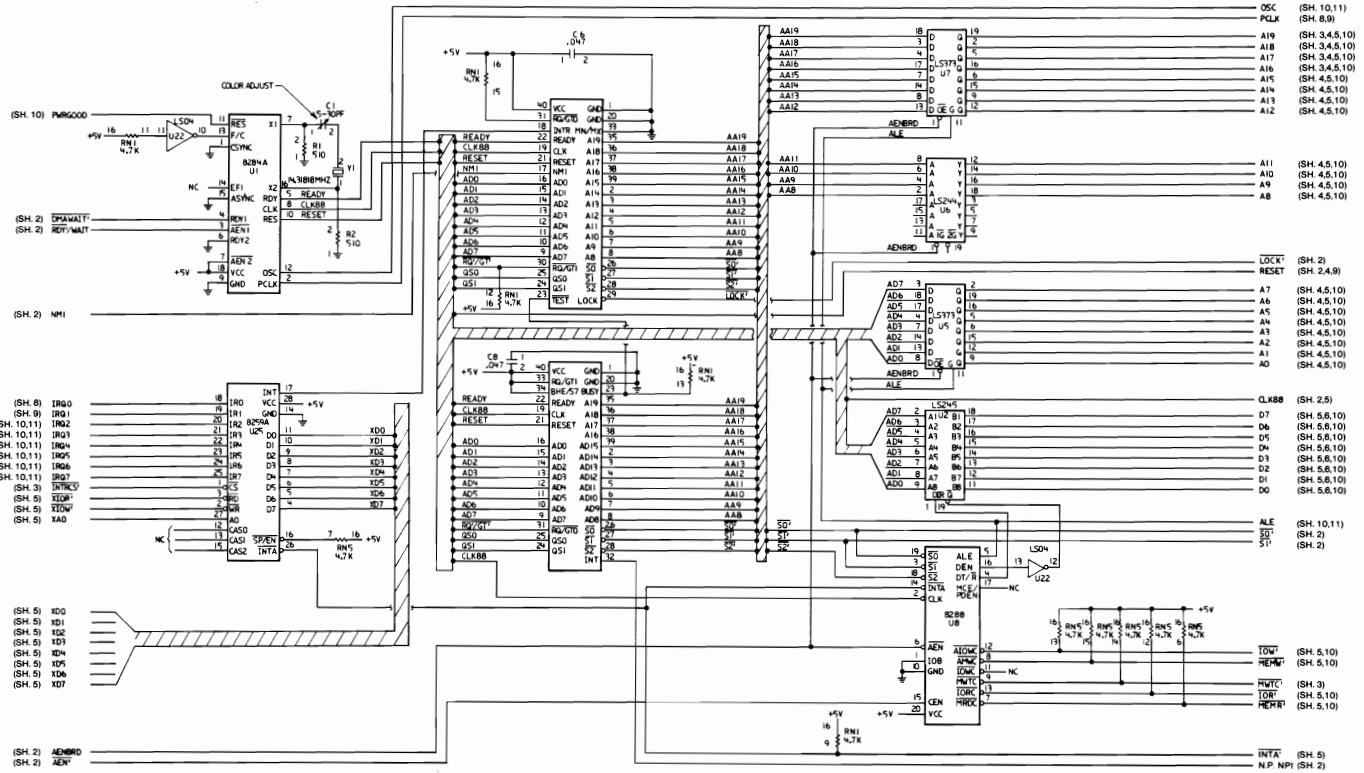


64/256K System Board (Sheet 11 of 11)

Logic Diagrams - 256/640K

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

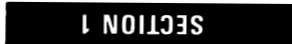


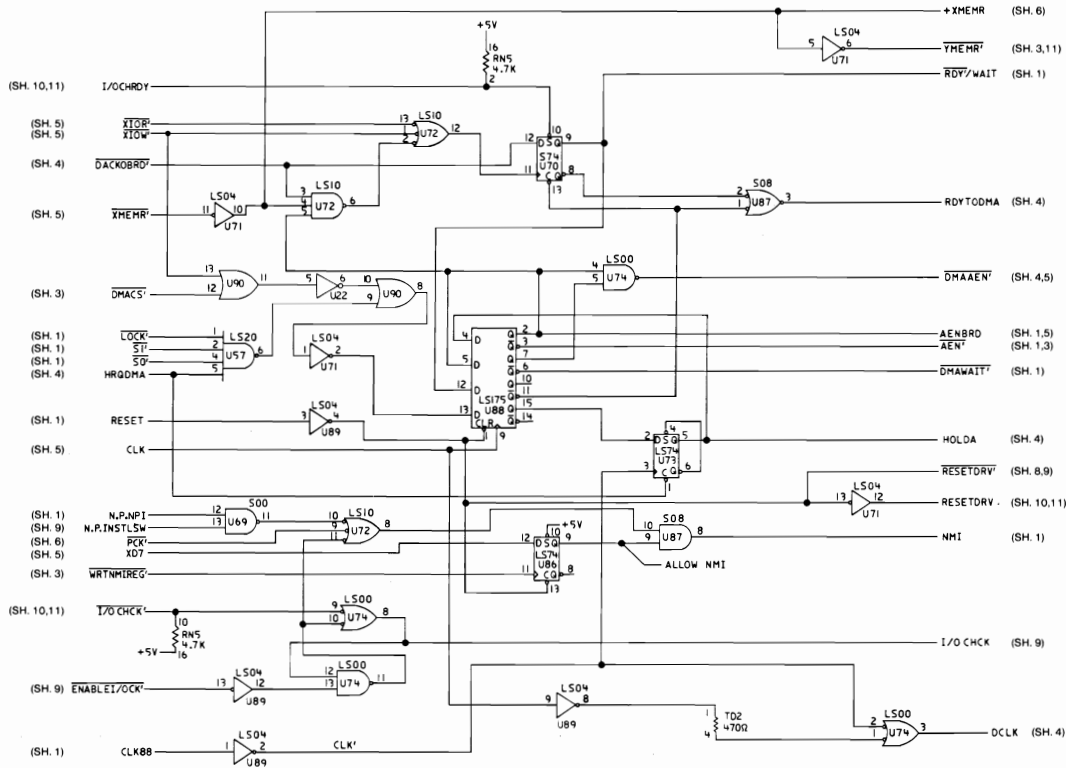


- (SH. 10) PAR/GOOD
- (SH. 2) SP/WAIT
- (SH. 2) RPT/WAIT
- (SH. 2) NMI
- (SH. 6) IRQ0
- (SH. 9) IRQ1
- (SH. 10,11) IRQ2
- (SH. 10,11) IRQ3
- (SH. 10,11) IRQ4
- (SH. 10,11) IRQ5
- (SH. 10,11) IRQ6
- (SH. 10,11) IRQ7
- (SH. 3) INTKCS
- (SH. 5) INTD
- (SH. 5) INTD
- (SH. 5) INTD
- (SH. 5) INTD
- (SH. 5) INTD
- (SH. 2) AENBRO
- (SH. 2) AEN

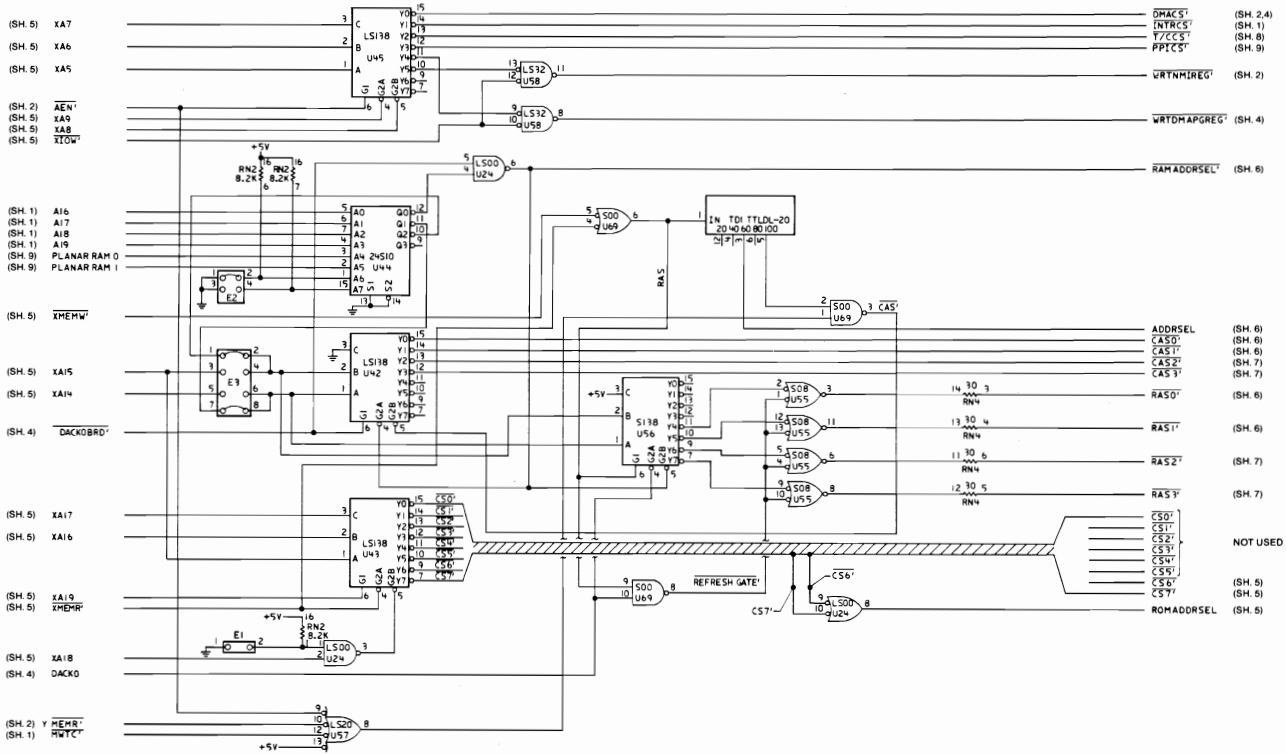
- OSC (SH. 10,11)
- PCLK (SH. 6,9)
- A10 (SH. 3,4,5,10)
- A18 (SH. 3,4,5,10)
- A17 (SH. 3,4,5,10)
- A16 (SH. 3,4,5,10)
- A15 (SH. 4,6,10)
- A14 (SH. 4,5,10)
- A13 (SH. 4,5,10)
- A12 (SH. 4,5,10)
- A11 (SH. 4,5,10)
- A10 (SH. 4,5,10)
- A9 (SH. 4,5,10)
- A8 (SH. 4,5,10)
- LOCK (SH. 2)
- RESET (SH. 2,4,9)
- A7 (SH. 4,5,10)
- A6 (SH. 4,5,10)
- A5 (SH. 4,5,10)
- A4 (SH. 4,5,10)
- A3 (SH. 4,5,10)
- A2 (SH. 4,5,10)
- A1 (SH. 4,5,10)
- CLKBB (SH. 2,5)
- D7 (SH. 5,6,10)
- D6 (SH. 5,6,10)
- D5 (SH. 5,6,10)
- D4 (SH. 5,6,10)
- D3 (SH. 5,6,10)
- D2 (SH. 5,6,10)
- D1 (SH. 5,6,10)
- D0 (SH. 5,6,10)
- ALE (SH. 10,11)
- ST (SH. 2)
- TOW (SH. 5,10)
- REWT (SH. 5,10)
- TORT (SH. 5,10)
- RETRT (SH. 5,10)
- INTA (SH. 5)
- N.P. NPI (SH. 2)

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

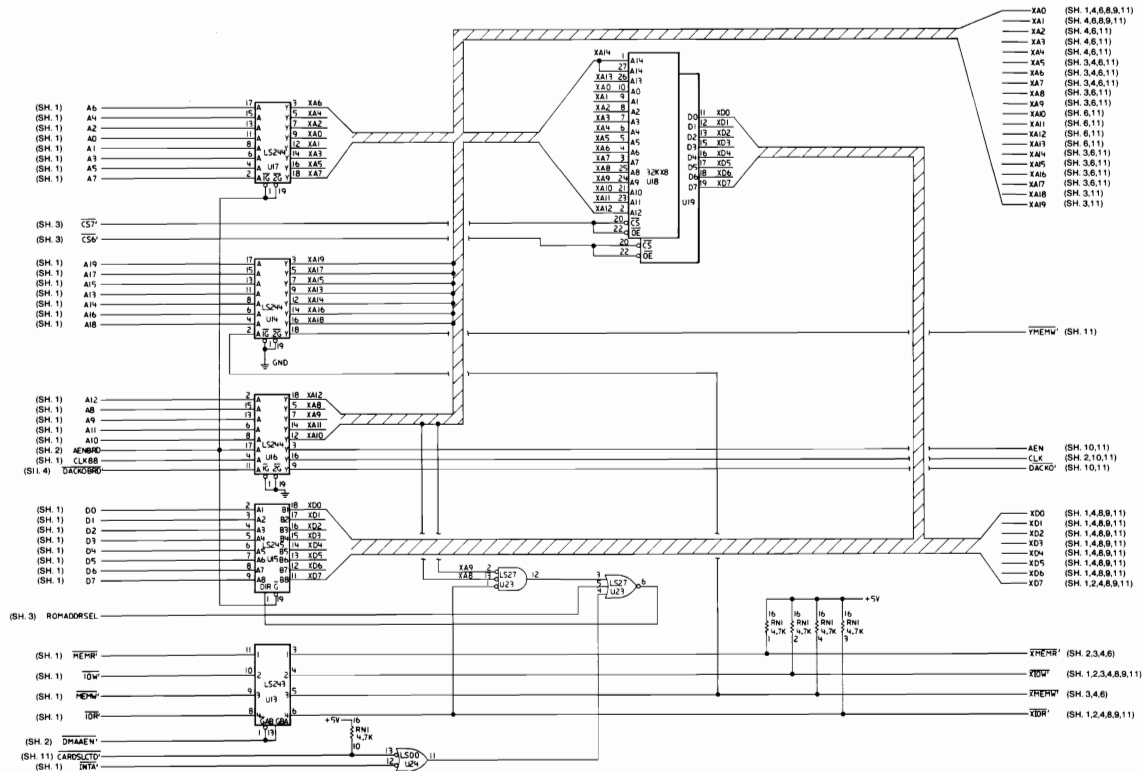




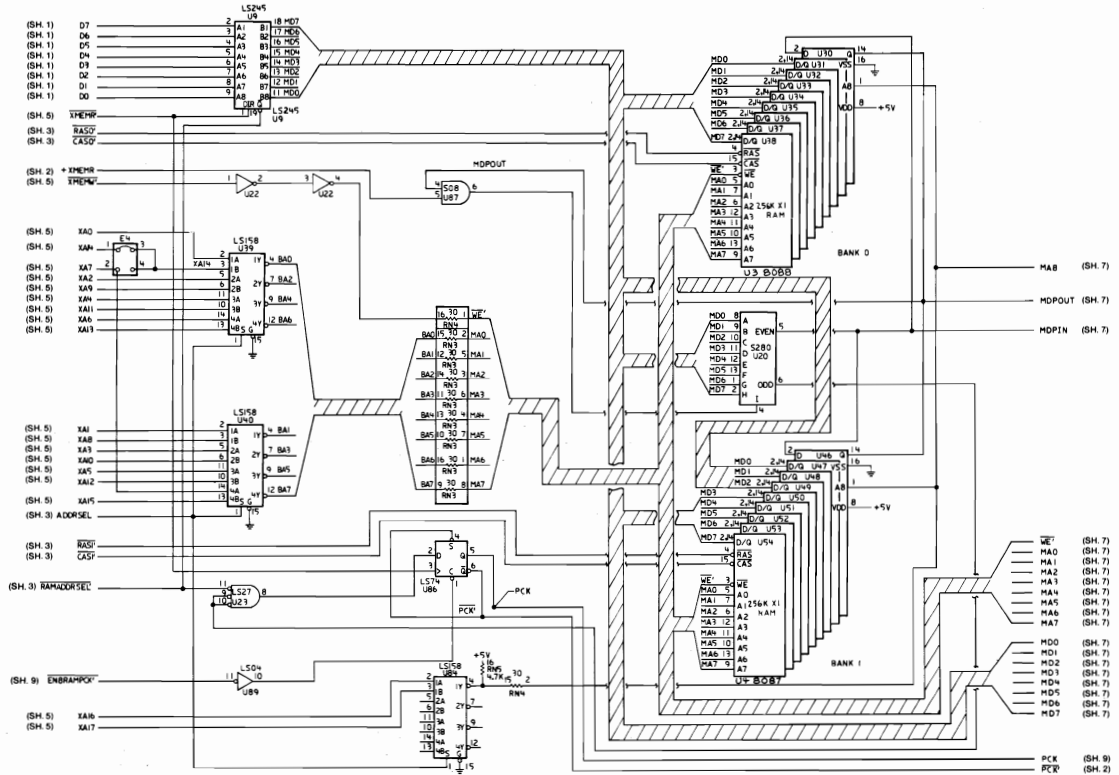
256/640K System Board (Sheet 2 of 11)



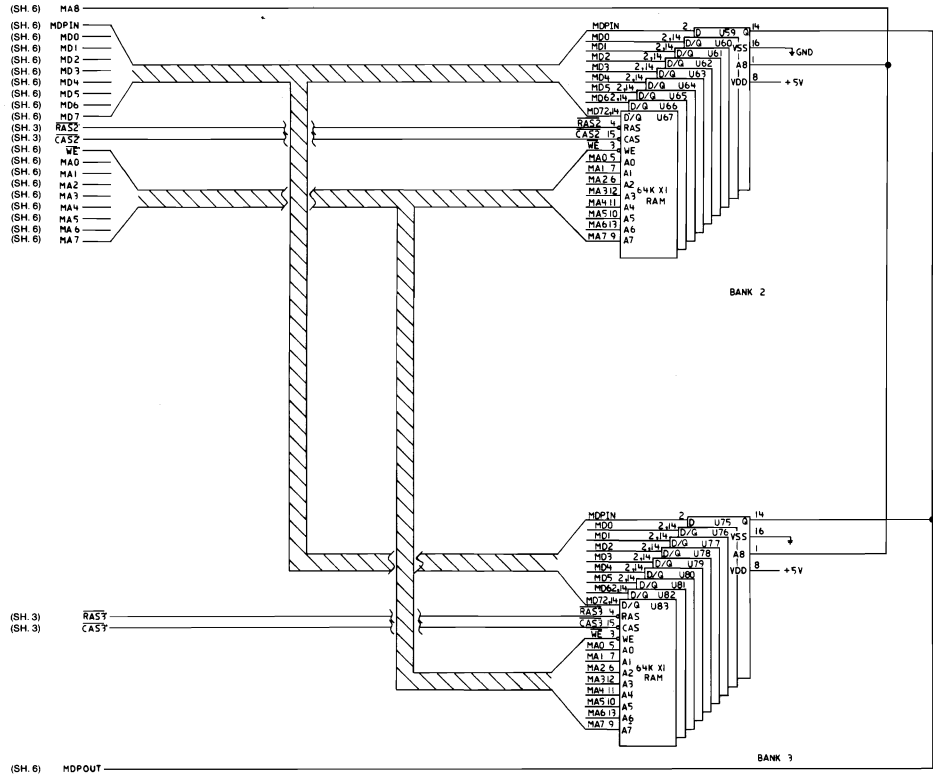
256/640K System Board (Sheet 3 of 11)



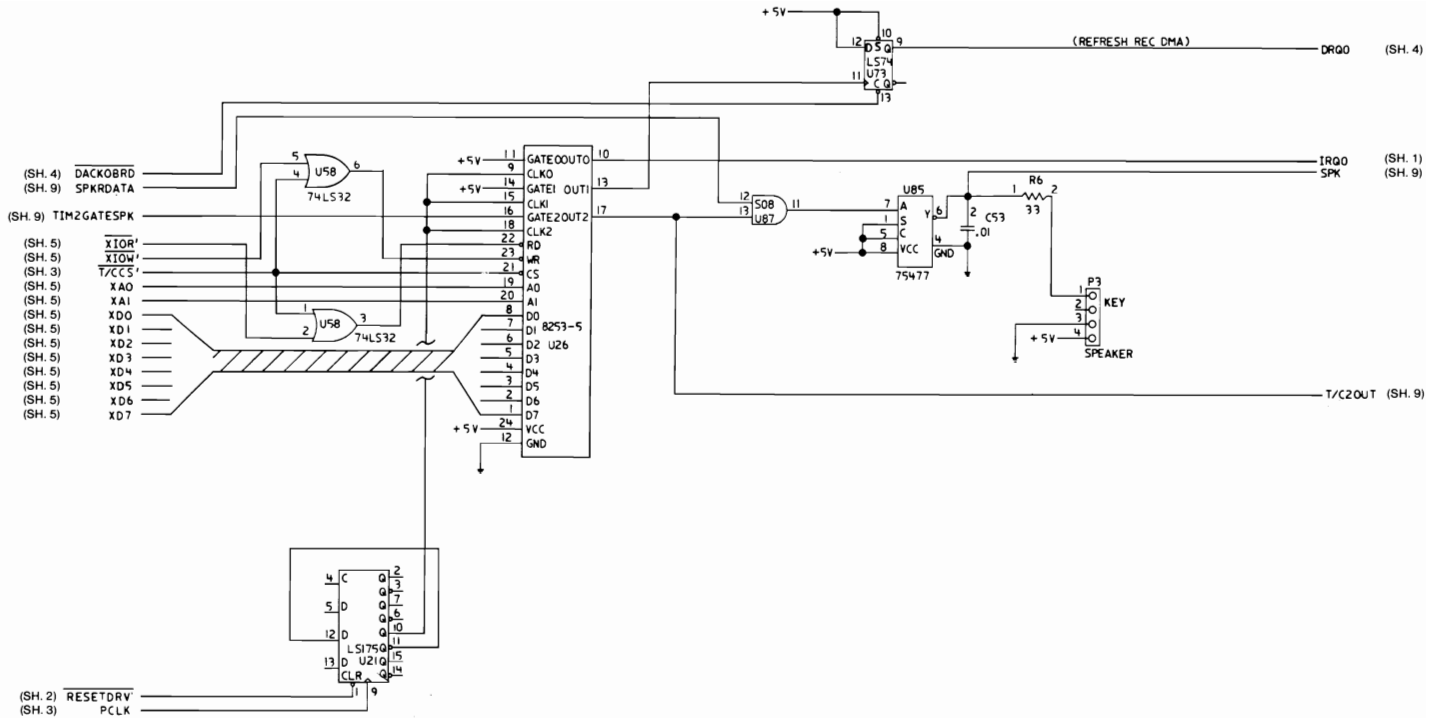
256/640K System Board (Sheet 5 of 11)



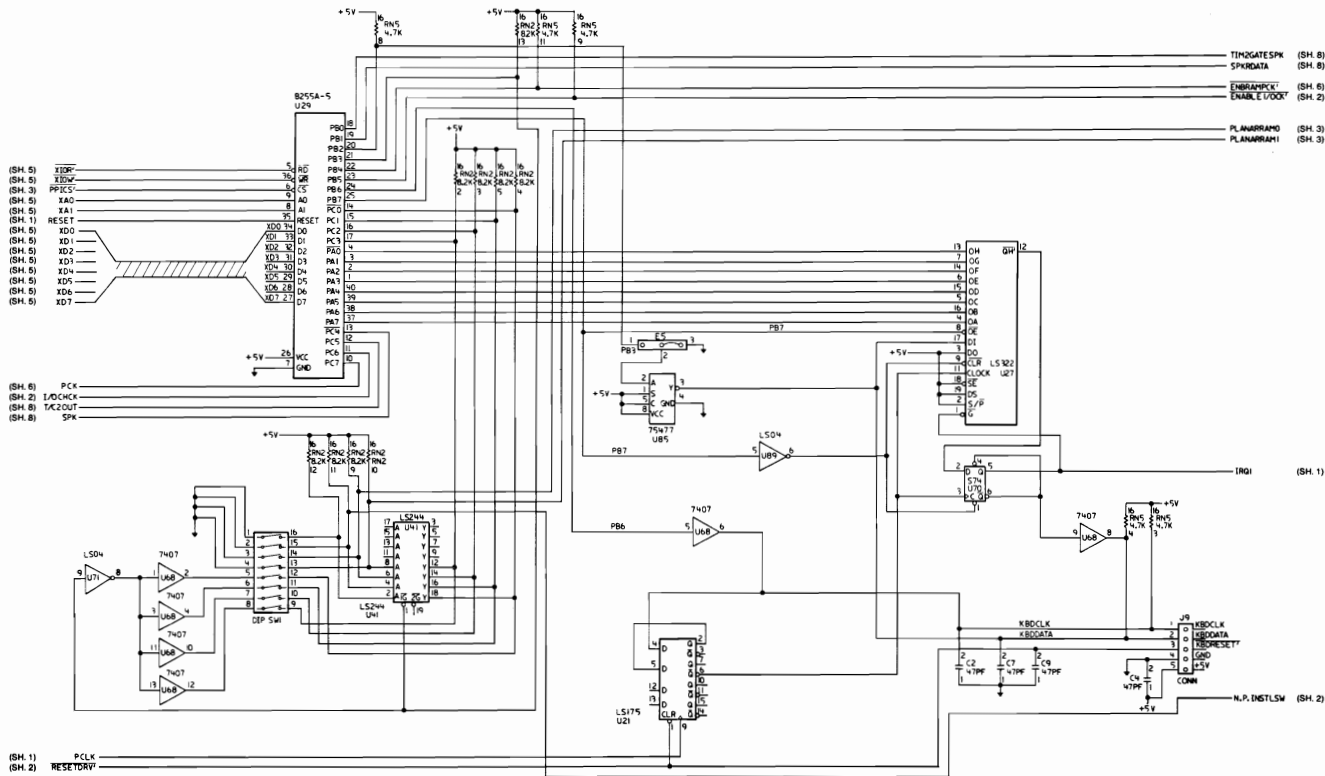
256/640K System Board (Sheet 6 of 11)



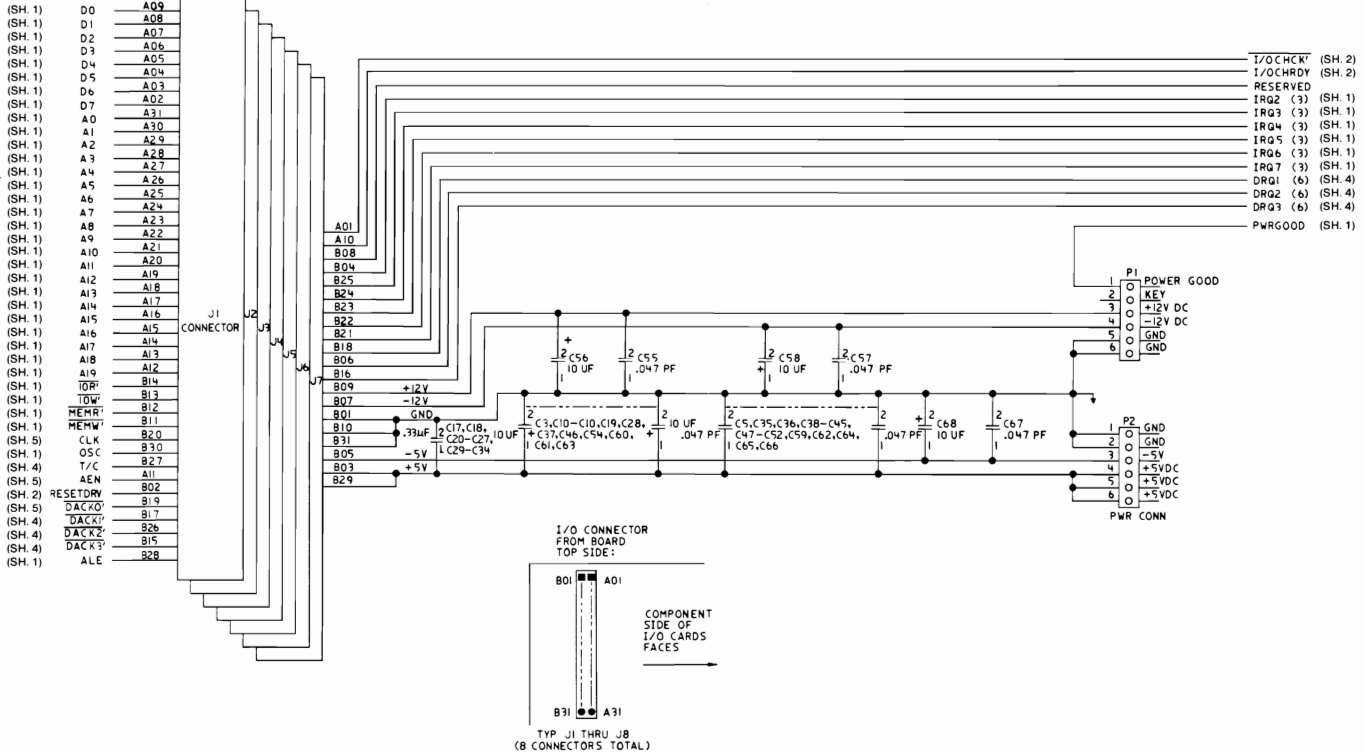
256/640K System Board (Sheet 7 of 11)



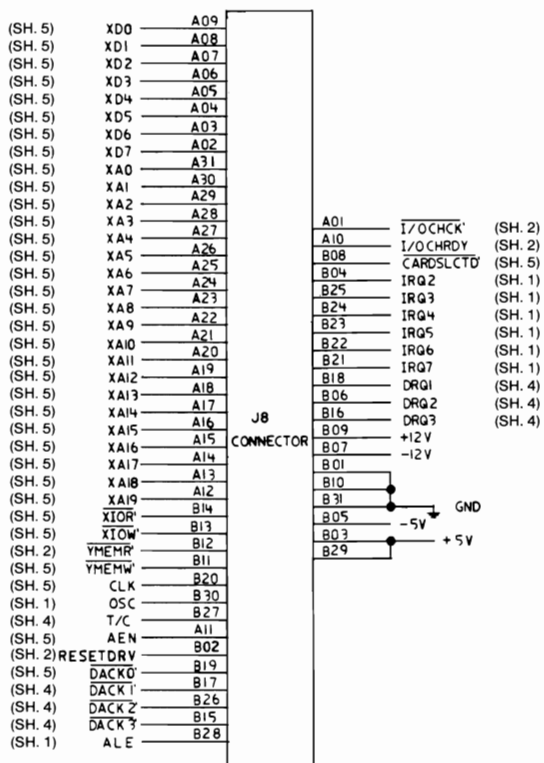
256/640K System Board (Sheet 8 of 11)



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



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



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

Notes:



SECTION 2. COPROCESSOR

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

Notes:



Description

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

The 8087 coprocessor works in parallel with the microprocessor. The parallel operation decreases operating time by allowing the coprocessor to do mathematical calculations while the microprocessor continues to do other functions.

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

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

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

Programming Interface

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

The coprocessor has eight 80-bit registers, which provide the equivalent capacity of the 40 16-bit registers found in the microprocessor. This register space allows constants and temporary results to be held in registers during calculations, thus reducing memory access and improving speed as well as bus availability. The register space can be used as a stack or as a fixed register set. When used as a stack, only the top two stack elements are operated on. The figure below shows representations of large and small numbers in each data type.

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

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

Data Types

Hardware Interface

The coprocessor uses the same clock generator and system bus interface components as the microprocessor. The microprocessor's queue status lines (QS0 and QS1) enable the coprocessor to obtain and decode instructions simultaneously with the microprocessor. The coprocessor's 'busy' signal informs the microprocessor that it is executing; the microprocessor's WAIT

instruction forces the microprocessor to wait until the coprocessor is finished executing (WAIT FOR NOT BUSY).

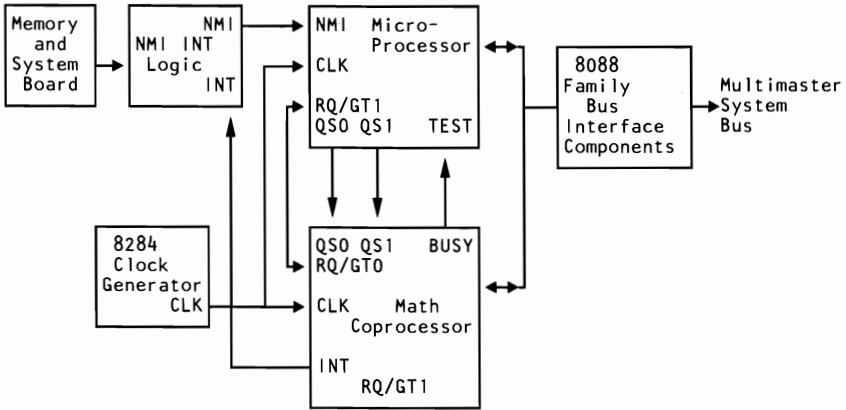
When an incorrect instruction is sent to the coprocessor (for example, divide by 0 or load a full register), the coprocessor can signal the microprocessor with an interrupt. There are three conditions that will disable the coprocessor interrupt to the microprocessor:

1. Exception and interrupt-enable bits of the control word are set to 1's
2. System-board switch-block 1, switch 2, set in the On position
3. Non-maskable interrupt register (NMI REG) is set to zero.

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

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

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



Coprocessor Interconnection

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

SECTION 3. POWER SUPPLIES

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

Notes:



IBM Personal Computer XT Power Supply

Description

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

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

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

Input Requirements

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

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

Input Requirements

Outputs

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

Vdc Output

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

Vac Output

The sense levels of the dc outputs are:

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

Vdc Sense Levels

Overvoltage/Overcurrent Protection

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

Voltage and Current Protection

Power Good Signal

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

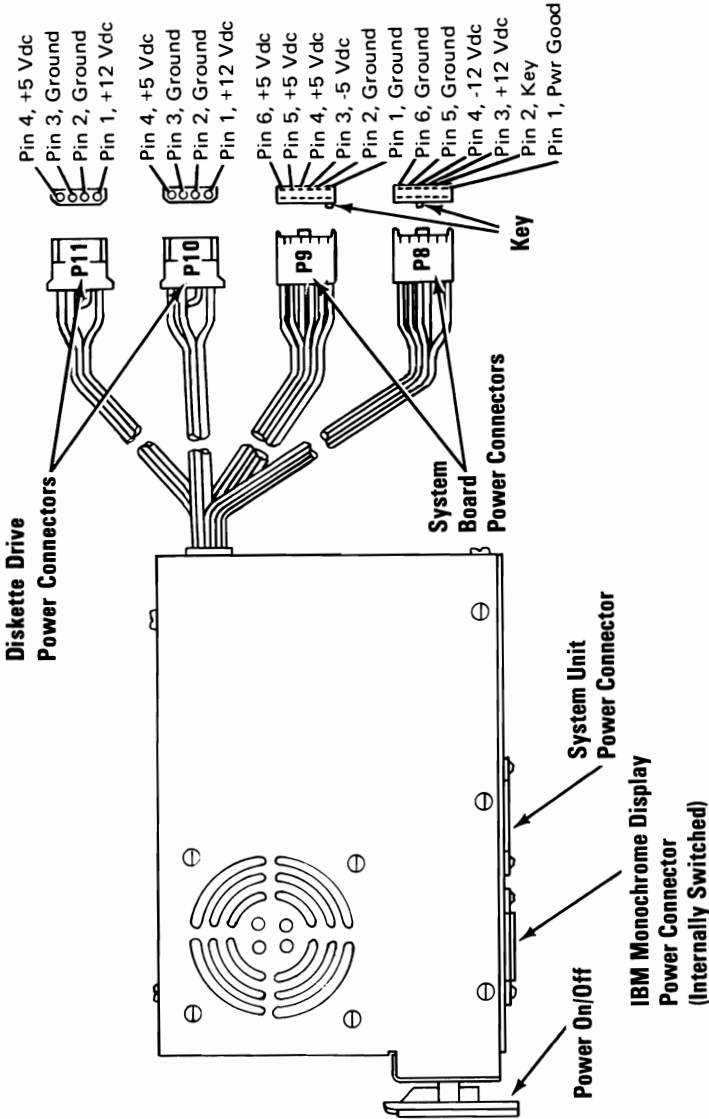
The 'power good' signal indicates that there is adequate power to continue processing. If the power goes below the specified levels, the 'power good' signal triggers a system shutdown.

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

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

Connector Specifications and Pin Assignments

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



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 \pm 3 Hz		
Nominal Vac	Minimum Vac	Maximum Vac
110 220/240	90 180	137 259
Current: 3.5 A max at 90 Vac		

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

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

Vdc Output

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

Vdc Sense Levels

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

Voltage and Current Protection

Power Good Signal

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

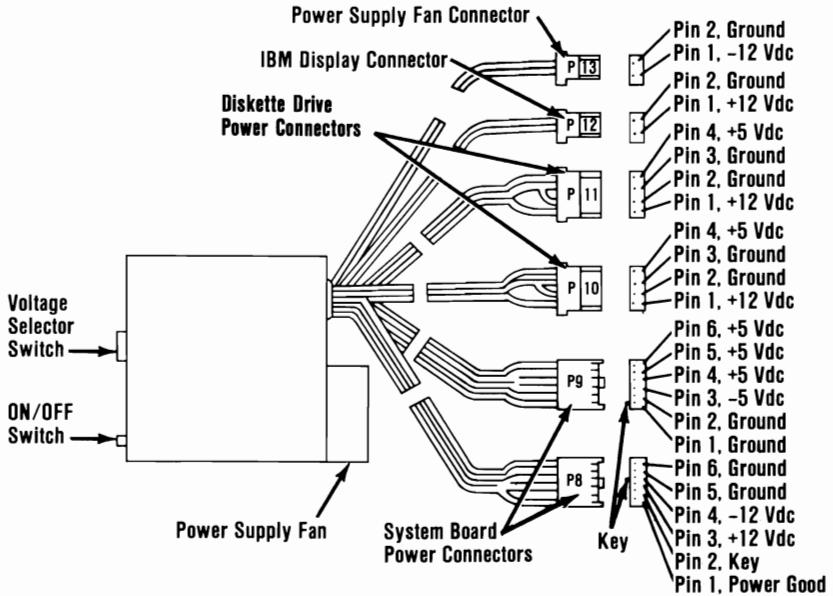
This signal is the logical **AND** of the dc output-voltage sense signal and the ac input-voltage fail signal. This signal is **TTL-compatible** up-level for normal operation or down-level for fault conditions. The ac 'fail' signal causes 'power good' to go to a down-level when any output voltage falls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

When power is switched on, the dc output-voltage sense signal holds the 'power good' signal at a down level until all output

voltages reach their minimum sense levels. The 'power good' signal has a turn-on delay of 100 to 500 milliseconds.

Connector Specifications and Pin Assignments

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



Power Supply and Connectors

Notes:



SECTION 4. KEYBOARDS

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

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

Introduction

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

83-Key Keyboard Description

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

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

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

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

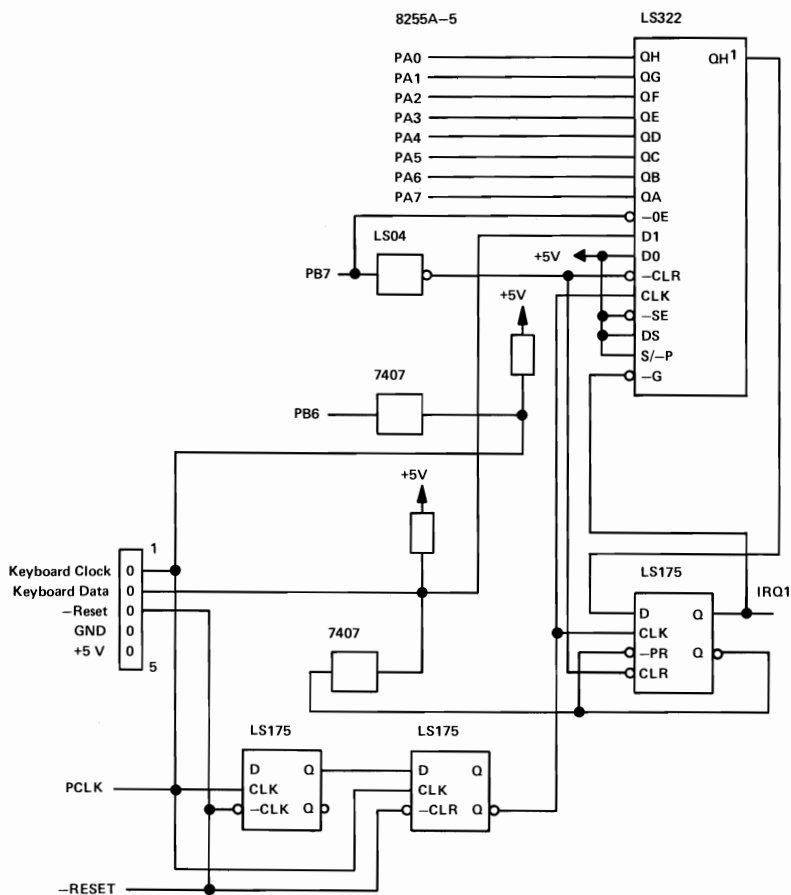
The keyboard has 83 keys arranged in three major groupings. The central portion of the keyboard is a standard typewriter keyboard layout. On the left side are 10 function keys. These keys are user-defined by the software. On the right is a 15-key keypad. These keys are also defined by the software, but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

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

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

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

Block Diagram



Keyboard Interface Block Diagram

Keyboard Encoding and Usage

Encoding

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

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

Character Codes

The following character codes are passed through the BIOS keyboard routine to the system or application program. A ‘-1’ means the combination is suppressed in the keyboard routine. The codes are returned in AL.

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

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

Character Codes (Part 1 of 2)

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

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

Character Codes (Part 2 of 2)

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

Extended Codes

Extended Functions

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

Shift States

Most shift states are handled within the keyboard routine and are not apparent to the system or application program. In any case, the current set of active shift states is available by calling an entry point in the ROM keyboard routine. The key numbers are shown on the keyboard diagrams beginning on page 4-12. The following keys result in altered shift states:

Shift

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

Ctrl

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

Alt

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

The Alt key has another use. This key allows the user to enter any ASCII character code from 1 to 255 into the system from the keyboard. The user holds down the Alt key and types the decimal value of the characters desired using the numeric keypad (keys 71–73, 75–77, and 79–82). The Alt key is then released. If more than three digits are typed, a modulo-256 result is created. These three digits are interpreted as a character code and are transmitted through the keyboard routine to the system or application program. Alt is handled within the keyboard routine.

Caps Lock

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

Scroll Lock

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

Shift Key Priorities and Combinations

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

Special Handling

System Reset

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

Break

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

Pause

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

Print Screen

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

Extended Functions

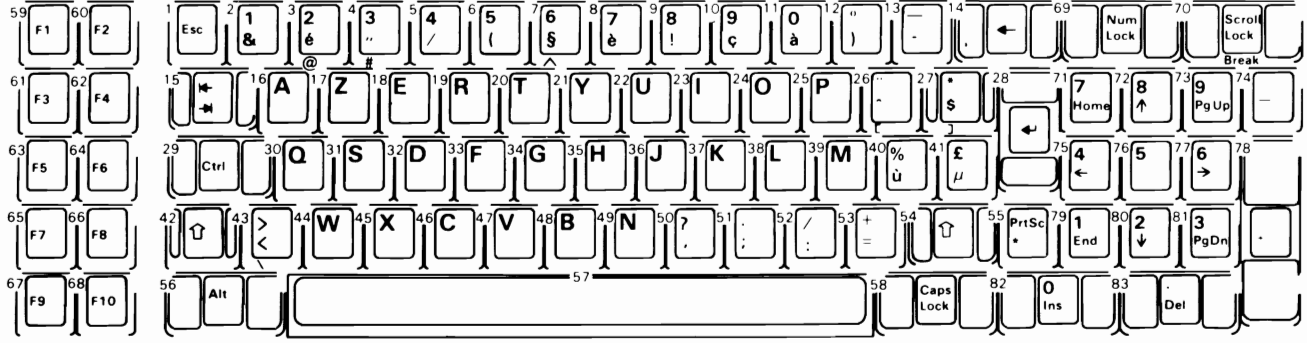
The keyboard routine does its own buffering. The keyboard buffer is large enough that few typists will ever fill it. However, if a key is pressed when the buffer is full, the key will be ignored and the “bell” will sound.

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

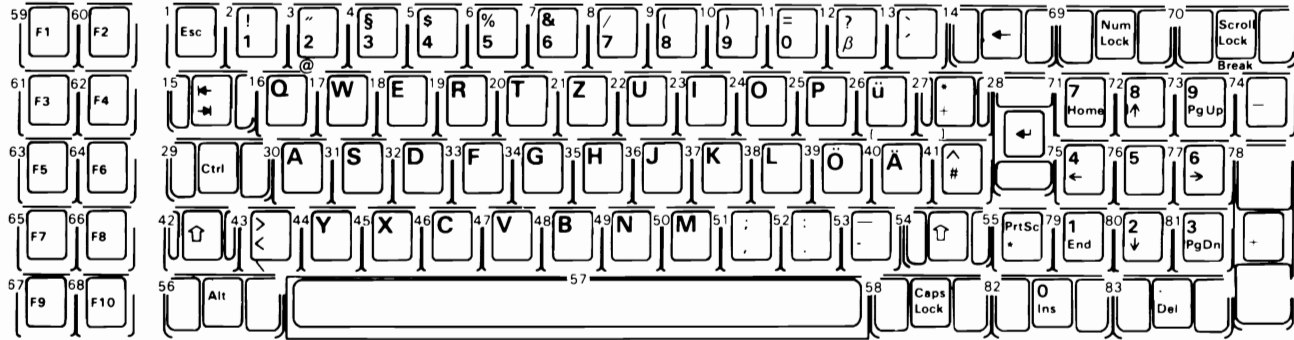
Keyboard Layouts

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

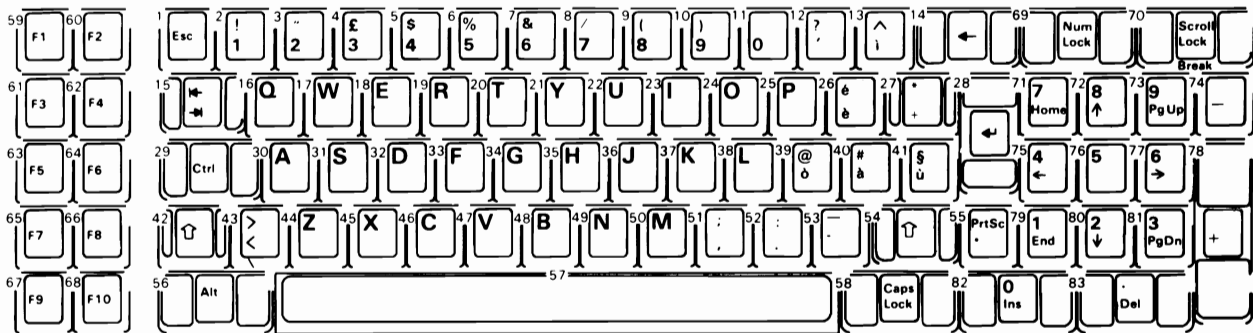
- French
- German
- Italian
- Spanish
- UK English
- US English



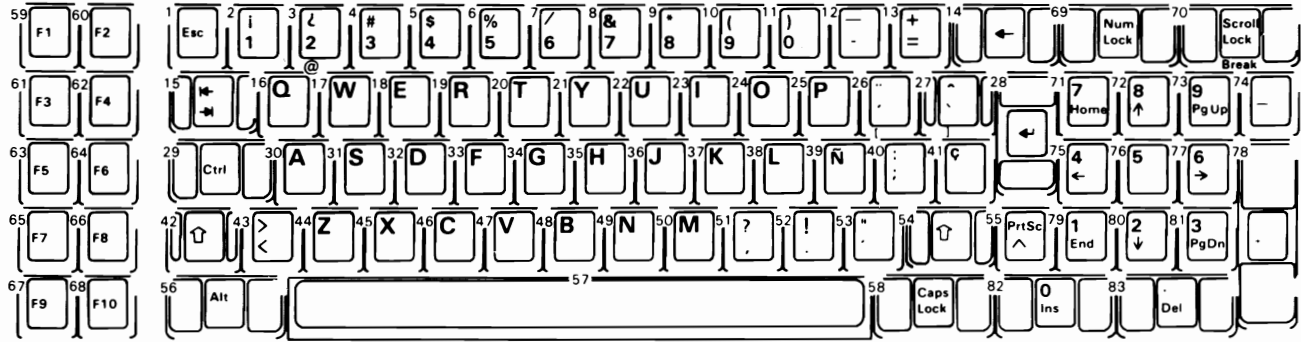
Note: Nomenclature is on both the top and front face of keybuttons as shown. The number to the upper left designates the button position.



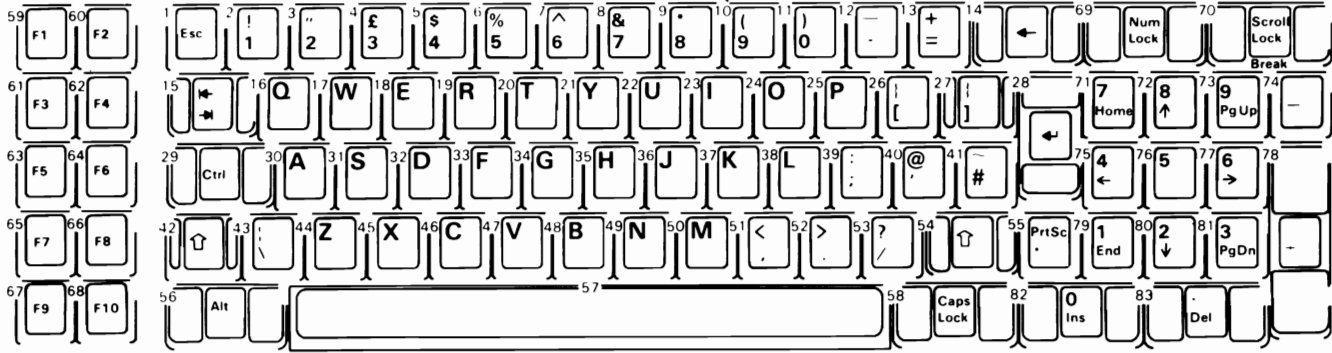
Note: Nomenclature is on both the top and front face of keybuttons as shown. The number to the upper left designates the button position.



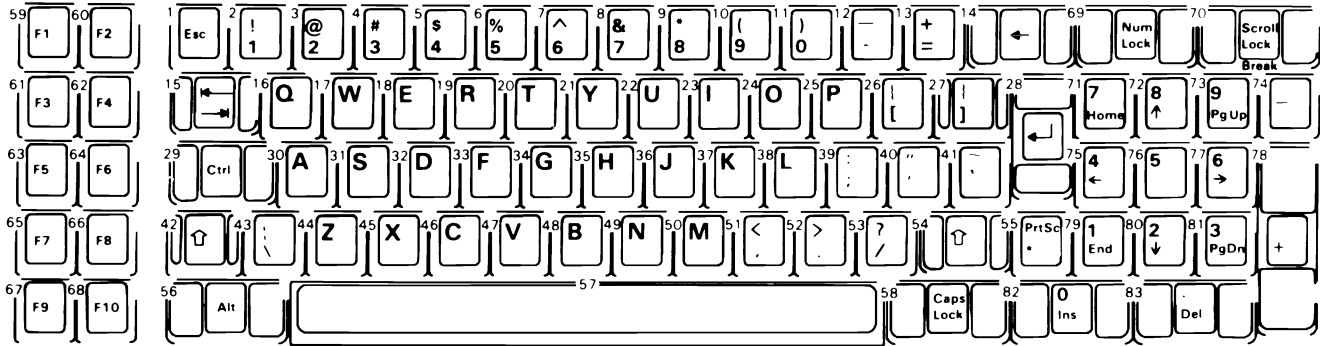
Note: Nomenclature is on both the top and front face of keybuttons as shown. The number to the upper left designates the button position.



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

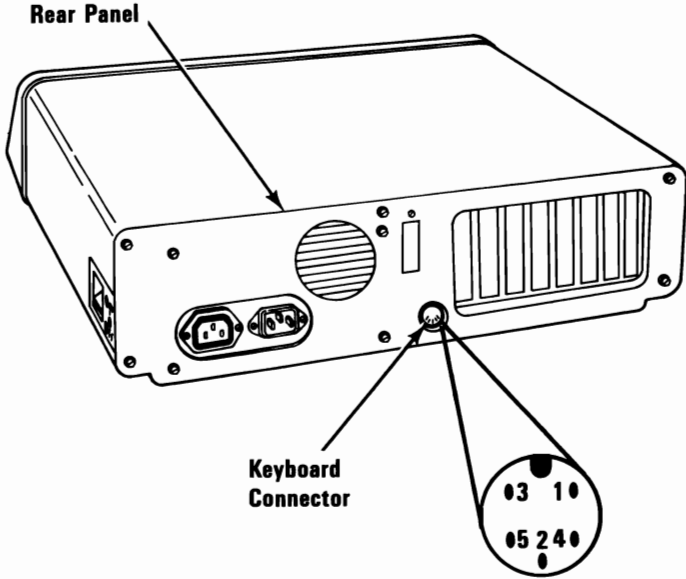


Note: Nomenclature is on both the top and front face of keybuttons as shown. The number to the upper left designates the button position.



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

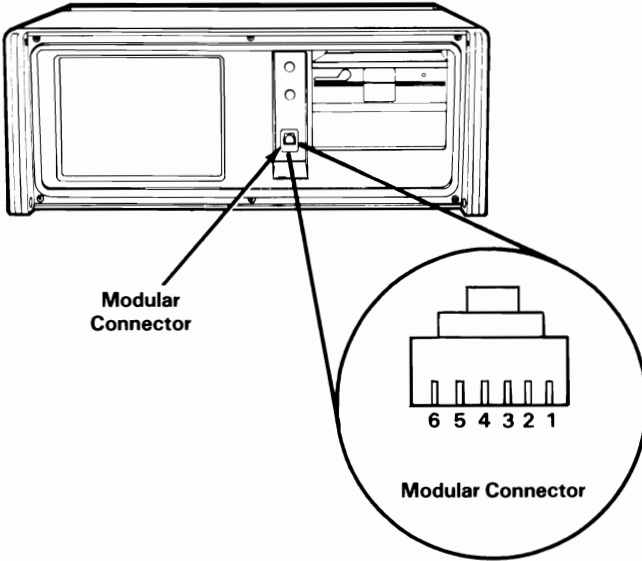
Connector Specifications



5-Pin DIN Connector

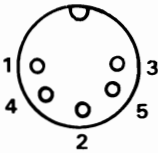
Pin	TTL Signal	Signal Level
1	+ Keyboard Clock	+ 5 Vdc
2	+ Keyboard Data	+ 5 Vdc
3	- Keyboard Reset (Not used by keyboard)	
Power Supply Voltages		Voltage
4	Ground	0
5	+ 5 Volts	+ 5 Vdc

Keyboard Interface Connector Specifications

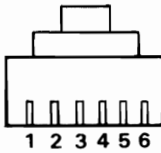


Keyboard Cable Connections

DIN Connector



Modular Connector



Keyboard Connector

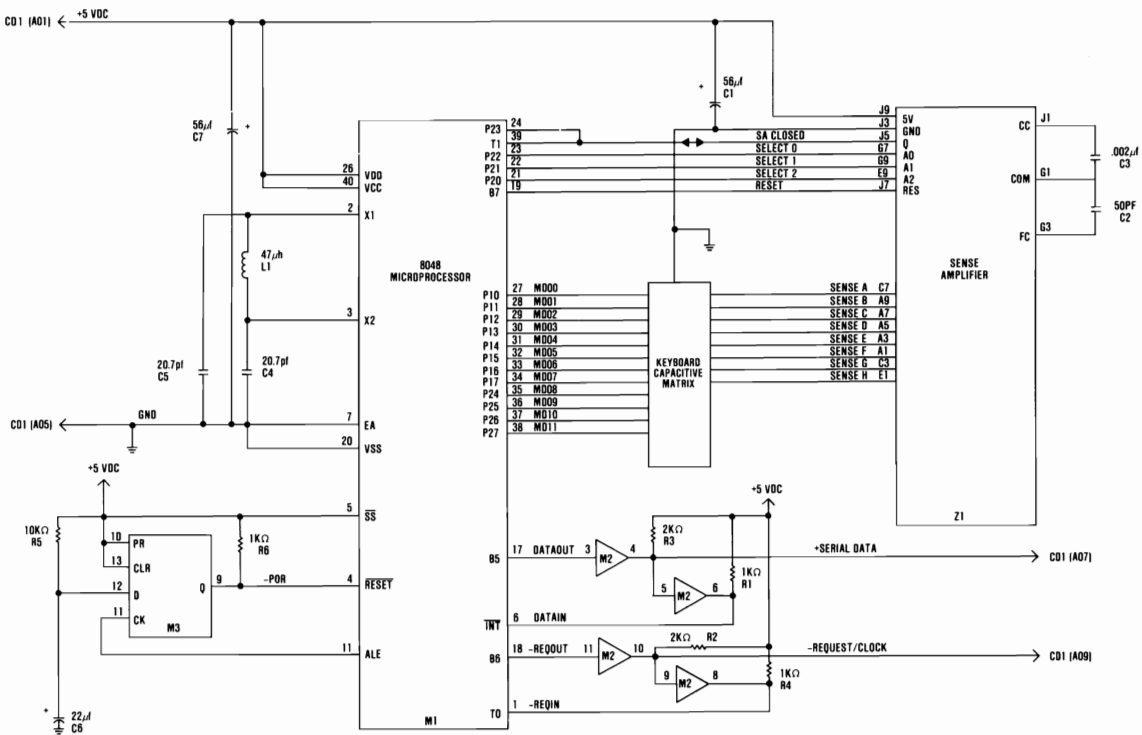


	Pin Side	Pin Side	Wire Side
Clock	1	4	6
Data	2	5	5
Ground	4	3	4
+5 Volts	5	2	2

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

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

Keyboard Logic Diagram



83-Key Keyboard

101/102-Key Keyboard

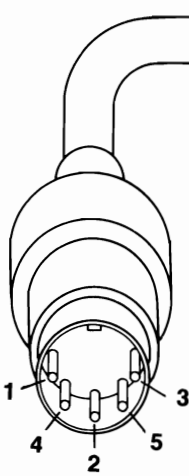
Description

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

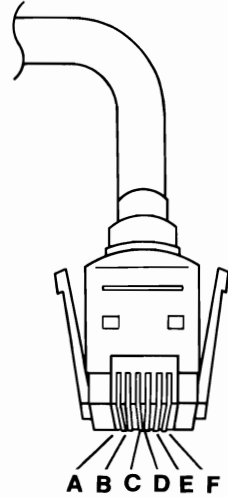


Cables and Connectors

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



DIN Connector



SDL Connector

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

Sequencing Key-Code Scanning

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

Keyboard Buffer

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

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

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

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

Keys

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

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

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

Power-On Routine

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

Power-On Reset

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

Basic Assurance Test

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

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

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

Commands from the System

Reset (Hex FF)

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

Commands to the System

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

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

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

BAT Completion Code (Hex AA)

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

BAT Failure Code (Hex FC)

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

Key Detection Error (Hex FF)

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

Overflow (Hex FF)

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

Keyboard Scan Codes

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

Scan Code Tables

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

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

* 101-key keyboard only.

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

** 102-key keyboard only.

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

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

Key No.	Base Case, or Shift+Num Lock Make/Break	Shift Case Make/Break *	Num Lock on Make/Break
75	E0 52 /E0 D2	E0 AA E0 52 /E0 D2 E0 2A	E0 2A E0 52 /E0 D2 E0 AA
76	E0 53 /E0 D3	E0 AA E0 53 /E0 D3 E0 2A	E0 2A E0 53 /E0 D3 E0 AA
79	E0 4B /E0 CB	E0 AA E0 4B /E0 CB E0 2A	E0 2A E0 4B /E0 CB E0 AA
80	E0 47 /E0 C7	E0 AA E0 47 /E0 C7 E0 2A	E0 2A E0 47 /E0 C7 E0 AA
81	E0 4F /E0 CF	E0 AA E0 4F /E0 CF E0 2A	E0 2A E0 4F /E0 CF E0 AA
83	E0 48 /E0 C8	E0 AA E0 48 /E0 C8 E0 2A	E0 2A E0 48 /E0 C8 E0 AA
84	E0 50 /E0 D0	E0 AA E0 50 /E0 D0 E0 2A	E0 2A E0 50 /E0 D0 E0 AA
85	E0 49 /E0 C9	E0 AA E0 49 /E0 C9 E0 2A	E0 2A E0 49 /E0 C9 E0 AA
86	E0 51 /E0 D1	E0 AA E0 51 /E0 D1 E0 2A	E0 2A E0 51 /E0 D1 E0 AA
89	E0 4D /E0 CD	E0 AA E0 4D /E0 CD E0 2A	E0 2A E0 4D /E0 CD E0 AA
* If the left Shift key is held down, the AA/2A shift break and make is sent with the other scan codes. If the right Shift key is held down, B6/36 is sent. If both Shift keys are down, both sets of codes are sent with the other scan code.			

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

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

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

Clock and Data Signals

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

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

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

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

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

Data Stream

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

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

Keyboard Data Output

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

Keyboard Encoding and Usage

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

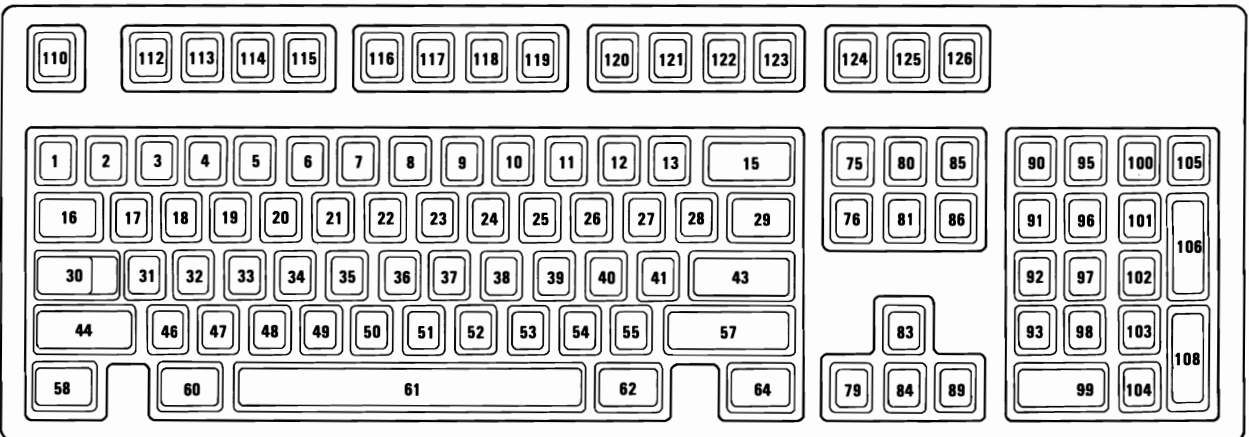
which resides in the read/write memory. This section discusses only the ROM routine.

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

Character Codes

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

The following figure shows the keyboard layout and key positions.



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

Notes:

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

Character Codes (Part 1 of 2)

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

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

Character Codes (Part 2 of 2)

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

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

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

Special Character Codes

Extended Functions

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

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

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

Keyboard Extended Functions (Part 1 of 2)

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

Keyboard Extended Functions (Part 2 of 2)

Shift States

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

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

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

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

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

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

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

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

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

Special Handling

System Reset

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

Break

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

Pause

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

Print Screen

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

System Request

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

Save the previous address.

Overlay interrupt vector hex 15.

Check AH for a value of hex 85:

If yes, process may begin.

If no, go to previous address.

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

Other Characteristics

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

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

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

Keyboard Layouts

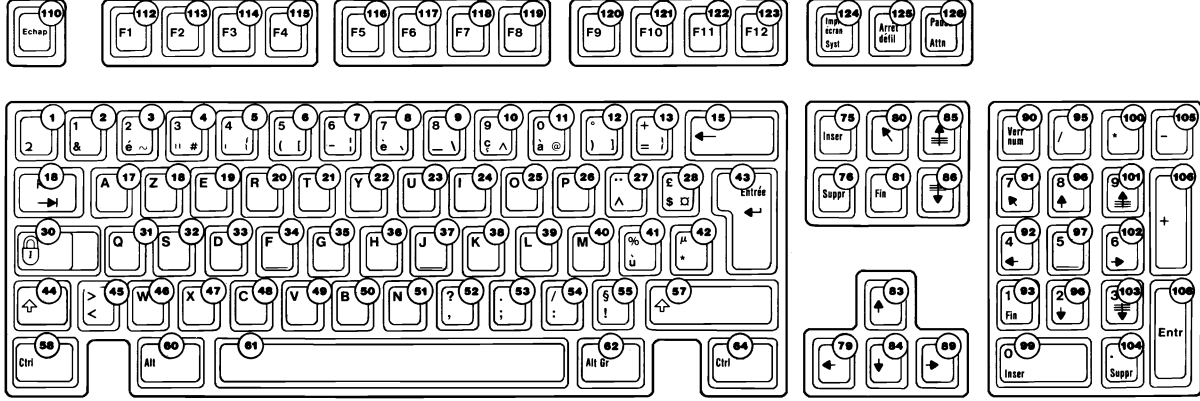
The keyboard is available in six layouts:

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

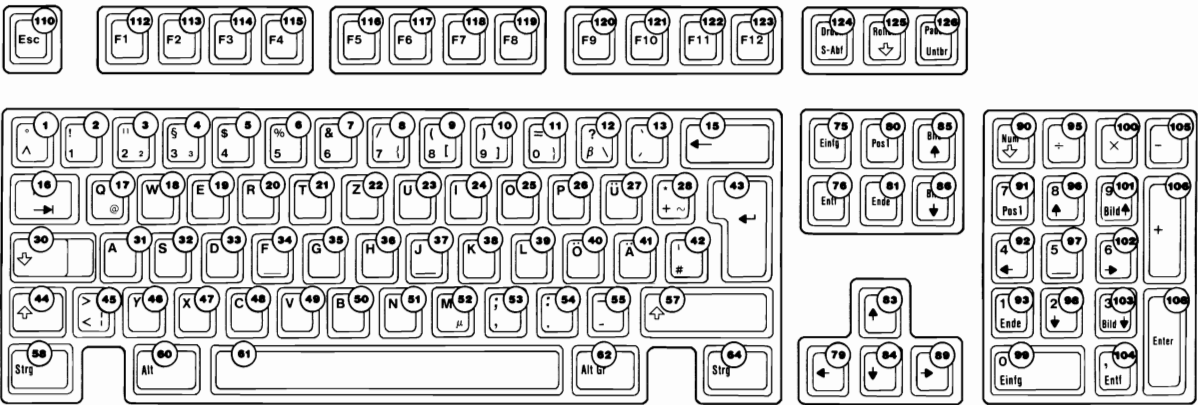
The various layouts are shown in alphabetic order on the following pages. Nomenclature is on both the top and front face of the keybuttons. The number to the upper right designates the keybutton position.



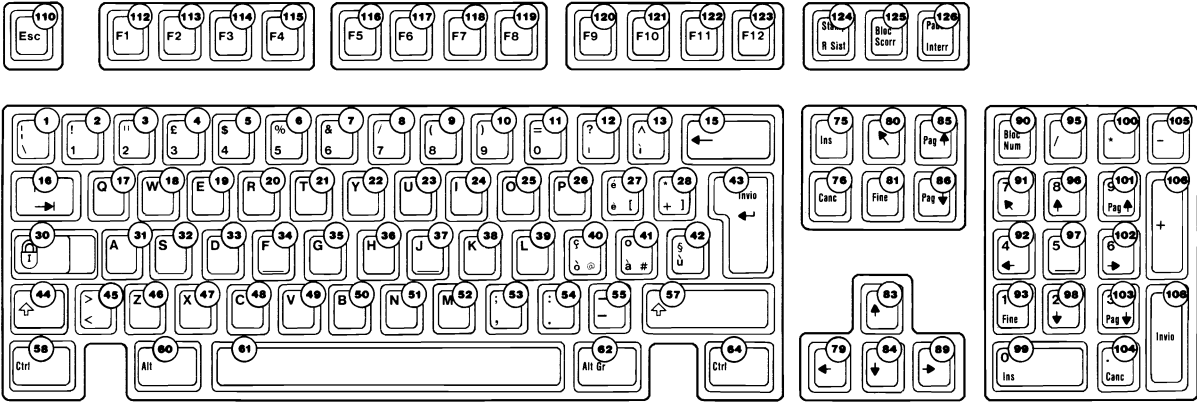
French Keyboard



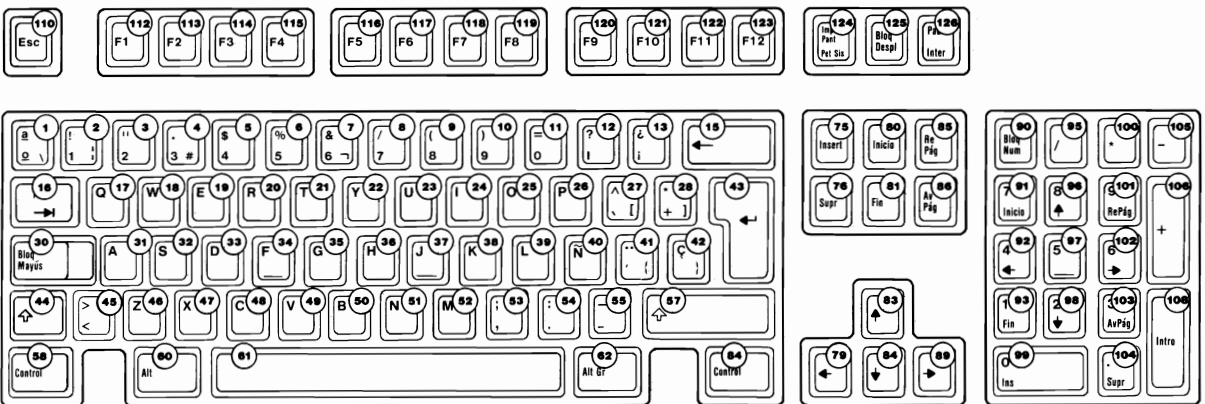
German Keyboard



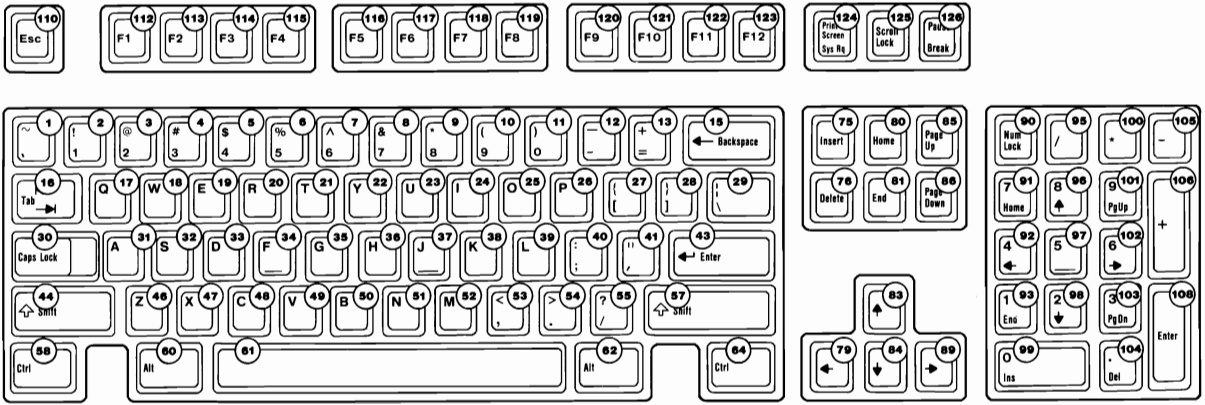
Italian Keyboard



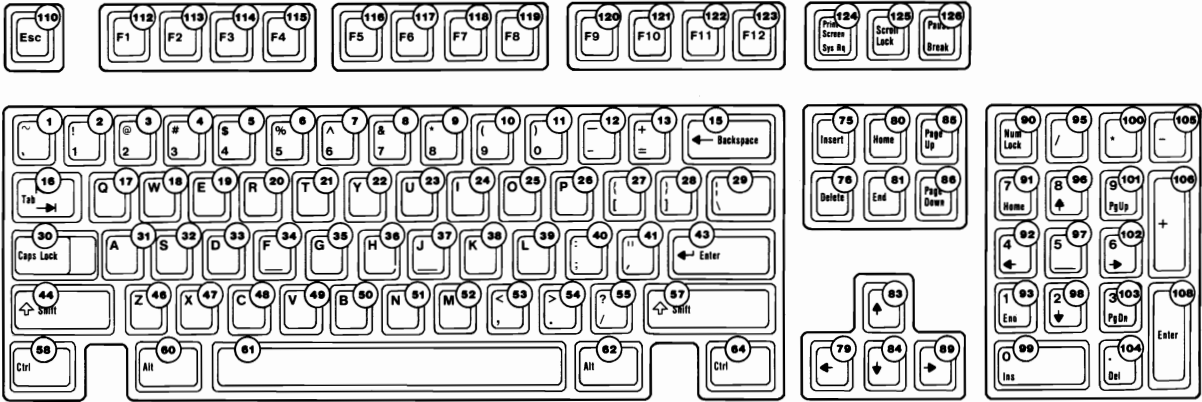
Spanish Keyboard



UK English Keyboard



US English Keyboard



Specifications

The specifications for the keyboard follow.

Power Requirements

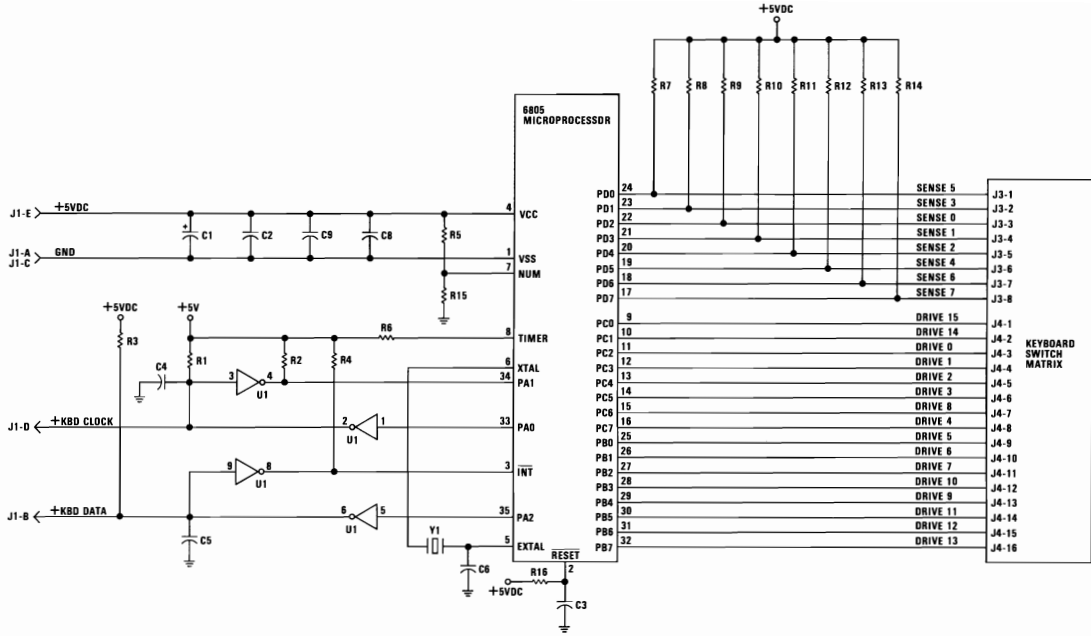
- +5 Vdc \pm 10%
- Current cannot exceed 275 mA.

Size

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

Weight

2.25 kilograms (5.0 pounds)



101/102-KEY KEYBOARD

SECTION 5. SYSTEM BIOS

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

Notes:



System BIOS Usage

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

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

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

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

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

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

INT 12H

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

Parameter Passing

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

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

```
MOV AH,1 ;function is to set time of day.
```

```
MOV CX,HIGH_COUNT ;establish the current time.
```

```
MOV DX,LOW_COUNT
```

```
INT 1AH ;set the time.
```

To read the time of day:

```
MOV AH,0 ;function is to read time of day.
```

```
INT 1AH ;read the timer.
```

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

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

8088 Software Interrupt Listing (Part 1 of 2)

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

8088 Software Interrupt Listing (Part 2 of 2)

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

Vectors with Special Meanings

Interrupt Hex 1B - Keyboard Break Address

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

Control may be retained by this routine, with the following problems. The Break may have occurred during interrupt

processing, so that one or more End of Interrupt commands must be sent to the 8259 Controller. Also, all I/O devices should be reset in case an operation was underway at that time.

Interrupt Hex 1C - Timer Tick

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

Interrupt Hex 1D - Video Parameters

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

Interrupt Hex 1E - Diskette Parameters

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

Interrupt Hex 1F - Graphics Character Extensions

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

Interrupt Hex 40 - Reserved

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

Interrupt Hex 41 - Fixed Disk Parameters

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

Other Read/Write Memory Usage

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

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

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

Hardware, Basic, and DOS Interrupts

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

Reserved Memory Locations

If you do DEF SEG (Default workspace segment):

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

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

Basic Workspace Variables

Example

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

L	H
Hex 64	Hex 00

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

BIOS Memory Map

BIOS Programming Hints

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

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

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

Adapter Cards with System-Accessible ROM Modules

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

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

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

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

System BIOS Listing - 01/10/86

Quick Reference - 256/640K Board

Map	5-13
Header	5-14
EQUATES	5-15
ABS0	5-19
DATA Segment	5-20
Diskette	5-23
INT 13H	5-23
Drive Type	5-25
Diskette __IO__ 1	5-25
DMA __Setup	5-36
Motor __On	5-40
Disk __Int	5-44
Diskette __Setup	5-45
Keyboard BIOS	5-46
Scan Tables	5-56
Printer BIOS	5-57
RS232 BIOS	5-59
Video BIOS	5-62
BIOS1	5-80
INT 15H	5-80
Joystick Support	5-82
POST	5-84
Determine Configuration	5-87
8259 Test	5-89
Keyboard Test	5-90
Expansion Test	5-91
Boot __Strap (INT 19H)	5-94
Time __of__ Day (INT 1AH)	5-95
Beep	5-96

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	Publica by Name	Address	Publica by Value
F000:E729	A1	F000:0000	HEADER
F000:15CC	ACT_DISP_PAGE	F000:10062	DISKETTE_IO_1
F000:6000	BASTC	F000:10A40	NEC_OUTPUT
F000:EC5C	BEEP	F000:10A64	SEEK
F000:1C4F	CASSETTE_IO_1	F000:0B32	RESULTS
F000:E73C	CONF_TBL	F000:10BC4	DISK_INT_1
F000:FA6E	CRT_CHAR_GEN	F000:10BDB	DSKETTE_SETUP
F000:FA12	DDS	F000:10C57	KEYBOARD_IO_1
F000:0062	DISKETTE_IO_1	F000:10718	KB_INT_1
F000:EFC7	DISK_BASE	F000:12BE	PRINTER_IO_1
F000:10BC4	DISK_INT_1	F000:1344	RS232_IO_1
F000:10BDB	DSKETTE_SETUP	F000:144E	VIDEO_IO_1
F000:1D37	FILL	F000:1485	SET_MODE
F000:0000	HEADER	F000:1563	VIDEO_RETURN
F000:1D78	KB_INT_1	F000:156C	SET_CTYPE
F000:1C57	KEYBOARD_IO_1	F000:158D	SET_CPOS
F000:F0E4	M5	F000:1586	READ_CURSOR
F000:F0EC	M6	F000:15CC	ACT_DISP_PAGE
F000:F0F4	M7	F000:15EE	SET_COLOR
F000:EF79	MD_TBL1	F000:1614	VIDEO_STATE
F000:EF86	MD_TBL2	F000:1635	SCROLL_UP
F000:EF93	MD_TBL3	F000:16D3	SCROLL_DOWN
F000:EFAD	MD_TBL4	F000:1725	READ_AC_CURRENT
F000:EFAD	MD_TBL5	F000:1782	WRITE_AC_CURRENT
F000:EFBA	MD_TBL6	F000:17B4	WRITE_C_CURRENT
F000:10A40	NEC_OUTPUT	F000:17E1	WRITE_STRING
F000:12BE	PRINTER_IO_1	F000:1866	READ_DOT
F000:FFF0	P_O_R	F000:1876	WRITE_DOT
F000:1725	READ_AC_CURRENT	F000:1B24	WRITE_TTY
F000:15B5	READ_CURSOR	F000:1BAB	READ_LPEN
F000:1865	READ_DOT	F000:1C4F	CASSETTE_IO_1
F000:1BAB	READ_LPEN	F000:1D37	FILL
F000:E05B	RESET	F000:1600	BASIC
F000:0B32	RESULTS	F000:E05B	RESET
F000:1344	RS232_IO_1	F000:E729	A1
F000:16D3	SCROLL_DOWN	F000:E73C	CONF_TBL
F000:1635	SCROLL_UP	F000:EC5C	BEEP
F000:10A64	SEEK	F000:ECA0	WAITF
F000:15EE	SET_COLOR	F000:EF79	MD_TBL1
F000:158D	SET_CPOS	F000:EF86	MD_TBL2
F000:156C	SET_CTYPE	F000:EF93	MD_TBL3
F000:1485	SET_MODE	F000:EFAD	MD_TBL4
F000:144E	VIDEO_IO_1	F000:EFAD	MD_TBL5
F000:F0A4	VIDEO_PARAMS	F000:EFBA	MD_TBL6
F000:1563	VIDEO_RETURN	F000:EFC7	DISK_BASE
F000:1614	VIDEO_STATE	F000:F0A4	VIDEO_PARAMS
F000:ECA0	WAITF	F000:F0E4	M5
F000:1782	WRITE_AC_CURRENT	F000:F0EC	M6
F000:17B4	WRITE_C_CURRENT	F000:F0F4	M7
F000:1876	WRITE_DOT	F000:FA12	DDS
F000:17E1	WRITE_STRING	F000:FA6E	CRT_CHAR_GEN
F000:1B24	WRITE_TTY	F000:FFF0	P_O_R

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

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

: BIOS I/O INTERFACE
:-----

: THESE LISTINGS PROVIDE INTERFACE INFORMATION FOR ACCESSING
: THE BIOS ROUTINES. THE POWER ON SELF TEST IS INCLUDED.

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

: MODULE REFERENCE
:-----

: HEADER.ASM --> DEFINITIONS
: DSEG.INC --> DATA SEGMENT LOCATIONS
: POSTEQU.INC --> COMMON EQUATES FOR POST AND BIOS
:
: DSKETTE.ASM --> DISKETTE BIOS
: DISKETTE_10_1 - INT 13H BIOS ENTRY (40H) -INT 13H
: DISK_INT_1 - HARDWARE INTERRUPT HANDLER -INT 0EH
: DSKETTE_SETUP - POST SETUP DRIVE TYPES
:
: KEYBRD.ASM --> KEYBOARD BIOS
: KEYBOARD_10_1 - INT 16H BIOS ENTRY -INT 16H
: KB_INT_1 - HARDWARE INTERRUPT -INT 09H
: SND_DATA - KEYBOARD TRANSMISSION
:
: PRT.ASM --> PRINTER ADAPTER BIOS -INT 17H
:
: RS232.ASM --> COMMUNICATIONS BIOS FOR RS232 -INT 14H
:
: VIDEO.ASM --> VIDEO BIOS -INT 10H
:
: BIOS1.ASM --> INTERRUPT 15H BIOS ROUTINES -INT 15H
: DEV_OPEN - NULL DEVICE OPEN HANDLER
: DEV_CLOSE - NULL DEVICE CLOSE HANDLER
: PROG_TERM - NULL PROGRAM TERMINATION
: JOY_STICK - JOYSTICK PORT HANDLER
: SYS_REQ - NULL SYSTEM REQUEST KEY
: EXT_MEMORY - EXTENDED MEMORY SIZE DETERMINE
: DEVICE_BUSY - NULL DEVICE BUSY HANDLER
: INT_COMPLETE - NULL INTERRUPT COMPLETE HANDLER
:
: POST.ASM --> BIOS INTERRUPT ROUTINES
: POST - POWER ON SELF TEST & INITIALIZATION
: TIME_OF_DAY_1 - TIME OF DAY ROUTINES -INT 1AH
: PRINT_SCREEN1 - PRINT SCREEN ROUTINE -INT 05H
: TIMER_INT_1 - TIMER1 INTERRUPT HANDLER ->INT 1CH
: DDS - LOAD (DSI) WITH DATA SEGMENT
: BEEP - SPEAKER BEEP CONTROL ROUTINE
: WAITF - FIXED TIME WAIT ROUTINE
:-----

.LIST

```
66 PAGE
67 C INCLUDE POSTEQU.INC
68 C |-----|
69 C | EQUATES USED BY POST AND BIOS |
70 C |-----|
71 C
72 = 0000 C SYSTEM EQU 0 ; 0 PC-XT, 1 PC-AT
73 = 0FB C MODEL_BYTE EQU 0FBH ; SYSTEM MODEL_BYTE
74 = 0000 C SUB_MODEL_BYTE EQU 000H ; SYSTEM SUB-MODEL_TYPE
75 = 0001 C BIOS_LEVEL EQU 001H ; BIOS REVISION LEVEL
76 C
77 C
78 C |-----| KEYBOARD INTERFACE AND DIAGNOSTIC CONTROL REGISTERS |-----|
79 = 0660 C PORT_A EQU 060H ; KEYBOARD SCAN CODE/CONTROL PORT
80 = 0661 C PORT_B EQU 061H ; PORT B READ/WRITE DIAGNOSTIC REGISTER
81 = 0662 C PORT_C EQU 062H ; 8255 PORT C ADDR
82 = 0663 C CMD_PORT EQU 063H
83 C
84 C
85 = 0660 C KB_DATA EQU 60H ; KEYBOARD SCAN CODE PORT
86 = 0661 C KB_CTL EQU 61H ; CONTROL BITS FOR KEYBOARD SENSE DATA
87 = 0664 C ID_2A EQU 054H ; ALTERNATE 2ND ID CHAR FOR KBX
88 = 06E0 C MC_EQ EQU 224 ; GENERAL MARKER CODE
89 = 06E1 C MC_E1 EQU 225 ; PAUSE KEY MARKER CODE
90 C
91 C
92 = 00F3 C RAM_PAR_ON EQU 11110011B ; AND MASK FOR PARITY CHECKING ENABLE ON
93 = 000C C RAM_PAR_OFF EQU 00001100B ; OR MASK FOR PARITY CHECKING ENABLE OFF
94 = 00C0 C PARITY_ERR EQU 11000000B ; R/W MEMORY - I/O CHANNEL PARITY ERROR
95 = 0001 C GATE2 EQU 00000001B ; TIMER 2 INPUT GATE CLOCK BIT
96 = 0002 C SPK2 EQU 00000010B ; SPEAKER OUTPUT DATA ENABLE BIT
97 = 0010 C REFRESH_BIT EQU 00010000B ; REFRESH TEST BIT
98 = 0020 C OUT2 EQU 00100000B ; SPEAKER TIMER OUT2 INPUT BIT
99 = 0040 C IO_CHECK EQU 01000000B ; I/O (MEMORY) CHECK OCCURRED BIT MASK
100 = 0080 C PARITY_CHECK EQU 10000000B ; MEMORY PARITY CHECK OCCURRED BIT MASK
101 C ENDIF
```

```

102 C PAGE
103 C ----- KEYBOARD RESPONSE -----
104 = 00AA C KB_OK EQU 0AAH ; RESPONSE FROM SELF DIAGNOSTIC
105 = 00FA C KB_ACK EQU 0FAH ; ACKNOWLEDGE FROM TRANSMISSION
106 = 00FE C KB_RESEND EQU 0FEH ; RESEND REQUEST
107 = 00FF C KB_OVER_RUN EQU 0FFH ; OVER RUN SCAN CODE
108
109 C ;----- FLAG EQUATES WITHIN *KB_FLAG -----
110 = 0001 C RIGHT_SHIFT EQU 00000001B ; RIGHT SHIFT KEY DEPRESSED
111 = 0002 C LEFT_SHIFT EQU 0000010B ; LEFT SHIFT KEY DEPRESSED
112 = 0004 C CTL_SHIFT EQU 00000100B ; CONTROL SHIFT KEY DEPRESSED
113 = 0008 C ALT_SHIFT EQU 00001000B ; ALTERNATE SHIFT KEY DEPRESSED
114 = 0010 C SCROLL_STATE EQU 00010000B ; SCROLL_LOCK STATE HAS BEEN TOGGLED
115 = 0020 C NUM_STATE EQU 00100000B ; NUM LOCK STATE HAS BEEN TOGGLED
116 = 0040 C CAPS_STATE EQU 01000000B ; CAPS LOCK STATE HAS BEEN TOGGLED
117 = 0080 C INS_STATE EQU 10000000B ; INSERT STATE IS ACTIVE
118
119 C ;----- FLAG EQUATES WITHIN *KB_FLAG_1 -----
120 = 0004 C SYS_SHIFT EQU 00000100B ; SYSTEM KEY DEPRESSED AND HELD
121 = 0008 C HOLD_STATE EQU 00001000B ; SUSPEND (MUST BE ZERO)
122 = 0010 C SCROLL_SHIFT EQU 00010000B ; SCROLL_LOCK KEY IS DEPRESSED
123 = 0020 C NUM_SHIFT EQU 00100000B ; NUM LOCK KEY IS DEPRESSED
124 = 0040 C CAPS_SHIFT EQU 01000000B ; CAPS LOCK KEY IS DEPRESSED
125 = 0080 C INS_SHIFT EQU 10000000B ; INSERT KEY IS DEPRESSED
126
127 C ;----- FLAG EQUATES WITHIN *KB_FLAG_2 -----
128 = 0007 C KB_LEDS EQU 00000111B ; KEYBOARD LED STATE BITS
129 C ; RESERVED (MUST BE ZERO)
130 = 0010 C KB_FA EQU 00010000B ; ACKNOWLEDGMENT RECEIVED
131 = 0020 C KB_FE EQU 00100000B ; RESEND RECEIVED FLAG
132 = 0040 C KB_PR_LED EQU 01000000B ; MODE INDICATOR UPDATE
133 = 0080 C KB_ERR EQU 10000000B ; KEYBOARD TRANSMIT ERROR FLAG
134
135 C ;----- FLAG EQUATES WITHIN *KB_FLAG_3 -----
136 = 0001 C LC_E1 EQU 00000001B ; LAST CODE WAS THE E1 HIDDEN CODE
137 = 0002 C LC_E0 EQU 00000010B ; LAST CODE WAS THE E0 HIDDEN CODE
138 = 0004 C R_CTL_SHIFT EQU 00000100B ; RIGHT CTL KEY DOWN
139 = 0008 C GRAPH_ON EQU 00001000B ; ALL GRAPHICS KEY DOWN (W.T. ONLY)
140 C ; RESERVED (MUST BE ZERO)
141 = 0010 C KBX EQU 00010000B ; KBX INSTALLED
142 = 0020 C SET_NUM_LK EQU 00100000B ; FORCE NUM LOCK IF READ ID AND KBX
143 = 0040 C LC_AB EQU 01000000B ; LAST CHARACTER WAS FIRST ID CHARACTER
144 = 0080 C RD_ID EQU 10000000B ; DOING A READ ID (MUST BE BIT0)
145
146 C ;----- KEYBOARD SCAN CODES -----
147 = 00AB C ID_1 EQU 0ABH ; 1ST ID CHARACTER FOR KBX
148 = 0041 C ID_2 EQU 041H ; 2ND ID CHARACTER FOR KBX
149 = 0038 C ALT_KEY EQU 56 ; SCAN CODE FOR ALTERNATE SHIFT KEY
150 = 001D C CTL_KEY EQU 29 ; SCAN CODE FOR CONTROL KEY
151 = 003A C CAPS_KEY EQU 58 ; SCAN CODE FOR SHIFT LOCK KEY
152 = 0053 C DEL_KEY EQU 83 ; SCAN CODE FOR DELETE KEY
153 = 0052 C INS_KEY EQU 82 ; SCAN CODE FOR INSERT KEY
154 = 002A C LEFT_KEY EQU 42 ; SCAN CODE FOR LEFT SHIFT
155 = 0045 C NUM_KEY EQU 69 ; SCAN CODE FOR NUMBER LOCK KEY
156 = 0036 C RIGHT_KEY EQU 54 ; SCAN CODE FOR RIGHT SHIFT
157 = 0046 C SCROLL_KEY EQU 70 ; SCAN CODE FOR SCROLL_LOCK KEY
158 = 0054 C SYS_KEY EQU 84 ; SCAN CODE FOR SYSTEM KEY
159 = 0057 C F11_M EQU 87 ; F11 KEY MAKE
160 = 0058 C F12_M EQU 88 ; F12 KEY MAKE

```

```

161 C PAGE
162 C ENDIF
163
164 C |----- DISKETTE EQUATES -----|
165 = 0050 C CARD_ID EQU 01010000B | CONTROLLER CARD I.D. BIT
166 = 0001 C DUAL EQU 00000010B | 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 00010000B | SET STATE DETERMINED IN STATE BITS
170 = 0010 C HOME EQU 00010000B | TRACK 0 MASK
171 = 0004 C SENSE_DRV_ST EQU 00000100B | SENSE DRIVE STATUS COMMAND
172 = 0030 C TRK_SLAP EQU 030H | CRASH STOP (48 TP1 DRIVES)
173 = 000A C QUIET_SEEK EQU 00AH | SEEK TO TRACK 10
174 = 0002 C MAX_DRV EQU 2 | MAX NUMBER OF DRIVES
175 = 000F C HD12_SETTLE EQU 15 | 1.2 M HEAD SETTLE TIME
176 = 0014 C HD320_SETTLE EQU 20 | 320 K HEAD SETTLE TIME
177 = 0025 C MOTOR_WAIT EQU 37 | 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 = 0020 C BAD_NEC EQU 020H | DISKETTE 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 = 0006 C MEDIA_CHANGE EQU 006H | MEDIA REMOVED ON DUAL ATTACH CARD
188 = 0004 C RECORD_NOT_FND EQU 004H | REQUESTED SECTOR 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 I/O
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 TRK_CAPA EQU 00000010B | 80 TRACK CAPABILITY
199 = 0002 C FMT_CAPA EQU 00000010B | MULTIPLE FORMAT CAPABILITY (1.2M)
200 = 0004 C DRV_DET EQU 00000100B | DRIVE DETERMINED
201 = 0010 C MED_DET EQU 00010000B | MEDIA DETERMINED BIT
202 = 0020 C DBL_STEP EQU 00100000B | DOUBLE STEP BIT
203 = 00C0 C RATE_500 EQU 11000000B | MASK FOR CLEARING ALL BUT RATE
204 = 0000 C RATE_500 EQU 00000000B | 500 KBS DATA RATE
205 = 0040 C RATE_300 EQU 01000000B | 300 KBS DATA RATE
206 = 0080 C RATE_250 EQU 10000000B | 250 KBS DATA RATE
207 = 000C C STRT_MSK EQU 00001100B | OPERATION START RATE MASK
208 = 00C0 C SEND_MSK EQU 11000000B | MASK FOR SEND RATE BITS
209
210 C |----- MEDIA/DRIVE STATE INDICATORS COMPATIBILITY -----|
211 = 0000 C M300U EQU 00000000B | 360 MEDIA/DRIVE NOT ESTABLISHED
212 = 0001 C M300I EQU 00000001B | 360 MEDIA,1.2DRIVE NOT ESTABLISHED
213 = 0002 C MIDIU EQU 00000010B | 1.2 MEDIA/DRIVE NOT ESTABLISHED
214 = 0007 C MED_UNK EQU 00000111B | NONE OF THE ABOVE

```

```

215 C PAGE
216 C |----- INTERRUPT EQUATES -----|
217 = 0020 C EQU EQU 020H ; END OF INTERRUPT COMMAND TO 8259
218 = 0020 C INTA00 EQU 020H ; 8259 PORT
219 = 0021 C INTA01 EQU 021H ; 8259 PORT
220 = 00A0 C INTB00 EQU 0A0H ; 2ND 8259
221 = 00A1 C INTB01 EQU 0A1H ;
222 = 0070 C INT_TYPE EQU 070H ; START OF 8259 INTERRUPT TABLE LOCATION
223 = 0010 C INT_VIDEO EQU 010H ; VIDEO VECTOR
224 C |-----|
225 = 0008 C DMA0B EQU 0008H ; DMA STATUS REGISTER PORT ADDRESS
226 = 0000 C DMA0B EQU 0000H ; DMA CH.0 ADDRESS REGISTER PORT ADDRESS
227 = 00D0 C DMA1B EQU 00D0H ; 2ND DMA STATUS PORT ADDRESS
228 = 00C0 C DMA1 EQU 00C0H ; 2ND DMA CH.0 ADDRESS REGISTER ADDRESS
229 C |-----|
230 = 0040 C TIMER EQU 040H ; 8253 TIMER - BASE ADDRESS
231 = 0043 C TIM_CTL EQU 043H ; 8253 TIMER CONTROL PORT ADDR
232 = 0040 C TIMER0 EQU 040H ; 8253 TIMER/CNTNR 0 PORT ADDR
233 C |-----|
234 C |----- MANUFACTURING PORT -----|
235 = 0080 C MFG_PORT EQU 80H ; MANUFACTURING AND POST CHECKPOINT PORT
236 C |-----|
237 C |-----|
238 C |----- MANUFACTURING BIT DEFINITION FOR *MFG_ERR_FLAG* -----|
239 = 0001 C MEM_FAIL EQU 00000001B ; STORAGE TEST FAILED (ERROR 20X)
240 = 0002 C PRO_FAIL EQU 00000002B ; VIRTUAL MODE TEST FAILED (ERROR 104)
241 = 0004 C LMCS_FAIL EQU 00000100B ; LOW MEG CHIP SELECT FAILED (ERROR 109)
242 = 0008 C KYCLK_FAIL EQU 00001000B ; KEYBOARD CLOCK TEST FAILED (ERROR 304)
243 = 0010 C KY_SYS_FAIL EQU 00010000B ; KEYBOARD OR SYSTEM FAILED (ERROR 303)
244 = 0020 C KYBD_FAIL EQU 00100000B ; KEYBOARD FAILED (ERROR 301)
245 = 0040 C DSK_FAIL EQU 01000000B ; DISKETTE TEST FAILED (ERROR 601)
246 = 0080 C KEY_FAIL EQU 10000000B ; KEYBOARD LOCKED (ERROR 302)
247 C |-----|
248 C |-----|
249 = 0081 C DMA_PAGE EQU 081H ; START OF DMA PAGE REGISTERS
250 = 008F C LAST_DMA_PAGE EQU 08FH ; LAST DMA PAGE REGISTER
251 C |-----|
252 C |-----|
253 C |X287 EQU 0F0H ; MATH COPROCESSOR CONTROL PORT
254 C |-----|
255 C |-----|
256 = 0000 C POST_SS EQU 000000H ; POST STACK SEGMENT
257 = 8000 C POST_SP EQU 800000H ; POST STACK POINTER
258 = 0030 C STACK_SS EQU 30H ; STACK SEGMENT USED DURING POST
259 = 0100 C TOS EQU 100H ; STACK -- USED DURING POST ONLY
260 C |-----|
261 C |-----|
262 C |-----|
263 C |-----|
264 = 000D C CR EQU 000DH ; CARRIAGE RETURN CHARACTER
265 = 000A C LF EQU 000AH ; LINE FEED CHARACTER
266 = 000B C RVRT EQU 000B0000B ; VIDEO VERTICAL RETRACE BIT
267 = 0001 C RHRZ EQU 00000001B ; VIDEO HORIZONTAL RETRACE BIT
268 = 0100 C H EQU 256 ; HIGH BYTE FACTOR (X 100H)
269 = 0101 C X EQU H+1 ; HIGH AND LOW BYTE FACTOR (X 101H)
270 C |-----|
271 .LIST
  
```

```

272          PAGE
273          C INCLUDE DSEG.INC
274          C ELSE
275          C :-----
276          C : 8088 INTERRUPT LOCATIONS :
277          C : REFERENCED BY POST & BIOS :
278          C :-----
279          C ENDIF
280
281          C AB50          SEGMENT AT 0          ; ADDRESS= 0000:0000
282          C
283          C *STG_LOCO      DB          ?          ; START OF INTERRUPT VECTOR TABLE
284          C
285          C *NMI_PTR        ORG          4*002H      ; NON-MASKABLE INTERRUPT VECTOR
286          C *NMI_PTR        DD          ?
287          C
288          C *INT5_PTR       ORG          4*005H      ; PRINT SCREEN INTERRUPT VECTOR
289          C *INT5_PTR       DD          ?
290          C
291          C *INT_PTR        ORG          4*008H      ; HARDWARE INTERRUPT POINTER (8-F)
292          C *INT_PTR        DD          ?
293          C
294          C *VIDEO_INT      ORG          4*010H      ; VIDEO I/O INTERRUPT VECTOR
295          C *VIDEO_INT      DD          ?
296          C
297          C *ORG_VECTOR     ORG          4*013H      ; DISKETTE/DISK INTERRUPT VECTOR
298          C *ORG_VECTOR     DD          ?
299          C
300          C *BASIC_PTR      ORG          4*018H      ; POINTER TO CASSETTE BASIC
301          C *BASIC_PTR      DD          ?
302          C
303          C *PARAM_PTR       ORG          4*01DH      ; POINTER TO VIDEO PARAMETERS
304          C *PARAM_PTR       DD          ?
305          C
306          C *DISK_POINTER    ORG          4*01EH      ; POINTER TO DISKETTE PARAMETER TABLE
307          C *DISK_POINTER    DD          ?
308          C
309          C *EXT_PTR        ORG          4*01FH      ; POINTER TO GRAPHIC CHARACTERS 128-255
310          C *EXT_PTR        DD          ?
311          C
312          C *DISK_VECTOR     ORG          4*040H      ; POINTER TO DISKETTE INTERRUPT CODE
313          C *DISK_VECTOR     DD          ?
314          C
315          C *HF_TBL_VEC      ORG          4*041H      ; POINTER TO FIRST DISK PARAMETER TABLE
316          C *HF_TBL_VEC      DD          ?
317          C
318          C *HF1_TBL_VEC     ORG          4*046H      ; POINTER TO SECOND DISK PARAMETER TABLE
319          C *HF1_TBL_VEC     DD          ?
320          C
321          C *SLAVE_INT_PTR   ORG          4*070H      ; POINTER TO SLAVE INTERRUPT HANDLER
322          C *SLAVE_INT_PTR   DD          ?
323          C
324          C *HDISK_INT      ORG          4*076H      ; POINTER TO FIXED DISK INTERRUPT CODE
325          C *HDISK_INT      DD          ?
326          C
327          C          ORG          400H
328          C          LABEL DATA_AREA          ; ABSOLUTE LOCATION OF DATA SEGMENT
329          C          LABEL DATA_WORD          ;
330          C
331          C *MFG_TEST_RTN   ORG          0500H      ; LOAD LOCATION FOR MANUFACTURING TESTS
332          C *MFG_TEST_RTN   LABEL FAR
333          C
334          C *BOOT_LOCN      ORG          TC00H      ; BOOT STRAP CODE LOAD LOCATION
335          C *BOOT_LOCN      LABEL FAR
336          C
337          C AB50          ENDS
  
```

```

338 C PAGE
339 C |-----|
340 C | ROM BIOS DATA AREAS |
341 C |-----|
342 C
343 C DATA SEGMENT AT 40H | ADDRESS= 0040:0000
344 C DATA40 LABEL BYTE
345 C *RS232_BASE DW ? | BASE ADDRESSES OF RS232 ADAPTERS
346 C 0002 ???? | SECOND LOGICAL RS232 ADAPTER
347 C 0004 ???? | RESERVED
348 C 0006 ???? | RESERVED
349 C *PRINTER_BASE DW ? | BASE ADDRESSES OF PRINTER ADAPTERS
350 C 000A ???? | SECOND LOGICAL PRINTER ADAPTER
351 C 000C ???? | THIRD LOGICAL PRINTER ADAPTER
352 C 000E ???? | RESERVED
353 C *EQUIP_FLAG DW ? | INSTALLED HARDWARE FLAGS
354 C *MFG_TST DB ? | INITIALIZATION FLAGS
355 C *MEMORY_SIZE DW ? | BASE MEMORY SIZE IN K BYTES (X 1024)
356 C *MFG_ERR_FLAG DB ? | SCRATCHPAD FOR MANUFACTURING
357 C 0016 ?? | ERROR CODES
358 C
359 C |-----|
360 C | KEYBOARD DATA AREAS |
361 C |-----|
362 C
363 C *KB_FLAG DB ? | KEYBOARD SHIFT STATE AND STATUS FLAGS
364 C *KB_FLAG_1 DB ? | SECOND BYTE OF KEYBOARD STATUS
365 C *AL_INPUT DB ? | STORAGE FOR ALTERNATE KEY PAD ENTRY
366 C *BUFFER_HEAD DW ? | POINTER TO HEAD OF KEYBOARD BUFFER
367 C *BUFFER_TAIL DW ? | POINTER TO TAIL OF KEYBOARD BUFFER
368 C
369 C |-----|
370 C | HEAD = TAIL INDICATES THAT THE BUFFER IS EMPTY
371 C *KB_BUFFER DW 16 DUP(?) | ROOM FOR 15 SCAN CODE ENTRIES
372 C
373 C |-----|
374 C | DISKETTE DATA AREAS |
375 C |-----|
376 C
377 C *SEEK_STATUS DB ? | DRIVE RECALIBRATION STATUS
378 C | BIT 3-0 = DRIVE 3-0 RECALIBRATION
379 C 003E ?? | BEFORE NEXT SEEK IF BIT 15 = 0
380 C *MOTOR_STATUS DB ? | MOTOR STATUS
381 C | BIT 3-0 = DRIVE 3-0 CURRENTLY RUNNING
382 C 003F ?? | BIT 7 = CURRENT OPERATION IS A WRITE
383 C | TIME OUT COUNTER FOR MOTOR(S) TURN OFF
384 C *MOTOR_COUNT DB ? | RETURN CODE STATUS BYTE
385 C 0040 ?? | CMD BLOCK IN STACK FOR DISK OPERATION
386 C *DISKETTE_STATUS DB ? | STATUS BYTES FROM DISKETTE OPERATION
387 C 0041 ??
388 C *NEC_STATUS DB 7 DUP(?)
389 C 0042 07 [ ?? ]
390 C
391 C |-----|
392 C | VIDEO DISPLAY DATA AREA |
393 C |-----|
394 C
395 C *CRT_MODE DB ? | CURRENT DISPLAY MODE (TYPE)
396 C 0049 ?? | NUMBER OF COLUMNS ON SCREEN
397 C *CRT_COLS DW ? | LENGTH OF REGEN BUFFER IN BYTES
398 C 004A ???? | *CRT_START DW ? | STARTING ADDRESS IN REGEN BUFFER
399 C *CRT_LEN DW ? | CURSOR FOR EACH OF UP TO 8 PAGES
400 C 004C ????
401 C *CURSOR_POSN DW 8 DUP(?)
402 C 0050 08 [ ???? ]
403 C
404 C *CURSOR_MODE DW ? | CURRENT CURSOR MODE SETTING
405 C 0060 ???? | CURRENT PAGE BEING DISPLAYED
406 C *ACTIVE_PAGE DB ? | BASE ADDRESS FOR ACTIVE DISPLAY CARD
407 C 0062 ?? | CURRENT SETTING OF THE 3x8 REGISTER
408 C *ADDR_6845 DW ? | CURRENT PALETTE SETTING - COLOR CARD
409 C *CRT_MODE_SET DB ?
410 C *CRT_PALETTE DB ?
411 C
412 C |-----|
413 C | POST AND BIOS WORK DATA AREA |
414 C |-----|
415 C
416 C *IO_ROM_INIT DW ? | STACK SAVE, ETC.
417 C 0067 ???? | POINTER TO ROM INITIALIZATION ROUTINE
418 C *IO_ROM_SEG DW ? | POINTER TO I/O ROM SEGMENT
419 C *INTR_FLAG DB ? | FLAG INDICATING AN INTERRUPT HAPPENED
420 C
421 C |-----|
422 C | TIMER DATA AREA |
423 C |-----|
424 C *TIMER_LOW DW ? | LOW WORD OF TIMER COUNT
425 C 006C ???? | HIGH WORD OF TIMER COUNT
426 C *TIMER_HIGH DW ? | TIMER HAS ROLLED OVER SINCE LAST READ
427 C 0070 ????
428 C *TIMER_OFL DB ?
429 C
430 C |-----|
431 C | SYSTEM DATA AREA |
432 C |-----|
433 C *BIOS_BREAK DB ? | BIT 7=1 IF BREAK KEY HAS BEEN PRESSED
434 C *RESET_FLAG DW ? | WORD=1234H IF KEYBOARD RESET UNDERWAY
435 C
436 C |-----|
437 C | FIXED DISK DATA AREAS |
438 C |-----|
439 C *DISK_STATUS1 DB ? | FIXED DISK STATUS
440 C 0074 ?? | COUNT OF FIXED DISK DRIVES
441 C *HF_NUM DB ? | HEAD CONTROL BYTE
442 C 0075 ?? | *CONTROL_BYTE DB ? | *PORT_OFF DB ? | RESERVED (PORT OFFSET)

```



```

443 C PAGE
444 |-----|
445 | TIME-OUT VARIABLES |
446 |-----|
448 0078 ?? C *PRINT_TIM_OUT DB ? | TIME OUT COUNTERS FOR PRINTER RESPONSE
449 0079 ?? C DB ? | SECOND LOGICAL PRINTER ADAPTER
450 007A ?? C DB ? | THIRD LOGICAL PRINTER ADAPTER
451 007B ?? C DB ? | RESERVED
452 007C ?? C *RS232_TIM_OUT DB ? | TIME OUT COUNTERS FOR RS232 RESPONSE
453 007D ?? C DB ? | SECOND LOGICAL RS232 ADAPTER
454 007E ?? C DB ? | RESERVED
455 007F ?? C DB ? | RESERVED
456
457 C |-----|
458 | ADDITIONAL KEYBOARD DATA AREA |
459 |-----|
460
461 C
462 0080 ???? C *BUFFER_START DW ? | BUFFER LOCATION WITHIN SEGMENT 40H
463 0082 ???? C *BUFFER_END DW ? | OFFSET OF KEYBOARD BUFFER START
| | | | | OFFSET OF END OF BUFFER
464
465 C |-----|
466 | EGA/PGA DISPLAY WORK AREA |
467 |-----|
468
469 0084 ?? C *ROWS DB ? | ROWS ON THE ACTIVE SCREEN (LESS 1)
470 0085 ???? C *POINTS DW ? | BYTES PER CHARACTER
471 0087 ?? C *INFO DB ? | MODE OPTIONS
472 0088 ?? C *INFO_3 DB ? | FEATURE BIT SWITCHES
473 0089 ?? C DB ? | RESERVED FOR DISPLAY ADAPTERS
474 008A ?? C DB ? | RESERVED FOR DISPLAY ADAPTERS
475
476 C |-----|
477 | ADDITIONAL MEDIA DATA |
478 |-----|
479
480 008B ?? C *LAstrate DB ? | LAST DISKETTE DATA RATE SELECTED
481 008C ?? C *HF_STATUS DB ? | STATUS REGISTER
482 008D ?? C *HF_ERROR DB ? | ERROR REGISTER
483 008E ?? C *HF_INT_FLAG DB ? | FIXED DISK INTERRUPT FLAG
484 008F ?? C *HF_CNTRL DB ? | BIT 0 -> PC-1/DUAL FDC ADAPTER CARD
485 0090 ?? C *DSK_STATE DB ? | DRIVE 0 MEDIA STATE
486 0091 ?? C DB ? | DRIVE 1 MEDIA STATE
487 0092 ?? C DB ? | DRIVE 0 OPERATION START STATE
488 0093 ?? C DB ? | DRIVE 1 OPERATION START STATE
489 0094 ?? C *DSK_TRK DB ? | DRIVE 0 PRESENT CYLINDER
490 0095 ?? C DB ? | DRIVE 1 PRESENT CYLINDER
491
492 C |-----|
493 | ADDITIONAL KEYBOARD FLAGS |
494 |-----|
495
496 0096 ?? C *KB_FLAG_3 DB ? | KEYBOARD MODE STATE AND TYPE FLAGS
497 0097 ?? C *KB_FLAG_2 DB ? | KEYBOARD LED FLAGS
498
499 C |-----|
500 | REAL TIME CLOCK DATA AREA |
501 |-----|
502
503 C
504 0098 ???? C *USER_FLAG DW ? | OFFSET ADDRESS OF USERS WAIT FLAG
505 009A ???? C *USER_FLAG_SEG DW ? | SEGMENT ADDRESS OF USER WAIT FLAG
506 009C ???? C *RTC_LOW DW ? | LOW WORD OF USER WAIT FLAG
507 009E ???? C *RTC_HIGH DW ? | HIGH WORD OF USER WAIT FLAG
508 00A0 ?? C *RTC_WAIT_FLAG DB ? | WAIT ACTIVE FLAG (01=BUSY, 80=POSTED)
509 ENDF | | | | | (00=POST ACKNOWLEDGED)
510
511 C |-----|
512 | AREA FOR NETWORK ADAPTER |
513 |-----|
514
515 00A1 07 [ ?? ] C *NET DB 7 DUP(?) | RESERVED FOR NETWORK ADAPTERS
516
517 C |-----|
518 | EGA/PGA PALETTE POINTER |
519 |-----|
520
521 C
522 C
523 00AB ???????? C *SAVE_PTR DD ? | POINTER TO EGA PARAMETER CONTROL BLOCK
524
525 C |-----|
526 | TIMER DATA |
527 |-----|
528
529 00CE C *DAY_COUNT ORG 0CEH | COUNT OF DAYS FROM 1-1-80
530 00CE ???? C DW ? |
531
532 C |-----|
533 | DATA AREA - PRINT SCREEN |
534 |-----|
535
536 C
537 C
538 ORG 100H | ADDRESS= 0040:0100 (REF 0050:1000)
539 0100 ?? C *STATUS_BYTE DB ? | PRINT SCREEN STATUS BYTE
540 | 00=READY/OK, 01=BUSY, FF=ERROR
541
542 0101 C DATA ENDS | END OF BIOS DATA SEGMENT
543
544 .LIST

```

```
545                                     PAGE
546 0000                               CODE   SEGMENT WORD PUBLIC
547                                     PUBLIC  HEADER
548
549                                     ASSUME  CS:CODE,DS:NOTHING,ES:NOTHING,SS:NOTHING
550
551                                     HEADER  PROC   NEAR
552 0000
553
554 = 0000                               BEGIN  EQU    $
555 0000 36 32 58 30 38 35              DB    '62X0854 CDPR. IBM CORP. 1981,1986 '
556                                     '
557                                     '
558                                     '
559                                     '
560                                     '
561                                     '
562 0022 20 20 20 20 20 20              EVEN  DB    '
563                                     '
564                                     '
565                                     '
566 0039 20 20 20 20 20 20              DB    '
567                                     '
568                                     '
569                                     '
570
571 0050                               HEADER  ENDP
572 0050                               CODE   ENDS
573                                     END
```

```

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

SECTION 5

```

115 :-----
116 : (AH)=16 DISK CHANGE LINE STATUS
117 : OUTPUT REGISTERS
118 : (AH) - 00 - DISK CHANGE LINE NOT ACTIVE
119 :       06 - DISK CHANGE LINE ACTIVE & CARRY BIT ON
120 : (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
121 :-----
122 :
123 : (AH)=17 SET DASD TYPE FOR FORMAT
124 : INPUT REGISTERS
125 : (AL) - 00 - NOT USED
126 :       01 - DISKETTE 320/360K IN 360K DRIVE
127 :       02 - DISKETTE 360K IN 1.2M DRIVE
128 :       03 - DISKETTE 1.2M IN 1.2M DRIVE
129 :       04 - DISKETTE 720K IN 720K DRIVE
130 : (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED);
131 :       DO NOT USE WHEN DISKETTE ATTACH CARD USED)
132 :-----
133 : (AH)=18 SET MEDIA TYPE FOR FORMAT
134 : INPUT REGISTERS
135 : (CH) - LOW ORDER 8 OF 10 BITS MAXIMUM NUMBER OF TRACKS
136 : (CL) - BITS 7 & 6 - HIGH ORDER TWO BITS OF MAXIMUM TRACKS
137 :       - BITS 5 THRU 0 - MAXIMUM SECTORS PER TRACK
138 : (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
139 : OUTPUT REGISTERS
140 : (ES:DI) - POINTER TO DRIVE PARAMETERS TABLE FOR THIS MEDIA TYPE,
141 :          UNCHANGED IF (AH) IS NON-ZERO
142 : (AH) - 00H, CY = 0, TRACK AND SECTORS/TRACK COMBINATION IS SUPPORTED
143 :       01H, CY = 1, FUNCTION IS NOT AVAILABLE
144 :       0CH, CY = 1, TRACK AND SECTORS/TRACK COMBINATION IS NOT SUPPORTED
145 :-----
146 : DISK CHANGE STATUS IS ONLY CHECKED WHEN A MEDIA SPECIFIED IS OTHER
147 : THAN 360 KB DRIVE. IF THE DISK CHANGE LINE IS FOUND TO BE
148 : ACTIVE THE FOLLOWING ACTIONS TAKE PLACE:
149 :   ATTEMPT TO RESET DISK CHANGE LINE TO INACTIVE STATE.
150 :   IF ATTEMPT SUCCEEDS SET DASD TYPE FOR FORMAT AND RETURN DISK
151 :   CHANGE ERROR CODE
152 :   IF ATTEMPT FAILS RETURN TIMEOUT ERROR CODE AND SET DASD TYPE
153 :   TO A PREDETERMINED STATE INDICATING MEDIA TYPE UNKNOWN.
154 :   IF THE DISK CHANGE LINE IN INACTIVE PERFORM SET DASD TYPE FOR FORMAT.
155 :-----
156 : DATA VARIABLE -- #DISK POINTER
157 : DOUBLE WORD POINTER TO THE CURRENT SET OF DISKETTE PARAMETERS
158 :-----
159 : OUTPUT FOR ALL FUNCTIONS
160 : AH = STATUS OF OPERATION
161 : STATUS BITS ARE DEFINED IN THE EQUATES FOR #DISKETTE_STATUS
162 : VARIABLE IN THE DATA SEGMENT OF THIS MODULE
163 : CY = 0 SUCCESSFUL OPERATION (AH=0 ON RETURN, EXCEPT FOR READ DASD
164 :     TYPE AH=(15))
165 : CY = 1 FAILED OPERATION (AH HAS ERROR REASON)
166 : FOR READ/WRITE/VERIFY
167 : DS,BX,DX,CX PRESERVED
168 : NOTE: IF AN ERROR IS REPORTED BY THE DISKETTE CODE, THE APPROPRIATE
169 : ACTION IS TO RESET THE DISKETTE, THEN RETRY THE OPERATION.
170 : ON READ ACCESSES, NO MOTOR START DELAY IS TAKEN, SO THAT
171 : THREE RETRIES ARE REQUIRED ON READS TO ENSURE THAT THE
172 : PROBLEM IS NOT DUE TO MOTOR START-UP.
173 :-----
174 : .LIST
175 : DISKETTE STATE MACHINE - ABSOLUTE ADDRESS 40:90 (DRIVE A) & 91 (DRIVE B)
176 : .LIST
177 :
178 : | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
179 : |---|---|---|---|---|---|---|---|
180 : |   |   |   |   |   |   |   |   |
181 : |   |   |   |   |   |   |   |   |
182 : |   |   |   |   |   |   |   |   |
183 : |   |   |   |   |   |   |   |   |
184 : |   |   |   |   |   |   |   |   |
185 : |   |   |   |   |   |   |   |   |
186 : |   |   |   |   |   |   |   |   |
187 : |   |   |   |   |   |   |   |   |
188 : |   |   |   |   |   |   |   |   |
189 : |   |   |   |   |   |   |   |   |
190 : |   |   |   |   |   |   |   |   |
191 : |   |   |   |   |   |   |   |   |
192 : |   |   |   |   |   |   |   |   |
193 : |   |   |   |   |   |   |   |   |
194 : |   |   |   |   |   |   |   |   |
195 : |   |   |   |   |   |   |   |   |
196 : |   |   |   |   |   |   |   |   |
197 : |   |   |   |   |   |   |   |   |
198 : |   |   |   |   |   |   |   |   |
199 : |   |   |   |   |   |   |   |   |
200 : |   |   |   |   |   |   |   |   |
201 : |-----|-----> DATA TRANSFER RATE FOR THIS DRIVE:
202 :
203 : |   |   |   |   |   |   |   |   |
204 : |   |   |   |   |   |   |   |   |
205 : |   |   |   |   |   |   |   |   |
206 : |   |   |   |   |   |   |   |   |
207 : |   |   |   |   |   |   |   |   |
208 : |   |   |   |   |   |   |   |   |
209 : |   |   |   |   |   |   |   |   |
210 : |-----|-----> STATE OPERATION STARTED - ABSOLUTE ADDRESS 40:192 (DRIVE A) & 93 (DRIVE B)
211 : .LIST
212 :
213 : |-----|-----> PRESENT CYLINDER NUMBER - ABSOLUTE ADDRESS 40:194 (DRIVE A) & 95 (DRIVE B)
214 : .LIST
    
```

```

215 PAGE
216
217 MD_STRUC STRUC
218 0000 ?? MD_SPEC1 DB ? ; SRT=D, HD UNLOAD#F - 1ST SPECIFY BYTE
219 0001 ?? MD_SPEC2 DB ? ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
220 0002 ?? MD_OFF_TIM DB ? ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
221 0003 ?? MD_BYT_SEC DB ? ; 512 BYTES/SECTOR
222 0004 ?? MD_SEC_TRK DB ? ; EOT (LAST SECTOR ON TRACK)
223 0005 ?? MD_GAP DB ? ; GAP LENGTH
224 0006 ?? MD_DTL DB ? ; DTL
225 0007 ?? MD_GAP3 DB ? ; GAP LENGTH FOR FORMAT
226 0008 ?? MD_FIL_BYT DB ? ; FILL BYTE FOR FORMAT
227 0009 ?? MD_HD_TIM DB ? ; HEAD SETTLE TIME (MILLISECONDS)
228 000A ?? MD_STR_TIM DB ? ; MOTOR START TIME (1/8 SECONDS)
229 000B ?? MD_MAX_TRK DB ? ; MAX. TRACK NUMBER
230 000C ?? MD_RATE DB ? ; DATA TRANSFER RATE
231 000D MD_STRUC ENDS
232
233 = 007F BITTOFF EQU 7FH
234 = 0080 BITTON EQU 80H
235
236 PUBLIC DISK_INT
237 PUBLIC DISKETTE_SETUP
238 PUBLIC DISKETTE_IO_1
239 PUBLIC NEC_OUTPUT
240 PUBLIC RESULTS
241 PUBLIC SEEK
242
243 EXTRN DDS:NEAR
244 EXTRN DISK_BASE:NEAR
245 EXTRN WAITF:NEAR
246 EXTRN MD_TBL1:NEAR
247 EXTRN MD_TBL2:NEAR
248 EXTRN MD_TBL3:NEAR
249 EXTRN MD_TBL4:NEAR
250 EXTRN MD_TBL5:NEAR
251 EXTRN MD_TBL6:NEAR
252
253 0000 CODE SEGMENT BYTE PUBLIC
254 ASSUME CS:CODE,DS:DATA,ES:DATA
255
256 -----
257 ; DRIVE TYPE TABLE
258 ; -----
259
260 OR_TYPE LABEL BYTE
261 0000 01 DB 01 ; 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 0007 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 0000 0000 E DW OFFSET MD_TBL5
271 000F 04 DB 04
272 0010 0000 E DW OFFSET MD_TBL6
273 = 0012 =# ; END OF TABLE
274 = 0006 DR_CMT EQU (DR_TYPE_E-DR_TYPE)/3 ; NUMBER OF DRIVE TYPES
275
276 0012 DISKETTE_IO_1 PROC FAR ;>>> ENTRY POINT FOR ORG 0EC59H
277 0012 FB STI ; INTERRUPTS BACK ON
278 0013 55 PUSH BP ; USER REGISTER
279 0014 57 PUSH DI ; USER REGISTER
280 0015 52 PUSH DX ; HEAD #, DRIVE # OR USER REGISTER
281 0016 53 PUSH BX ; BUFFER OFFSET PARAMETER OR REGISTER
282 0017 51 PUSH CX ; TRACK #-SECTOR # OR USER REGISTER
283 0018 8B EC MOV BP,SP ; BP => PARAMETER LIST DEP. ON AH
284 ; [BP] = SECTOR #
285 ; [BP+1] = TRACK #
286 ; [BP+2] = BUFFER OFFSET
287 ; FOR RETURN OF DRIVE PARAMETERS:
288 ; CL/[BP] = BITS 7:6 HI BITS OF MAX CYL
289 ; DI/[BP+6] = BITS 0-5 MAX SECTORS/TRACK
290 ; CH/[BP+1] = LOW 8 BITS OF MAX CYL.
291 ; BL/[BP+2] = BITS 7-4 = 0
292 ; ; BITS 3-0 = VALID CMOS TYPE
293 ; BH/[BP+3] = 0
294 ; DL/[BP+4] = # DRIVES INSTALLED
295 ; DH/[BP+5] = MAX HEAD #
296 ; DI/[BP+6] = OFFSET TO DISK BASE
297 001A 1E PUSH DS ; BUFFER SEGMENT PARM OR USER REGISTER
298 001B 56 PUSH SI ; USER REGISTERS
299 001C E8 FC 19 CALL DDS ; SEGMENT OF BIOS DATA AREA TO DS
300 001F 80 FC 19 CMP AH,(FNC_TAE-FNC_TAB)/2 ; CHECK FOR > LARGEST FUNCTION
301 0022 72 02 JB OK_FUNC ; FUNCTION OK
302 0024 B4 14 MOV AH,14H ; REPLACE WITH KNOWN INVALID FUNCTION
303 0026 OK_FUNC:
304 0026 80 FC 01 CMP AH,1 ; RESET OR STATUS ?
305 0029 76 0C JBE OK_DRV ; IF RESET OR STATUS DRIVE ALWAYS OK
306 002B 80 FC 08 CMP AH,8 ; READ DRIVE PARMS ?
307 002E 74 07 JZ OK_DRV ; IF SO DRIVE CHECKED LATER
308 0030 80 FA 03 CMP DL,3 ; DRIVES 0,1,2 AND 3 OK
309 0033 76 02 JBE OK_DRV ; IF 0 OR 1 THEN JUMP
310 0035 B4 14 MOV AH,14H ; REPLACE WITH KNOWN INVALID FUNCTION
311 0037 OK_DRV:
312 0037 8A CC MOV CL,AH ; CL = FUNCTION
313 0039 32 ED XOR CH,CH ; CX = FUNCTION
314 003B 00 E1 SHL CL,1 ; FUNCTION TIMES 2
315 003D BB 0060 R MOV BX,OFFSET FNC_TAB ; CHECK FOR > FUNCTION TABLE
316 0040 03 D9 ADD BX,CX ; ADD OFFSET INTO TABLE => ROUTINE
317 0042 8A E6 MOV AH,DH ; AX = HEAD #, # OF SECTORS OR DASD TYPE
318 0044 32 F6 XOR DH,DH ; DX = DRIVE #
319 0046 8B FA MOV SI,AX ; SI = HEAD #, # OF SECTORS OR DASD TYPE
320 0048 8B FA MOV DI,DX ; DI = DRIVE #
321 004A 8A 26 0041 R MOV AH,#DISKETTE_STATUS ; LOAD STATUS TO AH FOR STATUS FUNCTION
322 004E C6 06 0041 R MOV #DISKETTE_STATUS,0 ; INITIALIZE FOR ALL OTHERS
323
324 ; THROUGHOUT THE DISKETTE BIOS, THE FOLLOWING INFORMATION IS CONTAINED IN
325 ; THE FOLLOWING MEMORY LOCATIONS AND REGISTERS. NOT ALL DISKETTE BIOS
326 ; FUNCTIONS REQUIRE ALL OF THESE PARAMETERS.
327 ;
328 ; DI ; DRIVE #
    
```

```

329      ; SI-HI : HEAD #
330      ; SI-LOW : # OF SECTORS OR DASH TYPE FOR FORMAT
331      ; ES : BUFFER SEGMENT
332      ; [BP] : SECTOR #
333      ; [BP-1] : TRACK #
334      ; [BP+2] : BUFFER OFFSET
335      ;
336      ;
337      ; ACROSS CALLS TO SUBROUTINES THE CARRY FLAG (CY+1), WHERE INDICATED IN
338      ; SUBROUTINE PROLOGUES, REPRESENTS AN EXCEPTION RETURN (NORMALLY AN ERROR
339      ; CONDITION). IN MOST CASES, WHEN CY = 1, *DISKETTE_STATUS CONTAINS THE
340      ; SPECIFIC ERROR CODE.
341      0053 2E: FF 17      CALL   WORD PTR CS:[BX]      ; [AH] = *DISKETTE_STATUS
342                                     ; CALL THE REQUESTED FUNCTION
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 5F             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   DISK RESET      ; AH = 00: RESET
354      0062 00EA R        DW   DISK_STATUS      ; AH = 01: STATUS
355      0064 00F6 R        DW   DISK_READ        ; AH = 02: READ
356      0066 0102 R        DW   DISK_WRITE       ; AH = 03: WRITE
357      0068 010E R        DW   DISK_VERIFY      ; AH = 04: VERIFY
358      006A 011A R        DW   DISK_FORMAT      ; AH = 05: FORMAT
359      006C 017D R        FNC_ERR DW   FNC_ERR        ; AH = 06: INVALID
360      006E 017D R        DW   FNC_ERR        ; AH = 07: INVALID
361      0070 0187 R        DW   DISK_PARAMS     ; AH = 08: READ DRIVE PARAMETERS
362      0072 017D R        DW   FNC_ERR        ; AH = 09: INVALID
363      0074 017D R        DW   FNC_ERR        ; AH = 10: INVALID
364      0076 017D R        DW   FNC_ERR        ; AH = 0B: INVALID
365      0078 017D R        DW   FNC_ERR        ; AH = 0C: INVALID
366      007A 017D R        DW   FNC_ERR        ; AH = 0D: INVALID
367      007C 017D R        DW   FNC_ERR        ; AH = 0E: INVALID
368      007E 017D R        DW   FNC_ERR        ; AH = 0F: INVALID
369      0080 017D R        DW   FNC_ERR        ; AH = 10: INVALID
370      0082 017D R        DW   FNC_ERR        ; AH = 11: INVALID
371      0084 017D R        DW   FNC_ERR        ; AH = 12: INVALID
372      0086 017D R        DW   FNC_ERR        ; AH = 13: INVALID
373      0088 017D R        DW   FNC_ERR        ; AH = 14: INVALID
374      008A 0280 R        DW   DISK_TYPE      ; AH = 15: READ DASH TYPE
375      008C 0280 R        DW   DISK_CHANGE    ; AH = 16: CHANGE STATUS
376      008E 02E5 R        DW   FORMAT_SET    ; AH = 17: SET DASH TYPE
377      0090 034D R        DW   SET_MEDIA     ; AH = 18: SET MEDIA TYPE
378      0092             EQU     $
379      0092             DISKETTE_IO_1 ENDP
380
381      ;-----
382      ; DISK_RESET
383      ; RESET THE DISKETTE SYSTEM.
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                     ; NO INTERRUPTS
390      0096 A0 03F3 R    MOV    AL,*MOTOR_STATUS  ; GET DIGITAL OUTPUT REGISTER REFLECTION
391      0099 24 3F       AND    AL,00111111B       ; KEEP SELECTED AND MOTOR ON BITS
392      009B D0 C0       ROL    AL,1               ; MOTOR VALUE TO HIGH NIBBLE
393      009D D0 C0       ROL    AL,1               ; DRIVE SELECT TO LOW NIBBLE
394      00A1 D0 C0       ROL    AL,1
395      00A3 0C 08       OR     AL,00001000B       ; TURN ON INTERRUPT ENABLE
396      00A5 EE          OUT    DX,AL             ; RESET THE ADAPTER
397      00A6 C6 00 003E R 00  MOV    *NEC_STATUS,0     ; SET RECALIBRATE REQUIRED ON ALL DRIVES
398      00AB EB 00       JMP    $+2                 ; WAIT FOR I/O
399      00AD 0C 04       OR     AL,00000100B       ; TURN OFF RESET BIT
400      00AF EE          OUT    DX,AL             ; RESET THE ADAPTER
401      00B0 FB          STI                     ; ENABLE THE INTERRUPTS
402      00B1 E8 0ABA R    CALL   WAIT_INT           ; WAIT FOR THE INTERRUPT
403      00B4 72 2D       JC     DR_ERR              ; IF ERROR, RETURN IT
404      00B6 B9 00C0     MOV    CX,11000000B       ; CL = EXPECTED *NEC_STATUS
405
406      00B9          NXT_DRV:
407      00B9 51          PUSH   CX                  ; SAVE FOR CALL
408      00BA B8 00E2 R    MOV    AX,OFFSET DR_POP_ERR ; LOAD NEC_OUTPUT ERROR ADDRESS
409      00BD 50          PUSH   AX
410      00BE B4 08       MOV    AH,08H             ; SENSE INTERRUPT STATUS COMMAND
411      00C0 EB 09F0 R    CALL   NEC_OUTPUT
412      00C3 58          POP    AX
413      00C4 E8 0AE2 R    CALL   RESULTS           ; THROW AWAY ERROR RETURN
414      00C7 59          POP    CX                  ; READ IN THE RESULTS
415      00C8 72 19       JC     DR_ERR              ; RESTORE AFTER CALL
416      00CA 3A 0E 0042 R  CMP    CL,*NEC_STATUS     ; ERROR RETURN
417      00CE 75 13       JNZ    DR_ERR              ; TEST FOR DRIVE READY TRANSITION
418      00DD FE C1       INC    CL                  ; EVERYTHING OK
419      00DE 80 F9 C3     CMP    CL,11000011B       ; NEXT EXPECTED *NEC_STATUS
420      00D5 76 E2       JBE    NXT_DRV            ; ALL POSSIBLE DRIVES CLEARED
421                                     ; FALL THRU IF 1100100B OR >
422
423      ;----- SEND SPECIFY COMMAND TO NEC
424
425      00D7          JT:
426      00D7 E8 03D1 R    CALL   SEND_SPEC
427
428      00DA E8 0832 R    RESBAC: CALL   SETUP_END       ; VARIOUS CLEANUPS
429      00DD 8B DE       MOV    BX,SI              ; GET SAVED AL TO BL
430      00DF BA C3       MOV    AL,BL              ; PUT BACK FOR RETURN
431      00E1 C3          RET
432
433      00E2          DR_POP_ERR:
434      00E2 59          POP    CX                  ; CLEAR STACK
435      00E3          DR_ERR:
436      00E3 80 0E 0041 R 20 OR     *DISKETTE_STATUS,BAD_NEC ; SET ERROR CODE
437      00E8 EB F0       JMP    SHORT RESBAC       ; RETURN FROM RESET
438      00EA          DISK_RESET ENDP
439
440      ;-----
441      ; DISK_STATUS
442      ; DISKETTE STATUS.
443      ; ON ENTRY: AH = STATUS OF PREVIOUS OPERATION

```

```

443 ; ON EXIT: #DISKETTE_STATUS, CY REFLECT STATUS OF OPERATION ;
444 ;-----;
445 DISK_STATUS PROC NEAR
446 00EA ;
447 00EA 88 26 0041 R ; PUT BACK FOR SETUP_END
448 00EE E8 0832 R ; VARIOUS CLEANUPS
449 00F1 88 DE ; GET SAVED AL TO BL
450 00F3 8A C4 ; STORE STATUS IN AL
451 00F5 C3
452 00F6 ENDP
453 ;-----;
454 ; DISK_READ ;
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: #DISKETTE_STATUS, CY REFLECT STATUS OF OPERATION ;
465 ;-----;
466 00F6 DISK_READ PROC NEAR
467 00F6 80 26 003F R 7F ; #MOTOR STATUS,01111111B ; INDICATE A READ OPERATION
468 00FB B8 E446 ; AX = NEC COMMAND, DMA COMMAND
469 00FE E8 04B3 R ; CALL RD_WR_VF ; COMMON READ/WRITE/VERIFY
470 0101 C3
471 0102 ENDP
472 ;-----;
473 ; DISK_WRITE ;
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: #DISKETTE_STATUS, CY REFLECT STATUS OF OPERATION ;
484 ;-----;
485 0102 DISK_WRITE PROC NEAR
486 0102 B8 C54A ; MOV AX,0C54AH ; AX = NEC COMMAND, DMA COMMAND
487 0105 80 0E 003F R 80 ; #MOTOR STATUS,10000000B ; INDICATE WRITE OPERATION
488 010A E8 04B3 R ; CALL RD_WR_VF ; COMMON READ/WRITE/VERIFY
489 010D C3
490 010E ENDP
491 ;-----;
492 ; DISK_VERIFY ;
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: #DISKETTE_STATUS, CY REFLECT STATUS OF OPERATION ;
503 ;-----;
504 010E DISK_VERIFY PROC NEAR
505 010E 80 26 003F R 7F ; #MOTOR STATUS,01111111B ; INDICATE A READ OPERATION
506 0113 B8 E442 ; MOV AX,0E642H ; AX = NEC COMMAND, DMA COMMAND
507 0116 E8 04B3 R ; CALL RD_WR_VF ; COMMON READ/WRITE/VERIFY
508 0119 C3
509 011A ENDP
510 ;-----;
511 ; DISK_FORMAT ;
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: #DISKETTE_STATUS, CY REFLECT STATUS OF OPERATION ;
524 ;-----;
525 011A DISK_FORMAT PROC NEAR
526 011A E8 0404 R ; CALL XLAT_NEW ; TRANSLATE STATE TO PRESENT ARCH.
527 011D E8 05A0 R ; CALL FMT_INIT ; ESTABLISH STATE IF UNESTABLISHED
528 0120 80 0E 003F R 80 ; #MOTOR STATUS,10000000B ; INDICATE WRITE OPERATION
529 0125 F6 06 008F R 01 ; TEST #WF_CNTRL_DUAL ; TEST CONTROLLER I.D.
530 012A 74 05 ; JZ NO_CHG_CHECK ; NO CHG CHECK
531 012C E8 05F5 R ; CALL MED_CHANGE ; CHECK MEDIA CHANGE AND RESET IF SO
532 012F 72 41 ; JC FM_DON ; MEDIA CHANGED, SKIP
533 0131 ;
534 0131 NO_CHG_CHECK:
535 0131 E8 0658 R ; CALL CHK_LAstrate ; ZF=1 ATTEMPT RATE IS SAME AS LAST RATE
536 0134 74 06 ; JZ FM_WR ; YES, SKIP SPECIFY COMMAND
537 0136 E8 03D1 R ; CALL SEND_SPEC ; SEND SPECIFY COMMAND TO NEC
538 0139 E8 0637 R ; CALL SEND_RATE ; SEND DATA RATE TO CONTROLLER
539 013C ;
540 013C FM_WR:
541 013C 80 4A ; MOV AL,04AH ; WILL WRITE TO THE DISKETTE
542 013E E8 0668 R ; CALL DMA_SETUP ; SET UP THE DMA
543 0141 72 2F ; JC FM_DON ; RETURN WITH ERROR
544 0143 B4 4D ; MOV AH,04DH ; ESTABLISH THE FORMAT COMMAND
545 0145 E8 06CB R ; CALL NEC_INIT ; INITIALIZE THE NEC
546 0148 72 28 ; JC FM_DON
547 014A B8 0172 R ; MOV AX,OFFSEF FM_DON ; LOAD ERROR ADDRESS
548 014D 50 ; PUSH AX ; PUSH NEC_OUT ERROR RETURN
549 014E B2 04 ; MOV DL,3 ; BYTES/SECTOR VALUE TO NEC
550 0150 E8 08FE R ; CALL GET_PARM ; GET_PARM
551 0153 E8 09F0 R ; CALL NEC_OUTPUT ; SECTORS/TRACK VALUE TO NEC
552 0156 B2 04 ; MOV DL,4 ; GET_PARM
553 0158 E8 08FE R ; CALL GET_PARM ; GET_PARM
554 015B E8 09F0 R ; CALL NEC_OUTPUT ; GAP LENGTH VALUE TO NEC
555 015E B2 07 ; MOV DL,7 ; GET_PARM
556 0160 E8 08FE R ; CALL GET_PARM ; GET_PARM
557 0163 E8 09F0 R ; CALL NEC_OUTPUT ; FILLER BYTE TO NEC
558 0166 B2 08 ; MOV DL,8 ; FILLER BYTE TO NEC

```

SECTION 5

```

557 0168 E8 08FE R          CALL    GET_PARM
558 016B E8 09F0 R          CALL    NEC_OUTPUT
559 016E 58                 POP     AX
560 016F E8 0727 R          CALL    NEC_TERM          ; THROW AWAY ERROR
561 0172                 FM_DONE:                 ; TERMINATE, RECEIVE STATUS, ETC.
562 0172 E8 0432 R          CALL    XLAT_OLD          ; TRANSLATE STATE TO COMPATIBLE MODE
563 0175 E8 0832 R          CALL    SETUP_END        ; VARIOUS CLEANUPS
564 0178 BB DE             MOV     BX,S1            ; GET SAVED AL TO BL
565 017A BA C3             MOV     AL,BL           ; PUT BACK FOR RETURN
566 017C D3                 RET
567 017D                 DISK_FORMAT    ENDP
568
569
570
571
572
573
574
575 017D                 FNC_ERR PROC    NEAR          ; INVALID FUNCTION REQUEST
576 017D BB C6             MOV     AX,S1            ; RESTORE AL
577 017F B4 01             MOV     AH,BAD_CMD       ; SET BAD COMMAND ERROR
578 0181 88 26 0041 R      STC     #DSKETTE_STATUS,AH ; STORE IN DATA AREA
579 0185 F9                 MOV     STC              ; SET CARRY INDICATING ERROR
580 0186 C3                 RET
581 0187                 FNC_ERR ENDP
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605 0187                 DISK_PARMs    PROC    NEAR
606 0187 81 FF 0080        CMP     DI,BX            ; CHECK FOR FIXED MEDIA TYPE REQUEST
607 018B 72 06             JB      DISK_P2         ; CONTINUE IF NOT REQUEST FALL THROUGH
608
609
610
611 018D BB C6             MOV     AX,S1            ; RESTORE AL WITH CALLERS VALUE
612 018F B4 01             MOV     AH,BAD_CMD       ; SET BAD COMMAND ERROR IN (AH)
613 0191 F9                 STC
614 0192 C3                 RET
615
616 0193                 DISK_P2:
617 0193 E8 0404 R          CALL    XLAT_NEW        ; TRANSLATE STATE TO PRESENT ARCH.
618 0196 C7 46 02 0000    WORD_PTR [BP+2],0      ; DRIVE TYPE = 0
619 019B A1 0010 R          MOV     AX,EQUIP_FLAG   ; LOAD EQUIPMENT FLAG FOR # DISKETTES
620 019E 24 C1             AND    AL,11000001B    ; KEEP DISKETTE DRIVE BITS
621 01A0 D0 E8             SHR    SHR              ; ARE THERE ANY DRIVES INSTALLED?
622 01A2 73 7C             JNC    NON_DRV         ; NC-->NO DRIVES, ZERO PARAMETERS
623 01A4 D0 C0             ROL    AL,T            ; ROTATE TO ORIGINAL POSITION
624 01A6 D0 C0             ROL    AL,I            ; ROTATE BITS 6 AND 7 TO 0 AND 1
625 01A8 D0 C0             ROL    AL,I
626 01AA FE C0             INC    AL               ; CONVERT TO RELATIVE I
627 01AC 88 46 04         MOV     [BP+4],AL       ; STORE NUMBER OF DRIVES
628 01AF F6 06 008F R 01  TEST    #HF_CNTRL,DUAL ; CHECK CONTROLLER I.D.
629 01B4 75 03             JNZ    DP1_CONT        ; CONTINUE WITH USUAL PARMs CHECK
630 01B6 E9 0256 R          JMP     DET_PARMs      ; RETURN THIS CONTROLLERS PARMs
631 01B9                 DP1_CONT:
632 01B9 83 FF 01          CMP     DI,BX           ; CHECK FOR VALID DRIVE
633 01BC 77 66             JA      NON_DRV1       ; DRIVE INVALID
634 01BE C6 46 05 01      MOV     BYTE_PTR[BP+5],1 ; MAXIMUM HEAD NUMBER = 1
635 01C0 E8 08FC R          CALL    CMOS_TEST      ; RETURN DRIVE TYPE IN AL
636 01C5 72 16             JC      JC             ; ON CMOS BAD CHECK ESTABLISHED
637 01C7 0A C0           OR     AL,AL           ; TEST FOR NO DRIVE TYPE
638 01C9 74 12           JZ     CHK_EST         ; JUMP IF SO
639 01CB E8 03B1 R          CALL    DR_TYPE_CHECK  ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
640 01CE 72 0D           JC     CHK_EST         ; TYPE NOT IN TABLE (POSSIBLE BAD CMOS)
641 01D0 88 46 02         MOV     [BP+2],AL       ; STORE VALID CMOS DRIVE TYPE
642 01D3 2E: 8A 4F 04     MOV     CL,CS:[BX].MD_SEC_TRK ; GET SECTOR/TRACK
643 01D7 2E: 8A 6F 0B     MOV     CH,CS:[BX].MD_MAX_TRK ; GET MAX. TRACK NUMBER
644 01DB EB 32             JMP     SHORT STO_CX    ; CMOS GOOD, USE CMOS
645
646 01DD                 CHK_EST:
647 01DD 8A A5 0090 R      MOV     AH,#DSK STATE[D1] ; LOAD STATE FOR THIS DRIVE
648 01E1 F6 C4 10          TEST    AH,MD_DET      ; CHECK FOR ESTABLISHED STATE
649 01E4 74 3E             JZ     NON_DRV1       ; CMOS BAD/INVALID AND UNESTABLISHED
650
651
652
653
654
655
656
657
658 01EE B0 01             MOV     AL,01          ; ISOLATE STATE
659 01F0 E8 03B1 R          CALL    DR_TYPE_CHECK  ; RATE 250 ?
660 01F3 2E: 8A 4F 04     MOV     CL,CS:[BX].MD_SEC_TRK ; NO, GO CHECK OTHER RATE
661 01F7 2E: 8A 6F 0B     MOV     CH,CS:[BX].MD_MAX_TRK ;
662 01FB F6 15 0090 R 01  TEST    #DSK STATE[D1],TRK_CAPA ;
663 0200 74 0D             JZ     STO_CX         ; MUST BE 360KB DRIVE
664
665
666
667 0202                 PARM_HDR_80T:
668 0202 B0 04             MOV     AL,04          ; DRIVE TYPE 4
669 0204 E8 03B1 R          CALL    DR_TYPE_CHECK  ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
670 0207 2E: 8A 4F 04     MOV     CL,CS:[BX].MD_SEC_TRK ; GET SECTOR/TRACK

```



```

671 020B 2E: 8A 6F 0B          MOV     CH,CS:[BX].MD_MAX_TRK ; GET MAX. TRACK NUMBER
672
673 020F
674 020F 89 4E 00          STO_CX: MOV     [BP],CX ; SAVE IN STACK FOR RETURN
675 0212
676 0212 89 5E 06          ES_DI: MOV     [BP+6],BX ; ADDRESS OF MEDIA/DRIVE PARAM TABLE
677 0215 8C 8C          MOV     AX,CS ; SEGMENT MEDIA/DRIVE PARAMETER TABLE
678 0217 8E C0          MOV     ES,AX ; ES IS SEGMENT OF TABLE
679
680 0219
681 0219 EB 0432 R          DP_OUT: CALL    XLAT_OLD ; TRANSLATE STATE TO COMPATIBLE MODE
682 021C 33 C0          XOR     AX,AX ; CLEAR
683 021E F8          CLC
684 021F C3          RET
685
686
687
688
689 0220
690 0220 C6 46 04 00          NON_DRV: MOV     BYTE PTR [BP+4],0 ; CLEAR NUMBER OF DRIVES
691
692 0224 81 FF 0080          NON_DRV1: CMP     DI,80H ; CHECK FOR FIXED MEDIA TYPE REQUEST
693 0228 72 0F          JNB    NON_DRV2 ; CONTINUE IF NOT REQUEST FALL THROUGH
694
695
696
697 022A
698 022A EB 0432 R          FD_REQ_ERR: CALL   XLAT_OLD ; ELSE TRANSLATE TO COMPATIBLE MODE
699 022D 8B C6          MOV     AX,ST ; RESTORE AL
700 022F B4 01          MOV     AH,BAD_CMD ; SET BAD COMMAND ERROR
701 0231 F9          STC ; SET ERROR RETURN CODE
702 0232 C3          RET
703
704 0233
705 0233 33 C0          NON_DRV2: XOR     AX,AX ; CLEAR PARMS IF NO DRIVES OR CMOS BAD
706 0235 89 46 00          MOV     [BP],AX ; TRACKS, SECTORS/TRACK = 0
707 0238 88 66 05          MOV     [BP+5],AH ; HEAD = 0
708 023B 89 46 06          MOV     [BP+6],AX ; OFFSET TO DISK BASE = 0
709 023E 8E C0          MOV     ES,AX ; ES IS SEGMENT OF TABLE
710 0240 EB D7          JMP     SHORT DP_OUT
711
712
713
714 0242
715 0242 B0 02          USE_EST2: MOV     AL,02 ; DRIVE TYPE 2 (1.2MB)
716 0244 EB 03B1 R          CALL   DR_TYPE_CHECK ; RTN CS:[BX] = MEDIA/DRIVE PARAM TBL
717 0247 2E: 8A 4F 04          MOV     CL,CS:[BX].MD_SEC_TRK ; GET SECTOR/TRACK
718 0249 2E: 8A 6F 0B          MOV     CH,CS:[BX].MD_MAX_TRK ; GET MAX. TRACK NUMBER
719 024F 80 FC 40          CMP     AH,RATE_300 ; RATE 300 ?
720 0252 74 BF 0B          JBE    STO_CX ; MUST BE 1.2MB DRIVE
721 0254 EB AC          JMP     SHORT PARM_HDR_80T ; ELSE, HIGH DATA RATE/80 TRACK DRIVE
722 0256
723 0256 83 FF 03          DET_PARMS: CMP     DI,3 ; REQUEST FOR FIXED DISK?
724 0259 77 D8          JNA    NON_DRV2 ; YES-->DRIVE NUMBER INVALID
725 025B B1 09          MOV     CL,9
726 025D F6 85 0090 R 01    TEST   #DSK_STATE[DI],TRK_CAPA ; IS DRIVE 80 TRACKS?(RELATIVE ZERO)
727 0262 B0 01          MOV     AL,0 ; SET CMOS TYPE 1
728 0264 B5 27          MOV     CH,39 ; NUMBER OF TRACKS (RELATIVE ZERO)
729 0266 74 04          JZ     SET_TYP1 ; IF ZERO TYPE = 1
730 0268 B0 03          MOV     AL,3 ; SET CMOS TYPE 3
731 026A B5 4F          MOV     CH,19 ; NUMBER OF TRACKS (RELATIVE ZERO)
732 026C
733 026C 88 46 02          SET_TYP1: MOV     [BP+2],AL ; STORE TYPE
734 026F C6 46 03 00          MOV     BYTE PTR [BP+3],0 ; MAXIMUM HEAD NUMBER = 1
735 0273 C6 46 05 01          MOV     BYTE PTR [BP+5],1 ; ADDRESS OF DISK BASE
736 0277 EB 03B1 R          CALL   DR_TYPE_CHECK ; GO SET TRKS/SEC,CYL,ES:BX AND EXIT
737 027A EB 93          JMP
738
739 027C
740
741
742
743
744
745
746
747
748 027C
749 027C F6 06 008F R 01    DISK_TYPE: TEST   #HF_CNTRL,DUAL ; CHECK CONTROLLER I.D.
750 0281 74 22          JZ     NO_CHNG ; NO CHNG
751 0283 EB 0404 R          CALL   XLAT_NEW ; TRANSLATE STATE TO PRESENT ARCH.
752 0286 8A 85 0090 R      MOV     AL,#DSK_STATE[DI] ; GET PRESENT STATE INFORMATION
753 028A 0A C0          OR     AL,AL ; CHECK FOR NO DRIVE
754 028C 74 13          JZ     NO_DRV ; NO DRV
755 028E B4 01          MOV     AH,NOCHGLN ; NO CHANGE LINE FOR 40 TRACK DRIVE
756 0290 A8 01          TEST   AL,TRK_CAPA ; IS THIS DRIVE AN 80 TRACK DRIVE?
757 0292 74 02          JZ     DT_BACK ; IF NO JUMP
758 0294 B4 02          MOV     AH,CHGLN ; CHANGE LINE FOR 80 TRACK DRIVE
759
760 0296
761 0296 50          DT_BACK: PUSH   AX ; SAVE RETURN VALUE
762 0297 EB 0432 R          CALL   XLAT_OLD ; TRANSLATE STATE TO COMPATIBLE MODE
763 029A 58          POP    AX ; RESTORE RETURN VALUE
764 029B
765 029B F8          DISK_TYPE_EX: CLC ; EXIT DISK TYPE FUNCTION
766 029C 8B DE          MOV     BX,SI ; GET SAVED AL TO BL
767 029E 8A C3          MOV     AL,BL ; PUT BACK FOR RETURN
768 02A0 C3          RET
769 02A1
770 02A1 32 E4          NO_DRV: XOR     AH,AH ; NO DRIVE PRESENT OR UNKNOWN
771 02A3 EB F1          JMP
772 02A5
773 02A5 A1 0010 R          NO_CHNG: MOV     AX,#EQUIP_FLAG ; LOAD EQUIPMENT FLAG FOR # DISKETTES
774 02A8 D0 E8          SHR     AL,1 ; SHIFT DRIVES PRESENT BIT INTO CARRY
775 02AA 73 F5          JNC    NO_DRV ; NO DRIVE IN SYSTEM
776 02AC B4 01          MOV     AH,NO ; DISKETTE NO CHANGE LINE AVAILABLE
777 02AE EB EB          JMP     DISK_TYPE_EX
778 02B0
779
780
781
782
783
784

```

```

785          : ON EXIT:      AH = #DSKETTE STATUS          :
786          :             00 - DISK CHANGE LINE INACTIVE, CY = 0  :
787          :             06 - DISK CHANGE LINE ACTIVE, CY = 1    :
788          :-----:
789          02B0          PROC NEAR
790          02B0 F6 06 00BF R 01  DISK_CHANGE TEST #HF_CNTRL,DUAL ; TEST CONTROLLER I.D.
791          02B5 75 03          JNZ          DC1          ;
792          02B7 E9 017D R      JMP          FNC_ERR      ; ERROR FOR THIS KIND OF CONTROLLER
793          02BA          DC1:
794          02BA E8 0404 R      CALL XLAT_NEW      ; TRANSLATE STATE TO PRESENT ARCH.
795          02BD 8A 85 0090 R   MOV AL,#DSK_STATE[D1] ; GET MEDIA STATE INFORMATION
796          02C1 0A C0          OR AL,AL          ; DRIVE PRESENT ?
797          02C3 74 19          JZ DC_NON        ; JUMP IF NO DRIVE
798          02C5 A8 01          TEST AL,TRK_CAPA   ; 80 TRACK DRIVE ?
799          02C7 74 05          JZ IF_30         ; IF 30, CHECK CHANGE LINE
800          02C9 E8 0B21 R      DC0: CALL READ_DSKCHNG ; GO CHECK STATE OF DISK CHANGE LINE
801          02CC 74 05          JZ FINIS         ; CHECK LINE NOT ACTIVE
802          02CE C6 06 0041 R 06 SETIT: MOV #DSKETTE_STATUS,MEDIA_CHANGE ; INDICATE MEDIA REMOVED
803          02D0          FINIS: CALL XLAT_OLD      ; TRANSLATE STATE TO COMPATIBLE MODE
804          02D3 E8 0432 R      CALL SETUP_END    ; VARIOUS CLEANUPS
805          02D6 E8 0B32 R      MOV BX,S1        ; GET SAVED AL TO BL
806          02D9 8B DC          MOV AL,BL        ; PUT BACK FOR RETURN
807          02DB C3            RET
808          02DD C3            RET
809          02DE          DC_NON: OR #DSKETTE_STATUS,TIME_OUT ; SET TIMEOUT, NO DRIVE
810          02E0          JMP SHORT FINIS
811          02E3 EB EE          DISK_CHANGE ENDP
812          02E5
813          02E5
814          02E5
815          02E5
816          02E5
817          02E5
818          02E5
819          :-----:
820          : FORMAT SET :
821          : THIS ROUTINE IS USED TO ESTABLISH THE TYPE OF :
822          : MEDIA TO BE USED FOR THE FOLLOWING FORMAT OPERATION. :
823          :-----:
824          : ON ENTRY:   S1 LOW = DASD TYPE FOR FORMAT :
825          :             D1 = DRIVE # :
826          :-----:
827          : ON EXIT:   #DSKETTE_STATUS REFLECTS STATUS :
828          :             AH = #DSKETTE_STATUS :
829          :             CY = 1 IF ERROR :
830          :-----:
831          02E5          PROC NEAR
832          02E5 E8 0404 R      CALL XLAT_NEW      ; TRANSLATE STATE TO PRESENT ARCH.
833          02E8 56            PUSH          S1      ; SAVE DASD TYPE
834          02E9 0B C6          MOV AX,S1        ; AH = ? , AL = DASD TYPE
835          02EB 32 E4          XOR AH,AH       ; AH = 0 , AL = DASD TYPE
836          02ED 8B F0          MOV SI,AX       ; SI = DASD TYPE
837          02EF 80 A5 0090 R 0F AND #DSK_STATE[D1],NOT MED_DET+DBL_STEP+RATE_MSK ; CLEAR STATE
838          02F2 4E            DEC SI          ; CHECK FOR 320/360K MEDIA & DRIVE
839          02F5 75 07          JNZ NOT_320    ; BYPASS IF NOT
840          02F7 80 8D 0090 R 90 OR #DSK_STATE[D1],MED_DET+RATE_250 ; SET TO 320/360
841          02FC EB 3E          JMP SHORT S0
842          02FE          NOT_320:
843          02FE F6 06 00BF R 01 TEST #HF_CNTRL,DUAL ; TEST CONTROLLER I.D.
844          0303 74 0A          JZ S3          ;
845          0305 E8 05F5 R      CALL MED_CHANGE   ; CHECK FOR TIME_OUT
846          0308 80 3E 0041 R 80 CMP #DSKETTE_STATUS,TIME_OUT ;
847          030D 74 2D          JZ S0          ; IF TIME OUT TELL CALLER
848          030F 4E            DEC SI          ;
849          0310 75 07          JNZ NOT_320_12  ; CHECK FOR 320/360K IN 1.2M DRIVE
850          0312 80 8D 0090 R 70 OR #DSK_STATE[D1],MED_DET+DBL_STEP+RATE_300 ; BYPASS IF NOT
851          0317 EB 23          JMP SHORT S0    ; SET STATE
852          0319          NOT_320_12:
853          0319 4E            DEC SI          ; CHECK FOR 1.2M MEDIA IN 1.2M DRIVE
854          031A 75 07          JNZ NOT_12     ; BYPASS IF NOT
855          031C 80 8D 0090 R 10 OR #DSK_STATE[D1],MED_DET+RATE_500 ; SET STATE VARIABLE
856          0321 EB 19          JMP SHORT S0    ; RETURN TO CALLER
857          0323          NOT_12:
858          0323 4E            DEC SI          ;
859          0324 75 20          JNZ FS_ERR     ; CHECK FOR SET DASD TYPE 04
860          0326 F6 85 0090 R 04 TEST #DSK_STATE[D1],DRV_DET ; DRIVE DETERMINED ?
861          032B 74 09          JZ ASSUME      ; IF STILL NOT DETERMINED ASSUME
862          032D B0 50          MOV AL,MED_DET+RATE_300 ;
863          032F F6 85 0090 R 02 TEST #DSK_STATE[D1],FMT_CAPA ; MULTIPLE FORMAT CAPABILITY ?
864          0334 75 02          JNZ OR_IT_IN   ; IF 1.2 M THEN DATA RATE 300
865          0336          ASSUME: MOV AL,MED_DET+RATE_250 ; SET UP
866          0336 B0 90          MOV AL,MED_DET+RATE_250 ;
867          0338          OR_IT_IN: OR #DSK_STATE[D1],AL ; OR IN THE CORRECT STATE
868          0338
869          0338
870          0338
871          0338
872          0338
873          0338
874          0338
875          033C          S0: CALL XLAT_OLD      ; TRANSLATE STATE TO COMPATIBLE MODE
876          033C E8 0432 R      CALL SETUP_END    ; VARIOUS CLEANUPS
877          033F E8 0B32 R      MOV BX,S1        ; GET SAVED AL TO BL
878          0342 5B            POP BX          ; PUT BACK FOR RETURN
879          0343 8A C3          MOV AL,BL
880          0345 C3            RET
881          0347          FS_ERR: MOV #DSKETTE_STATUS,BAD_CMD ; UNKNOWN STATE,BAD COMMAND
882          0346 C6 06 0041 R 01 MOV SHORT S0
883          034B EB EF          JMP SHORT S0
884          034D          FORMAT_SET ENDP
885          034D
886          034D
887          034D
888          034D
889          034D
890          :-----:
891          : SET_MEDIA :
892          : THIS ROUTINE SETS THE TYPE OF MEDIA AND DATA RATE :
893          : TO BE USED FOR THE FOLLOWING FDMAT OPERATION. :
894          : ON ENTRY:   [BP] = SECTOR PER TRACK :
895          :             [BP+1] = TRACK # :
896          :             D1 = DRIVE # :
897          :-----:
898          : ON EXIT:   #DSKETTE_STATUS REFLECTS STATUS :
899          :             IF NO ERROR: :

```

```

899          : AH = 0          :
900          : CY = 0          :
901          : ES = SEGMENT OF MEDIA/DRIVE PARAMETER TABLE :
902          : DI/[BP+6] = OFFSET OF MEDIA/DRIVE PARAMETER TABLE :
903          : IF ERROR: :
904          : AH = #DISKETTE_STATUS :
905          : CY = 1          :
-----
907 034D          SET_MEDIA PROC NEAR
908 034D E8 0404 R CALL XLAT_NEW          ; TRANSLATE STATE TO PRESENT ARCH.
909 0350 33 DB      XOR BX,BX          ; ZERO INDEX POINTER
910 0352 80 7E 01 27 CMP BYTE PTR [BP+1],39 ; MAX. TRACK = 40 ?
911 0356 75 0B      JNE TBL_CHK1
912 0358 F6 85 0090 R 01 TEST #DSK_STATE[DI],TRK_CAPA ; 80 TRACK DRIVE ?
913 035D 74 16      JZ MD_FND          ; POINT TO TABLE ENTRY 1
914 035F B3 03      MOV BL,3          ; POINT TO TABLE ENTRY 2
915 0361 EB 12      JMP SHORT MD_FND
916 0363
917 0363 B3 06      TBL_CHK1: MOV BL,6          ; POINT TO TABLE ENTRY 3
918 0365 80 7E 00 F0 CMP BYTE PTR [BP],15 ; SECTORS/TRACK = 15 ?
919 0369 74 0A      JE MD_FND
920 036B B3 12      MOV BL,18         ; POINT TO TABLE ENTRY 6
921 036D 80 7E 00 12 CMP BYTE PTR [BP],18 ; SECTORS/TRACK = 18 ?
922 0371 74 02      JE MD_FND
923 0373 B3 0C      MOV BL,12         ; POINT TO TABLE ENTRY 4
924 0375
925 0375 2E: 8B 9F 0001 R MD_FND: MOV BX,CS:WORD PTR DR_TYPE[BX+1] ; DI = MEDIA/DRIVE PARAMETER TAB
          LE
926 037A 2E: 8A 47 04      MOV AL,CS:[BX].MD_SEC_TRK ; GET SECTOR/TRACK
927 037E 2E: 8A 67 0B      MOV AH,CS:[BX].MD_MAX_TRK ; GET MAX. TRACK #
928 0382 39 46 00      CMP [BP],AX        ; MATCH ?
929 0385 75 23      JNE ER_RTN        ; NOT SUPPORTED
930 0387 2E: 8A 47 0C      MOV AL,CS:[BX].MD_RATE ; GET RATE
931 038B 3C 40      CMP AL,RATE_300   ; DOUBLE STEP REQUIRED FOR RATE 300
932 038D 75 02      JNE OR
933 038F 0C 20      OR AL,DBL_STEP
934 0391
935 0391 89 5E 06      MD_SET: MOV [BP+6],BX      ; SAVE TABLE POINTER IN STACK
936 0394 0C 10      OR AL,MED_DET     ; SET MEDIA ESTABLISHED
937 0396 80 A5 0090 R 0F AND #DSK_STATE[DI],NOT MED_DET+DBL_STEP+RATE_MSK ; CLEAR STATE
938 039B 08 85 0090 R OR #DSK_STATE[DI],AL ; SET STATE
939 039F 8C C8      MOV AH,CS         ; SEGMENT MEDIA/DRIVE PARAMETER TABLE
940 03A1 8E C0      MOV ES,AX         ; ES IS SEGMENT OF TABLE
941 03A3
942 03A3 E8 0432 R      SM_RTN: CALL XLAT_OLD     ; TRANSLATE STATE TO COMPATIBLE MODE
943 03A6 E8 0832 R      CALL SETUP_END   ; VARIOUS CLEANUPS
944 03A9 C3
945 03AA
946 03AA C6 06 0041 R 0C ER_RTN: MOV #DISKETTE_STATUS,MED_NOT_FND ; ERROR, MEDIA TYPE NOT FOUND
947 03AF EB F2      JMP SM_RTN
948 03B1          SET_MEDIA ENDP
-----
949
950          : DR_TYPE_CHECK :
951          : CHECK IF THE GIVEN DRIVE TYPE IN REGISTER (AL) :
952          : IS SUPPORTED IN BIOS DRIVE TYPE TABLE :
953          : ON ENTRY: :
954          : AL = DRIVE TYPE :
955          : ON EXIT: :
956          : CS = SEGMENT OF MEDIA/DRIVE PARAMETER TABLE (CODE) :
957          : CY = DRIVE TYPE SUPPORTED :
958          : BX = OFFSET TO MEDIA/DRIVE PARAMETER TABLE :
959          : CY = 1 DRIVE TYPE NOT SUPPORTED :
960          : REGISTERS ALTERED: BX :
-----
963 03B1          DR_TYPE_CHECK PROC NEAR
964 03B1 50          PUSH AX
965 03B2 51          PUSH CX
966 03B3 33 DB      XOR BX,BX          ; BX = INDEX TO DR TYPE TABLE
967 03B5 B9 0006      MOV CX,DR_CNT     ; CX = LOOP COUNT
968 03B6
969 03B8 2E: 8A AT 0000 R TYPE_CHK: MOV AH,CS:DR_TYPE[BX] ; GET DRIVE TYPE
970 03BD 3A C4      CMP AL,AH         ; DRIVE TYPE MATCH ?
971 03BF 74 08      JE DR_TYPE_VALID ; YES, RETURN WITH CARRY RESET
972 03C1 83 C3 03      ADD BX,3          ; CHECK NEXT DRIVE TYPE
973 03C4 E2 F2      LOOP TYPE_CHK     ; DRIVE TYPE NOT FOUND IN TABLE
974 03C6 F9
975 03C7 EB 05      JMP SHORT TYPE_RTN
976 03C9
977 03C9 2E: 8B 9F 0001 R DR_TYPE_VALID: MOV BX,CS:WORD PTR DR_TYPE[BX+1] ; BX = MEDIA TABLE
978 03CE
979 03CE 59          POP CX
980 03CF 58          POP AX
981 03D0 C3
982 03D1          DR_TYPE_CHECK ENDP
-----
983
984          : SEND_SPEC :
985          : SEND THE SPECIFY COMMAND TO CONTROLLER USING DATA FROM :
986          : THE DRIVE PARAMETER TABLE POINTED BY #DISK_POINTER :
987          : ON ENTRY: #DISK_POINTER = DRIVE PARAMETER TABLE :
988          : ON EXIT: NONE :
989          : REGISTERS ALTERED: AX :
-----
991
992 03D1          SEND_SPEC PROC NEAR
993 03D1 B8 03EB R      MOV AX,OFFSET SPECBAC ; LOAD ERROR ADDRESS
994 03D4 50          PUSH AX          ; PUSH NEC_OUT ERROR RETURN
995 03D5 B4 03      MOV AH,03H      ; SPECIFY COMMAND
996 03D7 E8 09F0 R CALL NEC_OUTPUT   ; OUTPUT THE COMMAND
997 03DA 2A D2      SUB DL,DL       ; FIRST SPECIFY BYTE
998 03DC E8 08FE R CALL GET_PARM     ; GET PARAMETER TO AH
999 03DF E8 09F0 R CALL NEC_OUTPUT   ; OUTPUT THE COMMAND
1000 03E2 B2 01      MOV DL,1        ; SECOND SPECIFY BYTE
1001 03E4 E8 08FE R CALL GET_PARM     ; GET PARAMETER TO AH
1002 03E7 E8 09F0 R CALL NEC_OUTPUT   ; OUTPUT THE COMMAND
1003 03EA 58          POP AX          ; POP ERROR RETURN
1004 03EB
1005 03EB C3          SPECBAC: RET
1006 03EC          SEND_SPEC ENDP
-----
1007
1008          : SEND_SPEC_MD :
1009          : SEND THE SPECIFY COMMAND TO CONTROLLER USING DATA FROM :
1010          : THE MEDIA/DRIVE PARAMETER TABLE POINTED BY (CS:BX) :
1011

```

SECTION 5

```

1012 ; ON ENTRY: CS:BX = MEDIA/DRIVE PARAMETER TABLE ;
1013 ; ON EXIT: NONE ;
1014 ; REGISTERS ALTERED: AX,CX,DX ;
1015 -----
1016 03EC SEND_SPEC MD PROC NEAR
1017 03EC B8 0403 R MOV AX,OFFSET SPEC_ESBAC ; LOAD ERROR ADDRESS
1018 03EF 50 PUSH AX ; PUSH NEC_OUT ERROR RETURN
1019 03F0 B4 03 MOV AH,03H ; SPECIFY COMMAND
1020 03F2 E8 09F0 R CALL NEC_OUTPUT ; OUTPUT THE COMMAND
1021 03F5 2E: 8A 27 MOV AH,ES:[BX].MD_SPEC1 ; GET 1ST SPECIFY BYTE
1022 03F8 E8 09F0 R CALL NEC_OUTPUT ; OUTPUT THE COMMAND
1023 03FB 2E: 8A 67 01 MOV AH,CS:[BX].MD_SPEC2 ; GET SECOND SPECIFY BYTE
1024 03FF E8 09F0 R CALL NEC_OUTPUT ; OUTPUT THE COMMAND
1025 0402 58 POP AX ; POP ERROR RETURN
1026 0403 SPEC_ESBAC:
1027 0403 C3 RET
1028 0404 SEND_SPEC MD ENDP
1029 -----
1030 ; XLAT_NEW TRANSLATES DISKETTE STATE LOCATIONS FROM COMPATIBLE ;
1031 ; MODE TO NEW ARCHITECTURE. ;
1032 ; ;
1033 ; ;
1034 ; ON ENTRY: DI : DRIVE ;
1035 -----
1036 0404 XLAT_NEW PROC NEAR
1037 0404 F6 06 00BF R 01 TEST #HF_CNTRL,DUAL ; TEST CONTROLLER 1.D.
1038 0409 74 22 JZ XN_OUT ;
1039 040B 83 FF 01 CMP DI,1 ; VALID DRIVE ?
1040 040E 77 1D JA XN_OUT ; IF INVALID BACK
1041 0410 80 BD 0090 R 00 CMP #DSK_STATE[DI],0 ; NO DRIVE ?
1042 0415 74 17 JZ DO_DET ; IF NO DRIVE ATTEMPT DETERMINE
1043 0417 8B CF MOV CX,DI ; CX = DRIVE NUMBER
1044 0419 D0 E1 SHL CL,1 ; CL = SHIFT COUNT, A=0, B=4
1045 041B D0 E1 SHL CL,1 ;
1046 041D A0 00BF R MOV AL,#HF_CNTRL ; DRIVE INFORMATION
1047 0420 D2 C8 ROR AL,CL ; TO LOW NIBBLE
1048 0422 24 07 AND AL,DRV_DET+FM_T_CAPA+TRK_CAPA ; KEEP DRIVE BITS
1049 0424 80 A5 0090 R F8 AND #DSK_STATE[DI],NOT DRV_DET+FM_T_CAPA+TRK_CAPA ;
1050 0429 08 85 0090 R OR #DSK_STATE[DI],AL ; UPDATE DRIVE STATE
1051 042D XN_OUT:
1052 042D C3 RET ;
1053 ;
1054 042E DO_DET:
1055 042E E8 0B2B R CALL DRIVE_DET ; TRY TO DETERMINE
1056 0431 C3 RET ;
1057 ;
1058 0432 XLAT_NEW ENDP
1059 -----
1060 ; XLAT_OLD TRANSLATES DISKETTE STATE LOCATIONS FROM NEW ;
1061 ; ARCHITECTURE TO COMPATIBLE MODE. ;
1062 ; ;
1063 ; ;
1064 ; ON ENTRY: DI : DRIVE ;
1065 -----
1066 0432 XLAT_OLD PROC NEAR
1067 0432 F6 06 00BF R 01 TEST #HF_CNTRL,DUAL ; TEST CONTROLLER 1.D.
1068 0437 74 79 JZ XO_OUT ;
1069 0439 83 FF 01 CMP DI,1 ; VALID DRIVE ?
1070 043C 77 74 JA XO_OUT ; IF INVALID BACK
1071 043E 80 BD 0090 R 00 CMP #DSK_STATE[DI],0 ; NO DRIVE ?
1072 0443 74 6D JZ XO_OUT ; IF NO DRIVE TRANSLATE DONE
1073 ;
1074 ;----- TEST FOR SAVED DRIVE INFORMATION ALREADY SET
1075 ;
1076 0445 8B CF MOV CX,DI ; CX = DRIVE NUMBER
1077 0447 D0 E1 SHL CL,1 ; CL = SHIFT COUNT, A=0, B=4
1078 0449 D0 E1 SHL CL,1 ;
1079 044B B4 02 MOV AH,FM_T_CAPA ; LOAD MULTI DATA RATE BIT MASK
1080 044D D2 CC ROR AH,CL ; ROTATE BY MASK
1081 044F 84 26 00BF R JZ TEST #HF_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 #HF_CNTRL FOR THIS DRIVE
1085 ;
1086 0455 B4 07 MOV AH,DRV_DET+FM_T_CAPA+TRK_CAPA ; MASK TO KEEP
1087 0457 D2 CC ROR AH,CL ; FIX MASK TO KEEP
1088 0459 F6 04 AND NOT AH ; TRANSLATE MASK
1089 045B 20 26 00BF R AND #HF_CNTRL,AH ; KEEP BITS FROM OTHER DRIVE INTACT
1090 ;
1091 ;----- ACCESS CURRENT DRIVE BITS AND STORE IN #HF_CNTRL
1092 ;
1093 045F 8A 85 0090 R MOV AL,#DSK_STATE[DI] ; ACCESS STATE
1094 0463 24 07 AND AL,DRV_DET+FM_T_CAPA+TRK_CAPA ; KEEP DRIVE BITS
1095 0465 D2 CC ROR AL,CL ; FIX FOR THIS DRIVE
1096 0467 08 06 00BF R OR #HF_CNTRL,AL ; UPDATE SAVED DRIVE STATE
1097 ;
1098 ;----- TRANSLATE TO COMPATIBILITY MODE
1099 ;
1100 046B SAVE_SET:
1101 046B 8A A5 0090 R MOV AH,#DSK_STATE[DI] ; ACCESS STATE
1102 046F 8A FC MOV BH,AH ; TO BH FOR LATER
1103 0471 80 E4 C0 AND AH,RATE_MSK ; KEEP ONLY RATE
1104 0474 80 FC 00 CMP AH,RATE_500 ; RATE 500 ?
1105 0477 74 10 JZ CHK_HDR_80T ; YES 1./2,1./2 OR HIGH DATA RATE 80 TRK
1106 0479 B0 01 MOV AL,W3D1U ; 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 UNKNO:
1113 0485 B0 07 MOV AL,MED_UNK ; 'NONE OF THE ABOVE'
1114 0487 EB 20 JMP SHORT AL_SET ; PROCESS COMPLETE
1115 ;
1116 0489 CHK_HDR_80T:
1117 0489 E8 08CF R CALL CMDS_TYPE ; RETURN DRIVE TYPE IN (AL)
1118 048C 72 FF JC UNKNO ; SET 'NONE OF THE ABOVE'
1119 048E 3C 02 AND AL,02H ; 1.2MB DRIVE ?
1120 0490 75 F3 JNE UNKNO ; NO, GO SET 'NONE OF THE ABOVE'
1121 0492 B0 02 MOV AL,M3D1U ; AL = 1.2 IN 1.2 UNESTABLISHED
1122 0494 EB CC JMP SHORT TST_DET ;
1123 ;
1124 0496 CHK_250:
1125 0496 B0 00 MOV AL,W3D3U ; AL = 360 IN 360 UNESTABLISHED

```

```

1126 0498 80 FC 80          CMP      AH,RATE_250          ; RATE 250 ?
1127 0498 75 E8            JNZ     UNKN0                ; IF 50 FALL THRU
1128 049D F6 C7 01        TEST    BH,TRK_CAPA         ; 80 TRACK CAPABILITY ?
1129 04A0 75 E3            JNZ     UNKN0                ; IF 50 JUMP, FALL THRU TEST DET
1130
1131 04A2                    TST_DET:
1132 04A2 F6 C7 10        TEST    BH,MED_DET          ; DETERMINED ?
1133 04A5 74 02            JZ      AL_SET               ; IF NOT THEN SET
1134 04A7 04 03            ADD     AL,3                 ; MAKE DETERMINED/ESTABLISHED
1135
1136 04A9                    AL_SET:
1137 04A9 80 A5 0090 R F8   AND     #DSK_STATE[D1],NOT DRV_DET+RFLT_CAPA+TRK_CAPA ; CLEAR DRIVE
1138 04AE 08 85 0090 R    OR      #DSK_STATE[D1],AL    ; REPLACE WITH COMPATIBLE MODE
1139 04B2                    XO_OUT:
1140 04B2 C3                RET
1141 04B3                    XLAT_OLD:
1142                    ENDP
1143
1144                    -----
1145                    ; RD_WR_VF
1146                    ; COMMON READ, WRITE AND VERIFY;
1147                    ; MAIN LOOP FOR STATE RETRIES.
1148                    ;
1149                    ; ON ENTRY:  AH : READ/WRITE/VERIFY NEC PARAMETER
1150                    ; AL : READ/WRITE/VERIFY DMA PARAMETER
1151                    ;
1152                    ; ON EXIT:  #DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION ;
1153
1154 04B3                    RD_WR_VF:
1155 04B3 50                PUSHA
1156 04B4 E8 0404 R        CALL    XLAT_NEW             ; SAVE DMA, NEC PARAMETERS
1157 04B7 E8 0561 R        CALL    SETUP_STATE         ; TRANSLATE STATE TO PRESENT ARCH.
1158 04BA 55                POP     AX                   ; INITIALIZE START AND END RATE
1159                    ; RESTORE READ/WRITE/VERIFY
1160
1161 04BB                    DO_AGAIN:
1162 04BB F6 06 00BF R 01   TEST    #HF_CNTRL,DUAL      ; TEST CONTROLLER I.D.
1163 04C0 74 0A            JZ      RWV                  ;
1164 04C2 50                PUSHA
1165 04C3 E8 05F5 R        CALL    MED_CHANGE          ; SAVE READ/WRITE/VERIFY PARAMETER
1166 04C6 58                POP     AX                   ; MEDIA CHANGE AND RESET IF CHANGED
1167 04C7 73 03            JNC     RWV                  ; RESTORE READ/WRITE/VERIFY
1168 04C9 E9 0552 R        JMP     RWV_END             ;
1169
1170 04CC                    RWV:
1171 04CC 50                PUSHA
1172                    ; SAVE READ/WRITE/VERIFY PARAMETER
1173 04CD 8A B5 0090 R      MOV     DH,#DSK_STATE[D1]   ; GET RATE STATE OF THIS DRIVE
1174 04D1 80 E6 C0          AND     DH,RATE_MSK         ; KEEP ONLY RATE
1175
1176 04D2                    ;
1177 04D2 50                PUSHA
1178                    ; RETURN DRIVE TYPE IN (AL)
1179                    ; TEST FOR NO DRIVE
1180                    ; ASSUME TYPE, USE MAX TRACK
1181                    ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
1182                    ; TYPE NOT IN TABLE ASSUME DEFAULT
1183 04D3 50                PUSHA
1184 04D4 E8 08CF R        CALL    CMOS_TYPE           ;
1185 04D7 0A C0            OR      AL,AL                ;
1186 04D9 74 2F            JZ      RWV_ASSUME          ;
1187 04DB E8 03B1 R        CALL    DRN_TYPE_CHECK     ;
1188 04DE 72 2A            JC      RWV_ASSUME          ;
1189
1190                    ;--- SEARCH FOR MEDIA/DRIVE PARAMETER TABLE
1191
1192 04E0 57                PUSHA
1193 04E1 33 0B            XOR     BX,BX                ; SAVE DRIVE #
1194 04E3 B9 0006          MOV     CX,DR_CNT           ; BX = INDEX TO DR_TYPE TABLE
1195                    ; CX = LOOP COUNT
1196 04E6 2E: 8A A7 0000 R  RWV_DR_SEARCH:
1197 04E6 2E: 8A A7 0000 R  MOV     AH,CS:DR_TYPE[BX]   ; GET DRIVE TYPE
1198 04E8 80 E4 7F        AND     AH,BIT7OFF          ; MASK OUT MSB
1199 04EE 3A C4            CMP     AL,AH                ; MUST BE 40 TRACK
1200 04F0 75 0B            JNE     RWV_NXT_MD          ; NO, CHECK NEXT DRIVE TYPE
1201
1202 04F2                    RWV_DR_FND:
1203 04F2 2E: 8B BF 0001 R  MOV     DI,WORD PTR CS:DR_TYPE[BX+1] ; DI = MEDIA/DRIVE PARAMETER TABLE
1204
1205 04F5                    RWV_MD_SEARCH:
1206 04F5 2E: 3A 75 0C      MOV     DH,CS:[D1]_MD_RATE ; MATCH ?
1207 04F7 74 1D            JZ      RWV_MD_FND          ; YES, GO GET 1ST SPECIFY BYTE
1208
1209 04F9                    RWV_NXT_MD:
1210 04F9 4D                INC     BX,3                 ; CHECK NEXT DRIVE TYPE
1211 04FD 83 C3 03          LOOP   RWV_DR_SEARCH        ;
1212 0500 E2 E4            MOV     MOV     #DSKETTE_STATUS,OFFH ; FORCE IT TO RETRY
1213 0502 C6 06 0041 R FF  MOV     D1 ; RESTORE DRIVE #
1214 0507 5F                POP     D1                   ; GO RETRY
1215 0508 E8 3F            JMP     SHORT CHK_RET
1216
1217                    ;--- ASSUME PRIMARY DRIVE IS INSTALLED AS SHIPPED
1218
1219 050A                    RWV_ASSUME:
1220 050A 50A BB 0000 E      MOV     BX,OFFSET MD_TBL1   ; POINT TO 40 TK 250 KBS
1221 050D F6 85 0090 R 01  TEST    #DSK_STATE[DT],TRK_CAPA ; TEST FOR 80 TRACK
1222 0512 74 09            JZ      RWV_MD_FND1         ; MUST BE 40 TRACK
1223 0514 BB 0000 E        MOV     BX,OFFSET MD_TBL3   ; POINT TO 80 TK 500 KBS
1224 0517 E8 04 90        JMP     RWV_MD_FND1         ; GO SET SPECIFY PARAMETERS
1225
1226                    ;--- CS:BX POINTS TO MEDIA/DRIVE PARAMETER TABLE
1227
1228 051A                    RWV_MD_FND:
1229 051A 8B DF            MOV     BX,DI                ; BX = MEDIA/DRIVE PARAMETER TABLE
1230 051C 5F                POP     D1                   ; RESTORE DRIVE #
1231 051D                    RWV_MD_FND1:
1232
1233                    ;--- SEND THE SPECIFY COMMAND TO THE CONTROLLER
1234
1235 051D E8 03EC R        CALL    SEND_SPEC_MD        ;
1236 0520 E8 0658 R        CALL    CHK_LASTRATE       ; ZF=1 ATTEMPT RATE IS SAME AS LAST RATE
1237 0523 74 03            JZ      RWV_DBL             ; YES, SKIP SEND RATE COMMAND
1238 0525 E8 0637 R        CALL    SEND_RATE          ; SEND DATA RATE TO NEC
1239
1240 0528                    RWV_DBL:
1241 0528 53                PUSHA
1242 0529 E8 084C R        CALL    SETUP_DBL          ; SAVE MEDIA/DRIVE PARAM ADDRESS
1243 052C 5B                POP     BX                   ; CHECK FOR DOUBLE STEP
1244 052D 72 1A            JCH    CHK_RET             ; RESTORE ADDRESS
1245 052F 58                POP     AX                   ; ERROR FROM READ ID, POSSIBLE RETRY
1246 0530 50                PUSHA
1247 0531 53                POP     AX                   ; RESTORE NEC COMMAND
1248 0532 E8 0668 R        CALL    DMA_SETUP          ; SAVE MEDIA/DRIVE PARAM ADDRESS
1249 0535 58                POP     BX                   ; SET UP THE DMA
1250 0536 58                POP     AX                   ; RESTORE ADDRESS
1251 0537 72 1F            JCH    RWV_BAC             ; RESTORE NEC COMMAND
1252 0539 50                PUSHA
1253 053A 53                POP     BX                   ; CHECK FOR DMA BOUNDARY ERROR
1254 053B E8 06CB R        CALL    NEC_INIT           ; SAVE NEC COMMAND
1255 053E 5B                POP     BX                   ; SAVE MEDIA/DRIVE PARAM ADDRESS
1256                    ; INITIALIZE NEC
1257                    ; RESTORE ADDRESS
    
```

SECTION 5

```

1239 053F 72 08          JC      CHK_RET          ; IF ERROR DO NOT SEND MORE COMMANDS
1240 0541 EB 08 01 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 EB 07 27 R     CALL   NEC_TERM         ; TERMINATE, GET STATUS, ETC.
1243
1244 0549
1245 0549 EB 07BE R     CHK_RET: CALL   RETRY          ; CHECK FOR, SETUP RETRY
1246 054C 58            POP     AX              ; RESTORE READ/WRITE/VERIFY PARAMETER
1247 054D 73 03        JNC    RWV_END         ; CY = 0 NO RETRY
1248 054F E9 04BB R     JMP     DO_AGAIN       ; CY = 1 MEANS RETRY
1249
1250 0552
1251 0552 EB 077B R     RWV_END: CALL   DSTATE          ; ESTABLISH STATE IF SUCCESSFUL
1252 0555 EB 0805 R     CALL   NUM_TRANS      ; AL = NUMBER TRANSFERRED
1253
1254 0558
1255 0558 80            RWV_BAC: AX          ; BAD DMA ERROR ENTRY
1256 0559 EB 0432 R     CALL   XLAT_OLD       ; SAVE NUMBER TRANSFERRED
1257 055C 58            POP     AX              ; TRANSLATE STATE TO COMPATIBLE MODE
1258 055D EB 0832 R     CALL   SETUP_END     ; RESTORE NUMBER TRANSFERRED
1259 0560 C3            RET                    ; VARIOUS CLEANUPS
1260 0561
1261
1262
1263 -----
1264 0561
1265 0561 F6 06 00BF R 01  ; SETUP_STATE: INITIALIZES START AND END RATES.
1266 0566 74 37          ;-----
1267 0568 F6 85 0090 R 10  ; SETUP_STATE PROC NEAR
1268 056D 75 30          ; TEST CONTROLLER I.D.
1269 056F BB 4080        JZ     JIC              ; MEDIA DETERMINED ?
1270 0572 F6 85 0090 R 04  ; TEST
1271 0574 74 0C          JZ     JIC              ; NO STATES IF DETERMINED
1272 0579 80 00          MOV    AX,RATE_300*H+RATE_250 ; AH = START RATE, AL = END RATE
1273 057B F6 85 0090 R 02  ; TEST
1274 0580 75 03          JZ     AX_SET          ; DRIVE ?
1275 0582 BB 8080        MOV    AL,RATE_500     ; SET UP FOR 1.2 M END RATE
1276
1277 0585
1278 0585 80 A5 0090 R 1F  ; AND
1279 058A 08 A5 0090 R 1F  ; OR
1280 058E 80 26 00BB R F3  ; AND
1281 0593 D0 C8          ROR    AL,I            ; MSK+OBL_STEP ; TURN OFF THE RATE
1282 0595 D0 C8          ROR    AL,I            ; RATE FIRST TO TRY
1283 0597 D0 C8          ROR    AL,I            ; ERASE LAST TO TRY RATE BITS
1284 0599 D0 C8          ROR    AL,I            ; TO OPERATION LAST RATE LOCATION
1285 059B 08 06 00BB R 1F  ; OR
1286 059F
1287 059F C3            JIC:   OR     LASTRATE,AL ; LAST RATE
1288 05A0
1289
1290
1291 -----
1292 05A0
1293 05A0 F6 06 00BF R 01  ; FMT_INIT: ESTABLISH STATE IF UNESTABLISHED AT FORMAT TIME.
1294 05A5 74 49          ;-----
1295 05A7 F6 85 0090 R 10  ; FMT_INIT PROC NEAR
1296 05AC 75 42          ; TEST CONTROLLER I.D.
1297 05AE EB 08CF R     TEST  AH,CNTRL_DUAL   ; TEST CONTROLLER I.D.
1298 05B1 72 3E          JZ     FI_OUT         ; IF MEDIA ESTABLISHED
1299 05B3 FE C8          JC     FI_OUT         ; IF SO RETURN
1300 05B5 78 3A          CALL  CMDS_TYPE       ; RETURN DRIVE TYPE IN AL
1301 05B7 8A A5 0090 R 10  ; MOV    CL,DRV         ; ERROR IN CMOS ASSUME NO DRIVE
1302 05BB 80 E4 0F R     DEC    AL             ; MAKE ZERO ORIGIN
1303 05BE 0A C0          JS     CL_DRV         ; NO DRIVE IF AL 0
1304 05C0 75 05          OR     AH,DOSK_STATE[D1] ; AH = CURRENT STATE
1305 05C2 80 CC 90 R     AND    AH,NOT MED_DET+OBL_STEP+RATE_MSK ; CLEAR
1306 05C5 EB 25          OR     AL,AL         ; CHECK FOR 360
1307
1308 05C7
1309 05C7 FE C8          JNC   N_360         ; IF 360 WILL BE 0
1310 05C9 75 05          JZ     N_12         ; SET FORMAT RATE
1311 05CB 80 CC 10 R     OR     AH,MED_DET+RATE_500 ; SKIP OTHER STATE PROCESSING
1312 05CE EB 1C          JMP    SHORT_SKP_STATE
1313
1314 05D0
1315 05D0 FE C8          N_360: DEC    AL             ; 1.2 M DRIVE
1316 05D2 75 0F          JZ     JUMP_IF_NOT   ; JUMP IF NOT
1317 05D4 F6 C4 04        JZ     FI_RATE       ; RATE IF NOT
1318 05D7 74 10          OR     AH,MED_DET+RATE_500 ; SET FORMAT RATE
1319 05D9 F6 14 02        JZ     SHORT_SKP_STATE ; SKIP OTHER STATE PROCESSING
1320 05DC 74 0B          OR     AH,MED_DET+RATE_300 ; 1.2 M DRIVE
1321 05DE 80 CC 50 R     JZ     RATE_300     ; RATE 300
1322 05E1 EB 09          JMP    CONTINUE     ; CONTINUE
1323 05E3
1324 05E3 FE C8          N_12:  DEC    AL             ; CHECK FOR TYPE 3
1325 05E5 75 0A          JZ     JUMP_IF_NOT   ; JUMP IF NOT
1326 05E7 EB E2          JZ     FI_RATE       ; RATE 300
1327 05E9
1328 05E9 80 CC 90 R     OR     AH,MED_DET+RATE_250 ; MUST BE RATE 250
1329 05EC
1330 05EC 88 A5 0090 R 10  ; SKP_STATE: MOV    AH,DOSK_STATE[D1],AH ; STORE AWAY
1331 05F0
1332 05F0 C3            FI_OUT: RET
1333 05F1
1334 05F1 32 E4          CL_DRV: XOR    AH,AH         ; CLEAR STATE
1335 05F3 EB F7          JMP    SHORT_SKP_STATE ; SAVE IT
1336 05F5
1337 -----
1338
1339
1340
1341
1342
1343
1344
1345 05F5
1346 05F5 F6 06 00BF R 01  ; MED_CHANGE PROC NEAR
1347 05FA 74 37          ; TEST CONTROLLER I.D.
1348 05FC EB 0B21 R     JZ     READ_DSCHNG   ; READ DISK CHANGE LINE STATE
1349 05FF 74 34          JZ     MC_DET        ; BYPASS HANDLING DISK CHANGE LINE
1350 0601 80 A5 0090 R 1F  ; AND
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000

```

```

1353 ; ON THE NEXT OPERATION THE REQUIRED MOTOR START UP TIME WILL
1354 ; BE WAITED. (DRIVE MOTOR MAY GO OFF UPON DOOR OPENING).
1355
1356 0606 8B CF MOV CX,DI ; CL = DRIVE #
1357 0608 80 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 20 06 003F R AND #MOTOR_STATUS,AL ; TURN MOTOR OFF INDICATOR
1362 0613 FB STI ; INTERRUPTS ENABLED
1363 0614 E8 0913 R CALL MOTOR_ON ; TURN MOTOR ON
1364
1365 ;----- THIS SEQUENCE OF SEEEKS IS USED TO RESET DISKETTE CHANGE SIGNAL
1366
1367 0617 E8 0092 R CALL DISK_RESET ; RESET NEC
1368 061A 85 01 MOV CH,0TH ; MOVE TO CYLINDER 1
1369 061C E8 0A14 R CALL SEEK ; ISSUE SEEK
1370 061F 32 E0 XOR CH,CH ; MOVE TO CYLINDER 0
1371 0621 E8 0A14 R CALL SEEK ; ISSUE SEEK
1372 0624 C6 06 0041 R 06 MOV #DSKETTE_STATUS,MEDIA_CHANGE ; STORE IN STATUS
1373
1374 0629 E8 0B21 R OK1: CALL READ_DSKCHNG ; CHECK MEDIA CHANGED AGAIN
1375 062C 74 05 JZ OK2 ; IF ACTIVE, NO DISKETTE, TIMEOUT
1376
1377 062E C6 06 0041 R 80 OK4: MOV #DSKETTE_STATUS,TIME_OUT; TIMEOUT IF DRIVE EMPTY
1378
1379 0633 F9 OK2: STC ; MEDIA CHANGED, SET CY
1380 0634 C3 RET
1381 0635
MC_OUT: CLC ; NO MEDIA CHANGED, CLEAR CY
1382 0635 F8 RET
1383 0636 C3
1384 0637
MED_CHANGE ENDP
;-----
1385 ; SEND_RATE ;
1386 ; SENDS DATA RATE COMMAND TO NEC ;
1387 ; ON ENTRY: D1 = DRIVE # ;
1388 ; ON EXIT: NONE ;
1389 ; REGISTERS ALTERED: NONE ;
1390 ;-----
1391 SEND_RATE PROC NEAR ;
1392 0637 TEST #HF_CNTRL,DUAL ; TEST CONTROLLER I.D.
1393 0637 F6 06 008F R 01 JZ C_S_OUT ; SAVE REG.
1394 063C 74 19 PUSH AX ; ELSE CLEAR LAST RATE ATTEMPTED
1395 063E 50 AND #LAstrate,NOT SEND_MSK ; GET RATE STATE OF THIS DRIVE
1396 063F 80 26 008B R 3F MOV AL,#DSK_STATE[D1] ; KEEP ONLY RATE BITS
1397 0644 8A 85 0090 R AND AL,SEND_MSK ; SAVE NEW RATE FOR NEXT CHECK
1398 0648 24 C0 OR #LAstrate,AL ; MOVE TO BIT OUTPUT POSITIONS
1399 064A 08 06 008B R ROL AL,1
1400 064E D0 C0 ROL AL,1
1401 0650 D0 C0 MOV DX,03F7H ; OUTPUT NEW DATA RATE
1402 0652 BA 03F7 OUT DX,AL ; RESTORE REG.
1403 0655 EE POP AX
1404 0656 58
1405 0657 C_S_OUT: RET
1406 0657 C3 SEND_RATE ENDP
1407 0658
1408
1409 ;-----
1410 ; CHK_LAstrate ;
1411 ; CHECK PREVIOUS DATA RATE SENT TO THE CONTROLLER. ;
1412 ; ON ENTRY: ;
1413 ; D1 = DRIVE # ;
1414 ; ON EXIT: ;
1415 ; ZF = 1 DATA RATE IS THE SAME AS LAST RATE SENT TO NEC ;
1416 ; ZF = 0 DATA RATE IS DIFFERENT FROM LAST RATE ;
1417 ; REGISTERS ALTERED: NONE ;
1418 ;-----
1419 0658 CHK_LAstrate PROC NEAR ;
1420 0658 50 PUSH AX ; SAVE REG
1421 0659 8A 26 008B R MOV AH,#LAstrate ; GET LAST DATA RATE SELECTED
1422 065D 8A 85 0090 R MOV AL,#DSK_STATE[D1] ; GET RATE STATE OF THIS DRIVE
1423 0661 25 C0C0 AND AX,SEND_MSK*X ; KEEP ONLY RATE BITS OF BOTH
1424 0664 3A C4 CMP AL,AH ; COMPARE TO PREVIOUSLY TRIED
1425 ; ZF = 1 RATE IS THE SAME
1426 0666 58 POP AX ; RESTORE REG.
1427 0667 C3 RET
1428 0668 CHK_LAstrate ENDP
1429

```

```

1430 PAGE
1431 -----
1432 ; DMA_SETUP
1433 ; THIS ROUTINE SETS UP THE DMA FOR READ/WRITE/VERIFY
1434 ; OPERATIONS.
1435 ;
1436 ; ON ENTRY: AL = DMA COMMAND
1437 ;
1438 ; ON EXIT: 0DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION ;
1439 ;-----
1440 0668 DMA_SETUP PROC NEAR
1441 0668 FA CL I ; DISABLE INTERRUPTS DURING DMA SET-UP
1442 0669 E6 0C OUT DMA+12,AL ; SET THE FIRST/LAST F/F
1443 066B EB 00 JMP $+2 ; WAIT FOR I/O
1444 066D E6 0B OUT DMA+11,AL ; OUTPUT THE MODE BYTE
1445 066F C3 42 CMP AL,42H ; DMA VERIFY COMMAND
1446 0671 75 04 JNE NOT_VERIFY ; NO
1447 0673 33 C0 XOR AX,AX ; START ADDRESS
1448 0675 EB 15 JMP SHORT J33
1449 0677
1450 0677 BC 00 MOV AX,ES ; GET THE ES VALUE
1451 0679 D1 C0 ROL AX,1 ; ROTATE LEFT
1452 067B D1 C0 ROL AX,1
1453 067D D1 C0 ROL AX,1
1454 067F D1 C0 ROL AX,1
1455 0681 BA E8 MOV CH,AL ; GET HIGHEST NIBBLE OF ES TO CH
1456 0683 24 F0 AND AL,11110000B ; ZERO THE LOW NIBBLE FROM SEGMENT
1457 0685 03 46 02 ADD AX,[BP+2] ; TEST FOR CARRY FROM ADDITION
1458 0688 73 02 JNC J33
1459 068A FE C5 INC CH ; CARRY MEANS HIGH 4 BITS MUST BE INC
1460 068C
1461 068C 50 J33: PUSH AX ; SAVE START ADDRESS
1462 068D E6 04 OUT DMA+4,AL ; OUTPUT LOW ADDRESS
1463 068F EB 00 JMP $+2 ; WAIT FOR I/O
1464 0691 BA C4 MOV AL,AH ; OUTPUT HIGH ADDRESS
1465 0693 E6 04 OUT DMA+4,AL ; GET HIGH BITS
1466 0695 BA C5 MOV AL,AH ; I/O WAIT STATE
1467 0697 EB 00 JMP $+2
1468 0699 24 0F AND AL,00001111B
1469 069B E6 81 OUT 081H,AL ; OUTPUT HIGH 4 BITS TO PAGE REGISTER
1470
1471 ;----- DETERMINE COUNT
1472
1473 069D BB C6 MOV AX,S1 ; AL = # OF SECTORS
1474 069F B6 C4 XCHG AL,AH ; AH = # OF SECTORS
1475 06A1 2A C0 SUB AL,2A ; AL = 0, AX = # OF SECTORS * 256
1476 06A3 D1 E8 SHR AX,1 ; AX = # SECTORS * 128
1477 06A5 50 PUSH AX ; SAVE # OF SECTORS * 128
1478 06A6 B2 03 MOV DL,3 ; GET BYTES/SECTOR PARAMETER
1479 06A8 EB 08 FE R CALL GET_PARM
1480 06AB BA CC MOV CL,AH ; SHIFT COUNT (0=128, 1=256 ETC)
1481 06AD 58 POP AX ; AX = # OF SECTORS * 128
1482 06AE D3 E0 SHL AX,CL ; SHIFT BY PARAMETER VALUE
1483 06B0 48 DEC AX ; -1 FOR DMA VALUE
1484 06B1 50 PUSH AX ; SAVE COUNT VALUE
1485 06B2 E6 05 OUT DMA+5,AL ; LOW BYTE OF COUNT
1486 06B4 EB 00 JMP $+2 ; WAIT FOR I/O
1487 06B6 BA C4 MOV AL,AH ; HIGH BYTE OF COUNT
1488 06B8 E6 05 OUT DMA+5,AL ; RE-ENABLE INTERRUPTS
1489 06BA FB STI ; RECOVER COUNT VALUE
1490 06BB 59 POP CX ; RECOVER ADDRESS VALUE
1491 06BC 58 POP AX ; RECOVER ADDRESS VALUE
1492 06BD 03 C1 ADD AX,CX ; ADD, TEST FOR 64K OVERFLOW
1493 06BF B0 02 MOV AL,2 ; MODE FOR 8237
1494 06C1 E6 0A OUT DMA+10,AL ; INITIALIZE THE DISKETTE CHANNEL
1495
1496 06C3 73 05 JNC NO_BAD ; CHECK FOR ERROR
1497 06C5 C6 06 0041 R 09 MOV 0DSKETTE_STATUS,DMA_BOUNDARY ; SET ERROR
1498
1499 06CA NO_BAD: RET ; CY SET BY ABOVE IF ERROR
1500 06CA C3
1501 06CB DMA_SETUP ENDP
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 NEC_INIT PROC NEAR
1513 06CB 50 PUSH AX ; SAVE NEC COMMAND
1514 06CC EB 0913 R CALL MOTOR_ON ; TURN MOTOR ON FOR SPECIFIC DRIVE
1515
1516 ;----- DO THE SEEK OPERATION
1517
1518 06CF 8A 4E 01 MOV CH,[BP+1] ; CH = TRACK #
1519 06D0 E8 0A14 R CALL SEEK ; MOVE TO CORRECT TRACK
1520 06D5 58 POP AX ; RECOVER COMMAND
1521 06D6 72 18 JC ER_1 ; ERROR ON SEEK
1522 06D8 BB 06F0 R MOV BX,OFFSET ER_1 ; LOAD ERROR ADDRESS
1523 06DB 53 PUSH BX ; PUSH NEC_OUT ERROR RETURN
1524
1525 ;----- SEND OUT THE PARAMETERS TO THE CONTROLLER
1526
1527 06DC E8 09F0 R CALL NEC_OUTPUT ; OUTPUT THE OPERATION COMMAND
1528 06DF 8B C6 MOV AX,S1 ; AH = HEAD #
1529 06E1 8B DF MOV BX,D1 ; BL = DRIVE #
1530 06E3 D0 E4 SAL AH,1 ; MOVE IT TO BIT 2
1531 06E5 D0 E4 SAL AH,1
1532 06E7 80 E4 04 AND AH,00000100B ; ISOLATE THAT BIT
1533 06EA 0A E3 OR AH,BL ; OR IN THE DRIVE NUMBER
1534 06EC E8 09F0 R CALL NEC_OUTPUT ; FALL THRU CY SET IF ERROR
1535 06EF 5B POP BX ; THROW AWAY ERROR RETURN
1536 06F0
1537 06F0 C3 ER_1: RET
1538 06F1 NEC_INIT ENDP
1539
1540 ;----- RWV_CMD
1541 ; THIS ROUTINE SENDS PARAMETERS TO THE NEC SPECIFIC
1542 ; TO THE READ/WRITE/VERIFY OPERATIONS.
1543 ;

```



```

1544      ; ON ENTRY:      CS:BX = ADDRESS OF MEDIA/DRIVE PARAMETER TABLE ;
1545      ; ON EXIT :      #DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION ;
1546      -----
1547 06F1      RWV_COM PROC      NEAR
1548 06F1 8B 0726 R      MOV     AX,OFFSET ER_2      ; LOAD ERROR ADDRESS
1549 06F4 50          PUSH    AX                    ; PUSH NEC_OUT ERROR RETURN
1550 06F5 8A 66 01    MOV     AH,[BP+1]           ; OUTPUT TRACK #
1551 06F8 EB 09F0 R    CALL    NEC_OUTPUT         ;
1552 06FB 8B C6      MOV     AX,S1                ;
1553 06FD EB 09F0 R    CALL    NEC_OUTPUT         ; OUTPUT HEAD #
1554 0700 8A 66 00    MOV     AH,[BP]            ; OUTPUT SECTOR #
1555 0703 EB 09F0 R    CALL    NEC_OUTPUT         ;
1556 0706 B2 03      MOV     DL,3                ; BYTES/SECTOR PARAMETER FROM BLOCK
1557 0708 EB 08FE R    CALL    GET_PARM           ; . TO THE NEC
1558 070B EB 09F0 R    CALL    NEC_OUTPUT         ; OUTPUT TO CONTROLLER
1559 070E B2 04      MOV     DL,4                ; EOT PARAMETER FROM BLOCK
1560 0710 EB 08FE R    CALL    GET_PARM           ; . TO THE NEC
1561 0713 EB 09F0 R    CALL    NEC_OUTPUT         ; OUTPUT TO CONTROLLER
1562 0716 2E: 8A 67 05 MOV     AH,CS:[BX].MD_GAP   ; GET GAP LENGTH
1563 071A          R15:
1564 071A EB 09F0 R    CALL    NEC_OUTPUT         ;
1565 071D B2 06      MOV     DL,6                ; DTL PARAMETER FROM BLOCK
1566 071F EB 08FE R    CALL    GET_PARM           ; . TO THE NEC
1567 0722 EB 09F0 R    CALL    NEC_OUTPUT         ; OUTPUT TO CONTROLLER
1568 0725 58          POP     AX                    ; THROW AWAY ERROR EXIT
1569 0726          ER_2: RET
1570 0726 C3          RWV_COM ENDP
1571 0727          -----
1572          ; NEC_TERM
1573          ; THIS ROUTINE WAITS FOR THE OPERATION THEN ACCEPTS
1574          ; THE STATUS FROM THE NEC FOR THE READ/WRITE/VERIFY/
1575          ; FORMAT OPERATION.
1576          ;
1577          ; ON EXIT:      #DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION ;
1578          -----
1579          NEC_TERM PROC      NEAR
1580 0727          I----- LET THE OPERATION HAPPEN
1581          I-----
1582          PUSH    S1          ; SAVE HEAD #, # OF SECTORS
1583          CALL    WAIT_INT    ; WAIT FOR THE INTERRUPT
1584 0727 56          CALL    PUSHF              ;
1585 0728 EB 0ABA R    CALL    RESULTS           ; GET THE NEC STATUS
1586 072B 9C          JC     SET_END_POP        ;
1587 072C EB 0AE2 R    POPF                      ;
1588 072F 72 47      JC     SET_END             ;
1589 0731 9D          JNC    SET_END             ;
1590 0732 72 3C      JC     SET_END             ; LOOK FOR ERROR
1591          I----- CHECK THE RESULTS RETURNED BY THE CONTROLLER
1592          I-----
1593          CLD                ; SET THE CORRECT DIRECTION
1594 0734 FC          MOV     SI,OFFSET #NEC_STATUS ; POINT TO STATUS FIELD
1595 0735 BE 0042 R   LODS    #NEC_STATUS        ; GET ST0
1596 0738 AC          AND     AL,11000000B      ; TEST FOR NORMAL TERMINATION
1597 0739 24 C0      JZ     SET_END            ;
1598 073B 74 33      CMP     AL,01000000B      ; TEST FOR ABNORMAL TERMINATION
1599 073D 3C 40      JNB    SET_END            ; NOT ABNORMAL, BAD NEC
1600 073F 75 29      JNB    SET_END            ;
1601          I----- ABNORMAL TERMINATION, FIND OUT WHY
1602          I-----
1603          LODS    #NEC_STATUS ; GET ST1
1604 0741 AC          SAL     AL,1              ; TEST FOR EOT FOUND
1605 0742 D0 E0      MOV     AH,RECORD_NOT_FND ;
1606 0744 B4 04      JC     J19                ;
1607 0746 72 24      SAL     AL,1              ;
1608 0748 D0 E0      MOV     AH,BAD_CRC        ;
1609 074A D0 E0      SAL     AL,1              ; TEST FOR DMA OVERRUN
1610 074C B4 10      JC     J19                ;
1611 074E 72 1C      MOV     AH,BAD_DMA        ;
1612 0750 D0 E0      SAL     AL,1              ; TEST FOR RECORD NOT FOUND
1613 0752 B4 08      JC     J19                ;
1614 0754 72 16      MOV     AH,RECORD_NOT_FND ;
1615 0756 D0 E0      SAL     AL,1              ;
1616 0758 D0 E0      MOV     AH,WRITE_PROTECT  ; TEST FOR WRITE_PROTECT
1617 075A B4 04      JC     J19                ;
1618 075C 72 0E      MOV     AH,BAD_ADDR_MARK ; TEST MISSING ADDRESS MARK
1619 075E D0 E0      SAL     AL,1              ;
1620 0760 B4 03      JC     J19                ;
1621 0762 72 08      MOV     AH,BAD_ADDR_MARK ;
1622 0764 D0 E0      SAL     AL,1              ;
1623 0766 B4 02      JC     J19                ;
1624 0768 72 02      J19:
1625          I----- NEC MUST HAVE FAILED
1626          I-----
1627 076A          J19: MOV     AH,BAD_NEC
1628 076A B4 20      SET_END:OR    #DSKETTE_STATUS,AH
1629 076C          I-----
1630 076C 8B 26 0041 R CMP     #DSKETTE_STATUS,1 ; SET ERROR CONDITION
1631 0770          CMC
1632 0770 80 3E 0041 R POP
1633 0775 F5          RET
1634 0776 5E          ; RESTORE HEAD #, # OF SECTORS
1635 0777 C3          SET_END_POP:
1636          I-----
1637 0778          POPF
1638 0778 9D          JMP     SHORT SET_END
1639 0779 EB F5      NEC_TERM ENDP
1640 077B          -----
1641          ; DSTATE: ESTABLISH STATE UPON SUCCESSFUL OPERATION.
1642          ;
1643          ;
1644 077B          DSTATE PROC      NEAR
1645 077B F6 06 00BF R I TEST CONTROLLER I.D.
1646 077D 74 3B      JZ     SETBAC              ;
1647 0782 80 3E 0041 R CMP     #DSKETTE_STATUS,0 ; CHECK FOR ERROR
1648 0787 75 34      JNB    SETBAC              ; IF ERROR JUMP
1649 0789 80 8D 0090 R TEST    #DSK_STATE[D1],MED_DET ; NO ERROR, MARK MEDIA AS DETERMINED
1650 078E F6 85 0090 R JNZ     #DSK_STATE[D1],DRV_DET ; DRIVE DETERMINED ?
1651 0793 75 28      JNB    SETBAC              ; IF DETERMINED NO TRY TO DETERMINE
1652 0795 8A 85 0090 R MOV     AL,#DSK_STATE[D1] ; LOAD STATE
1653 0799 24 C0      AND     AL,RATE_MSK        ; KEEP ONLY RATE
1654 079B 3C 80      CMP     AL,RATE_250        ; RATE 250 ?
1655 079D 75 19      JNE    M_12                ; NO MUST BE 1.2M OR HI DATA RATE 80 TRK
1656          I-----
1657          I--- CHECK FOR HIGH DATA RATE 80 TRACK
    
```

SECTION 5

```

1658
1659 079F E8 08CF R      CALL    CMOS_TYPE           ; RETURN DRIVE TYPE IN (AL)
1660 07A2 72 14          JC      M_12                ; CMOS BAD ASSUME DEFAULT
1661 07A4 3C 02          CMP     AL,2                ; TYPE 2 DRIVE ?
1662 07A6 74 10          JE      M_12                ; YES-->ASSUME MULTI FORMAT CAPABILITY
1663 07AB 3C 04          CMP     AL,4                ; TYPE 4 DRIVE ?
1664 07AA 74 0C          JE      M_12                ; YES-->ASSUME MULTI FORMAT CAPABILITY
1665 07AC
M_720:
1666 07AC 80 A5 0090 R FD  AND     #DSK_STATE[D1],NOT_FMT_CAPA ; TURN OFF FORMAT CAPABILITY
1667 07B1 80 8D 0090 R 04  OR      #DSK_STATE[D1],DRV_DET     ; MARK DRIVE DETERMINED
1668 07B6 EB 05          JMP     SHORT_SETBAC        ; BACK
1669
1670 07B8
M_12:
1671 97B8 80 8D 0090 R 06  OR      #DSK_STATE[D1],DRV_DET+_FMT_CAPA ; TURN ON DETERMINED & FMT CAPA
1672
SETBAC:
1673 07BD
1674 07BD C3              RET
1675 07BE
DSTATE ENDP
-----
1676
; RETRY
1677
; DETERMINES WHETHER A RETRY IS NECESSARY. IF RETRY IS
1678
; REQUIRED THEN STATE INFORMATION IS UPDATED FOR RETRY.
1679
;
1680
; ON EXIT: CY = 1 FOR RETRY, CY = 0 FOR NO RETRY
1681
;
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 NO RETRY
1687 07CA 74 37          JZ      NO_RETRY              ;
1688 07CC 8A A5 0090 R    MOV     AH,#DSK_STATE[D1]      ; GET MEDIA STATE OF DRIVE
1689 07D0 F6 C4 10       TEST    AH,00FF              ; 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 07D8 BA 2E 008B R   MOV     CH,#LASTRATE         ; GET START OPERATION STATE
1693 07DC D0 C5          ROL     CH,1                  ; TO CORRESPONDING BITS
1694 07DE D0 C5          ROL     CH,1
1695 07E0 D0 C5          ROL     CH,1
1696 07E2 D0 C5          ROL     CH,1
1697 07E4 80 5E C0       AND     CH,RATE_MSK          ; ISOLATE RATE BITS
1698 07E7 3A EC          CMP     CH,AH                 ; ALL RATES TRIED
1699 07E9 74 18          JE      NO_RETRY              ; IF YES, THEN TRUE ERROR
1700
;
; SETUP STATE INDICATOR FOR RETRY ATTEMPT TO NEXT RATE
1701
;
; 00000000B (500) -> 10000000B (250)
1702
; 10000000B (250) -> 01000000B (300)
1703
; 01000000B (300) -> 00000000B (500)
1704
;
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[D1],NOT_RATE_MSK+DBL_STEP ; RATE, DBL STEP OFF
1710 07F8 08 A5 0090 R   OR      #DSK_STATE[D1],AH    ; TURN ON NEW RATE
1711 07FC 06 06 0041 R 00 MOV     OV,#DSKETTE_STATUS,0 ; RESET STATUS FOR RETRY
1712 0801 F9             STC
1713 0802 C3              RET
1714
NO_RETRY:
1715 0803
1716 0803 F8              CLC
1717 0804 C3              RET
1718 0805
RETRY ENDP
-----
1719
; NUM_TRANS
1720
; THIS ROUTINE CALCULATES THE NUMBER OF SECTORS THAT
1721
; WERE ACTUALLY TRANSFERRED TO/FROM THE DISKETTE.
1722
;
1723
; ON ENTRY: [BP+1] = TRACK
1724
; SI-HI = HEAD
1725
; [BP] = START SECTOR
1726
;
1727
; ON EXIT: AL = NUMBER ACTUALLY TRANSFERRED
1728
;
1729
-----
1730 0805
NUM_TRANS PROC NEAR
1731 0805 32 C0          XOR     AL,AH                 ; CLEAR FOR ERROR
1732 0807 80 3E 0041 R 00  CMP     #DSKETTE_STATUS,0      ; CHECK FOR ERROR
1733 080C 75 23          JNZ     NT_OUT                ; IF ERROR 0 TRANSFERRED
1734 080E B2 04          MOV     DL,4                  ; SECTORS/TRACK OFFSET TO DL
1735 0810 E8 08FE R     CALL    GET_PARM              ; AH = SECTORS/TRACK
1736 0813 8A 1E 0047 R   MOV     BL,#NEC_STATUS+5      ; GET ENDING SECTOR
1737 0817 8B EC          MOV     CX,SI                 ; CH = HEAD # STARTED
1738 0819 3A 2E 0046 R   CMP     CH,#NEC_STATUS+4      ; GET HEAD ENDED UP ON
1739 081D 75 09          JNZ     DIF_HD                 ; IF ON SAME HEAD, THEN NO ADJUST
1740
1741 081F 8A 2E 0045 R   MOV     CH,#NEC_STATUS+3      ; GET TRACK ENDED UP ON
1742 0823 3A 6E 01        CMP     CH,[BP+1]             ; 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
1747 082A 02 DC          ADD     BL,AH                 ; ADD SECTORS/TRACK
1748 082C
SAME_TRK:
1749 082C 2A 5E 00       SUB     BL,[BP]                ; SUBTRACT START FROM END
1750 082F 8A C3          MOV     AL,BL                 ; TO AL
1751
NT_OUT:
1752 0831
1753 0831 C3              RET
1754 0832
NUM_TRANS ENDP
-----
1755
; SETUP_END
1756
; 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
-----
1762 0832
SETUP_END PROC NEAR
1763 0832 B2 02          MOV     DL,2                  ; GET THE MOTOR WAIT PARAMETER
1764 0834 50             AX     PUSH                     ; SAVE NUMBER TRANSFERRED
1765 0835 E8 08FE R     CALL    GET_PARM              ; STORE UPON RETURN
1766 0838 8B 26 0040 R   MOV     #MOTOR_COUNT,AH
1767 083C 58             POP     AX                     ; RESTORE NUMBER TRANSFERRED
1768 083D 8A 26 0041 R   MOV     AH,#DSKETTE_STATUS    ; GET STATUS OF OPERATION
1769 0841 0A E4          OR      AH,AH                 ; CHECK FOR ERROR
1770 0843 74 02          JZ      UN_ERR                ; NO ERROR
1771 0845 32 C0          XOR     AL,AL                 ; CLEAR NUMBER RETURNED

```

```

1772
1773 0847
1774 0847 80 FC 01
1775 084A F5
1776 084B C3
1777 084C
1778
1779
1780
1781
1782
1783
1784
1785 084C
1786 084C F6 04 008F R 01
1787 0851 74 65
1788 0853 8A A5 0090 R
1789 0857 F6 C4 10
1790 085A 75 5C
1791
1792
1793
1794 085C C6 06 003E R 00
1795 0861 E8 0913 R
1796 0864 B5 00
1797 0866 E8 0A14 R
1798 0869 E8 088A R
1799 086C 72 35
1800
1801
1802
1803 086E B9 0450
1804 0871 F6 85 0090 R 01
1805 0876 74 02
1806 0878 B1 A0
1807
1808
1809
1810
1811
1812 087A
1813 087A 51
1814 087B C6 06 0041 R 00
1815 0880 33 C0
1816 0882 D0 ED
1817 0884 D0 D0
1818 0886 D0 D0
1819 0888 D0 D0
1820 088A 50
1821 088B E8 0A14 R
1822 088E 58
1823 088F 08 F8
1824 0891 E8 088A R
1825 0894 9C
1826 0895 81 E7 00FB
1827 0899 9D
1828 089A 59
1829 089B 73 08
1830 089D FE C5
1831 089F 3A 09
1832 08A1 75 D7
1833
1834
1835
1836 08A3
1837 08A3 F9
1838 08A4 C3
1839
1840 08A5
1841 08A5 8A 0E 0045 R
1842 08A9 88 8D 0094 R
1843 08AD D0 ED
1844 08AF 3A E9
1845 08B1 74 05
1846 08B3 80 8D 0090 R 20
1847
1848 08B8
1849 08B8 F8
1850 08B9 C3
1851 08BA
1852
1853
1854
1855
1856
1857
1858
1859
1860 08BA
1861 08BA B8 08CE R
1862 08BD 50
1863 08BE B4 4A
1864 08C0 E8 09F0 R
1865 08C3 B8 C7
1866 08C5 8A E0
1867 08C7 E8 09F0 R
1868 08CA E8 0727 R
1869 08CD 58
1870 08CE
1871 08CE C3
1872 08CF
1873
1874
1875
1876
1877
1878
1879
1880
1881 08CF
1882 08CF A0 0010 R
1883 08D2 24 C1
1884 08D4 D0 E8
1885 08D6 73 20

```

```

NUN_ERR:
    CMP AH,1
    CMC
    RET
    ; SET THE CARRY FLAG TO INDICATE
    ; SUCCESS OR FAILURE

SETUP_END ENDP

-----
; SETUP_DBL
; CHECK DOUBLE STEP.
; ON ENTRY: DI = DRIVE
; ON EXIT: CY = 1 MEANS ERROR
;
SETUP_DBL PROC NEAR
    TEST #HF_CNTRL_DUAL
    JZ NO_DBL
    MOV AH,#DSK_STATE[DI]
    TEST AH,MED_DET
    JNZ NO_DBL
    ; TEST CONTROLLER I.D.
    ; NO DOUBLE STEPPING REQUIRED
    ; ACCESS STATE
    ; ESTABLISHED STATE ?
    ; IF ESTABLISHED THEN DOUBLE DONE
;-----
; CHECK FOR TRACK 0 TO SPEED UP ACKNOWLEDGE OF UNFORMATTED DISKETTE
;
    MOV #SEEK_STATUS,0
    CALL MOTOR_ON
    MOV CH,0
    CALL SEEK
    CALL READ_ID
    JC SD_ERR
    ; SET RECALIBRATE REQUIRED ON ALL DRIVES
    ; ENSURE MOTOR STAY ON
    ; LOAD TRACK 0
    ; SEEK TO TRACK 0
    ; READ ID FUNCTION
    ; IF ERROR NO TRACK 0

;----- INITIALIZE START AND MAX TRACKS (TIMES 2 FOR BOTH HEADS)
;
    MOV CX,0450H
    TEST #DSK_STATE[DI],TRK_CAPA
    JZ CNT_OK
    MOV CL,DA0H
    ; START, MAX TRACKS
    ; TEST FOR 80 TRACK CAPABILITY
    ; IF NOT COUNT 15 SETUP
    ; MAXIMUM TRACK 1.2 MB

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

CNT_OK:
    PUSH CX
    MOV #DSKETTE_STATUS,0
    XOR AX,AX
    SHR CH,1
    RCL AL,1
    RCL AL,1
    RCL AL,1
    PUSH AX
    CALL SEEK
    POP AX
    OR DI,AX
    CALL READ_ID
    PUSHF
    AND DI,11111011B
    POPF
    POP CX
    JNC DO_CHK
    INC CH
    CMP CH,CL
    JNZ CNT_OK
    ; SAVE TRACK, COUNT
    ; CLEAR STATUS, EXPECT ERRORS
    ; CLEAR AX
    ; HALVE TRACK, CY = HEAD
    ; AX = HEAD IN CORRECT BIT
    ; SAVE HEAD
    ; SEEK TO TRACK
    ; RESTORE HEAD
    ; DI = HEAD OR'ED DRIVE
    ; READ ID HEAD 0
    ; SAVE RETURN FROM READ_ID
    ; TURN OFF HEAD 1 BIT
    ; RESTORE ERROR RETURN
    ; RESTORE COUNT
    ; IF OK, ASKED = RETURNED TRACK ?
    ; INC FOR NEXT TRACK
    ; REACHED MAXIMUM YET
    ; CONTINUE TILL ALL TRIED

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

SD_ERR:
    STC
    RET
    ; SET CARRY FOR ERROR
    ; SETUP_DBL ERROR EXIT

DO_CHK:
    MOV CL,#NEC_STATUS+3
    MOV #DSK_TRK[DI],CL
    SHR CH,1
    CMP CH,CL
    NO_DBL
    OR #DSK_STATE[DI],DBL_STEP
    ; LOAD RETURNED TRACK
    ; STORE TRACK NUMBER
    ; HALVE TRACK
    ; IS IT THE SAME AS ASKED FOR TRACK
    ; IF SAME THEN NO DOUBLE STEP
    ; TURN ON DOUBLE STEP REQUIRED

NO_DBL:
    CLC
    RET
    ; CLEAR ERROR FLAG

SETUP_DBL ENDP

-----
; READ_ID
; READ ID FUNCTION.
; ON ENTRY: DI = BIT 2 = HEAD; BITS 1,0 = DRIVE
; ON EXIT: DI = BIT 2 IS RESET, BITS 1,0 = DRIVE
; #DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
;
READ_ID PROC NEAR
    MOV AX,OFFSET_ER_3
    PUSH AX
    MOV AH,4AH
    CALL NEC_OUTPUT
    MOV AX,DI
    MOV AH,AL
    CALL NEC_OUTPUT
    CALL NEC_TERM
    POP AX
    ; MOVE NEC OUTPUT ERROR ADDRESS
    ; READ ID COMMAND
    ; TO CONTROLLER
    ; DRIVE # TO AH, HEAD 0
    ; TO CONTROLLER
    ; WAIT FOR OPERATION, GET STATUS
    ; THROW AWAY ERROR ADDRESS

ER_3:
    RET

READ_ID ENDP

-----
; CMOS_TYPE
; RETURNS DISKETTE TYPE FROM CMOS
;
; ON ENTRY: DI = DRIVE #
; ON EXIT: AL = TYPE; CY REFLECTS STATUS
;
CMOS_TYPE PROC NEAR
    MOV AL,BYTE PTR #EQUIP_FLAG
    AND AL,11000011B
    SHR AL,1
    JNC TYP_ZERO
    ; LOAD EQUIPMENT FLAG FOR # DISKETTES
    ; KEEP DISKETTE DRIVE BITS
    ; ARE THERE ANY DRIVES INSTALLED?
    ; NC-->NO DRIVES TYPE ZERO

```

```

1886 08D8 D0 C0          ROL    AL,1          ; ROTATE TO ORIGINAL POSITION
1887 08DA D0 C0          ROL    AL,1          ; ROTATE BITS 6 AND 7 TO 0 AND 1
1888 08DC D0 C0          ROL    AL,1
1889 08DE 32 E4          XOR    AH,AH         ; AX=NUMBER OF DRIVES
1890 08E0 3B C7          CMP    AX,DI         ; IS DRIVE REQUESTED PRESENT
1891 08E2 72 14          JC     TYP_ZERO      ; C->REQUESTED DRIVE NOT PRESENT
1892 08E4 F6 06 00BF R 01  ; TEST  #HF_CNTRL_DUAL ; TEST CONTROLLER I.O.
1893 08E9 75 10          JNZ   CR2
1894 08EB F6 85 0090 R 01 ; TEST  #DSK_STATE[DI],TRK_CAPA ; TEST FOR 80 TRACKS
1895 08F0 B0 01          MOV    AL,1          ; DRIVE TYPE HAS 40 TRACKS
1896 08F2 74 06          JZ     CR1
1897 08F4 B0 03          MOV    AL,3          ; DRIVE TYPE HAS 80 TRACKS
1898 08F6 EB 02          JMP    SHORT CR1
1899 08F8
1900 08FB 32 C0          XOR    AL,AL         ; DRIVE TYPE 0
1901 08FA
1902 08FA C3          CR1:  RET             ; EXIT WITH AL=TYPE ACCORDING TO TRACKS
1903 08FB
1904 08FB F9          CR2:  STC             ;
1905 08FC EB FC          JMP    CR1           ; EXIT WITH CARRY IF DUAL CARD
1906 08FE
1907
1908 -----
1909 | GET_PARM |
1910 | THIS ROUTINE FETCHES THE INDEXED POINTER FROM THE |
1911 | DISK BASE BLOCK POINTED TO BY THE DATA VARIABLE |
1912 | #DISK_POINTER, A BYTE FROM THAT TABLE IS THEN MOVED |
1913 | INTO AH, THE INDEX OF THAT BYTE BEING THE PARAMETER |
1914 | IN DL. |
1915 | ON ENTRY: DL = INDEX OF BYTE TO BE FETCHED |
1916 | ON EXIT: AH = THAT BYTE FROM BLOCK |
1917 | AL,OH IS DESTROYED |
1918 | |
1919 -----
1920 08FE          GET_PARM  PROC  NEAR
1921 08FE IE          PUSH  DS
1922 08FF 56          PUSH  SI
1923 0900 2B C0          SUB  AX,AX           ; DS = 0 , BIOS DATA AREA
1924 0902 BE D8          MOV  DS,AX
1925 0904 87 D3          XCHG DX,BX          ; BL = INDEX
1926 0906 2A FF          SUB  BH,BH           ; BX = INDEX
1927
1928 0908 C5 36 0078 R  ; ASSUME DS:ABS0
1929 090C BA 20          LDS  SI,#DISK_POINTER ; POINT TO BLOCK
1930 090E 87 D3          MOV  AH,[SI-BX]     ; GET THE WORD
1931 0910 5E          POP  SI             ; RESTORE BX
1932 0911 1F          POP  DS
1933 0912 C3          RET
1934
1935 0913          GET_PARM  ENDP
1936
1937 -----
1938 | MOTOR_ON |
1939 | TURN MOTOR ON AND WAIT FOR MOTOR START UP TIME. THE #MOTOR COUNT |
1940 | IS REPLACED WITH A SUFFICIENTLY HIGH NUMBER (0FFH) TO ENSURE |
1941 | THAT THE MOTOR DOES NOT GO OFF DURING THE OPERATION. IF THE |
1942 | MOTOR NEEDED TO BE TURNED ON, THE MULTITASKING HOOK FUNCTION |
1943 | (AX=90FDH, INT 15H) IS CALLED TELLING THE OPERATING SYSTEM |
1944 | THAT THE BIOS IS ABOUT TO WAIT FOR MOTOR START UP. IF THIS |
1945 | FUNCTION RETURNS WITH CY = 1, IT MEANS THAT THE MINIMUM WAIT |
1946 | HAS BEEN COMPLETED. AT THIS POINT A CHECK IS MADE TO ENSURE |
1947 | THAT THE MOTOR WASN'T TURNED OFF BY THE TIMER. IF THE HOOK DID |
1948 | NOT WAIT, THE WAIT FUNCTION (AH=068H) IS CALLED TO WAIT THE |
1949 | PRESCRIBED AMOUNT OF TIME. IF THE CARRY FLAG IS SET ON RETURN, |
1950 | IT MEANS THAT THE FUNCTION IS IN USE AND DID NOT PERFORM THE |
1951 | WAIT. A TIMER ! WAIT LOOP WILL THEN DO THE WAIT. |
1952 | ON ENTRY: DI = DRIVE # |
1953 | |
1954 | ON EXIT: AX,CX,DX DESTROYED |
1955 -----
1956 0913          MOTOR_ON  PROC  NEAR
1957 0913 53          PUSH  BX             ; SAVE REG.
1958 0914 EB 095E R    CALL  TURN_ON       ; TURN ON MOTOR
1959 0917 72 43          JC     IF_CY1_NO_WAIT ; IF CY1 NO WAIT
1960 0919 EB 0432 R    CALL  XLAT_OLD      ; TRANSLATE STATE TO COMPATIBLE MODE
1961 091C BE 90FDH    MOV  AX,#90FDH      ; LOAD WAIT CODE & TYPE
1962 091F CD 15          INT  15H            ; TELL OPERATING SYSTEM ABOUT TO DO WAIT
1963 0921 9C          PUSHF               ; SAVE CY FOR TEST
1964 0922 EB 0404 R  CALL  XLAT_NEW      ; TRANSLATE STATE TO PRESENT ARCH.
1965 0925 90          POPF                ; RESTORE CY FOR TEST
1966 0926 73 05          JNC   BYPASS_LOOP  ; BYPASS LOOP IF OP SYSTEM HANDLED WAIT
1967 0928 EB 095E R    CALL  TURN_ON       ; CHECK AGAIN IF MOTOR ON
1968 092B 72 2F          JC     IF_NO_WAIT   ; IF NO WAIT MEANS IT IS ON
1969
1970 092D          M_WAIT:
1971 092D B2 0A          MOV  DL,10          ; GET THE MOTOR WAIT PARAMETER
1972 092F EB 08FE R    CALL  GET_PARM      ;
1973 0932 BA C4          MOV  AL,AH          ; AL = MOTOR WAIT PARAMETER
1974 0934 32 E4          XOR  AH,AH          ; AX = MOTOR WAIT PARAMETER
1975 0936 3C 08          CMP  AL,8           ; SEE IF AT LEAST A SECOND IS SPECIFIED
1976 0938 73 02          JAE  GP2            ; IF YES, CONTINUE
1977 093A B0 08          MOV  AL,8           ; ONE SECOND WAIT FOR MOTOR START UP
1978
1979 -----
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             ; MULTIPLY BY HALF OF WHAT'S NECESSARY
1984 0942 B8 CA          MOV  CX,DX          ; CX = HIGH WORD
1985 0944 B8 D0          MOV  DX,AX          ; CX,DX = 1/2 * (# OF MICROSECONDS)
1986 0946 F8          CLC                 ; CLEAR CARRY FOR ROTATE
1987 0947 D1 D2          RCL  DX,1           ; DOUBLE LOW WORD, CY CONTAINS OVERFLOW
1988 0949 D1 D1          RCL  CX,1           ; DOUBLE HI, INCLUDING LOW WORD OVERFLOW
1989 094B B4 86          MOV  AH,#86H       ; LOAD WAIT CODE
1990 094D CD 15          INT  15H            ; PERFORM WAIT
1991 094F 58          POP  AX             ; RESTORE WAIT PARAMETER
1992 0950 73 0A          JNC   MOT_IS_ON     ; CY MEANS WAIT COULD NOT BE DONE
1993
1994 -----
1995 | FOLLOWING LOOPS REQUIRED WHEN RTC WAIT FUNCTION IS ALREADY IN USE |
1996 0952          J13:
1997 0952 B9 205E    MOV  CX,8286        ; WAIT FOR 1/8 SECOND PER (AL)
1998 0955 EB 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 75 F6                                JNZ     J13                                ; ARE WE DONE YET
2001
2002 095C                                MOT_IS_ON: POP     BX                                ; RESTORE REG.
2003 095C 5B
2004 095D C3                                RET
2005 095E                                MOTOR_ON  ENDP
2006
2007                                ;-----
2008                                ; TURN_ON
2009                                ; TURN MOTOR ON AND RETURN WAIT STATE.
2010                                ;
2011                                ; ON ENTRY:  DI = DRIVE #
2012                                ;
2013                                ; ON EXIT:   CY = 0 MEANS WAIT REQUIRED
2014                                ;           CX = 1 MEANS NO WAIT REQUIRED
2015                                ;           AX,BX,CX,DX DESTROYED
2016                                ;-----
2017 095E 8B DF                                TURN_ON PROC NEAR
2018 0960 8A CB                                MOV     BX,DI                                ; BX = DRIVE #
2019 0962 D0 C3                                MOV     CL,BL                                ; CL = DRIVE #
2020 0964 D0 C3                                ROL     BL,1                                ; BL = DRIVE SELECT
2021 0966 D0 C3                                ROL     BL,1
2022 0968 D0 C3                                ROL     BL,1
2023 096A FA                                CLI
2024 096B C6 06 0040 R FF                    MOV     #MOTOR_COUNT,OFFH                ; NO INTERRUPTS WHILE DETERMINING STATUS
2025 0970 A0 003F R                            MOV     AL,#MOTOR_STATUS                 ; GET DIGITAL OUTPUT REGISTER REFLECTION
2026 0973 24 30                                AND     AL,00110000B                     ; KEEP ONLY DRIVE SELECT BITS
2027 0975 B4 01                                MOV     SHL,AH,1                          ; MASK FOR DETERMINING MOTOR BIT
2028 0977 D2 E4                                AND     AH,CL                                ; AH = MOTOR ON, A=00000001, B=00000010
2029
2030                                ; AL = DRIVE SELECT FROM #MOTOR_STATUS
2031                                ; BL = DRIVE SELECT DESIRED
2032                                ; AH = MOTOR ON MASK DESIRED
2033
2034 0979 3A C2                                CMP     AL,BL                                ; REQUESTED DRIVE ALREADY SELECTED ?
2035 097B 75 06                                JNZ     TURN_IT_ON                          ; IF NOT SELECTED JUMP
2036 097D 84 26 003F R                        TEST    AH,#MOTOR_STATUS                 ; TEST MOTOR ON BIT
2037 0981 75 31                                JNZ     NO_MOT_WAIT                         ; JUMP IF MOTOR ON AND SELECTED
2038
2039 0983                                TURN_IT_ON:
2040 0983 0A E3                                OR      AH,BL                                ; AH = DRIVE SELECT AND MOTOR ON
2041 0985 8A 3E 003F R                        MOV     BH,#MOTOR_STATUS                 ; SAVE COPY OF #MOTOR STATUS BEFORE
2042 0987 0E 0F                                AND     BH,00001111B                     ; KEEP ONLY MOTOR BITS
2043 098C 80 26 003F R C0                    AND     #MOTOR_STATUS,11000000B        ; CLEAR OUT DRIVE SELECT AND MOTORS
2044 0991 08 26 003F R                        OR      #MOTOR_STATUS,AH                 ; OR IN DRIVE SELECTED
2045 0995 A0 003F R                            MOV     AL,#MOTOR_STATUS                 ; GET DIGITAL OUTPUT REGISTER REFLECTION
2046 0998 8A D8                                MOV     BL,AL                                ; BL=#MOTOR STATUS AFTER, BH=BEFORE
2047 099A 80 E3 0F                            AND     BL,00001111B                     ; KEEP ONLY MOTOR BITS
2048 099D FB                                STI
2049 099E 24 3F                                AND     AL,00111111B                     ; STRIP AWAY UNWANTED BITS
2050 09A0 D0 C0                                ROL     AL,1                                ; PUT BITS IN DESIRED POSITIONS
2051 09A2 D0 C0                                ROL     AL,1
2052 09A4 D0 C0                                ROL     AL,1
2053 09A6 D0 C0                                ROL     AL,1
2054 09A8 00 C0                                OR      AL,00001100B                     ; NO RESET, ENABLE DMA/INTERRUPT
2055 09AA BA 03F2                             MOV     DX,03F2H                          ; SELECT DRIVE AND TURN ON MOTOR
2056 09AD EE                                OUT     DX,AL
2057 09AE 3A DF                                CMP     BL,BH                                ; NEW MOTOR TURNED ON ?
2058 09B0 74 02                                JZ      NO_MOT_WAIT                       ; NO WAIT REQUIRED IF JUST SELECT
2059 09B2 F8                                CLC
2060 09B3 C3                                RET
2061
2062 09B4                                NO_MOT_WAIT:
2063 09B4 F9                                STC
2064 09B5 FB                                STC
2065 09B6 C3                                RET
2066 09B7                                TURN_ON ENDP
2067                                ;-----
2068                                ; HD_WAIT
2069                                ; WAIT FOR HEAD SETTLE TIME.
2070                                ;
2071                                ; ON ENTRY:  DI = DRIVE #
2072                                ;
2073                                ; ON EXIT:   AX,BX,CX,DX DESTROYED
2074                                ;-----
2075 09B7                                HD_WAIT  PROC NEAR
2076 09B7 B2 09                                MOV     DL,9                                ; POINT TO HEAD SETTLE PARAMETER
2077 09B9 E8 08FE R                            CALL    GET_PARM                           ; GET PARAMETER
2078 09BC F6 06 003F R 80                    TEST    #MOTOR_STATUS,10000000B         ; SEE IF A WRITE OPERATION
2079 09C1 74 09                                JZ      !SNT_WRITE                         ; IF NOT, DO NOT ENFORCE ANY VALUES
2080 09C3 80 FC 0F                            CMP     AH,15                              ; IS WAIT 15 MILLI SECONDS OR GREATER?
2081 09C6 73 08                                JAE     DO_WAIT                            ; IF THERE DO NOT ENFORCE
2082 09C8 B4 0F                                MOV     AH,15                              ; HEAD SETTLE MINIMUM
2083 09CA EB 04                                JMP     SHORT DO_WAIT                      ; DO WAIT OPERATION
2084 09CC                                !SNT_WRITE:
2085 09CC 0A E4                                OR      AH,AH                                ; CHECK FOR WAIT TO BE ZERO
2086 09CE 74 1F                                JZ      HW_DONE                            ; IF NOT WRITE AND 0 THEN EXIT
2087
2088                                ;----- AH CONTAINS NUMBER OF MILLISECDS TO WAIT
2089
2090 09D0                                DO_WAIT:
2091 09D0 8A C4                                MOV     AL,AH                                ; AL = # MILLISECDS
2092 09D2 32 E4                                XOR     AH,AH                                ; AX = # MILLISECDS
2093 09D4 50                                PUSH   AX                                ; SAVE HEAD SETTLE PARAMETER
2094 09D5 BA 03EB                             MOV     DX,1000                            ; SET UP FOR MULTIPLY TO MICROSECONDS
2095 09D8 F7 E2                                MUL     DX,AX                                ; DX,AX = # MICROSECONDS
2096 09DA 8B CA                                MOV     CX,DX                                ; CX,AX = # MICROSECONDS
2097 09DC 8B D0                                MOV     DX,AX                                ; CX,DX = # MICROSECONDS
2098 09DE B4 86                                MOV     AH,86H                             ; LOAD WAIT CODE
2099 09E0 CD 15                                INT     15H                                ; PERFORM WAIT
2100 09E2 58                                POP     AX                                ; RESTORE HEAD SETTLE PARAMETER
2101 09E3 73 0A                                JNC     HW_DONE                            ; CHECK FOR EVENT WAIT ACTIVE
2102
2103 09E5                                J29:
2104 09E5 89 0042                             MOV     CX,66                                ; 1 MILLISECOND LOOP
2105 09E8 EB 0000 E                            CALL    WAITF                              ; COUNT AT 15.085737 US PER COUNT
2106 09EB FE C4                                DEC     CX                                ; DELAY FOR 1 MILLISECOND
2107 09ED 75 F6                                JNZ     J29                                ; DECREMENT THE COUNT
2108 09EF                                ; DO AL MILLISECOND # OF TIMES
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
2131 09F0 53
2132 09F1 BA 03F4
2133 09F4 B3 02
2134 09F6 33 C9
2135
2136 09F8 EC
2137 09F9 24 C0
2138 09FB 3C 80
2139 09FD 74 0F
2140 09FF E2 F7
2141
2142 0A01 FE CB
2143 0A03 75 F3
2144
2145
2146
2147 0A05 80 0E 0041 R 80
2148
2149 0A0A 5B
2150
2151 0A0B 5B
2152 0A0C F9
2153 0A0D C3
2154
2155
2156
2157 0A0E 8A C4
2158 0A10 42
2159 0A11 EE
2160
2161 0A12 5B
2162 0A13 C3
2163 0A14
-----
2164
2165
2166
2167 ; THIS ROUTINE WILL MOVE THE HEAD ON THE NAMED DRIVE ;
2168 ; TO THE NAMED TRACK. IF THE DRIVE HAS NOT BEEN ACCESSED ;
2169 ; SINCE THE DRIVE RESET COMMAND WAS ISSUED, THE DRIVE ;
2170 ; WILL BE RECALIBRATED. ;
2171 ; ;
2172 ; ON ENTRY: DI = DRIVE # ;
2173 ; CH = TRACK # ;
2174 ; ;
2175 ; ON EXIT: *DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION. ;
2176 ; AX,BX,CX,DX DESTROYED ;
-----
2177 0A14
2178 0A14 8B DF
2179 0A16 BA 0A7B R
2180 0A19 52
2181 0A1A 80 01
2182 0A1C 86 CB
2183 0A1E D2 C0
2184 0A20 86 CB
2185 0A22 84 06 003E R
2186 0A26 75 21
2187
2188 0A28 08 06 003E R
2189 0A2C E8 0A7C R
2190 0A2F 73 0A
2191
2192
2193
2194 0A31 C6 06 0041 R 80
2195 0A36 E8 0A7C R
2196 0A39 72 3F
2197
2198 0A3B
2199 0A3B 83 FF 01
2200 0A3E 77 21
2201 0A40 C6 85 0094 R 80
2202 0A45 0A E9
2203 0A47 74 2C
2204
2205
2206
2207 0A49
2208 0A49 83 FF 01
2209 0A4C 77 13
2210 0A4E F6 85 0090 R 20
2211 0A53 74 02
2212 0A55 D0 E5
2213
2214 0A57 3A AD 0094 R
2215 0A5B 74 1D
2216
2217 0A5D 88 AD 0094 R
2218 0A61
2219 0A61 51
2220 0A62 84 0F
2221 0A64 E8 09F0 R
2222 0A67 8B DF
2223 0A69 BA E3
2224 0A6B E8 09F0 R
2225 0A6E 58
2226 0A6F E8 09F0 R
2227 0A72 E8 0A93 R
-----
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

```

```

2228
2229
2230
2231 0A75
2232 0A75 9C
2233 0A76 E8 09B7 R
2234 0A79 9D
2235 0A7A
2236 0A7A 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 0A86 8B 0F
2255 0A88 8A 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 0AB1 R
2274 0A96 50
2275 0A97 E8 0ABA R
2276 0A9A 72 14
2277 0A9C B4 08
2278 0A9E E8 09F0 R
2279 0AA1 E8 0AE2 R
2280 0AA4 72 0A
2281 0AA6 A0 0042 R
2282 0AA9 24 60
2283 0AAB 3C 60
2284 0AAD 74 03
2285 0AAF F8
2286 0AB0
2287 0AB0 58
2288 0AB1
2289 0AB1 C3
2290
2291 0AB2
2292 0AB2 80 0E 0041 R 40
2293 0AB7 F9
2294 0AB8 EB F6
2295 0ABA
2296
2297
2298
2299
2300
2301
2302
2303
2304 0ABA
2305 0ABA FB
2306 0ABB F8
2307 0ABC B8 9001
2308 0ABF CD 15
2309 0AC1 72 11
2310
2311 0AC3 B3 04
2312 0AC5 33 C9
2313 0AC7
2314 0ACT F6 06 003E R 80
2315 0ACC 75 0C
2316 0ACE E2 F7
2317 0AD0 FE CB
2318 0AD2 75 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 0AE0 9D
2326 0AE1 C3
2327 0AE2
2328
2329
2330
2331
2332
2333
2334
2335
2336 0AE2 57
2337 0AE2 57
2338 0AE3 BF 0042 R
2339 0AE6 B3 07
2340 0AE8 BA 03F4
2341

```

```

;----- WAIT FOR HEAD SETTLE
DO_WAIT:
    PUSHF                ; SAVE STATUS
    CALL HD_WAIT         ; WAIT FOR HEAD SETTLE TIME
    POPF                 ; RESTORE STATUS
RB:
    POP AX               ; CLEAR ERROR RETURN FROM NEC_OUTPUT
NEC_ERR:
    RET                 ; RETURN TO CALLER
SEEK
    ENDP
;-----
; RECAL
; 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                ; RECALIBRATE COMMAND
    MOV NEC_OUTPUT
    CALL BX,DI             ; BX = DRIVE #
    MOV AH,BL
    CALL NEC_OUTPUT        ; OUTPUT THE DRIVE NUMBER
    CALL CHK_STAT_2        ; GET THE INTERRUPT AND SENSE INT STATUS
    POP AX                 ; THROW AWAY ERROR
RC_BACK:
    POP CX
    RET
RECAL
    ENDP
;-----
; CHK_STAT_2
; THIS ROUTINE HANDLES THE INTERRUPT RECEIVED AFTER
; RECALIBRATE, SEEK, OR RESET TO THE ADAPTER. THE
; INTERRUPT IS WAITED FOR, THE INTERRUPT STATUS SENSED,
; AND THE RESULT RETURNED TO THE CALLER.
; ON EXIT:   *DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION.
;-----
CHK_STAT_2
    PROC NEAR
    MOV AX,OFFSET CS_BACK ; LOAD NEC_OUTPUT ERROR ADDRESS
    PUSH AX
    CALL WAIT_INT         ; WAIT FOR THE INTERRUPT
    JC J34                ; IF ERROR, RETURN IT
    MOV AH,0BH           ; SENSE INTERRUPT STATUS COMMAND
    CALL NEC_OUTPUT
    CALL RESULTS          ; READ IN THE RESULTS
    JC J34
    MOV AL,*NEC_STATUS    ; GET THE FIRST STATUS BYTE
    AND AL,0110000B      ; ISOLATE THE BITS
    CMP AL,0110000B      ; TEST FOR CORRECT VALUE
    JZ J35                ; IF ERROR, GO MARK IT
    CLC                   ; GOOD RETURN
J34:
    POP AX                ; THROW AWAY ERROR RETURN
CS_BACK:
    RET
J35:
    OR *DSKETTE_STATUS,BAD_SEEK ; ERROR RETURN CODE
    STC
    SHORT J34
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                   ; TURN ON INTERRUPTS, JUST IN CASE
    CLC                   ; CLEAR TIMEOUT INDICATOR
    MOV AX,09001H         ; LOAD WAIT CODE AND TYPE
    INT J36A              ; PERFORM OTHER FUNCTION
    JC J36A               ; BYPASS TIMING LOOP IF TIMEDOUT DONE
    MOV BL,4              ; CLEAR THE COUNTERS
    XOR CX,CX             ; FOR 2 SECOND WAIT
J36:
    TEST *SEEK_STATUS,INT_FLAG ; TEST FOR INTERRUPT OCCURRING
    JNZ J37
    LOOP J36              ; COUNT DOWN WHILE WAITING
    DEC BL                ; SECOND LEVEL COUNTER
    JNZ J36
J36A:
    OR *DSKETTE_STATUS,TIME_OUT ; NOTHING HAPPENED
    STC                   ; ERROR RETURN
J37:
    PUSHF                ; SAVE CURRENT CARRY
    AND *SEEK_STATUS,NOT_INT_FLAG ; TURN OFF INTERRUPT FLAG
    POPF                 ; RECOVER CARRY
    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
    DI
    PUSH DI
    MOV DI,OFFSET *NEC_STATUS ; POINTER TO DATA AREA
    MOV BL,7                 ; MAX STATUS BYTES
    MOV DX,03F4H             ; STATUS PORT

```

SECTION 5

```

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 0AEE                J39:                ; WAIT FOR MASTER
2347 0AEF EC            IN   AL,DX           ; GET STATUS
2348 0AF0 24 C0          AND  AL,11000000B      ; KEEP ONLY STATUS AND DIRECTION
2349 0AF2 3C C0          CMP  AL,11000000B      ; STATUS 1 AND DIRECTION 0 ?
2350 0AF4 74 0E          JZ   J42           ; STATUS AND DIRECTION OK
2351 0AF6 E2 F7          LOOP J39           ; LOOP TILL TIMEOUT
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                ; 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:                ;
2363 0B04 42            INC  DX           ; POINT AT DATA PORT
2364 0B05 EC            IN   AL,DX           ; GET THE DATA
2365 0B06 88 05          MOV  [DI],AL       ; STORE THE BYTE
2366 0B08 47            INC  DI           ; INCREMENT THE POINTER
2367
2368 0B09 B9 0002        MOV  CX,2           ; MINIMUM 12 MICROSECONDS FOR NEC
2369 0B0C E8 0000 E      CALL WAITF          ; WAIT 15 TO 30 MICROSECONDS
2370 0B0F 41 A          DEC  DX           ; POINT AT STATUS PORT
2371 0B10 EC            IN   AL,DX           ; GET STATUS
2372 0B11 A8 10          TEST AL,00010000B  ; 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                ; SET ERROR FLAG
2379
2380          I----- RESULT OPERATION IS DONE
2381
2382 0B1F          POPRES:          POP  DI           ; RETURN WITH CARRY SET
2383 0B1F 5F          RET
2384 0B20 C3          RESULTS ENDP
2385 0B21
2386          I-----
2387 0B21          READ_DSKCHNG  PROC NEAR
2388          I READS THE STATE OF THE DISK CHANGE LINE.
2389          I
2390          I ON ENTRY:  DI = DRIVE #
2391          I
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          I-----
2397 0B21          READ_DSKCHNG  PROC NEAR
2398 0B21 E8 0913 R      CALL MOTOR_ON      ; TURN ON THE MOTOR IF OFF
2399 0B24 BA 03F7 H      MOV  DX,03F7H      ; ADDRESS DIGITAL INPUT REGISTER
2400 0B27 EC            IN   DX             ; INPUT DIGITAL INPUT REGISTER
2401 0B28 AB 80          TEST AL,DSK_CHG   ; CHECK FOR DISK CHANGE LINE ACTIVE
2402 0B2A C3          RET               ; RETURN TO CALLER WITH ZERO FLAG SET
2403 0B2B          READ_DSKCHNG  ENDP
2404          I-----
2405          I DRIVE_DET
2406          I DETERMINES WHETHER DRIVE IS 80 OR 40 TRACKS AND
2407          I UPDATES STATE INFORMATION ACCORDINGLY.
2408          I
2409          I ON ENTRY:  DI = DRIVE #
2410          I-----
2411 0B2B          DRIVE_DET  PROC NEAR
2412 0B2B E8 0913 R      CALL MOTOR_ON      ; TURN ON MOTOR IF NOT ALREADY ON
2413 0B2E E8 0A7C R      CALL RECAL         ; RECALIBRATE DRIVE
2414 0B31 72 3E          JC   DD_BAC        ; ASSUME NO DRIVE PRESENT
2415 0B33 B5 30          MOV  CH,TRK_SLAP  ; SEEK TO TRACK 48
2416 0B35 E8 0A14 R      CALL SEEK          ;
2417 0B38 72 37          JC   DD_BAC        ; ERROR NO DRIVE
2418 0B3A B5 0B          MOV  CH,QUIET_SEEK+1 ; SEEK TO TRACK 10
2419 0B3C          SK_GIN:
2420 0B3C FE CD          DEC  CH           ; DECREMENT TO NEXT TRACK
2421 0B3E 78 26          JS   IS_40        ; END LOOP IF CYLINDER COUNT NEGATIVE
2422 0B40 51            PUSH CX           ; SAVE TRACK
2423 0B41 E8 0A14 R      CALL SEEK          ;
2424 0B44 72 2C          JC   POP_BAC      ; POP AND RETURN
2425 0B46 B8 0B71 R      MOV  AX,OFFSET DD_BAC ; LOAD NEC OUTPUT ERROR ADDRESS
2426 0B49 50            PUSH AX           ;
2427 0B4A B4 04          MOV  AH,SENSE_DRV_ST ; SENSE DRIVE STATUS COMMAND BYTE
2428 0B4C E8 09F0 R      CALL NEC_OUTPUT    ; OUTPUT TO NEC
2429 0B4F 8B C7          MOV  AX,DI         ; AL = DRIVE
2430 0B51 8A E0          MOV  AH,AL         ; AH = DRIVE
2431 0B53 E8 09F0 R      CALL NEC_OUTPUT    ; OUTPUT TO NEC
2432 0B56 E8 0AEE R      CALL RESULTS       ; GO GET STATUS
2433 0B59 58            POP  AX            ; THROW AWAY ERROR ADDRESS
2434 0B5A 59            POP  CX            ; RESTORE TRACK
2435 0B5B F6 06 0042 R 10 TEST #NEC_STATUS,HOME ; TRACK 0 ?
2436 0B60 74 DA          JZ   SK_GIN        ; GO TILL 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
2441          I DRIVE IS A 360; SET DRIVE TO DETERMINED;
2442          I SET MEDIA TO DETERMINED AT RATE 250.
2443          I
2444          I IS_40:
2445          OR   #DSK_STATE[DI],DRV_DET+MED_DET+RATE_250
2446          RET     ; ALL INFORMATION SET
2447          I
2448          I IS_80:
2449          OR   #DSK_STATE[DI],TRK_CAPA ; SETUP 80 TRACK CAPABILITY
2450          RET
2451          I
2452          I POP_BAC:
2453          POP  CX           ; THROW AWAY
2454          RET
2455 0B74          DRIVE_DET  ENDP
    
```



```

2456                                     ;-----;
2457 ; DISK_INT                               ;
2458 ; THIS ROUTINE HANDLES THE DISKETTE INTERRUPT. ;
2459 ;
2460 ; ON EXIT: THE INTERRUPT FLAG IS SET IN #SEEK STATUS. ;
2461 ;-----;
2462 0B74 DISK_INT PROC FAR ; ENTRY POINT FOR ORG 0EF57H
2463 0B74 FB STI ; RE-ENABLE INTERRUPTS
2464 0B75 50 PUSH AX ; SAVE WORK REGISTER
2465 0B76 1E PUSH DS ; SAVE REGISTERS
2466 0B77 EB 0000 E CALL DDS ; SETUP DATA ADDRESSING
2467 0B7A 80 0E 003E R 80 OR #SEEK_STATUS,INT_FLAG ; TURN ON INTERRUPT OCCURRED
2468 0B7F 1F POP DS ; RESTORE USER (DS)
2469 0B80 B0 20 MOV AL,EOI ; END OF INTERRUPT MARKER
2470 0B82 E6 20 OUT INTA0,AL ; INTERRUPT CONTROL PORT
2471 0B84 B8 9101 MOV AX,09101H ; INTERRUPT POST CODE AND TYPE
2472 0B87 CD 15 INT ISH ; GO PERFORM OTHER TASK
2473 0B89 58 POP AX ; RECOVER REGISTER
2474 0B8A CF IRET ; RETURN FROM INTERRUPT
2475 0B8B
2476 DISK_INT ENDP
2477 ;-----;
2478 ; DSKETTE_SETUP
2479 ; THIS ROUTINE DOES A PRELIMINARY CHECK TO SEE WHAT TYPE ;
2480 ; OF DISKETTE DRIVES ARE ATTACHED TO THE SYSTEM. ;
2481 ;-----;
2481 0B8B DSKETTE_SETUP PROC NEAR
2482 0B8B 50 PUSH AX ; SAVE REGISTERS
2483 0B8C 53 PUSH BX
2484 0B8D 51 PUSH CX
2485 0B8E 52 PUSH DX
2486 0B8F 57 PUSH DI
2487 0B90 56 PUSH SI
2488 0B91 1E PUSH DS
2489 0B92 EB 0000 E CALL DDS ; POINT DATA SEGMENT TO BIOS DATA AREA
2490 0B95 80 0E 00A0 R 01 OR #RTC_WAIT_FLAG,0 ; NO RTC WAIT, FORCE USE OF LOOP
2491 0B9A C7 06 0090 R 0000 MOV WORD_PTR #DSK_STATE,0 ; INITIALIZE STATES
2492 0BA0 B0 26 008B R 33 AND #LAstrate,NOT_STRT_MSK+SEND_MSK ; CLEAR START & SEND
2493 0BA5 80 0E 008B R C0 OR #LAstrate,SEND_MSK ; INITIALIZE SENT TO IMPOSSIBLE
2494 0BA8 A6 C6 06 003E R 00 MOV #SEEK_STATUS,0 ; INDICATE RECALIBRATE NEEDED
2495 0BAF C6 C6 0040 R 00 MOV #MOTOR_COUNT,0 ; INITIALIZE MOTOR COUNT
2496 0BB4 C6 C6 003F R 00 MOV #MOTOR_STATUS,0 ; INITIALIZE DRIVES TO OFF STATE
2497 0BB9 C6 C6 0041 R 00 MOV #DSKETTE_STATUS,0 ; NO ERRORS
2498 0BBE A0 0010 R MOV AL,BYTE_PTR #EQUIP_FLAG ; GET EQUIPMENT STATUS
2499 0BC1 D0 C0 ROL AL,I ; SHIFT BITS 7,6 TO 1,0
2500 0BC3 D0 C0 ROL AL,I
2501 0BC5 24 03 AND AL,3 ; MASK DRIVE BITS
2502 0BC7 32 E4 XOR AH,AH ; AX=NUMBER OF DRIVES(RELATIVE ZERO)
2503 0BC9 33 FF XOR DI,DI ; DI=INITIAL DRIVE TO BE ESTABLISHED
2504 0BCB BE 0010 MOV SI,HOME ; SI=HOME MASK FOR ALL DRIVES
2505 0BCE
2506 0BCE F6 06 008F R 01 SUP0: TEST #HF_CNTRL,DUAL ; TEST CONTROLLER TYPE
2507 0BD3 75 05 JNZ SUPT
2508 0BD5 C6 85 0090 R 94 MOV #DSK_STATE[DI],DRV_DET+MED_DET+RATE_250
2509 0BDA SUP1:
2510 0BDA 50 PUSH AX ; SAVE DRIVE COUNT
2511 0BD0 EB 0B2B R CALL DRIVE_DET ; DETERMINE DRIVE
2512 0BDE EB 0432 R CALL XLAT_GLD ; TRANSLATE STATE TO COMPATIBLE MODE
2513 0BE1 23 36 0042 R AND SI,WORD_PTR #NEC_STATUS ; AND #NEC STATUS WITH HOME MASK
2514 0BE5 58 POP AX ; RESTORE DRIVE COUNT
2515 0BE6 47 INC DI ; POINT TO NEXT DRIVE
2516 0BE7 3B F8 CMP DI,AX
2517 0BE9 76 E3 JNA SUP0 ; REPEAT FOR EACH DRIVE
2518 0BEB SUP2:
2519 0BEB C6 06 003E R 00 MOV #SEEK_STATUS,0 ; FORCE RECALIBRATE
2520 0BF0 B0 26 00A0 R FE AND #RTC_WAIT_FLAG,0FEH ; ALLOW FOR RTC WAIT
2521 0BF5 EB 0832 R CALL SETUP_END ; VARIOUS CLEANUPS
2522 0BF8 72 05 JC HOME_OK ; EXIT WITH CY FLAG FROM SETUP_END
2523 0BFA 0B F6 OR SI,ST ; TEST HOME INDICATORS FOR ALL DRIVES
2524 0BFC 75 01 JNZ HOME_OK
2525 0BFE F9 STC ; ERROR-->HOME INDICATOR BAD
2526 0BFF HOME_OK:
2527 0BFF 1F POP DS ; RESTORE CALLERS REGISTERS
2528 0C00 5E POP SI
2529 0C01 5F POP DI
2530 0C02 5A POP DX
2531 0C03 59 POP CX
2532 0C04 5B POP BX
2533 0C05 58 POP AX
2534 0C06 C3 RET
2535 0C07 DSKETTE_SETUP ENDP
2536 0C07 CODE ENDS
2537 2537 END
    
```

```

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

```

INPUT

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

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

OUTPUT

AS NOTED ABOVE, ONLY AX AND FLAGS CHANGED
 ALL OTHER REGISTERS PRESERVED

```

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

PUBLIC KEYBOARD_IO_1
PUBLIC KB_INT_1

.LIST

CODE SEGMENT BYTE PUBLIC
ASSUME CS:CODE,DS:DATA
KEYBOARD_IO_1 PROC FAR
;
; INTERRUPTS BACK ON
; SAVE CURRENT DS
; SAVE BX TEMPORARILY
; ESTABLISH POINTER TO DATA REGION
; ASCII_READ
; AH=1
; ASCII_STATUS
; AH=2
; SHIFT_STATUS
; AH=3
; AH=5
; KEYBOARD_WRITE
; AH=10
; EXTENDED_ASCII_READ
; AH=11
; EXTENDED_ASCII_STATUS
; AH=12
; EXTENDED_SHIFT_STATUS
;
; KIO_EXIT:
; POP CX
; POP BX
; POP DS
; RECOVER REGISTER
; RECOVER SEGMENT
; INVALID COMMAND
;
;----- ASCII CHARACTER
KIE: CALL KIS
CALL KIO_E_XLAT
JMP KIO_EXIT
; GET A CHARACTER FROM THE BUFFER (EXTENDED)
; ROUTINE TO XLATE FOR EXTENDED CALLS
; GIVE IT TO THE CALLER
;
;
K1: CALL KIS
CALL KIO_S_XLAT
JC K1
JMP KIO_EXIT
; GET A CHARACTER FROM THE BUFFER
; ROUTINE TO XLATE FOR STANDARD CALLS
; CARRY SET MEANS THROW CODE AWAY

```

```

115                                     |----- ASCII STATUS
116
117 003B EB 00C4 R                       K2E: CALL K25           | TEST FOR CHARACTER IN BUFFER (EXTENDED)
118 003E 74 18                             JZ K2B             | RETURN IF BUFFER EMPTY
119 0040 9C                               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 74 0D                             JZ K2B           | RETURN IF BUFFER EMPTY
125 004B 9C                               PUSHF            | SAVE ZF FROM TEST
126 004C EB 00DC R                       CALL K10_5_XLAT | ROUTINE TO XLATE FOR STANDARD CALLS
127 004F 73 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                               POPF            | RESTORE ZF FROM TEST
133 0058 59                               POP             |
134 0059 5B                               POP             | RECOVER REGISTER
135 005A 1F                               POP             | RECOVER SEGMENT
136 005B CA 0002                          RET 2           | THROW AWAY FLAGS
137
138                                     |----- SHIFT STATUS
139
140 005E                                     K3E:
141 005E 8A 26 0018 R                     MOV AH,*KB_FLAG | GET THE EXTENDED SHIFT STATUS FLAGS
142 0062 80 E4 04                         AND AH,SYS_SHIFT| GET SYSTEM SHIFT KEY STATUS
143 0065 B1 05                             MOV CL,5        | MASK ALL BUT SYS KEY BIT
144 0067 02 E4                             SHL AH,CL      | SHIFT THE SYSTEM KEY BIT OVER TO
145 0069 A0 0018 R                       MOV AL,*KB_FLAG | BIT 7 POSITION
146 006C 24 73                             AND AL,01110011| GET SHIFT STATES BACK
147 006E 0A E0                             OR AH,AL       | ELIMINATE SYS_SHIFT, HOLD_STATE, AND INS_SHIFT
148 0070 A0 0096 R                       MOV AL,*KB_FLAG | MERGE THE REMAINING BITS INTO AH
149 0073 24 0C                             AND AL,00001100| GET RIGHT CTL AND ALT
150 0075 0A E0                             OR AH,AL       | ELIMINATE LC ED AND LC E1
151 0077 A0 0017 R                       MOV AL,*KB_FLAG | OR THE SHIFT_FLAGS TOGETHER
152 007A EA A9                             JMP K10_EXIT    | GET THE SHIFT STATUS FLAGS
153                                         | RETURN TO CALLER
154
155                                     |----- WRITE TO KEYBOARD BUFFER
156
157 007C 56                               PUSH SI
158 007D FA                               CLI
159 007E BB 1E 001C R                     MOV SI,BX       | GET THE "IN TO" POINTER TO THE BUFFER
160 0082 BB F3                             MOV SI,BX       | SAVE A COPY IN CASE BUFFER NOT FULL
161 0084 EB 0114 R                       CALL K4         | BUMP THE POINTER TO SEE IF BUFFER IS FULL
162 0087 3B 1E 001A R                     CMP K502        | WILL THE BUFFER OVERRUN IF WE STORE THIS?
163 008B 74 0B                             JE K502         | YES - INFORM CALLER OF ERROR
164 008D 89 0C                             MOV [SI],CX    | NO - PUT THE ASCII/SCAN CODE INTO BUFFER
165 008F 89 1E 001C R                     MOV [SI],CX    | ADJUST "IN TO" POINTER TO REFLECT CHANGE
166 0093 2A C0                             SUB AL,AL      | TELL CALLER THAT OPERATION WAS SUCCESSFUL
167 0095 EB 03 90                             JMP K504       | SUB INSTRUCTION ALSO RESETS CARRY FLAG
168 0098
169 0098 B0 01                             MOV AL,01H    | BUFFER FULL INDICATION
170 009A
171 009A FB                             STI
172 009B 5E                             POP SI
173 009C EB 87                             JMP K10_EXIT   | RETURN TO CALLER WITH STATUS IN AL
174
175 009E                                     KEYBOARD_IO_1 ENDP

```

```

176 PAGE
177 |----- READ THE KEY TO FIGURE OUT WHAT TO DO -----
178
179 009E K15 PROC NEAR
180 009E 9B 1E 001A R MOV BX,®BUFFER_HEAD ; GET POINTER TO HEAD OF BUFFER
181 00A2 3B 1E 001C R CMP BX,®BUFFER_TAIL ; TEST END OF BUFFER
182 00A6 75 05 JNE KIT ; IF ANYTHING IN BUFFER DONT DO INTERRUPT
183
184 00A8 8B 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 00AE 90 NOP ; ALLOW AN INTERRUPT TO OCCUR
189 00AF FA CLI ; INTERRUPTS BACK OFF
190 00B0 8B 1E 001A R MOV BX,®BUFFER_HEAD ; GET POINTER TO HEAD OF BUFFER
191 00B4 3B 1E 001C R CMP BX,®BUFFER_TAIL ; TEST END OF BUFFER
192 00B8 74 F3 JECXZ KIT ; LOOP UNTIL SOMETHING IN BUFFER
193 00BA 8B 07 MOV AX,[BX] ; GET SCAN CODE AND ASCII CODE
194 00BC EB 0114 R CALL K4 ; MOVE POINTER TO NEXT POSITION
195 00BF 89 1E 001A R MOV ®BUFFER_HEAD,BX ; STORE VALUE IN VARIABLE
196 00C3 C3 RET ; RETURN
197 00C4 K15 ENDP
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 8B 1E 001A R MOV BX,®BUFFER_HEAD ; GET HEAD POINTER
205 00C9 3B 1E 001C R CMP BX,®BUFFER_TAIL ; IF EQUAL (Z=1) THEN NOTHING THERE
206 00CD 8B 07 MOV AX,[BX]
207 00CF FB STI ; INTERRUPTS BACK ON
208 00D0 C3 RET ; RETURN
209 00D1 K2S ENDP
210
211
212 |----- ROUTINE TO TRANSLATE SCAN CODE PAIRS FOR EXTENDED CALLS
213
214 00D1 K10_E_XLAT:
215 00D1 3C F0 CMP AL,0F0h ; IS IT ONE OF THE FILL-INs?
216 00D3 75 06 JNE K10_E_RET ; NO, PASS IT ON
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 00D9 32 C0 XOR AL,AL ; OTHERWISE SET AL = 0
220 00DB K10_E_RET:
221 00DB C3 RET ; GO BACK
222
223
224 |----- ROUTINE TO TRANSLATE SCAN CODE PAIRS FOR STANDARD CALLS
225
226 00DC K10_S_XLAT:
227 00DC 80 FC E0 CMP AH,0E0h ; IS IT KEYPAD ENTER OR / ?
228 00DF 75 12 JNE K10_S2 ; NO, CONTINUE
229 00E1 3C 0D CMP AL,0Dh ; KEYPAD ENTER CODE?
230 00E3 74 09 JE K10_S1 ; YES, MESSAGE A BIT
231 00E5 3C 0A CMP AL,0Ah ; CTRL KEYPAD ENTER CODE?
232 00E7 74 05 JE K10_S1 ; YES, MESSAGE THE SAME
233 00E9 84 35 JECXZ MOV AH,35h ; NO, MUST BE KEYPAD /
234 00EB EB 23 90 JMP K10_USE ; GIVE TO CALLER
235 00EE 84 1C CMP AH,Tch ; CONVERT TO COMPATIBLE OUTPUT
236 00F0 EB 1E 90 JMP K10_USE ; GIVE TO CALLER
237
238 00F3 80 FC 84 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 00FC 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 75 09 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 JZ K10_USE ; JUMP IF AH = 0
251 010B 3C 0D CMP AL,0Dh ; CONVERT TO COMPATIBLE OUTPUT
252 010D EB 01 90 JMP K10_USE ; PASS IT ON TO CALLER
253
254 0110 K10_USE:
255 0110 F8 CLC ; CLEAR CARRY TO INDICATE GOOD CODE
256 0111 C3 RET ; RETURN
257 0112 K10_DIS:
258 0112 F9 STC ; SET CARRY TO INDICATE DISCARD CODE
259 0113 C3 RET ; RETURN

```

```

260
261
262
263
264
265 0114
266 0114 43
267 0115 43
268
269 0116 3B IE 0082 R
270 011A 72 04
271 011C 8B IE 0080 R
272 0120 C3
273 0121
274
275
276
277
278 0121
279 0121 50
280 0122 53
281 0123 51
282 0124 52
283 0125 56
284 0126 57
285 0127 IE
286 0128 06
287 0129 FC
288 012A E8 0000 E
289 012D E4 60
290 012F 93
291
292
293
294
295 0130 E4 61
296 0132 8A E0
297 0134 0C 80
298 0136 E6 61
299 0138 86 E0
300 013A E6 61
301 013C FB
302 013D 93
303
304
305
306 013E B4 4F
307 0140 F9
308 0141 CD 15
309
310 0143 72 03
311 0145 E9 02CA R
312
313 0148
314 0148 8A E0
315
316
317
318 014A 3C FF
319 014C 75 03
320 014E E9 0540 R
321
322 0151 0E
323 0152 07
324 0153 8A 3E 0096 R
325
326 0157
327 0157 3C E0
328 0159 75 07
329 015B 80 0E 0096 R 12
330 0160 EB 09
331
332 0162
333 0162 3C E1
334 0164 75 08
335 0166 80 0E 0096 R 11
336 016B E9 02CF R
337
338 016E
339 016E 24 7F
340 0170 F6 C7 02
341 0173 74 0C
342
343 0175 B9 0002
344 0178 BF 0558 R
345 017B F2/ AE
346 017D 75 54
347 017F EB 3D
348
349 0181
350 0181 F6 C7 01
351 0184 74 16
352
353 0186 B9 0004
354 0189 BF 0553 R
355 018C F2/ AE
356 018E 74 DB
357
358 0190 3C 45
359 0192 75 2A
360 0194 F6 C4 80
361 0197 75 25
362 0199 E9 03FF R

```

PAGE

INCREMENT BUFFER POINTER ROUTINE

```

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

```

----- KEYBOARD INTERRUPT ROUTINE

```

KB_INT_1 PROC FAR
PUSH AX ; SAVE THE STI UNTIL AFTER KEYBOARD RESET
PUSH BX
PUSH CX
PUSH DX
PUSH SI
PUSH DI
PUSH DS
PUSH ES
CLD ; FORWARD DIRECTION
CALL DD5 ; SET UP ADDRESSING TO DATA SEGMENT
IN AL,KB_DATA ; READ IN THE CHARACTER
XCHG BX,AX ; SAVE IT

```

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

```

IN AL,KB_CTL ; GET THE CONTROL PORT
MOV AH,AL ; SAVE VALUE
OR AL,80H ; RESET BIT FOR KEYBOARD
OUT KB_CTL,AL
XCHG AH,AL ; GET BACK ORIGINAL CONTROL
OUT KB_CTL,AL ; KB HAS BEEN RESET
STI
XCHG AX,BX ; RESTORE DATA IN

```

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

```

MOV AH,04FH ; SYSTEM INTERCEPT - KEY CODE FUNCTION
STC
INT 15H ; CASSETTE CALL (AL)= KEY SCAN CODE
; RETURNS CY=1 FOR INVALID FUNCTION
JC KB_INT_PC ; CONTINUE IF CARRY FLAG SET ((AL)=CODE)
JMP K26 ; EXIT IF SYSTEM HANDLED SCAN CODE
; EXIT HANDLES HARDWARE EOI AND ENABLE

```

KB_INT_PC:

```

MOV AH,AL ; SAVE SCAN CODE IN AH ALSO

```

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

```

CMP AL,OFFH ; IS THIS AN OVERRUN CHAR
JNZ K16 ; WAS LAST CODE THE E0 MARKER CODE?
JMP K62 ; BUFFER_FULL_BEEP

```

K16: PROC NEAR

```

PUSH CS ; ESTABLISH ADDRESS OF TABLES
POP ES ; LOAD FLAG FOR TESTING
MOV BH,#KB_FLAG_3

```

TEST_E0:

```

CMP AL,MC_E0 ; IS THIS THE GENERAL MARKER CODE?
JNE TEST_E1 ; CHECK IT
OR #KB_FLAG_3,LC_E0+KBX ; SET FLAG BIT, SET KBX, AND
JMP SHORT EXIT_K ; THROW AWAY THIS CODE

```

TEST_E1:

```

CMP AL,MC_E1 ; IS THIS THE PAUSE KEY?
JNE NOT_HC ; CHECK IT
OR #KB_FLAG_3,LC_E1+KBX ; SET FLAG, PAUSE KEY MARKER CODE
JMP SHORT EXIT_K ; THROW AWAY THIS CODE

```

EXIT_K: PROC NEAR

```

JMP K26A

```

NOT_HC:

```

AND AL,07FH ; TURN OFF THE BREAK BIT
TEST BH,LC_E0 ; WAS LAST CODE THE E0 MARKER CODE?
JZ NOT_LC_E0 ; JUMP IF NOT

```

MOV CX,2 ; LENGTH OF SEARCH
MOV DI,OFFSET K6+6 ; IS THIS A SHIFT KEY?
SCASB ; CHECK IT
JNE K16A ; NO, CONTINUE KEY PROCESSING
JMP SHORT K16B ; YES, THROW AWAY & RESET FLAG

NOT_LC_E0:

```

TEST BH,LC_E1 ; WAS LAST CODE THE E1 MARKER CODE?
JZ T_SYS_KEY ; JUMP IF NOT

```

MOV CX,4 ; LENGTH OF SEARCH
MOV DI,OFFSET K6+4 ; IS THIS AN ALT, CTL, OR SHIFT?
SCASB ; CHECK IT
JNE EXIT_K ; THROW AWAY IF SO

CMP AL,NUM_KEY ; IS IT THE PAUSE KEY?
JNE K16B ; NO, THROW AWAY & RESET FLAG
TEST AH,80H ; YES, IS IT THE BREAK OF THE KEY?
JNZ K16B ; YES, THROW THIS AWAY, TOO
JMP K39P ; NO, THIS IS THE REAL PAUSE STATE

```

363                PAGE
364                I----- TEST FOR SYSTEM KEY
365
366 019C            T_SYS_KEY:
367 019C 3C 54      CMP     AL,SYS_KEY
368 019E 75 33      JNE     K16A
369
370 01A0 F6 C4 80   TEST    AH,080H
371 01A3 75 1C      JNZ     K16C
372
373 01A5 F6 06 0018 R 04 TEST    *KB_FLAG_1,SYS_SHIFT
374 01AA 75 12      JNZ     K16B
375
376 01AC 80 0E 0018 R 04 OR      *KB_FLAG_1,SYS_SHIFT
377 01B1 B0 20      MOV     AL,E01
378 01B3 E6 20      OUT    020H,AL
379
380 01B5 B8 8500    MOV     AX,08500H
381 01B8 FB        STI
382 01B9 CD 15     INT    15H
383 01BB E9 02D4 R JMP     K27
384
385 01BE E9 02CA R K16B: JMP     K26
386
387 01C1 80 26 0018 R FB K16C: AND    *KB_FLAG_1,NOT SYS_SHIFT;
388 01C6 B0 20      MOV     AL,E01;
389 01C8 E6 20      OUT    020H,AL;
390
391 01CA B8 8501    MOV     AX,08501H;
392 01CD FB        STI;
393 01CE CD 15     INT    15H;
394 01D0 E9 02D4 R JMP     K27;
395
396 I----- TEST FOR SHIFT KEYS
397
398 01D3 8A 1E 0017 R K16A: MOV     BL,*KB_FLAG
399 01D7 BF 064F R   MOV     D1,OFFSET K6
400 01DA B9 0008 90 MOV     CX,K6+1
401 01DE F2/ AE     REPNE SCASB
402 01E0 8A 04     MOV     AL,AH
403 01E3 74 03     JE      K17
404 01E4 E9 02B6 R JMP     K25
405
406 I----- SHIFT KEY FOUND
407
408 01E7 81 EF 0550 R K17: SUB     D1,OFFSET K6+1
409 01EB 2E1 8A 5 0557 R MOV     AH,CS:K7[D1]
410 01F0 B1 02     MOV     CL,2
411 01F2 A8 80     TEST    AL,80H
412 01F4 74 03     JZ      K17C
413 01F6 EB 6E 90 JMP     K23
414
415 I----- 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 I----- FLAIN SHIFT KEY, SET SHIFT ON
421
422 01FE 08 26 0017 R OR      *KB_FLAG,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 020A F6 C7 02 K17D: TEST    BH,LC_E0
427 020D 74 07     JZ      K17E
428 020F 08 26 0096 R OR      *KB_FLAG_3,AH
429 0213 E9 02CA R JMP     K26
430 0216 D2 EC     K17E: SHR     AH,CL
431 0218 08 26 0018 R OR      *KB_FLAG_1,AH
432 021C E9 02CA R JMP     K26
433
434 I----- TOGGLED SHIFT KEY, TEST FOR 1ST MAKE OR NOT
435
436 021F K18:
437 021F F6 C3 04 TEST    BL,CTL_SHIFT
438 0222 74 03     JZ      K18A
439 0224 E9 02B6 R JMP     K25
440 0227 3C 52     CMP     AL,INS_KEY
441 0229 75 21     JNE     K22
442 022B F6 C3 08 TEST    BL,ALT_SHIFT
443 022E 74 03     JZ      K18B
444 0230 E9 02B6 R JMP     K25
445 0233 F6 C7 02 K18B: TEST    BH,LC_E0
446 0236 75 14     JNZ     K22
447 0238 F6 C3 20 K19: TEST    BL,NUM_STATE
448 023B 75 0A     JNZ     K21
449 023D F6 C3 03 K19: TEST    BL,LEFT_SHIFT+RIGHT_SHIFT
450 0240 74 0A     JZ      K22
451 0242
452 0242 8A E0     MOV     AH,AL
453 0244 EB 70 90 JMP     K25
454
455 0247 F6 C3 03 K21: TEST    BL,LEFT_SHIFT+RIGHT_SHIFT
456 024A 74 F6     JZ      K20
457
458 024C K22:
459 024C 84 26 0018 R TEST    AH,*KB_FLAG_1
460 0250 74 03     JZ      K22A
461 0252 EB 76 90 JMP     K26
462 0255 08 26 0018 R K22A: OR      *KB_FLAG_1,AH
463 0259 30 26 0017 R XOR     *KB_FLAG,AH
464 025D 3C 52     CMP     AL,INS_KEY
465 025F 75 69     JNE     K26
466 0261 8A E0     MOV     AH,AL
467 0263 EB 78 90 JMP     K26
468
469 I----- BREAK SHIFT FOUND
470
471 0266 K23:
472 0266 80 FC 10 CMP     AH,SCROLL_SHIFT
473 0269 F6 D4     NOT    AH
474 026B 73 43     JAE     K24
475 026D 20 26 0017 R AND    *KB_FLAG,AH
476 0271 80 FC 10 CMP     AH,NOT CTL_SHIFT

```

```

477 0274 77 26          JA      K23D          ; NO, ALL DONE
478
479 0276 F6 C7 02      TEST     BH,LC_E0          ; 2ND ALT OR CTL?
480 0279 74 06          JZ      K23A          ; NO, HANDLE NORMALLY
481 027B 20 26 0096 R  AND     *KB_FLAG_3,AH    ; RESET BIT FOR RIGHT ALT OR CTL
482 027F EB 06          JMP     SHORT K23B       ; CONTINUE
483 0281 D2 FC          SAR     AH,CL           ; MOVE THE MASK BIT TWO POSITIONS
484 0283 20 26 0018 R  AND     *KB_FLAG_1,AH    ; RESET BIT FOR LEFT ALT OR CTL
485 0287 8A E0          MOV     AH,AL          ; SAVE SCAN CODE
486 0289 AD 0096 R     MOV     AL,*KB_FLAG_3   ; GET RIGHT ALT & CTRL FLAGS
487 028C D2 E8          SHR     AH,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+80H  ; IS THIS ALTERNATE SHIFT RELEASE
495 029E 75 2A          JNE     K26            ; INTERRUPT_RETURN
496
497 ;----- ALTERNATE SHIFT KEY RELEASED, GET THE VALUE INTO BUFFER
498
499 02A0 A0 0019 R     MOV     AL,*ALT_INPUT   ;
500 02A3 B4 00          MOV     AH,0           ; SCAN CODE OF 0
501 02A5 88 26 0019 R  MOV     *ALT_INPUT,AH  ; ZERO OUT THE FIELD
502 02A9 3C 00          CMP     AL,0           ; WAS THE INPUT = 0?
503 02AB 74 1D          JE      K26            ; INTERRUPT_RETURN
504 02AD E9 0519 R     JMP     K61            ; IT WASN'T, SO PUT IN BUFFER
505
506 02B0          K24:          ; BREAK-TOGGLE
507 02B2 20 26 0018 R  AND     *KB_FLAG_1,AH  ; INDICATE NO LONGER DEPRESSED
508 02B4 EB 14          JMP     SHORT K26       ; INTERRUPT_RETURN
509
510 ;----- TEST FOR HOLD STATE
511
512 02B6          K25:          ; AL, AH = SCAN CODE
513 02B8 3C 80          CMP     AL,80H        ; NO-SHIFT-FOUND
514 02BB 73 10          JAE     K26            ; TEST FOR BREAK KEY
515 02BA F6 06 0018 R 0B TEST     *KB_FLAG_1,HOLD_STATE ; NOTHING FOR BREAK CHARS FROM HERE ON
516 02BF 74 1C          JZ      K28            ; ARE WE IN HOLD STATE
517 02C1 3C 45          CMP     AL,NUM_KEY     ; BRANCH AROUND TEST IF NOT
518 02C3 74 05          JE      K26            ; CAN'T END HOLD ON NUM LOCK
519 02C5 8A 26 0018 R F7 AND     *KB_FLAG_1,NOT_HOLD_STATE ; TURN OFF THE HOLD STATE BIT
520
521 02CA          K26:          ; RESET LAST CHAR H.C. FLAG
522 02CC 8A 26 0096 R FC AND     *KB_FLAG_3,NOT_LC_E0+LC_E1
523
524 02CF          K26A:         ; INTERRUPT_RETURN
525 02CF FA          CLI     ; TURN OFF INTERRUPTS
526 02D0 B0 20          MOV     AL,E01        ; END OF INTERRUPT COMMAND
527 02D2 E6 20          OUT    020H,AL       ; SEND COMMAND TO INTERRUPT CONTROL PORT
528
529 02D4          K27:          ; INTERRUPT_RETURN-NO-E01
530 02D4 07          POP     ES            ; RESTORE REGISTERS
531 02D5 1F          POP     DS
532 02D6 5F          POP     DI
533 02D7 5E          POP     SI
534 02D8 5A          POP     DX
535 02D9 59          POP     CX
536 02DA 5B          POP     BX
537 02DB 58          POP     AX
538 02DC CF          IRET                ; RETURN, INTERRUPTS BACK ON
539                                     ; WITH FLAG CHANGE

```

```

540                PAGE
541                I----- NOT IN HOLD STATE
542
543 02DD                K28:                I AL, AH = SCAN CODE (ALL MAKES)
544 02DD 3C 58          CMP AL,88          I NO-HOLD-STATE
545 02DF 77 E9          JA K26                I TEST FOR OUT-OF-RANGE SCAN CODES
546                                     I IGNORE IF OUT-OF-RANGE
547 02E1 F6 C3 08      TEST BL,ALT_SHIFT      I ARE WE IN ALTERNATE SHIFT?
548 02E4 74 0C          JZ K28A              I JUMP IF NOT ALTERNATE
549
550 02E6 F6 C7 10      TEST BH,KBX           I IS THIS THE ENHANCED KEYBOARD?
551 02E9 74 0A          JZ K29                I NO, ALT STATE IS REAL
552
553 02EB F6 06 0018 R 04 TEST 0KB_FLAG_1,SYS_SHIFT I YES, IS SYSREQ KEY DOWN?
554 02F0 74 03          JZ K29                I NO, ALT STATE IS REAL
555 02F2 E9 03CC R     K28A: JMP K38             I YES, THIS IS PHONY ALT STATE
556                                     I DUE TO PRESSING SYSREQ
557
558 I----- TEST FOR RESET KEY SEQUENCE (CTL ALT DEL)
559
560 02F5                K29:                I TEST-RESET
561 02F5 F6 C3 04      TEST BL,CTL_SHIFT      I ARE WE IN CONTROL SHIFT ALSO?
562 02F8 74 37          JZ K31                I NO-RESET
563 02FA 3C 53          CMP AL,DEL_KEY        I SHIFT STATE IS THERE, TEST KEY
564 02FC 75 33          JNE K31              I NO-RESET, IGNORE
565
566 I----- CTL-ALT-DEL HAS BEEN FOUND, DO I/O CLEANUP
567
568 02FE C7 06 0072 R 1234H MOV 0RESET_FLAG,1234H I SET FLAG FOR RESET FUNCTION
569 0304 81 26 0096 R 0010H AND WORD PTR 0KB_FLAG_3,KBX I CLEAR ALT FLAG BITS EXCEPT KBX
570 030A E9 0000 E     JMP RESET            I JUMP TO POWER ON DIAGNOSTICS
571
572 I----- ALT-INPUT-TABLE
573 030D                K30                I LABEL BYTE
574 030D 52 4F 50 51 48 DB 82,79,80,81,75     I
575 0312 4C 4D 47 48 49 DB 76,77,71,72,73     I 10 NUMBERS ON KEYPAD
576 I----- SUPER-SHIFT-TABLE
577 0317 10 11 12 13 14 15 DB 16,17,18,19,20,21 I
578 031D 16 17 18 19 1E 1F DB 22,23,24,25,30,31 I A-Z TYPEWRITER CHARS
579 0323 20 21 22 23 24 25 DB 32,33,34,35,36,37 I
580 032B 26 27 28 2D 2E 2F 30 DB 38,44,45,46,47,48 I
581 032F 31 32          DB 49,50             I
582
583 I----- IN ALTERNATE SHIFT, RESET NOT FOUND
584
585 0331                K31:                I NO-RESET
586 0331 3C 39          CMP AL,57            I TEST FOR SPACE KEY
587 0333 75 05          JNE K311            I NOT THERE
588 0335 B0 20          MOV AL,' '          I SET SPACE CHAR
589 0337 E9 050D R     JMP K57            I BUFFER_FILL
590 033A                K311:               I TEST FOR TAB KEY
591 033A 3C 0F          CMP AL,15            I NOT THERE
592 033C 75 06          JNE K312            I SET SPECIAL CODE FOR ALT-TAB
593 033E BB A500H      MOV AX,0A500H        I BUFFER_FILL
594 0341 E9 050D R     JMP K57
595 0344                K312:               I TEST FOR KEYPAD -
596 0344 3C 4A          CMP AL,74            I GO PROCESS
597 0346 74 79          JE K37B             I TEST FOR KEYPAD +
598 0348 3C 4E          CMP AL,78            I GO PROCESS
599 034A 74 75          JE K37B
600
601 I----- LOOK FOR KEY PAD ENTRY
602
603 034C                K32:                I ALT-KEY-PAD
604 034C BF 030D R     MOV D1,OFFSET K30   I ALT-INPUT-TABLE
605 034F B9 000A      MOV CX,10            I LOOK FOR ENTRY USING KEYPAD
606 0352 F2 / AE      REPNE SCASB          I LOOK FOR MATCH
607 0354 75 18          JNE K33              I NO-ALT-KEYPAD
608 0356 F6 C7 02      TEST BH,LC_E0        I IS THIS ONE OF THE NEW KEYS?
609 0359 75 6B          JNZ K37C             I YES, JUMP, NOT NUMPAD KEY
610 035B 81 EF 030E R SUB D1,OFFSET K30+1 I D1 NOW HAS ENTRY VALUE
611 035F A0 0019 R   MOV AL,0ALT_INPUT   I GET THE CURRENT BYTE
612 0362 B4 0A          MOV AH,10            I MULTIPLY BY 10
613 0364 F6 E4          MUL AH               I
614 0366 03 C7          ADD AX,D1            I ADD IN THE LATEST ENTRY
615 0368 A2 0019 R   MOV 0ALT_INPUT,AL   I STORE IT AWAY
616 036B E9 02CA R     JMP K26              I THROW AWAY THAT KEYSTROKE
617
618 I----- LOOK FOR SUPERSHIFT ENTRY
619
620 036E                K33:                I NO-ALT-KEYPAD
621 036E C6 06 0019 R 00 MOV 0ALT_INPUT,0     I ZERO ANY PREVIOUS ENTRY INTO INPUT
622 0373 B9 001A      MOV CX,26            I D1,ES ALREADY POINTING
623 0376 F2 / AE      REPNE SCASB          I LOOK FOR MATCH IN ALPHABET
624 0378 74 42          JE K37A              I MATCH FOUND, GO FILL THE BUFFER
625
626 I----- LOOK FOR TOP ROW OF ALTERNATE SHIFT
627
628 037A                K34:                I ALT-TOP-ROW
629 037A 3C 02          CMP AL,2             I KEY WITH 'I' ON IT
630 037C 72 43          JB K37B             I NOT ONE OF INTERESTING KEYS
631 037E 3C 0D          CMP AL,13            I IS IT IN THE REGION
632 0380 77 AE          JA K35              I ALT-FUNCTION
633 0382 B0 C4 76      ADD AH,118           I CONVERT PSEUDO SCAN CODE TO RANGE
634 0385 EB 35          JMP SHORT K37A       I GO FILL THE BUFFER
635
636 I----- TRANSLATE ALTERNATE SHIFT PSEUDO SCAN CODES
637
638 0387                K35:                I ALT-FUNCTION
639 0387 3C 57          CMP AL,F11_M         I IS IT F11?
640 0389 72 09          JB K35A             I NO, BRANCH
641 038B 3C 58          CMP AL,F12_M         I IS IT F12?
642 038D 77 05          JA K35A             I NO, BRANCH
643 038F 80 C4 34      ADD AH,52            I CONVERT TO PSEUDO SCAN CODE
644 0392 EB 28          JMP SHORT K37A       I GO FILL THE BUFFER
645
646 0394 F6 C7 02      TEST BH,LC_E0        I DO WE HAVE ONE OF THE NEW KEYS?
647 0397 74 18          JZ K37              I NO, JUMP
648 0399 3C 1C          CMP AL,28            I TEST FOR KEYPAD ENTER
649 039B 75 06          JNE K35B            I NOT THERE
650 039D BB A600H      MOV AX,0A600H        I SPECIAL CODE
651 03A0 E9 050D R     JMP K57              I BUFFER_FILL
652 03A3 3C 53          CMP AL,83            I TEST FOR DELETE KEY
653 03A5 74 1F          JE K37C             I HANDLE WITH OTHER EDIT KEYS

```



```

654 03A7 3C 35          CMP     AL,53          ; TEST FOR KEYPAD /
655 03A9 75 C0          JNE     K32A          ; NOT THERE, NO OTHER E0 SPECIALS
656 03AB BB A400       MOV     AX,0A400h     ; SPECIAL CODE
657 03AE E9 050D R     JMP     K5T           ; BUFFER_FILL
658
659 03B1 3C 3B          CMP     AL,59          ; TEST FOR IN TABLE
660 03B3 72 0C          JB     K37B          ; ALT-CONTINUE
661 03B5 3C 44          CMP     AL,68          ; IN KEYPAD REGION?
662                                ; OR NUMLOCK, SCROLLLOCK?
663 03B7 77 92          JA     K32A          ; IF SO, IGNORE
664 03B9 80 C4 2D      ADD     AH,45         ; CONVERT TO PSEUDO SCAN CODE
665
666 03BC B0 00          K37A: MOV     AL,0         ; ASCII CODE OF ZERO
667 03BE E9 050D R     JMP     K5T           ; PUT IT IN THE BUFFER
668
669 03C1 B0 F0          K37B: MOV     AL,0F0h   ; USE SPECIAL ASCII CODE
670 03C3 E9 050D R     JMP     K5T           ; PUT IT IN THE BUFFER
671
672 03C6 04 50          K37C: ADD     AL,80     ; CONVERT SCAN CODE (EDIT KEYS)
673 03C8 8A E0          MOV     AH,AL         ; (SCAN CODE NOT IN AH FOR INSERT)
674 03CA EB F0          JMP     K37A          ; PUT IT IN THE BUFFER
675
676 I----- NOT IN ALTERNATE SHIFT
677
678 K38:                                ; NOT-ALT-SHIFT
679                                ; BL STILL HAS SHIFT FLAGS
680 03CC F6 C3 04       TEST    BL,CTL_SHIFT ; ARE WE IN CONTROL SHIFT?
681 03CF 75 03          JNZ    K38A          ; YES, START PROCESSING
682 03D1 E9 0454 R     JMP     K44           ; NOT-CTL-SHIFT
683
684 I----- CONTROL SHIFT, TEST SPECIAL CHARACTERS
685 I----- TEST FOR BREAK
686
687 K38A: CMP     AL,SCROLL_KEY ; TEST FOR BREAK
688 03D4 3C 46          JNE     K39           ; JUMP, NO-BREAK
689 03D6 75 1E          JNE     TEST          ; IS THIS THE ENHANCED KEYBOARD?
690 03D8 F4 07 10      TEST    BH,KBX       ; NO, BREAK_VALID
691 03DB 74 05          JZ     K38B          ; NO, BREAK_VALID
692 03DD F6 C7 02      TEST    BH,LC_E0     ; YES, WAS LAST CODE AN E0?
693 03E0 74 14          JZ     K39           ; NO-BREAK, TEST FOR PAUSE
694
695 03E2 8B 1E 001A R   K38B: MOV     BX,#BUFFER_HEAD ; RESET BUFFER TO EMPTY
696 03E6 89 1E 001C R   MOV     #BUFFER_TAIL,BX ; TURN ON BIOS BREAK BIT
697 03EA C6 0E 0071 R 80 MOV     #BIOS_BREAK,80H ; BREAK INTERRUPT VECTOR
698 03EF CD 1B          INT     IBH           ; PUT OUT DUMMY CHARACTER
699 03F1 28 C0          SUB     AX,AX         ; PUT OUT DUMMY CHARACTER
700 03F3 E9 050D R     JMP     K5T           ; BUFFER_FILL
701
702 I----- TEST FOR PAUSE
703
704 K39:                                ; NO-BREAK
705 03F6 F6 C7 10      TEST    BH,KBX       ; IS THIS THE ENHANCED KEYBOARD?
706 03F9 75 25          JNZ    K41           ; YES, THEN 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 OR     #KB_FLAG_1,HOLD_STATE ; TURN ON THE HOLD FLAG
710 7404 B0 20          MOV     AL,E0I       ; END OF INTERRUPT TO CONTROL PORT
711 0406 E6 20          OUT    020H,AL       ; ALLOW FURTHER KEYSTROKE INTS
712
713 I----- DURING PAUSE INTERVAL, TURN CRT BACK ON
714
715 0408 80 3E 0049 R 07 CMP     #CRT_MODE,7   ; IS THIS BLACK AND WHITE CARD
716 040D 74 07 10      JBE    K40           ; YES, NOTHING TO DO
717 040F BA 03D8       MOV     DX,03D8h     ; PORT FOR COLOR CARD
718 0412 A0 0665 R     MOV     AL,#CRT_MODE_SET ; GET THE VALUE OF THE CURRENT MODE
719 0415 EE            OUT    DX,AL         ; SET THE CRT MODE, SO THAT CRT IS ON
720 0416 F6 06 0018 R 08 K40: TEST    #KB_FLAG_1,HOLD_STATE ; PAUSE-LOOP
721 041B 75 F9          JNZ    K40           ; LOOP UNTIL FLAG TURNED OFF
722 041D E9 02D4 R     JMP     K2T           ; INTERRUPT_RETURN_NO_E0I
723
724 I----- TEST SPECIAL CASE KEY 55
725
726 K41:                                ; NO-PAUSE
727 0420                                ; TEST FOR * /PRTSK KEY
728 0420 3C 37          CMP     AL,55         ; NOT-KEY-55
729 0422 75 10          JNE     K42           ; NOT-KEY-55
730 0424 F6 C7 10      TEST    BH,KBX       ; IS THIS THE ENHANCED KEYBOARD?
731 0427 74 05          JZ     K41A          ; NO, CTL-PRTSK IS VALID
732 0429 F6 C7 02      TEST    BH,LC_E0     ; NO, NO MORE SPECIAL CASES
733 042C 74 20          JZ     K42B          ; YES, IS IT FROM THE KEYPAD?
734 042E B6 7200       MOV     AX,114*256   ; NO, JUST TRANSLATE
735 0431 E9 050D R     JMP     K5T           ; START/STOP TRANSLATE SWITCH
736                                ; BUFFER_FILL
737
738 I----- SET UP TO TRANSLATE CONTROL SHIFT
739
740 K42:                                ; NOT-KEY-55
741 0434 3C 0F          CMP     AL,15         ; IS IT THE TAB KEY?
742 0436 74 16          JBE    K42B          ; YES, XLATE TO FUNCTION CODE
743 0438 3C 35          CMP     AL,53         ; IS IT THE / KEY?
744 043C F6 C7 02      TEST    BH,LC_E0     ; NO, NO MORE SPECIAL CASES
745 043F 74 06          JZ     K42A          ; YES, IS IT FROM THE KEYPAD?
746 0441 B9 9500       MOV     AX,9500h     ; NO, JUST TRANSLATE
747 0444 E9 050D R     JMP     K5T           ; YES, SPECIAL CODE FOR THIS ONE
748                                ; BUFFER_FILL
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 I----- NOT IN CONTROL SHIFT
756
757 K44:                                ; PRINT SCREEN KEY?
758 0454 3C 37          CMP     AL,55         ; NOT-PRINT-SCREEN
759 0456 75 1F          JNE     K45           ; NOT-PRINT-SCREEN
760 0458 F6 C7 10      TEST    BH,KBX       ; IS THIS ENHANCED KEYBOARD?
761 045B 74 07          JZ     K44A          ; NO, TEST FOR SHIFT STATE
762 045D F6 C7 02      TEST    BH,LC_E0     ; YES, LAST CODE A MARKER?
763 0460 75 07          JNZ    K44B          ; YES, IS PRINT SCREEN
764 0462 EB 34          JMP     SHORT K45C    ; NO, XLATE TO "*" CHARACTER
765 0464 F6 C3 03      K44A: TEST    BL,LEFT_SHIFT+RIGHT_SHIFT ; NOT 101 KBD, SHIFT KEY DOWN?
766 0467 74 2F          JZ     K45C          ; NO, XLATE TO "*" CHARACTER
767
768 I----- ISSUE INTERRUPT TO PERFORM PRINT SCREEN FUNCTION

```

```

768 0469 B0 20      K44B:  MOV     AL,E01          ; 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     0KB_FLAG_3,NOT LC,E0+LC ; ZERO OUT THESE FLAGS
772 0474 E9 02D4 R  JMP     K27            ; GO BACK WITHOUT EOI OCCURRING
773
774
775
776 0477          ;----- HANDLE THE IN-CORE KEYS
777 0477 3C 3A      K45:   CMP     AL,58          ; NOT-PRINT-SCREEN
778 0479 77 2C      JA     K46            ; TEST FOR IN-CORE AREA
779                          ; JUMP IF NOT
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     DI,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 SURE
792 0493 F6 C3 03   K45B:  TEST   BL,LEFT_SHIFT+RIGHT_SHIFT ; ARE WE IN SHIFT STATE?
793 0496 75 04      JNZ    K45E           ; YES, UPPERCASE
794                          ; NO, LOWERCASE
795 0498 BB 05B7 R  K45C:  MOV     BX,OFFSET K10  ; TRANSLATE TO LOWERCASE LETTERS
796 049B EB 50      JMP     SHORT K56
797 049D          ;-----
798 049D F6 C3 03   K45D:  TEST   BL,LEFT_SHIFT+RIGHT_SHIFT ; ALMOST-CAPS-STATE
799 04A0 75 F6      JNZ    K45C           ; CL 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      JMP     SHORT K56      ; TRANSLATE TO UPPERCASE LETTERS
802
803
804
805 04A7          ;----- TEST FOR KEYS F1 - F10
806 04A7 3C 44      K46:   CMP     AL,68          ; NOT IN-CORE AREA
807 04A9 77 02      JA     K47            ; TEST FOR F1 - F10
808 04AB EB 36      JMP     SHORT K53      ; JUMP IF NOT
809                          ; YES, GO DO FN KEY PROCESS
810
811
812
813 04AD          ;----- HANDLE THE NUMERIC PAD KEYS
814 04AD 3C 53      K47:   CMP     AL,83          ; NOT F1 - F10
815 04AF 77 2C      JA     K52            ; TEST FOR NUMPAD KEYS
816                          ; JUMP IF NOT
817
818 04B1 3C 4A      ;----- KEYPAD KEYS, MUST TEST NUM LOCK FOR DETERMINATION
819 04B2 74 ED      K48:   CMP     AL,74          ; SPECIAL CASE FOR MINUS
820 04B5 3C 4E      JE     K45E           ; GO TRANSLATE
821 04B7 74 E9      CMP     AL,78          ; SPECIAL CASE FOR PLUS
822 04B9 F6 C7 02   JE     K45E           ; GO TRANSLATE
823 04BC 75 0A      TEST   BH,LC_E0        ; IS THIS ONE OF THE NEW KEYS?
824                          ; YES, TRANSLATE TO BASE STATE
825 04BE F6 C3 20   TEST   BL,NUM_STATE    ; ARE WE IN NUM LOCK?
826 04C1 75 13      JNZ    K50            ; TEST FOR SURE
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
831 04C8 3C 4C      ;----- BASE CASE FOR KEYPAD
832 04CA 75 05      K49:   CMP     AL,76          ; SPECIAL CASE FOR BASE STATE 5
833 04CC B0 F0      JNE    K49A           ; CONTINUE IF NOT KEYPAD 5
834 04CE BD 3D 90   MOV     AL,0F0h        ; SPECIAL ASCII CODE
835 04D1 BB 05B7 R  K49A:  JMP     K51            ; BUFFER FILL
836 04D4 EB 26      MOV     BX,OFFSET K10  ; BASE CASE TABLE
837                          ; CONVERT TO PSEUDO SCAN
838
839 04D6 F6 C3 03   ;----- MIGHT BE NUM LOCK, TEST SHIFT STATUS
840 04D9 75 0D      K50:  TEST   BL,LEFT_SHIFT+RIGHT_SHIFT ; ALMOST-NUM-STATE
841 04DB EB C5      JNZ    K49            ; SHIFTED TEMP OUT OF NUM STATE
842                          ; REALLY_NUM_STATE
843
844
845
846 04DD          ;----- TEST FOR THE NEW KEY ON WT KEYBOARDS
847 04DD 3C 56      K52:   CMP     AL,86          ; NOT A NUMPAD KEY
848 04DF 75 02      JNE    K53            ; IS IT THE NEW WT KEY?
849 04E1 EB 80      JMP     SHORT K45B     ; JUMP IF NOT
850                          ; HANDLE WITH REST OF LETTER KEYS
851
852
853
854 04E3 F6 C3 03   ;----- MUST BE F11 OR F12
855 04E6 74 E0      K53:  TEST   BL,LEFT_SHIFT+RIGHT_SHIFT ; F1 - F10 COME HERE, TOO
856                          ; TEST SHIFT STATE
857 04E8 BB 060F R  JZ     K49            ; JUMP, LOWERCASE PSEUDO SC'S
858 04EB 0F         MOV     BX,OFFSET K11  ; UPPERCASE PSEUDO SCAN CODES
859                          ; TRANSLATE_SCAN
860
861
862 04ED          ;----- TRANSLATE THE CHARACTER
863 04ED FE C8      K56:  DEC     AL             ; TRANSLATE-CHAR
864 04EF 2E: D7     XLAT   CS:K11         ; CONVERT ORIGIN
865 04F1 F6 06 0096 R 02 AND     0KB_FLAG_3,LC_E0 ; CONVERT THE SCAN CODE TO ASCII
866 04F6 74 15      JZ     K57            ; IS THIS A NEW KEY?
867 04F8 B4 E0      MOV     AH,MC_E0       ; NO, GO FILL BUFFER
868 04FA EB 11      JMP     SHORT K57       ; YES, PUT SPECIAL MARKER IN AH
869                          ; PUT IT INTO THE BUFFER
870
871
872 04FC          ;----- TRANSLATE SCAN FOR PSEUDO SCAN CODES
873 04FC FE C8      K64:  DEC     AL             ; TRANSLATE-SCAN-ORGD
874 04FE 2E: D7     XLAT   CS:K8         ; CONVERT ORIGIN
875 0500 8A E0      MOV     AH,AL          ; CTL TABLE SCAN
876 0502 B0 00      MOV     AL,0           ; PUT VALUE INTO AH
877 0504 F6 06 0096 R 02 AND     0KB_FLAG_3,LC_E0 ; ZERO ASCII CODE
878 0509 74 02      TEST   JZ             ; IS THIS A NEW KEY?
879 050B B0 E0      MOV     AL,MC_E0       ; NO, GO FILL BUFFER
880                          ; YES, PUT SPECIAL MARKER IN AL
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

```

882
883 050D                                K57:                                ; BUFFER-FILL
884 050D 3C FF                            CMP     AL,-1                        ; IS THIS AN IGNORE CHAR
885 050F 74 05                            JE      K59                          ; YES, DO NOTHING WITH IT
886 0511 80 FC FF                          CMP     AH,-1                        ; LOOK FOR -1 PSEUDO SCAN
887 0514 75 03                            JNE    K61                          ; NEAR_INTERRUPT_RETURN
888
889 0516                                K59:                                ; NEAR_INTERRUPT_RETURN
890 0516 E9 02CA R                          JMP     K26                          ; INTERRUPT_RETURN
891
892 0519                                K61:                                ; TURN OFF INTERRUPTS
893 0519 FA                                CLI
894 051A 8B 1E 001C R                       MOV     BX,#BUFFER_TAIL            ; GET THE END POINTER TO THE BUFFER
895 051E 8B F3                               MOV     SI,BX                      ; SAVE THE VALUE
896 0520 E8 0114 R                           CALL    K4                          ; ADVANCE THE TAIL
897 0523 3B 1E 001A R                       CMP     BX,#BUFFER_HEAD            ; HAS THE BUFFER WRAPPED AROUND
898 0527 74 17                               JE      K62                          ; BUFFER FULL BEEP
899 0529 89 04                               MOV     [SI],AX                    ; STORE THE VALUE
900 052B 89 1E 001C R                       MOV     #BUFFER_TAIL,BX           ; MOVE THE POINTER UP
901 052F 80 20                               MOV     AL,E01                    ; END OF INTERRUPT COMMAND
902 0531 E6 20                               OUT    020H,AL                    ; SEND COMMAND TO INTERRUPT CONTROL PORT
903 0533 88 9102                            MOV     AX,09102H                 ; MOVE IN POST CODE & TYPE
904 0536 CD 15                               INT    15H                        ; PERFORM OTHER FUNCTION
905 0538 80 26 0096 R FC                    AND     #KB_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:                                ; ENABLE INTERRUPT CONTROLLER CHIP
911 0540 80 20                               MOV     AL,E01
912 0542 E6 20                               OUT    INTA00,AL
913 0544 B9 02A6                            MOV     CX,678                     ; DIVISOR FOR 1760 HZ
914 0547 B3 04                               MOV     BL,4                       ; SHORT BEEP COUNT (1/16 + 1/64 DELAY)
915 0549 E8 0000 E                          CALL    BEEP                       ; GO TO COMMON BEEP HANDLER
916 054C E9 02D4 R                          JMP     K27                          ; EXIT
917
918 054F                                KB_INT_1 ENDP

```

```

919                                     PAGE
920 -----
921                                     |
922                                     | KEY IDENTIFICATION SCAN TABLES |
923 -----
924                                     |
925                                     |----- TABLE OF SHIFT KEYS AND MASK VALUES -----|
926                                     | LABEL BYTE |
927 054F 52                                K6 |
928 0550 3A 45 46 38 1D                DB | INS KEY | INSERT KEY
929 0555 2A 36                          DB | CAPS_KEY,NUM_KEY,SCROLL_KEY,ALT_KEY,CTL_KEY
930 = 0008                                K6L | EQU | *K
931
932                                     |----- MASK TABLE -----|
933                                     | LABEL BYTE |
934 0557 80                                K7 |
935 0558 40 20 10 08 04                DB | INS SHIFT | INSERT MODE SHIFT
936 0550 02 01                          DB | CAPS_SHIFT,NUM_SHIFT,SCROLL_SHIFT,ALT_SHIFT,CTL_SHIFT
937                                     DB | LEFT_SHIFT,RIGHT_SHIFT
938
939                                     |----- TABLES FOR CTRL CASE -----|
940                                     |
941 055F 1B FF 00 FF FF FF                K8 | LABEL BYTE | CHARACTERS
942 0565 1E FF FF FF FF FF                DB | 27,-1,00,-1,-1,-1 | Esc,1,2,3,4,5
943 056B FF 7F 94 11 17 05              DB | 30,-1,-1,-1,-1,31 | 6,7,8,9,0
944 0571 12 14 19 15 09 0F              DB | -1,127,148,17,23,5 | =,Backsp,Tab,Q,W,E
945 0577 10 1B 1D 0A FF 01              DB | 18,20,25,21,09,15 | R,T,Y,U,I,O
946 057D 13 04 06 07 08 0A              DB | 16,27,29,10,-1,01 | F,[ ],Enter,Ctrl,A
947 0583 0B 0C FF FF FF FF              DB | 19,04,06,07,08,10 | S,D,F,G,H,J
948 0589 1C 1A 18 03 16 02              DB | 11,12,-1,-1,-1,-1 | K,L,I, , ,LShift
949 058F 0E 0D FF FF FF FF              DB | 28,26,24,03,22,02 | J,Z,X,C,V,B
950 0595 9E FF 20 FF                    DB | 14,13,-1,-1,-1,-1 | N,M, , , , /,RShift
951                                     DB | 150,-1, , , -1 | * , Alt, Space, CL
952                                     |----- FUNCTIONS -----|
953 0599 5E 5F 60 61 62 63              DB | 94,95,96,97,98,99 | F1 - F6
954 059F 64 65 66 67 FF FF              DB | 100,101,102,103,-1,-1 | F7 - F10, NL, SL
955 05A5 77 80 84 8E 73 8F              DB | 119,141,132,142,115,143 | Home,Up,PgUp,-,Left,Pad5
956 05AB 74 90 75 91 76 92              DB | 116,144,117,145,118,146 | Right, , End,Down,PgDn,Ins
957 05B1 93 FF FF FF 89 8A              DB | 147,-1,-1,-1,137,138 | Del,SysReq,Undef,WT,F11,F12
958
959                                     |----- TABLES FOR LOWER CASE -----|
960 05B7 1B 31 32 33 34 35              K10 | LABEL BYTE |
961 05BD 36 37 38 39 30 2D              DB | 27,'12345'
962 05C3 3D 08 09 71 77 65              DB | '67890-'
963 05C9 72 74 75 69 6F                DB | '=,'08,09,'qwe'
964 05CF 70 5B 5D 5D FF 61              DB | 'rtyuiop'
965 05D5 73 64 66 67 68 6A              DB | 'p[ ],ODH,-1,'a' | LETTERS, Return, Ctrl
966 05DB 6B 6C 3B 27 60 FF              DB | 'sdghj'
967 05E1 6C 7A 78 63 76 62              DB | 'kl;~,-1 | LETTERS, L Shift
968 05E7 6E 6D 2C 2E 2F                DB | '\xcvb?'
969 05EC FF 2A FF 20 FF                DB | 'nm;/'
970                                     DB | -1,' , , -1,-1 | R Shift,*, Alt, Space, CL
971
972                                     |----- LC TABLE SCAN -----|
973 05F1 3B 3C 3D 3E 3F                K11 | LABEL BYTE |
974 05F6 40 41 42 43 44                DB | 59,60,61,62,63 | BASE STATE OF F1 - F10
975 05FB FF FF                          DB | 64,65,66,67,68 |
976                                     DB | -1,-1 | NL, SL
977
978                                     |----- KEYPAD TABLE -----|
979 05FD 47 48 49 FF 4B FF              K15 | LABEL BYTE |
980 0603 4D FF 4F 50 51 52              DB | 77,-1,79,80,81,82 | BASE STATE OF KEYPAD KEYS
981 0609 53                              DB | 83 |
982 060A FF FF 5C 85 86                DB | -1,-1,'\'',133,134 | SysRq,Undef,WT,F11,F12
983
984                                     |----- TABLES FOR UPPER CASE -----|
985 060F 1B 21 40 23 24 25              K11 | LABEL BYTE |
986 0615 5E 26 2A 2B 29 5F              DB | 27,'!@#$%'
987 061B 2B 08 00 51 57 45              DB | '\A'(') |
988 0621 52 54 59 55 49 4F              DB | '=,'08,00,'QWE'
989 0627 50 7B 7D 0D FF 41              DB | 'RTYUIO'
990 062D 53 44 46 47 48 4A              DB | 'P[ ],ODH,-1,'A' | LETTERS, Return, Ctrl
991 0633 4B 4C 3A 2E 7E FF              DB | 'SDFGHJ'
992 0639 7C 5A 58 43 56 42              DB | 'KL;~,-1 | LETTERS, L Shift
993 063F 4E 4D 3C 3E 3F                DB | '\XCVB?'
994 0644 FF 2A FF 20 FF                DB | 'NMc??'
995                                     DB | -1,' , , -1,-1 | R Shift,*, Alt, Space, CL
996
997                                     |----- UC TABLE SCAN -----|
998 0649                                K12 | LABEL BYTE |
999 0649 54 55 56 57 58                DB | 84,85,86,87,88 | SHIFTED STATE OF F1 - F10
1000 064E 59 5A 5B 5C 5D                DB | 89,90,91,92,93 |
1001 0653 FF FF                          DB | -1,-1 | NL, SL
1002
1003                                     |----- NUM STATE TABLE -----|
1004 0655                                K14 | LABEL BYTE |
1005 0655 37 38 39 2D 34 35              DB | '789-456+1230.' | NUMLOCK STATE OF KEYPAD KEYS
1006                                     DB | 36 2B 31 32 33 30 |
1007                                     DB | ZE |
1008 0662 FF FF 7C 87 88                DB | -1,-1,'|',135,136 | SysRq,Undef,WT,F11,F12
1009 0667
1010 CODE ENDS
1011 END
    
```



```

96 PAGE
97
98 |----- CHECK FOR PRINTER BUSY
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 80 TEST AL,B0H ; 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 B40:
112 003F 51 PUSH CX ; SAVE CALLERS (CX) REGISTER
113 0040 2B C9 SUB CX,CX ; INNER LOOP (64K)
114 0042
115 0042 EC IN AL,DX ; GET STATUS
116 0043 8A E0 MOV AH,AL ; STATUS TO (AH) ALSO
117 0045 A8 80 TEST AL,B0H ; 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,0F9H ; TURN OFF THE UNUSED BITS
128 0056 EB 15 JMP SHORT B70 ; RETURN WITH ERROR FLAG SET
129
130 0058 B50:
131 0058 59 POP CX ; SEND STROBE PULSE
132 0059 B0 0D MOV AL,0DH ; RESTORE (CX) WITH CALLERS VALUE
133 005B 42 INC DX ; SET THE STROBE LOW (BIT 0N)
134 005C FA CLJ ; OUTPUT STROBE TO CONTROL PORT
135 005D EE OUT DX,AL ; PREVENT INTERRUPT PULSE STRETCHING
136 005E EB 00 JMP $+2 ; OUTPUT STROBE BIT > 1µs < 5µs
137 ; 1/10 DELAY TO ALLOW FOR LINE LOADING
138 ; AND FOR CORRECT PULSE WIDTH
139 0060 B0 0C MOV AL,0CH ; SET THE -STROBE HIGH
140 0062 EE OUT DX,AL
141 0064 4A DEC DX ; INTERRUPTS BACK ON
142 0065 4A DEC DX ; ADJUST BACK TO BASE ADDRESS
143 ; FOR STATUS ROUTINE EXIT
144
145 |----- PRINTER STATUS
146
147 0066 B60:
148 0066 42 INC DX ; POINT TO CONTROL PORT
149 0067 EC IN AL,DX ; PRE-CHARGE +BUSY LINE IF FLOATING
150 0068 EC IN AL,DX ; GET PRINTER STATUS HARDWARE BITS
151 0069 24 F8 AND AL,0F8H ; TURN OFF UNUSED BITS
152 006B 8A E0 MOV AH,AL ; SAVE
153 006D B70:
154 006D 8A C7 MOV AL,BH ; RECOVER CHARACTER INTO (AL) REGISTER
155 006F 80 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 |----- INITIALIZE THE PRINTER PORT
159
160 B80:
161 0074 INC DX ; POINT TO OUTPUT PORT
162 0074 42 INC DX
163 0075 42 INC DX
164 0076 B0 08 MOV AL,8 ; SET INIT LINE LOW
165 0078 EE OUT DX,AL
166 0079 B8 03EB MOV AX,1000 ; ADJUST FOR INITIALIZATION DELAY LOOP
167 007C B90:
168 007C 48 DEC AX ; DECREMENT DELAY COUNTER
169 007D 75 FD JNZ B90 ; LOOP FOR RESET TO TAKE
170
171 007F B0 0C MOV AL,0CH ; NO INTERRUPTS, NON AUTO LF, INIT HIGH
172 0081 EE OUT DX,AL ; SET DEFAULT INITIAL OUTPUTS
173 0082 4A DEC DX ; ADJUST BACK TO BASE ADDRESS
174 0083 4A DEC DX ; FOR STATUS ROUTINE EXIT
175 0084 EB E0 JMP B60 ; EXIT THROUGH STATUS ROUTINE
176
177 PRINTER_IO_1 ENDP
178
179 0086 CODE ENDS
180

```

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114

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

PUBLIC RS232_IO_1
EXTRN AH:NEAR
EXTRN DS:NEAR

INT 14 H
RS232_IO_1
THIS ROUTINE PROVIDES BYTE STREAM I/O TO THE COMMUNICATIONS
PORT ACCORDING TO THE PARAMETERS:

(AH) = 00H INITIALIZE THE COMMUNICATIONS PORT
(AL) HAS PARAMETERS FOR INITIALIZATION

7 6 5 4 3 2 1 0
---- BAUD RATE -- -PARITY-- STOPBIT -WORD LENGTH--
000 - 110 X0 - NONE 0 - 1 10 - 7 BITS
001 - 150 01 - ODD 1 - 2 11 - 8 BITS
010 - 300 11 - EVEN
011 - 600
100 - 1200
101 - 2400
110 - 4800
111 - 9600
ON RETURN, CONDITIONS SET AS IN CALL TO COMMO STATUS (AH=03H)

(AH) = 01H SEND THE CHARACTER IN (AL) OVER THE COMMO LINE
(AL) REGISTER IS PRESERVED
ON EXIT, BIT 7 OF AH IS SET IF THE ROUTINE WAS UNABLE TO
TO TRANSMIT THE BYTE OF DATA OVER THE LINE.
IF BIT 7 OF AH IS NOT SET, THE
REMAINDER OF (AH) IS SET AS IN A STATUS REQUEST,
REFLECTING THE CURRENT STATUS OF THE LINE.
(AH) = 02H RECEIVE A CHARACTER IN (AL) FROM COMMO LINE BEFORE
RETURNING TO CALLER
ON EXIT, (AH) HAS THE CURRENT LINE STATUS, AS SET BY THE
THE STATUS ROUTINE, EXCEPT THAT THE ONLY BITS
LEFT ON ARE THE ERROR BITS (7,4,3,2,1)
IF (AH) HAS BIT 7 ON (TIME OUT) THE REMAINING
BITS ARE NOT PREDICTABLE
THUS, (AH) IS NON ZERO ONLY WHEN AN ERROR OCCURRED.
(AH) = 03H RETURN THE COMMO PORT STATUS IN (AX)
(AH) CONTAINS THE LINE CONTROL STATUS
BIT 7 = TIME OUT
BIT 6 = TRANSMIT SHIFT REGISTER EMPTY
BIT 5 = TRANSMIT HOLDING REGISTER EMPTY
BIT 4 = BREAK DETECT
BIT 3 = FRAMING ERROR
BIT 2 = PARITY ERROR
BIT 1 = OVERRUN ERROR
BIT 0 = DATA READY
(AL) CONTAINS THE MODEM STATUS
BIT 7 = RECEIVE LINE SIGNAL DETECT
BIT 6 = RING INDICATOR
BIT 5 = DATA SET READY
BIT 4 = CLEAR TO SEND
BIT 3 = DELTA RECEIVE LINE SIGNAL DETECT
BIT 2 = TRAILING EDGE RING DETECTOR
BIT 1 = DELTA DATA SET READY
BIT 0 = DELTA CLEAR TO SEND

(DX) = PARAMETER INDICATING WHICH RS232 CARD (0,1 ALLOWED)

DATA AREA #RS232_BASE CONTAINS THE BASE ADDRESS OF THE 8250 ON THE CARD
LOCATION 400H CONTAINS UP TO 4 RS232 ADDRESSES POSSIBLE
DATA AREA LABEL #RS232_TIM_OUT (BYTE) CONTAINS OUTER LOOP COUNT
VALUE FOR TIMEOUT (DEFAULT=1)

OUTPUT AX MODIFIED ACCORDING TO PARAMETERS OF CALL
ALL OTHERS UNCHANGED

ASSUME CS:CODE,DS:DATA

RS232_IO_1 PROC FAR
STI ; INTERRUPTS BACK ON
PUSH DS ; SAVE SEGMENT
PUSH DX
PUSH SI
PUSH DI
PUSH CX
PUSH BX
CMP DX,03H ; CHECK FOR ADAPTER NUMBER VALID 0-3
JAE A3E ; ERROR EXIT IF OUT OF RANGE
MOV SI,DX ; RS232 VALUE TO (SI)
MOV DI,DX ; AND TO (DI) (FOR TIMEOUTS)
SHL SI,1 ; WORD OFFSET
CALL DDS ;
MOV DX,#RS232_BASE[SI] ; GET BASE ADDRESS
OR DX,DX ; TEST FOR 0 BASE ADDRESS
JZ A3E ; RETURN
OR AH,AH ; TEST FOR (AH) = 00H
JZ A4 ; COMMO INITIALIZATION
DEC AH ; TEST FOR (AH) = 01H
JZ A5 ; SEND (AL)
DEC AH ; TEST FOR (AH) = 02H
JZ A12 ; RECEIVE INTO (AL)
DEC AH ; TEST FOR (AH) = 03H
JNZ A3E ; ERROR IF BAD COMMAND
JMP A18 ; COMMUNICATION STATUS

A2:
DEC AH ;
JNZ A3E ;
JMP A18 ;

A3E:
MOV AH,080H ; SET ERROR RETURN CODE

A3:
POP BX ; RETURN FROM RS232
POP CX
POP DI
POP SI
POP DX
POP DS
IRET ; RETURN TO CALLER, NO ACTION
  
```

SECTION 5

```

115 PAGE
116 |----- INITIALIZE THE COMMUNICATIONS PORT
117
118 0039 A4:
119 0039 8A E0 MOV AH,AL ; SAVE INITIALIZATION PARAMETERS IN (AH)
120 003B 83 C2 03 ADD DX,3 ; POINT TO 8250 CONTROL REGISTER
121 003E B0 80 MOV AL,80H
122 0040 EE OUT DX,AL ; SET DLAB=1
123
124 |----- DETERMINE BAUD RATE DIVISOR
125
126 0041 8A D4 MOV DL,AH ; GET PARAMETERS TO (DL)
127 0043 B1 04 MOV CL,4
128 0045 D2 C2 ROL DL,CL
129 0047 81 E2 000E AND DX,0EH ; ISOLATE THEM
130 004B BF 0000 E MOV DI,OFFSET A1 ; BASE OF TABLE
131 004E 03 FA ADD DI,DX ; PUT INTO INDEX REGISTER
132 0050 8B 94 0000 R MOV DX,®RS232_BASE[SI] ; POINT TO HIGH ORDER OF DIVISOR
133 0054 42 INC DX
134 0055 2E: 8A 45 01 MOV AL,CS:[DI]+1 ; GET HIGH ORDER OF DIVISOR
135 0059 EE OUT DX,AL ; SET ms OF DIVISOR TO 0
136 005A 4A DEC DX
137 005B 90 NOP ; I/O DELAY
138 005C 2E: 8A 05 MOV AL,CS:[DI] ; GET LOW ORDER OF DIVISOR
139 005F EE OUT DX,AL ; SET LOW OF DIVISOR
140 0060 83 C2 03 ADD DX,3 ; SET LOW OF DIVISOR
141 0063 8A C4 MOV AL,AH ; GET PARAMETERS BACK
142 0065 2A 1F AND AL,01FH ; STRIP OFF THE BAUD BITS
143 0067 EE OUT DX,AL ; LINE CONTROL TO 8 BITS
144 0068 4A DEC DX
145 0069 4A DEC DX
146 006A 90 NOP ; I/O DELAY
147 006B B0 00 MOV AL,0
148 006D EE OUT DX,AL ; INTERRUPT ENABLES ALL OFF
149 006E EB 4B JMP SHORT A18 ; COM_STATUS
150
151 |----- SEND CHARACTER IN (AL) OVER COMMO LINE
152
153 0070 A5:
154 0070 50 PUSH AX ; SAVE CHAR TO SEND
155 0071 83 C2 04 ADD DX,4 ; MODEM CONTROL REGISTER
156 0074 B0 03 MOV AL,3 ; DTR AND RTS
157 0076 EE OUT DX,AL ; DATA TERMINAL READY, REQUEST TO SEND
158 0077 42 INC DX ; MODEM STATUS REGISTER
159 0078 42 INC DX
160 0079 B7 30 MOV BH,30H ; DATA SET READY & CLEAR TO SEND
161 007B E8 00CA R CALL WAIT_FOR_STATUS ; ARE BOTH TRUE
162 007E 74 08 JE A9 ; YES, READY TO TRANSMIT CHAR
163 0080 A7:
164 0080 59 POP CX
165 0081 8A C1 MOV AL,CL ; RELOAD DATA BYTE
166 0083 A8:
167 0083 B0 CC 80 OR AH,80H ; INDICATE TIME OUT
168 0086 EB AA JMP A3 ; RETURN
169
170 0088 A9:
171 0088 4A DEC DX ; CLEAR TO SEND
172 0089 A10:
173 0089 B7 20 MOV BH,20H ; LINE STATUS REGISTER
174 008B E8 00CA R CALL WAIT_FOR_STATUS ; WAIT SEND
175 008E 75 F0 JNZ A7 ; IS TRANSMITTER READY
176 0090 A11:
177 0090 83 EA 05 SUB DX,5 ; TEST FOR TRANSMITTER READY
178 0093 59 POP CX ; RETURN WITH TIME OUT SET
179 0094 8A C1 MOV AL,CL ; OUT CHAR
180 0096 EE OUT DX,AL ; DTR PORT
181 0097 EB 99 JMP A3 ; RECOVER IN CX TEMPORARILY
182 ; MOVE CHAR TO AL FOR OUT, STATUS IN AH
183 ; OUTPUT CHARACTER
184 ; RETURN
185 |----- RECEIVE CHARACTER FROM COMMO LINE
186
187 0099 A12:
188 0099 83 C2 04 ADD DX,4 ; MODEM CONTROL REGISTER
189 009C B0 01 MOV AL,1 ; DATA TERMINAL READY
190 009F 42 INC DX ; MODEM STATUS REGISTER
191 00A0 42 INC DX
192 00A1 B7 20 MOV BH,20H ; WAIT_DSR
193 00A3 E8 00CA R CALL WAIT_FOR_STATUS ; DATA SET READY
194 00A6 75 DB JNZ A8 ; TEST FOR DSR
195 00A8 A13:
196 00A8 4A DEC DX ; RETURN WITH ERROR
197 00A9 A15:
198 00A9 B7 01 MOV BH,1 ; WAIT_DSR_END
199 00AB E8 00CA R CALL WAIT_FOR_STATUS ; LINE STATUS REGISTER
200 00AE 75 D3 JNZ A8 ; WAIT RECV
201 00B0 A16:
202 00B0 B0 E4 IE MOV AL,1 ; RECEIVE BUFFER FULL
203 00B3 8B 94 0000 R MOV DX,®RS232_BASE[SI] ; TEST FOR RECEIVE BUFFER FULL
204 00B7 EC IN AL,DX ; SET TIME OUT ERROR
205 00B8 E9 0032 R JMP A3 ; GET CHAR
206 ; TEST FOR ERROR CONDITIONS ON RECEIVE
207
208 |----- COMMO PORT STATUS ROUTINE
209
210 00BB A18:
211 00BB 8B 94 0000 R MOV DX,®RS232_BASE[SI] ; CONTROL PORT
212 00BF 83 C2 05 ADD DX,5 ; GET LINE CONTROL STATUS
213 00C2 EC IN AL,AL ; PUT IN (AH) FOR RETURN
214 00C3 8A E0 MOV AL,DX ; POINT TO MODEM STATUS REGISTER
215 00C5 42 INC DX ; GET MODEM CONTROL STATUS
216 00C6 EC IN AL,DX ; RETURN
217 00C7 E9 0032 R JMP A3
    
```



```

218                                     PAGE
219                                     :-----:
220                                     : WAIT FOR STATUS ROUTINE :
221 : ENTRY: (BH)= STATUS BIT(S) TO LOOK FOR :
222 : (DX)= ADDRESS OF STATUS REG :
223 :EXIT: ZERO FLAG ON = STATUS FOUND :
224 : ZERO FLAG OFF = TIMEOUT. :
225 : (AH)= LAST STATUS READ :
226                                     :-----:
227
228 00CA                                WAIT_FOR_STATUS PROC NEAR
229
230 00CA 8A 9D 007C R                    MOV     BL,0RS232_TIM_OUT[D1] ; LOAD OUTER LOOP COUNT
231 00CE                                WFS0:  SUB     CX,CX
232 00CE 2B C9                          WFS1:
233 00D0                                IN      AL,DX ; GET STATUS
234 00D0 EC                              MOV     AH,AL ; MOVE TO (AH)
235 00D1 8A E0                            AND     AL,BH ; ISOLATE BITS TO TEST
236 00D3 22 C7                            CMP     AL,BH ; EXACTLY = TO MASK
237 00D5 3A C7                            JE      WFS_END ; RETURN WITH ZERO FLAG ON
238 00D7 74 08
239
240 00D9 E2 F5                            LOOP   WFS1 ; TRY AGAIN
241
242 00DB FE CB                            DEC     BL ; DECREMENT LOOP COUNTER
243 00DD 75 EF                            JNZ    WFS0
244
245 00DF 0A FF                            OR      BH,BH ; SET ZERO FLAG OFF
246 00E1                                WFS_END: RET
247 00E1 C3
248
249 00E2                                WAIT_FOR_STATUS ENDP
250
251 00E2                                RS232_IO_1 ENDP
252
253 00E2                                CODE ENDS
254                                END
  
```

```

1          PAGE 118,121
2          TITLE VIDEO ---- 01/10/86 VIDEO DISPLAY BIOS
3          .LIST
4          0000
5          CODE SEGMENT BYTE PUBLIC
6          PUBLIC ACT_DISP_PAGE
7          PUBLIC READ_AC_CURRENT
8          PUBLIC READ_CURSOR
9          PUBLIC READ_DOT
10         PUBLIC READ_LPEN
11         PUBLIC SCROLL_DOWN
12         PUBLIC SCROLL_UP
13         PUBLIC SET_COLOR
14         PUBLIC SET_CPOS
15         PUBLIC SET_CTYPE
16         PUBLIC SET_MODE
17         PUBLIC WRITE_AC_CURRENT
18         PUBLIC WRITE_C_CURRENT
19         PUBLIC WRITE_DOT
20         PUBLIC WRITE_TTY
21         PUBLIC VIDEO_IO_1
22         PUBLIC VIDEO_STATE
23
24         PUBLIC SET_MODE
25         PUBLIC SET_CTYPE
26         PUBLIC SET_CPOS
27         PUBLIC READ_CURSOR
28         PUBLIC READ_LPEN
29         PUBLIC ACT_DISP_PAGE
30         PUBLIC SCROLL_UP
31         PUBLIC SCROLL_DOWN
32         PUBLIC READ_AC_CURRENT
33         PUBLIC WRITE_AC_CURRENT
34         PUBLIC WRITE_C_CURRENT
35         PUBLIC SET_COLOR
36         PUBLIC WRITE_DOT
37         PUBLIC READ_DOT
38         PUBLIC WRITE_TTY
39         PUBLIC VIDEO_STATE
40         PUBLIC VIDEO_RETURN
41         PUBLIC VIDEO_RETURN
42         PUBLIC VIDEO_RETURN
43         PUBLIC WRITE_STRING
44
45         EXTRN BEEP:NEAR          ; SPEAKER BEEP ROUTINE
46         EXTRN CRT_CHAR_GEN:NEAR ; CHARACTER GENERATOR GRAPHICS TABLE
47         EXTRN DDST:NEAR         ; LOAD (DS) WITH DATA SEGMENT SELECTOR
48         EXTRN M5:WORD           ; REGEN BUFFER LENGTH TABLE
49         EXTRN M6:BYTE           ; COLUMNS PER MODE TABLE
50         EXTRN M7:BYTE           ; MODE SET VALUE PER MODE TABLE
51
52         ;----- INT 10 H -----
53         ;
54         VIDEO_IO
55         ; THESE ROUTINES PROVIDE THE CRT DISPLAY INTERFACE
56         ; THE FOLLOWING FUNCTIONS ARE PROVIDED:
57
58         (AH) = 00H SET MODE (AL) CONTAINS MODE VALUE
59         (AL) = 00H 40X25 BW MODE (POWER ON DEFAULT)
60         (AL) = 01H 40X25 COLOR
61         (AL) = 02H 80X25 BW
62         (AL) = 03H 80X25 COLOR
63
64         (AL) = 04H GRAPHICS 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 4-0 = END LINE FOR CURSOR
78
79         (AH) = 02H SET CURSOR POSITION
80         (DH,DL) = ROW,COLUMN (00H,00H) IS UPPER LEFT
81         (BH) = PAGE NUMBER (MUST BE 00H FOR GRAPHICS MODES)
82
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) = RASTER LINE (0-199)
94         (BX) = PIXEL COLUMN (0-219,639)
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 2&3)
98
99         (AH) = 06H SCROLL ACTIVE PAGE UP
100        (AL) = NUMBER OF LINES, ( LINES BLANKED AT BOTTOM OF WINDOW )
101        (AL) = 00H MEANS BLANK ENTIRE WINDOW
102        (CH,CL) = ROW,COLUMN OF UPPER LEFT CORNER OF SCROLL
103        (DH,DL) = ROW,COLUMN OF LOWER RIGHT CORNER OF SCROLL
104        (BH) = ATTRIBUTE TO BE USED ON BLANK LINE
105
106        (AH) = 07H SCROLL ACTIVE PAGE DOWN
107        (AL) = NUMBER OF LINES, INPUT LINES BLANKED AT TOP OF WINDOW
108        (AL) = 00H MEANS BLANK ENTIRE WINDOW
109        (CH,CL) = ROW,COLUMN OF UPPER LEFT CORNER OF SCROLL
110        (DH,DL) = ROW,COLUMN OF LOWER RIGHT CORNER OF SCROLL
111        (BH) = ATTRIBUTE TO BE USED ON BLANK LINE
112
113        CHARACTER HANDLING ROUTINES
114
115        (AH) = 08H READ ATTRIBUTE/CHARACTER AT CURRENT CURSOR POSITION
116        (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY)
117        ON EXIT:
118        (AL) = CHAR READ
119        (AH) = ATTRIBUTE OF CHARACTER READ (ALPHA MODES ONLY)
120
121        (AH) = 09H WRITE ATTRIBUTE/CHARACTER AT CURRENT CURSOR POSITION
122        (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY)

```

```

115      ; (CX) = COUNT OF CHARACTERS TO WRITE
116      ; (AL) = CHAR TO WRITE
117      ; (BL) = ATTRIBUTE OF CHARACTER (ALPHA)/COLOR OF CHAR (GRAPHICS)
118      ; SEE NOTE ON WRITE DOT FOR BIT 7 OF BL = 1.
119      ; (AH) = 0AH WRITE CHARACTER ONLY AT CURRENT CURSOR POSITION
120      ; (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY)
121      ; (CX) = COUNT OF CHARACTERS TO WRITE
122      ; (AL) = CHAR TO WRITE
123      ; NOTE: USE FUNCTION (AH) = 09H IN GRAPHICS MODES
124      ; FOR READ/WRITE CHARACTER INTERFACE IN GRAPHICS MODE. THE
125      ; CHARACTERS ARE FORMED FROM A CHARACTER GENERATOR IMAGE
126      ; MAINTAINED IN THE SYSTEM ROM. ONLY THE 1ST 128 CHARS
127      ; ARE CONTAINED THERE. TO READ/WRITE THE SECOND 128 CHARS,
128      ; THE USER MUST INITIALIZE THE POINTER AT INTERRUPT 1FH
129      ; (LOCATION 0007CH) TO POINT TO THE 1K BYTE TABLE CONTAINING
130      ; THE CODE POINTS FOR THE SECOND 128 CHARS (128-255).
131      ; FOR WRITE CHARACTER INTERFACE IN GRAPHICS MODE, THE REPLICATION FACTOR
132      ; CONTAINED IN (CX) ON ENTRY WILL PRODUCE VALID RESULTS ONLY
133      ; FOR CHARACTERS CONTAINED ON THE SAME ROW. CONTINUATION TO
134      ; SUCCEEDING LINES WILL NOT PRODUCE CORRECTLY.
135
136      GRAPHICS INTERFACE
137
138      ; (AH) = 0BH SET COLOR PALETTE
139      ; (BH) = PALETTE COLOR ID BEING SET (0-127)
140      ; (BL) = COLOR VALUE TO BE USED WITH THAT COLOR ID
141      ; NOTE: FOR THE CURRENT COLOR CARD, THIS ENTRY POINT HAS
142      ; MEANING ONLY FOR 320x200 GRAPHICS.
143      ; COLOR ID = 0 SELECTS THE BACKGROUND COLOR (0-15)
144      ; COLOR ID = 1 SELECTS THE PALETTE TO BE USED:
145      ; 0 = GREEN(1)/RED(2)/YELLOW(3)
146      ; 1 = CYAN(11)/MAGENTA(2)/WHITE(3)
147      ; IN 40X25 OR 80X25 ALPHA MODES, THE VALUE SET FOR
148      ; PALETTE COLOR 0 INDICATES THE BORDER COLOR
149      ; TO BE USED (VALUES 0-31, WHERE 16-31 SELECT
150      ; THE HIGH INTENSITY BACKGROUND SET.
151
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      ; OR'ed 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
171      ; (AH) = 0FH CURRENT VIDEO STATE
172      ; (AL) = MODE CURRENTLY SET (SEE (AH) = 00H FOR EXPLANATION)
173      ; (AH) = NUMBER OF CHARACTER COLUMNS ON SCREEN
174      ; (BH) = CURRENT ACTIVE DISPLAY PAGE
175
176      ; (AH) = 10H RESERVED
177      ; (AH) = 11H RESERVED
178      ; (AH) = 12H RESERVED
179      ; (AH) = 13H WRITE STRING
180      ; ES:BP - POINTER TO STRING TO BE WRITTEN
181      ; CX - LENGTH OF CHARACTER STRING TO WRITTEN
182      ; DX - CURSOR POSITION FOR STRING TO BE WRITTEN
183      ; BH - PAGE NUMBER
184      ; (AL) = 00H WRITE CHARACTER STRING
185      ; BL - ATTRIBUTE
186      ; STRING IS <CHAR,CHAR, ... ,CHAR>
187      ; CURSOR NOT MOVED
188      ; (AL) = 01H WRITE CHARACTER STRING AND MOVE CURSOR
189      ; BL - ATTRIBUTE
190      ; STRING IS <CHAR,CHAR, ... ,CHAR>
191      ; CURSOR IS MOVED
192      ; (AL) = 02H WRITE CHARACTER AND ATTRIBUTE STRING
193      ; (VALID FOR ALPHA MODES ONLY)
194      ; STRING IS <CHAR,ATTR,CHAR,ATTR .. ,CHAR,ATTR>
195      ; CURSOR IS NOT MOVED
196      ; (AL) = 03H WRITE CHARACTER AND ATTRIBUTE STRING AND MOVE CURSOR
197      ; (VALID FOR ALPHA MODES ONLY)
198      ; STRING IS <CHAR,ATTR,CHAR,ATTR .. ,CHAR,ATTR>
199      ; CURSOR IS MOVED
200      ; NOTE: CARRIAGE RETURN, LINE FEED, BACKSPACE, AND BELL ARE
201      ; TREATED AS COMMANDS RATHER THAN PRINTABLE CHARACTERS.
202
203      ; BX,CX,DX,S1,D1,BP,SP,DS,ES,SS PRESERVED DURING CALLS EXCEPT FOR
204      ; BX,CX,DX RETURN VALUES ON FUNCTIONS 03H,04H,0DH AND 0DH. ON ALL CALLS
205      ; AX IS MODIFIED.
206
207      -----
208
209      ASSUME CS:CODE,DS:DATA,ES:NOTHING
210
211      MI DW OFFSET SET_MODE ; TABLE OF ROUTINES WITHIN VIDEO I/O
212      DW OFFSET SET_CTYPE
213      DW OFFSET SET_CPOS
214      DW OFFSET READ_CURSOR
215      DW OFFSET READ_LINEN
216      DW OFFSET ACT_DISP_PAGE
217      DW OFFSET SCROLL_UP
218      DW OFFSET SCROLL_DOWN
219      DW OFFSET READ_AC_CURRENT
220      DW OFFSET WRITE_AC_CURRENT
221      DW OFFSET WRITE_C_CURRENT
222      DW OFFSET SET_COLOR
223      DW OFFSET WRITE_DOT
224      DW OFFSET READ_DOT
225      DW OFFSET WRITE_TTY
226      DW OFFSET VIDEO_STATE
227      DW OFFSET VIDEO_RETURN ; RESERVED
228      DW OFFSET VIDEO_RETURN ; RESERVED
229      DW OFFSET VIDEO_RETURN ; RESERVED
230      DW OFFSET WRITE_STRING ; CASE 13H, WRITE STRING
231
232      M1L EQU $-MI

```

```

229 0028          VIDEO_10_1      PROC    NEAR          ; ENTRY POINT FOR ORG 0F065H
230 0028 FB      SETI          ; INTERRUPTS BACK ON
231 0029 FC      CLD              ; SET DIRECTION FORWARD
232 002A 80 FC 14 CMP    AH,M1L/2      ; TEST FOR WITHIN TABLE RANGE
233 002D 73 2F  JNB    M4              ; BRANCH TO EXIT IF NOT A VALID COMMAND
234
235 002F 06      PUSH    ES              ;
236 0030 1E      PUSH    DS              ; SAVE WORK AND PARAMETER REGISTERS
237 0031 52      PUSH    DX              ;
238 0032 51      PUSH    CX              ;
239 0033 53      PUSH    BX              ;
240 0034 56      PUSH    SI              ;
241 0035 57      PUSH    DI              ;
242 0036 55      PUSH    BP              ;
243 0037 BE ---- R MOV    SI,DATA          ; POINT DS: TO DATA SEGMENT
244 003A 8E DE    MOV    DS,SI          ;
245 003C 8F 00    MOV    SI,AX          ; SAVE COMMAND/DATA INTO (SI) REGISTER
246 003E AD 0010 R MOV    AL,BYTE PTR @EQUIP_FLAG ; GET THE EQUIPMENT FLAG VIDEO BITS
247 0041 24 30    AND    AL,30H         ; ISOLATE CRT SWITCHES
248 0043 3C 30    CMP    AL,30H         ; IS SETTING FOR MONOCHROME CARD?
249 0045 BF B800 MOV    DI,0B800H      ; GET SEGMENT FOR COLOR CARD
250 0048 75 03    JNE    M2              ; SKIP IF NOT MONOCHROME CARD
251 004A BF B000 MOV    DI,0B000H      ; ELSE GET SEGMENT FOR MONOCHROME CARD
252 004D
253 004D BE C7    MOV    ES,DI          ; SET UP TO POINT AT VIDEO MEMORY AREAS
254 004F 8A C4    MOV    AL,AH          ; PLACE COMMAND IN LOW BYTE OF (AX)
255 0051 98      CBW                    ; AND FORM BYTE OFFSET WITH COMMAND
256 0052 D1 E0    SAL    AX,1           ; TIMES 2 FOR WORD TABLE LOOKUP
257 0054 96      XCHG   SI,AX          ; MOVE OFFSET INTO LOOK UP REGISTER (SI)
258
259 0055 8A 26 0049 R MOV    AH,@CRT_MODE    ; AND RESTORE COMMAND/DATA INTO (AX)
260
261 0059 2E: FF A4 0000 R JMP    WORD PTR CS:[SI-OFFSET M1] ; GO TO SELECTED FUNCTION
262
263 005E          M4:          ; COMMAND NOT VALID
264 005E CF          ; DO NOTHING IF NOT IN VALID RANGE
265 005F
266
267          ; SET_MODE          ;
268          ; THIS ROUTINE INITIALIZES THE ATTACHMENT TO ;
269          ; THE SELECTED MODE. THE SCREEN IS BLANKED. ;
270          ; INPUT          ;
271          ; @EQUIP_FLAG BITS 5-4 = MODE/WIDTH          ;
272          ; 0 = MONOCHROME (FORCES MODE 7)          ;
273          ; 01 = COLOR ADAPTER 40x25 (MODE 0 DEFAULT) ;
274          ; 10 = COLOR ADAPTER 80x25 (MODE 2 DEFAULT) ;
275          ; (AL) = COLOR MODE REQUESTED ( RANGE 0 - 6 ) ;
276          ; OUTPUT          ;
277          ; NONE          ;
278
279 005F          SET_MODE    PROC    NEAR          ; ADDRESS OF COLOR CARD
280 005F BA 03D4    MOV    DX,03D4H        ; GET EQUIPMENT FLAGS SETTING
281 0062 8B 3E 0010 R MOV    DI,@EQUIP_FLAG ; ISOLATE CRT SWITCHES
282 0066 81 E7 0030 AND    DI,30H          ; IS BW CARD, INSTALLED AS PRIMARY
283 006A 83 FF 30  CMP    DI,30H          ; SKIP AND CHECK IF COLOR
284 006D 75 06    JNE    M5              ; ELSE INDICATE INTERNAL BW CARD MODE
285 006F B0 07    MOV    AL,7           ; ELSE INDICATE INTERNAL BW CARD MODE
286 0071 B2 B4    MOV    DL,0B4H        ; SET ADDRESS OF BW (MONOCHROME) CARD
287 0073 EB 0D    JMP    SHORT M8        ; CONTINUE WITH FORCED MODE 7
288
289 0075 3C 07    CMP    AL,7           ; CHECK FOR VALID COLOR MODES 0-6
290 0077 72 09    JB    M8              ; CONTINUE IF BELOW MODE 7
291 0079 B0 00    MOV    AL,0           ; FORCE DEFAULT 40x25 BW MODE
292 007B 83 FF 20  CMP    DI,20H        ; CHECK FOR @EQUIP_FLAG AT 80x25 BW
293 007E 74 02    JE    M8              ; CONTINUE WITH MODE 0 IF NOT
294 0080 B0 02    MOV    AL,2           ; ELSE FORCE MODE 2
295
296 0082 A2 0049 R MOV    @CRT_MODE,AL    ; SAVE MODE IN GLOBAL VARIABLE
297 0085 89 16 0063 R MOV    @ADDR_6845,DX   ; SAVE ADDRESS OF BASE
298 0089 C6 06 0084 R 18 MOV    @ROWS,25-1     ; INITIALIZE DEFAULT_ROW COUNT OF 25
299 008E 1E          PUSH    DS             ; SAVE POINTER TO DATA SEGMENT
300 008F 50          PUSH    AX             ; SAVE MODE NUMBER (AL)
301 0091 98      CBW                    ; CLEAR HIGH BYTE OF MODE
302 0091 8B F0    MOV    SI,AX          ; SET TABLE POINTER, INDEXED BY MODE
303 0093 2E: 8A 84 0000 E MOV    AL,CS:[SI + OFFSET M7] ; GET THE MODE SET VALUE FROM TABLE
304 0098 A2 0065 R MOV    @CRT_MODE_SET,AL ; SAVE THE MODE SET VALUE
305 009B 24 37    AND    AL,037H        ; VIDEO OFF, SAVE HIGH RESOLUTION BIT
306 009D 52      PUSH    DS             ; SAVE OUTPUT PORT VALUE
307 009E 83 C2 04 ADD    DX,4           ; POINT TO CONTROL REGISTER
308 00A1 EE      OUT    DX,AL          ; RESET VIDEO TO OFF TO SUPPRESS ROLLING
309 00A2 5A      POP    DX             ; BACK TO BASE REGISTER
310
311 00A3 2B 0B    SUB    BX,BX          ; SET UP FOR AB50 SEGMENT
312 00A5 8E DB    MOV    DS,BX          ; ESTABLISH VECTOR TABLE ADDRESSING
313 00A7 C5 1E 0074 R LDS    DS:@PARAM_PTR  ; GET POINTER TO VIDEO PARAMS
314
315 00AB 58          ASSUME DS:CODE        ;
316 00AC B9 0010 MOV    CX,16          ; RECOVER MODE NUMBER IN (AL)
317 00AF 3E 02    CMP    AL,2           ; LENGTH OF EACH ROW OF TABLE
318 00B1 72 0E    JC    M9              ; DETERMINE WHICH ONE TO USE
319 00B3 03 D9    ADD    BX,CX          ; MODE IS 0 OR 1
320 00B5 3C 04    CMP    AL,4           ; NEXT ROW OF INITIALIZATION TABLE
321 00B7 72 08    JC    M9              ;
322 00B9 03 D9    ADD    BX,CX          ; MODE IS 2 OR 3
323 00BB 3C 07    CMP    AL,7           ; MOVE TO GRAPHICS ROW OF INIT_TABLE
324 00BD 72 02    JC    M9              ; MODE IS 4,5, OR 6
325 00BF 03 D9    ADD    BX,CX          ; MOVE TO BW CARD ROW OF INIT_TABLE
326
327          ; ---- BX POINTS TO CORRECT ROW OF INITIALIZATION TABLE
328
329 00C1          M9:          ; OUT_INIT          ;
330 00C1 50      PUSH    AX            ; SAVE MODE IN (AL)
331 00C2 8B 47 0A MOV    AX,[BX+10]     ; GET THE CURSOR MODE FROM THE TABLE
332 00C5 86 E0    XCHG   AH,AL         ; PUT CURSOR MODE IN CORRECT POSITION
333 00C7 1E      PUSH    DS            ; SAVE TABLE SEGMENT POINTER
334
335 00C8 83 00    ASSUME DS:DATA        ;
336 00CA E8 0000 E CALL    DDS            ; POINT DS TO DATA SEGMENT
337 00CB A3 0060 R MOV    @CURSOR_MODE,AX ; PLACE INTO BIOS DATA SAVE AREA
338 00CE 1E      ASSUME DS:CODE        ;
339 00CF 32 E4    POP    DS            ; RESTORE THE TABLE SEGMENT POINTER
340 00CF 3F E4    XOR    AH,AH          ; AH IS REGISTER NUMBER DURING LOOP
341
342          ; ---- LOOP THROUGH TABLE, OUTPUTTING REGISTER ADDRESS, THEN VALUE FROM TABLE
    
```

```

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

```

```

457 0165 EE          OUT    DX,AL
458 0166 C3          RET
459 0167          SET_CTYPE    ENDP
-----
461
462 | SET_CPOS
463 | THIS ROUTINE SETS THE CURRENT CURSOR POSITION TO THE
464 | NEW X-Y VALUES PASSED
465 |
466 | INPUT
467 |     DX - ROW,COLUMN OF NEW CURSOR
468 |     BH - DISPLAY PAGE OF CURSOR
469 | OUTPUT
470 |     CURSOR IS SET AT 6845 IF DISPLAY PAGE IS CURRENT DISPLAY
-----
471 0167          SET_CPOS    PROC    NEAR
472 0167 8A C7      MOV    AL,BH          ; MOVE PAGE NUMBER TO WORK REGISTER
473 0169 98        CBW          ; CONVERT PAGE TO WORD VALUE
474 016A D1 E0     SAL    AX,1          ; WORD OFFSET
475 016C 96       XCHG   AX,S1         ; USE INDEX REGISTER
476 016D 89 94 0050 R  MOV    [S1-OFFSET+CURSOR_POSN],DX ; SAVE THE POINTER
477 0171 38 3E 0062 R  CMP    #ACTIVE_PAGE,BH        ; SET CPOS_RETURN
478 0175 75 05     JNZ    M18              ; GET_ROW/COLUMN TO AX
479 0177 8B C2     MOV    AX,DX          ; CURSOR_SET
480 0179 E8 017E R  CALL   M18            ; SET_CPOS_RETURN
481 017C          M18:
482 017C EB BF     JMP    VIDEO_RETURN
483 017E          SET_CPOS    ENDP
-----
484
485 |----- SET CURSOR POSITION, AX HAS ROW/COLUMN FOR CURSOR
486
487 017E          M18  PROC    NEAR
488 017E E8 0200 R  CALL   POSITION        ; DETERMINE LOCATION IN REGEN BUFFER
489 0181 8B C8     MOV    CX,AX          ;
490 0183 03 0E 004E R  ADD    CX,#CRT_START    ; ADD IN THE START ADDRESS FOR THIS PAGE
491 0187 D1 F9     SAR    CX,1           ; DIVIDE BY 2 FOR CHAR ONLY COUNT
492 0189 B4 0E     MOV    AH,14          ; REGISTER NUMBER FOR CURSOR
493 018B E8 0151 R  CALL   M16            ; OUTPUT THE VALUE TO THE 6845
494 018E C3       RET
495 018F          M18  ENDP
-----
496
497 | 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 018F          READ_CURSOR  PROC    NEAR
507 018F 8A DF      MOV    BL,BH          ;
508 0191 32 FF     PUSH   BH,BH         ;
509 0193 D1 E3     SAL    BX,1          ; WORD OFFSET
510 0195 8B 97 0050 R  MOV    DX,[BX+OFFSET+CURSOR_POSN] ;
511 0199 8B 0E 0060 R  MOV    CX,#CURSOR_MODE ;
512 019D 5D       POP    BP
513 019E 5F       POP    DI
514 019F 5E       POP    SI
515 01A0 5B       POP    BX
516 01A1 58       POP    AX            ; DISCARD SAVED CX AND DX
517 01A2 58       POP    AX
518 01A3 5F       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 | OUTPUT
530 |     THE 6845 IS RESET TO DISPLAY THAT PAGE
-----
531 01A6          ACT_DISP_PAGE  PROC    NEAR
532 01A6 A2 0062 R  MOV    #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   ; SAVE START ADDRESS FOR LATER
537 01B2 8B C8     MOV    CX,AX          ; START ADDRESS TO CX
538 01B4 D1 F9     SAR    CX,1           ; DIVIDE BY 2 FOR 6845 HANDLING
539 01B6 B4 0C     MOV    AH,12         ; 6845 REGISTER FOR START ADDRESS
540 01B8 E8 0151 R  CALL   M16            ; RECOVER PAGE VALUE
541 01BB 5B       POP    BX            ; *2 FOR WORD OFFSET
542 01BC D1 E3     SAL    BX,1          ;
543 01BE 8B 07 0050 R  MOV    AX,[BX + OFFSET+CURSOR_POSN] ; GET CURSOR FOR THIS PAGE
544 01C2 E8 017E R  CALL   M18            ; SET THE CURSOR POSITION
545 01C5 E9 013D R  JMP    VIDEO_RETURN
546 01C8          ACT_DISP_PAGE  ENDP
-----
547
548 | 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 LOW BIT OF BL
558 |     0 = GREEN, RED, YELLOW FOR COLORS 1,2,3
559 |     1 = BLUE, CYAN, MAGENTA FOR COLORS 1,2,3
560 | OUTPUT
561 |     THE COLOR SELECTION IS UPDATED
-----
562
563 01C8          SET_COLOR    PROC    NEAR
564 01C8 8B 15 0063 R  MOV    DX,#ADDR_6845 ; I/O PORT FOR PALETTE
565 01CC 83 C2 05   ADD    DX,5           ; OVERSCAN PORT
566 01CF A0 0066 R  MOV    AL,#CRT_PALETTE ; GET THE CURRENT PALETTE VALUE
567 01D2 0A FF     OR    BH,BH          ; IS THIS COLOR 0?
568 01D4 75 0E     JNZ    M20            ; OUTPUT COLOR 1
569
570 |----- HANDLE COLOR 0 BY SETTING THE BACKGROUND COLOR

```

```

571                                     AND    AL,0E0H          ; TURN OFF LOW 5 BITS OF CURRENT
572 01D6 24 E0                          OR     BL,01FH          ; TURN OFF HIGH 3 BITS OF INPUT VALUE
573 01D8 80 E3 1F                        OR     AL,BL          ; PUT VALUE INTO REGISTER
574 01DB 0A C3                            M19:   AND    AL,BL          ; OUTPUT THE PALETTE
575 01DD                                OUT    DX,AL          ; OUTPUT COLOR SELECTION TO 3D9 PORT
576 01DD EE                                MOV    %CRT_PALETTE,AL ; SAVE THE COLOR VALUE
577 01DE A2 0066 R                        JMP    VIDEO_RETURN
578 01E1 E9 013D R
579
580 ;----- HANDLE COLOR 1 BY SELECTING THE PALETTE TO BE USED
581
582 01E4 M20:                             AND    AL,0DFH          ; TURN OFF PALETTE SELECT BIT
583 01E4 24 DF                            SHR    BL,1            ; TEST THE LOW ORDER BIT OF BL
584 01E6 D0 EB                            JNC   M19              ; ALREADY DONE
585 01E8 73 F3                            OR     AL,20H          ; TURN ON PALETTE SELECT BIT
586 01EA 0C 20                            JMP    M19              ; GO DO IT
587 01EC EB EF
588 01EE
589
590 ; 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 PROC NEAR
597 01EE 8A 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 01F9 5F                                POP    BP                 ; RECOVER REGISTERS
601 01FA 5F                                POP    DI
602 01FB 5E                                POP    SI
603 01FC 59                                POP    CX                 ; DISCARD SAVED BX
604 01FD E9 0141 R                           JMP    M15                ; RETURN TO CALLER
605
606 VIDEO_STATE ENDP
607
608 ; POSITION
609 ; THIS SERVICE ROUTINE CALCULATES THE REGEN BUFFER ADDRESS
610 ; OF A CHARACTER IN THE ALPHA MODE
611 ; INPUT
612 ; AX = ROW, COLUMN POSITION
613 ; AX = OFFSET OF CHAR POSITION IN REGEN BUFFER
614
615 0200 POSITION PROC NEAR
616 0200 53                                PUSH   BX                 ; SAVE REGISTER
617 0201 93                                XCHG  BX,AX              ; SAVE ROW/COLUMN POSITION IN (BX)
618 0202 A0 004A R                           MOV    AL,BYTE PTR %CRT_COLS ; GET COLUMNS PER ROW COUNT
619 0205 F4 E7                                MUL   BL,BL               ; DETERMINE BYTES TO ROW
620 0207 32 FF                                XOR   BH,BH
621 0209 03 C3                                ADD   AX,BX               ; ADD IN COLUMN VALUE
622 020B D1 E0                                SAL   AX,1                ; * 2 FOR ATTRIBUTE BYTES
623 020D 5B                                POP    BP
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 ; INPUT
631 ; (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
640 ; NONE -- THE REGEN BUFFER IS MODIFIED
641
642 020F ASSUME DS:DATA,ES:DATA
643 SCROLL_UP PROC NEAR
644 020F EB 02EA R                           CALL   TEST_LINE_COUNT    ; TEST FOR GRAPHICS MODE
645 0212 80 FC 04                            CMP    AH,4               ; HANDLE SEPARATELY
646 0215 72 08                                JC     N1                  ; TEST FOR BW CARD
647 0217 80 FC 07                            CMP    AH,7
648 021A 74 03                                JE     N1
649 021C E9 044C R                           JMP    GRAPHICS_UP
650 021F
651 021F 53 N1:                             PUSH   BX                 ; UP CONTINUE
652 0220 8B C1                               MOV    AX,CX              ; SAVE FILL ATTRIBUTE IN BH
653 0222 EB 025C R                           CALL   SCROLL_POSITION    ; UPPER LEFT POSITION
654 0225 74 31                               JC     N2                  ; DO SETUP FOR SCROLL
655 0227 03 F0                               JZ     N7                  ; BLANK FIELD
656 0229 8A E6                               MOV    SI,AX              ; FROM ADDRESS
657 022B 2A E3                               MOV    AH,DH              ; # ROWS IN BLOCK
658 022D                                SUB    AH,BL              ; # ROWS TO BE MOVED
659 022D EB 029D R                           ROW_LOOP:                ; ROW_LOOP
660 0230 03 F5                               ADD    SI,BP              ; MOVE ONE ROW
661 0232 03 FD                               ADD    D1,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                                CLEAR_ENTRY:             ; CLEAR ENTRY
665 0238 58 N3:                             POP    AX                 ; RECOVER ATTRIBUTE IN AH
666 0239 B0 20                               MOV    AL,' '             ; FILL WITH BLANKS
667 023B                                CLEAR_LOOP:              ; CLEAR_LOOP
668 023B EB 02A6 R                           CALL   N11                ; CLEAR THE ROW
669 023E 03 FD                               ADD    D1,BP              ; POINT TO NEXT LINE
670 0240 FE CB                               DEC    BL                 ; COUNTER OF LINES TO SCROLL
671 0242 75 F7                               JNZ   N4                  ; CLEAR_LOOP
672 0244                                SCROLL_END:              ; SCROLL_END
673 0244 EB 0000 E N5:                             CALL   DDS                 ; IS THIS THE BLACK AND WHITE CARD
674 0247 80 3E 0049 R 07                   CMP    %CRT_MODE,7
675 024C 74 07                               JE     N6                  ; IF SO, SKIP THE MODE RESET
676 024E A0 0065 R                           MOV    AL,%CRT_MODE_SET  ; GET THE VALUE OF THE MODE SET
677 0251 BA 03D8 R                           MOV    DX,03D8H           ; ALWAYS SET COLOR CARD PORT
678 0254 EE                                OUT    DX,AL
679 0255                                N6:                       ; VIDEO_RET_HERE
680 0255 E9 013D R                           JMP    VIDEO_RETURN
681 0258 N7:                             MOV    BL,DH              ; BLANK_FIELD
682 025B 8A DE                               MOV    BL,0               ; GET ROW COUNT
683 025A EB DC                               JMP    N3                  ; GO CLEAR THAT AREA
684 025C SCROLL_UP ENDP
    
```

SECTION 5

```

685
686
687
688 025C
689 025C E8 0200 R
690 025F 03 06 004E R
691 0263 8B F8
692 0265 8F F0
693 0267 2B D1
694 0269 FE C6
695 026B FE C2
696 026D 32 ED
697 026F 8B 2E 004A R
698 0273 03 ED
699 0275 A0 004A R
700 0278 FA E3
701 027A 03 C0
702 027C 50
703 027D A0 0049 R
704 0280 06
705 0281 1F
706 0282 3C 02
707 0284 72 13
708 0286 3C 03
709 0288 77 0F
710
711 028A 52
712 028B BA 03DA
713 028E
714 028E EC
715 028F A8 08
716 0291 74 FB
717 0293 80 25
718 0295 82 D8
719 0297 EE
720 0298 5A
721 0299
722 0299 58
723 029A 0A DB
724 029C C3
725 029D
726
727
728 029D
729 029D 8A CA
730 029F 86
731 02A0 57
732 02A1 F3/ A5
733 02A3 5F
734 02A4 5E
735 02A5 C3
736 02A6
737
738
739 02A6
740 02A6 8A CA
741 02AB 57
742 02A9 F3/ AB
743 02AB 5F
744 02AC C3
745 02AD
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762 02AD
763 02AD FD
764 02AE E8 02EA R
765 02B1 80 FC 04
766 02B4 72 08
767 02B6 80 FC 07
768 02B9 74 03
769 02BB E9 0503 R
770 02BE
771 02BE 53
772 02BF 8B C2
773 02C1 E8 025C R
774 02C4 74 20
775 02C6 2B F0
776 02C8 8A E6
777 02CA 2A E3
778 02CC
779 02CC E8 029D R
780 02CF 2B F5
781 02D1 2B FD
782 02D3 FE CC
783 02D5 75 F5
784 02D7
785 02D7 58
786 02D8 80 20
787 02DA
788 02DA E8 02A6 R
789 02DD 2B FD
790 02DF FE CB
791 02E1 75 F7
792 02E3 E9 0244 R
793 02E6
794 02E6 8A DE
795 02E8 EB ED
796 02EA

```

```

          I----- HANDLE COMMON SCROLL SET UP HERE
SCROLL_POSITION PROC NEAR
          CALL POSITION NEAR
          ADD AX,#CRT_START
          MOV D1,AX
          MOV S1,AX
          SUB DX,CX
          INC DH
          INC DL
          XOR CH,CH
          MOV BP,#CRT_COLS
          BP,#BP
          MOV AL,BYTE PTR #CRT_COLS
          MUL BL
          ADD AX,AX
          PUSH AX
          MOV AL,#CRT_MODE
          PUSH ES
          POP DS
          CMP AL,2
          JNB N9
          CMP AL,3
          JA N9
          I-----
          PUSH DX
          MOV DX,3DAH
          N8:
          IN AL,DX
          TEST AL,RVRT
          JZ N8
          MOV AL,25H
          MOV DL,0DBH
          OUT DX,AL
          POP DX
          I 80X25 COLOR CARD SCROLL
          I GUARANTEED TO BE COLOR CARD HERE
          I WAIT_DISP_ENABLE
          I GET_PORT
          I WAIT FOR VERTICAL RETRACE
          I WAIT_DISP_ENABLE
          N9:
          POP AX
          OR BL,BL
          RET
          I RESTORE LINE COUNT
          I 0 SCROLL MEANS BLANK FIELD
          I RETURN WITH FLAGS SET
SCROLL_POSITION ENDP
          I----- MOVE_ROW
          N10 PROC NEAR
          MOV CL,DL
          PUSH SI
          PUSH D1
          REP MOVSW
          POP D1
          POP SI
          RET
          I GET # OF COLS TO MOVE
          I SAVE START ADDRESS
          I MOVE THAT LINE ON SCREEN
          I RECOVER ADDRESSES
          N10 ENDP
          I----- CLEAR_ROW
          N11 PROC NEAR
          MOV CL,DL
          PUSH D1
          REP STOSW
          POP D1
          RET
          I GET # COLUMNS TO CLEAR
          I STORE THE FILL CHARACTER
          N11 ENDP
          I-----
          I SCROLL_DOWN
          I THIS ROUTINE MOVES THE CHARACTERS WITHIN A DEFINED
          I BLOCK DOWN ON THE SCREEN, FILLING THE TOP LINES
          I WITH A DEFINED CHARACTER
          I INPUT
          I (AH) = CURRENT CRT MODE
          I (AL) = NUMBER OF LINES TO SCROLL
          I (CX) = UPPER LEFT CORNER OF REGION
          I (DX) = LOWER RIGHT CORNER OF REGION
          I (BH) = FILL CHARACTER
          I (DS) = DATA SEGMENT
          I (ES) = REGEN SEGMENT
          I OUTPUT
          I NONE -- SCREEN IS SCROLLED
          I-----
          SCROLL_DOWN PROC NEAR
          STD
          CALL TEST_LINE_COUNT
          CMP AH,4
          JC N12
          CMP AH,7
          JE N12
          JMP GRAPHICS_DOWN
          I DIRECTION FOR SCROLL DOWN
          I TEST FOR GRAPHICS
          I TEST FOR BW CARD
          N12:
          PUSH BX
          MOV AX,DX
          CALL SCROLL_POSITION
          JZ N16
          SUB SI,AX
          MOV AH,DH
          SUB AH,BL
          I CONTINUE DOWN
          I SAVE ATTRIBUTE IN BH
          I LOWER RIGHT CORNER
          I GET REGEN LOCATION
          I SI IS FROM ADDRESS
          I GET TOTAL # ROWS
          I COUNT TO MOVE IN SCROLL
          N13:
          CALL N10
          SUB SI,BP
          SUB D1,BP
          DEC AH
          JNZ N13
          I MOVE ONE ROW
          N14:
          POP AX
          MOV AL,' '
          I RECOVER ATTRIBUTE IN AH
          N15:
          CALL N11
          SUB D1,BP
          DEC BL
          JNZ N15
          I CLEAR ONE ROW
          I GO TO NEXT ROW
          N16:
          JMP N5
          I SCROLL_END
          MOV BL,DH
          JMP N14
          SCROLL_DOWN ENDP

```



```

797
798
799
800
801 02EA
802
803 02EA 8A D8
804 02EC 0A C0
805 02EE 74 0E
806 02F0 50
807 02F1 8A C6
808 02F3 2A C6
809 02F5 FE C0
810 02F7 3A C3
811 02F9 58
812 02FA 75 02
813 02FC 2A DB
814 02FE
815 02FE C3
816 02FF
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833 02FF
834 02FF 80 FC 04
835 0302 72 08
836
837 0304 80 FC 07
838 0307 74 03
839
840 0309 E9 063E R
841 030C
842 030C E8 0328 R
843 030F 8B F1
844 0311 06
845 0312 1F
846
847
848
849 0313 0A DB
850 0315 75 0D
851 0317
852 0317 FB
853 0318 90
854 0319 FA
855 031A EC
856 031B 8A 01
857 031D 75 F8
858 031F
859 031F EC
860 0320 8A 09
861 0322 74 FB
862 0324
863 0324 AD
864 0325 E9 013D R
865
866 0328
867
868
869
870 0328
871 0328 86 E3
872 032A 8B E8
873 032C 80 EB 02
874 032F 00 EB
875 0331 8A C7
876 0333 98
877 0334 8B F8
878 0336 D1 E7
879 0338 8B 95 0050 R
880 033C 74 09
881
882 033E 33 FF
883 0340
884 0340 03 3E 004C R
885 0344 48
886 0345 75 F9
887
888 0347
889 0347 A0 004A R
890 034A F6 E6
891 034C 32 F6
892 034E 03 C2
893 0350 D1 E0
894 0352 03 F8
895 0354 8B 16 0063 R
896 0358 83 C2 06
897 035B C3
898
899 035C

```

```

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

TEST_LINE_COUNT PROC NEAR
    MOV BL,AL          ; SAVE LINE COUNT IN BL
    OR AL,AL          ; TEST IF AL IS ALREADY ZERO
    JZ BL_SET         ; IF IT IS THEN RETURN...
    PUSH AX           ; SAVE AX
    MOV AL,DH         ; SUBTRACT LOWER ROW FROM UPPER ROW
    SUB AL,CH
    INC AL            ; ADJUST DIFFERENCE BY 1
    CMP AL,BL        ; LINE COUNT = AMOUNT OF ROWS IN WINDOW?
    POP AX            ; RESTORE AX
    JNE BL_SET       ; IF NOT THEN WE'RE ALL SET
    SUB BL,BL        ; OTHERWISE SET BL TO ZERO
BL_SET: RET
TEST_LINE_COUNT ENDP

```

```

-----
; READ_AC_CURRENT
; THIS ROUTINE READS THE ATTRIBUTE AND CHARACTER AT THE CURRENT
; CURSOR POSITION AND RETURNS THEM TO THE CALLER
; INPUT
; (AH) = CURRENT CRT MODE
; (BH) = DISPLAY PAGE ( ALPHA MODES ONLY )
; (DS) = DATA SEGMENT
; (ES) = REGEN SEGMENT
; OUTPUT
; (AL) = CHARACTER READ
; (AH) = ATTRIBUTE READ
;-----
ASSUME DS:DATA,ES:DATA

READ_AC_CURRENT PROC NEAR
    CMP AH,4          ; IS THIS GRAPHICS
    JC P10
    CMP AH,7          ; IS THIS BW CARD
    JE P10
    JMP GRAPHICS_READ

P10: CALL FIND_POSITION ; READ AC CONTINUE
    MOV SI,DI         ; GET REGEN LOCATION AND PORT ADDRESS
    PUSH ES           ; ESTABLISH ADDRESSING IN SI
    POP DS            ; GET REGEN SEGMENT FOR QUICK ACCESS

I----- WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
    OR BL,BL          ; CHECK MODE FLAG FOR COLOR CARD IN 80
    P11: JNZ P13       ; ELSE SKIP RETRACE.WAIT - DO FAST READ
           ; WAIT FOR HORIZ RETRACE LOW OR VERTICAL
           ; ENABLE INTERRUPTS FIRST
           ; ALLOW FOR SMALL INTERRUPT WINDOW
           ; BLOCK INTERRUPTS FOR SINGLE LOOP
           ; GET STATUS FROM THE ADAPTER
           ; IS HORIZONTAL RETRACE LOW
           ; WAIT UNTIL IT IS
           ; NOW WAIT FOR EITHER RETRACE HIGH
           ; GET STATUS
           ; IS HORIZONTAL OR VERTICAL RETRACE HIGH
           ; WAIT UNTIL EITHER IS ACTIVE
    P12: IN AL,DX
           ; GET STATUS
           ; TEST AL,RVRT+RHRZ
           ; IS HORIZONTAL OR VERTICAL RETRACE HIGH
           ; JZ P12
    P13: LODSW VIDEO_RETURN ; GET THE CHARACTER AND ATTRIBUTE
           ; EXIT WITH (AX)

READ_AC_CURRENT ENDP

```

```

FIND_POSITION PROC NEAR
    ; SETUP FOR BUFFER READ OR WRITE
    XCHG AH,BL
    MOV BP,AX
    SUB BL,2
    SHR BL,1
    MOV AL,BH
    MOV CBW
    MOV DI,AX
    SAL DI,1
    MOV DX,DI+OFFSET *CURSOR_POSN ; GET ROW/COLUMN OF THAT PAGE
    JZ P21 ; SKIP BUFFER ADJUSTMENT IF PAGE ZERO
    XOR DI,DI ; ELSE SET BUFFER START ADDRESS TO ZERO
P20: ADD DI,*CRT_LEN ; ADD LENGTH OF BUFFER FOR ONE PAGE
    DEC AX ; DECREMENT PAGE COUNT
    JNZ P20 ; LOOP TILL PAGE COUNT EXHAUSTED
P21: ; DETERMINE LOCATION IN REGEN IN PAGE
    MOV AL,BYTE PTR *CRT_COLS ; GET COLUMNS PER ROW COUNT
    MUL DH ; DETERMINE BYTES TO ROW
    XOR AH,DX
    ADD AX,DX
    ; ADD IN COLUMN VALUE
    ; * 2 FOR ATTRIBUTE BYTES
    ADD LOCATION TO START OF REGEN PAGE
    ; GET BASE ADDRESS OF ACTIVE DISPLAY
    ; DX = STATUS PORT ADDRESS OF ADAPTER
    ; BP = ATTRIBUTE/CHARACTER (FROM BL/AL)
    ; DI = POSITION (OFFSET IN REGEN BUFFER)
    ; BL = MODE FLAG (ZERO FOR 80X25 COLOR)
FIND_POSITION ENDP

```

```

900 PAGE
901 -----
902 | WRITE_AC_CURRENT
903 | THIS ROUTINE WRITES THE ATTRIBUTE AND CHARACTER
904 | AT THE CURRENT CURSOR POSITION
905 |
906 | INPUT
907 | (AH) = CURRENT CRT MODE
908 | (BH) = DISPLAY PAGE
909 | (CX) = COUNT OF CHARACTERS TO WRITE
910 | (AL) = CHAR TO WRITE
911 | (BL) = ATTRIBUTE OF CHAR TO WRITE
912 | (DS) = DATA SEGMENT
913 | (ES) = REGEN SEGMENT
914 |
915 | OUTPUT
916 | DISPLAY REGEN BUFFER UPDATED
917 -----
918 038C      WRITE_AC_CURRENT      PROC    NEAR
919 038C 80 FC 04      CMP     AH,4      | IS THIS GRAPHICS
920 038F 72 08      JC     P30       |
921 0361 80 FC 07      CMP     AH,7      | IS THIS BW CARD
922 0364 74 03      JE     P30       |
923 0369      JMP     GRAPHICS_WRITE
924 0369 E8 0328 R    P30:      CALL    FIND_POSITION | WRITE_AC_CONTINUE
925      | GET REGEN LOCATION AND PORT ADDRESS
926 036C 0A DB      OR     BL,BL     | ADDRESS IN (DI) REGISTER
927 036E 74 06      JZ     P32       | CHECK MODE FLAG FOR COLOR CARD AT 80
928      | SKIP TO RETRACE WAIT IF COLOR AT 80
929 0370 95      XCHG  AX,BP     | GET THE ATTR/CHAR SAVED FOR FAST WRITE
930 0371 F3/ AB      REP   STOSW    | STRING WRITE THE ATTRIBUTE & CHARACTER
931 0373 EB 16      JMP   SHORT P35 | EXIT FAST WRITE ROUTINE
932
933 |----- WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
934
935 0375      P31:      XCHG  BP,AX     | LOOP FOR EACH ATTR/CHAR WRITE
936 0375 95      XCHG  BP,AX     | PLACE ATTR/CHAR BACK IN SAVE REGISTER
937 0376      P32:      STI     | ENABLE INTERRUPTS FIRST
938 0376 FB      NOP     | ALLOW FOR INTERRUPT WINDOW
939 0377 90      CLI     | BLOCK INTERRUPTS FOR SINGLE LOOP
940 0378 FA      IN     AL,DX    | GET STATUS FROM THE ADAPTER
941 0379 EC      TEST  AL,RVRT  | CHECK FOR VERTICAL RETRACE FIRST
942 037A A8 08      JNZ   P34       | DO FAST WRITE NOW IF VERTICAL RETRACE
943 037C 75 09      TEST  AL,RHRZ  | IS HORIZONTAL RETRACE LOW THEN
944 037E A8 01      JNZ   P32       | WAIT UNTIL IT IS
945 0380 75 F4      | WAIT FOR EITHER RETRACE HIGH
946 0382      P33:      IN     AL,DX    | GET STATUS AGAIN
947 0382 EC      TEST  AL,RVRT+RHRZ | IS HORIZONTAL OR VERTICAL RETRACE HIGH
948 0383 A8 09      JZ     P33       | WAIT UNTIL EITHER IS ACTIVE
949 0385 74 FB      P34:      XCHG  AX,BP     | GET THE ATTR/CHAR SAVED IN (BP)
950 0387      STOSW    | WRITE THE ATTRIBUTE AND CHARACTER
951 0387 95      LOOP  P31       | AS MANY TIMES AS REQUESTED - TILL CX=0
952 0388 AB      P35:      JMP     VIDEO_RETURN | EXIT
953 0389 E2 EA
954 038B
955 038B E9 013D R
956
957 038E      WRITE_AC_CURRENT      ENDP
958
959 |-----
960 | WRITE_C_CURRENT
961 | THIS ROUTINE WRITES THE CHARACTER AT
962 | THE CURRENT CURSOR POSITION, ATTRIBUTE UNCHANGED
963 |
964 | INPUT
965 | (AH) = CURRENT CRT MODE
966 | (BH) = DISPLAY PAGE
967 | (CX) = COUNT OF CHARACTERS TO WRITE
968 | (AL) = CHAR TO WRITE
969 | (DS) = DATA SEGMENT
970 | (ES) = REGEN SEGMENT
971 |
972 | OUTPUT
973 | DISPLAY REGEN BUFFER UPDATED
974 -----
975 038E      WRITE_C_CURRENT      PROC    NEAR
976 038E 80 FC 04      CMP     AH,4      | IS THIS GRAPHICS
977 0391 72 08      JC     P40       |
978 0393 80 FC 07      CMP     AH,7      | IS THIS BW CARD
979 0396 74 03      JE     P40       |
980 0398      JMP     GRAPHICS_WRITE
981 0398 E8 0328 R    P40:      CALL    FIND_POSITION | GET REGEN LOCATION AND PORT ADDRESS
982      | ADDRESS OF LOCATION IN (DI)
983
984 |----- WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
985
986 039E      P41:      STI     | WAIT FOR HORZ RETRACE LOW OR VERTICAL
987 039E FB      OR     BL,BL     | ENABLE INTERRUPTS FIRST
988 039F 0A DB      OR     BL,BL     | CHECK MODE FLAG FOR COLOR CARD IN 80
989 03A1 75 0F      JNZ   P43       | ELSE SKIP RETRACE WAIT - DO FAST WRITE
990 03A3 FA      CLI     | BLOCK INTERRUPTS FOR SINGLE LOOP
991 03A4 EC      IN     AL,DX    | GET STATUS FROM THE ADAPTER
992 03A5 A8 08      TEST  AL,RVRT  | CHECK FOR VERTICAL RETRACE FIRST
993 03A7 75 09      JNZ   P43       | DO FAST WRITE NOW IF VERTICAL RETRACE
994 03A9 A8 01      JNZ   AL,RHRZ  | IS HORIZONTAL RETRACE LOW THEN
995 03AB 75 F1      JNZ   P41       | WAIT UNTIL IT IS
996 03AD      P42:      IN     AL,DX    | WAIT FOR EITHER RETRACE HIGH
997 03AD EC      TEST  AL,RVRT+RHRZ | GET STATUS AGAIN
998 03AE A8 09      JZ     P42       | IS HORIZONTAL OR VERTICAL RETRACE HIGH
999 03B0 74 FB      | WAIT UNTIL EITHER RETRACE ACTIVE
1000 03B2
1001 03B2 8B C5      P43:      MOV    AX,BP     | GET THE CHARACTER SAVED IN (BP)
1002 03B4 AA      STOSB    | PUT THE CHARACTER INTO REGEN BUFFER
1003 03B5 47      INC    DI       | BUMP POINTER PAST ATTRIBUTE
1004 03B6 E2 E6      LOOP  P41       | AS MANY TIMES AS REQUESTED
1005
1006 03B8 E9 013D R    JMP     VIDEO_RETURN
1007
1008 03BB      WRITE_C_CURRENT      ENDP
    
```

```

1009 PAGE
1010 -----
1011 | WRITE_STRING THIS ROUTINE WRITES A STRING OF CHARACTERS TO THE CRT. |
1012 | INPUT |
1013 | |
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 03BB 55 PUSH BP ; SAVE BUFFER OFFSET (BP) IN STACK
1026 03BC 8B EC MOV BP,SP ; GET POINTER TO STACKED REGISTERS
1027 03BE 8E 46 10 MOV ES,[BP]+14+2 ; RECOVER ENTRY (ES) SEGMENT REGISTER
1028 03C1 5D POP BP ; RESTORE BUFFER OFFSET
1029 03C2 98 CBW ; CLEAR (AH) REGISTER
1030 03C3 8B F8 MOV DI,AX ; SAVE (AL) COMMAND IN (DI) REGISTER
1031 03C5 3C 04 CMP AL,04 ; TEST FOR INVALID WRITE STRING OPTION
1032 03C7 73 73 JNB P59 ; IF OPTION INVALID THEN RETURN
1033 |
1034 03C9 E3 71 JCXZ P59 ; IF ZERO LENGTH STRING THEN RETURN
1035 |
1036 03CB 8B F3 MOV SI,BX ; SAVE CURRENT CURSOR PAGE
1037 03CD 8A 0F MOV BL,BH ; MOVE PAGE TO LOW BYTE
1038 03CF 32 FF XOR BH,BH ; CLEAR HIGH BYTE
1039 03D1 87 F3 XCHG SI,BX ; MOVE OFFSET AND RESTORE PAGE REGISTER
1040 03D3 D1 E6 SAL SI,1 ; CONVERT TO PAGE OFFSET (SI* PAGE)
1041 03D5 FF B4 0050 R PUSH AX,[SI+OFFSET *CURSOR_POSN] ; SAVE CURRENT CURSOR POSITION IN STACK
1042 03D9 B8 0200 MOV AX,0200H ; SET NEW CURSOR POSITION
1043 03DC CD 10 INT 10H
1044 03DE P50: MOV AL,ES:[BP] ; GET CHARACTER FROM INPUT STRING
1045 03E0 26: 8A 46 00 INC BP ; BUMP POINTER TO CHARACTER
1046 03E2 45
1047 |
1048 |----- TEST FOR SPECIAL CHARACTER'S
1049 |
1050 03E3 3C 08 CMP AL,08H ; IS IT A BACKSPACE
1051 03E5 74 0C JE P51 ; BACK SPACE
1052 03E7 3C 0D CMP AL,CR ; IS IT CARRIAGE RETURN
1053 03E9 74 08 JE P51 ; GAT RET
1054 03EB 3C 0A CMP AL,LF ; IS IT A LINE FEED
1055 03ED 74 04 JE P51 ; LINE FEED
1056 03EF 3C 07 CMP AL,07H ; IS IT A BELL
1057 03F1 75 04 JNE P52 ; IF NOT THEN DO WRITE CHARACTER
1058 03F3 P51: MOV AH,0EH ; TTY_CHARACTER WRITE
1059 03F5 B4 0E INT 10H ; WRITE TTY CHARACTER TO THE CRT
1060 03F7 CD 10 MOV DX,[SI+OFFSET *CURSOR_POSN] ; GET CURRENT CURSOR POSITION
1061 03F7 8B 94 0050 R MOV SHORT P54 ; SET CURSOR POSITION AND CONTINUE
1062 03FB EB 2D JMP
1063 |
1064 03FD P52: PUSH CX
1065 03FD 51 PUSH BX
1066 03FE 53 MOV CX,1 ; SET CHARACTER WRITE AMOUNT TO ONE
1067 03FF B9 0001 MOV D1,2 ; IS THE ATTRIBUTE IN THE STRING
1068 0402 83 FF 02 CMP D1,2 ; IF NOT THEN SKIP
1069 0405 72 05 JB P53 ; ELSE GET NEW ATTRIBUTE
1070 0407 26: 8A SE 00 MOV BL,ES:[BP] ; ESTABLISH NEW CURSOR POSITION
1071 040B 45 INC BP ; BUMP STRING POINTER
1072 040C P53: MOV AH,09H ; GOT CHARACTER
1073 040C B4 09 INT 10H ; WRITE CHARACTER TO THE CRT
1074 040E CD 10 MOV BX,0 ; RESTORE REGISTERS
1075 0410 5B POP BX
1076 0411 59 POP CX
1077 0412 FE C2 INC DL ; INCREMENT COLUMN COUNTER
1078 0414 3A 16 004A R CMP DL,BYTE PTR *CRT_COLS ; IF COLS ARE WITHIN RANGE FOR THIS MODE
1079 0418 72 16 JB P54 ; THEN GO TO COLUMNS SET
1080 041A FE C6 INC DH ; BUMP ROW COUNTER BY ONE
1081 041C 2A D2 SUB DL,DL ; SET COLUMN COUNTER TO ZERO
1082 041E 80 FE 19 CMP DH,25 ; IF ROWS ARE LESS THAN 25 THEN
1083 0421 72 07 JB P54 ; GO TO ROWS_COLUMNS_SET
1084 |
1085 0423 B8 0E0A MOV AX,0E0AH ; ELSE SCROLL SCREEN ONE LINE
1086 0426 CD 10 INT 10H ; RESET ROW COUNTER TO 24
1087 0428 FE CE DEC DH
1088 042A P54: MOV AX,0200H ; ROW COLUMNS SET
1089 042A B8 0200 INT 10H ; SET NEW CURSOR POSITION COMMAND
1090 042D CD 10 MOV LOOP P50 ; ESTABLISH NEW CURSOR POSITION
1091 042F E2 AD LOOP P50 ; DO IT ONCE MORE UNTIL (CX) = ZERO
1092 |
1093 0431 5A POP DX ; RESTORE OLD CURSOR COORDINATES
1094 0432 97 XCHG AX,DI ; RECOVER WRITE STRING COMMAND
1095 0433 A8 01 TEST AL,01H ; IF CURSOR WAS NOT TO BE MOVED THEN
1096 0435 75 05 JNZ P59 ; THEN EXIT WITHOUT RESETTING OLD VALUE
1097 0437 B8 0200 MOV AX,0200H ; ELSE RESTORE OLD CURSOR POSITION
1098 043A CD 10 INT 10H
1099 043C P59: MOV VIDEO_RETURN ; DONE - EXIT WRITE STRING
1100 043C E9 013D R JMP ; RETURN TO CALLER
1101 |
1102 043F WRITE_STRING ENDP
    
```

```

1103 PAGE
1104 -----
1105 ; READ DOT = WRITE DOT
1106 ; THESE ROUTINES WILL WRITE A DOT, OR READ THE
1107 ; DOT AT THE INDICATED LOCATION
1108 ; ENTRY --
1109 ; DX = ROW (0-199) (THE ACTUAL VALUE DEPENDS ON THE MODE)
1110 ; CX = COLUMN ( 0-639) ( THE VALUES ARE NOT RANGE CHECKED )
1111 ; AL = DOT VALUE TO WRITE ( 1,2 OR 4 BITS DEPENDING ON MODE,
1112 ; REQUIRED FOR WRITE DOT ONLY, RIGHT JUSTIFIED)
1113 ; BIT 7 OF AL = 1 INDICATES XOR THE VALUE INTO THE LOCATION
1114 ; DS = DATA SEGMENT
1115 ; ES = REGEN SEGMENT
1116
1117 ;
1118 ; EXIT AL = DOT VALUE READ, RIGHT JUSTIFIED, READ ONLY
1119 -----
1120 ASSUME DS:DATA,ES:DATA
1121
1122 READ_DOT PROC NEAR
1123 CALL R3
1124 AND AL,ES:[SI]
1125 SHL AL,CL
1126 MOV CL,DH
1127 ROL AL,CL
1128 JMP VIDEO_RETURN
1129 ENDP
1130
1131 WRITE_DOT PROC NEAR
1132 PUSH AX
1133 PUSH AX
1134 CALL R3
1135 SHR AL,CL
1136 AND AL,AH
1137 MOV CL,ES:[SI]
1138 POP BX
1139 TEST EB,80H
1140 JNZ R2
1141 NOT AH
1142 AND CL,AH
1143 OR AL,CL
1144
1145 MOV ES:[SI],AL
1146 POP AX
1147 JMP VIDEO_RETURN
1148
1149 XOR AL,CL
1150 JMP R1
1151 ENDP
1152
1153 ; THIS SUBROUTINE DETERMINES THE REGEN BYTE LOCATION OF THE
1154 ; INDICATED ROW COLUMN VALUE IN GRAPHICS MODE.
1155 ; ENTRY --
1156 ; DX = ROW VALUE (0-199)
1157 ; CX = COLUMN VALUE (0-639)
1158 ; EXIT --
1159 ; SI = OFFSET INTO REGEN BUFFER FOR BYTE OF INTEREST
1160 ; AH = MASK TO STRIP OFF THE BITS OF INTEREST
1161 ; CL = BITS TO SHIFT TO RIGHT JUSTIFY THE MASK IN AH
1162 ; DH = # BITS IN RESULT
1163 ; BX = MODIFIED
1164 -----
1165 R3 PROC NEAR
1166
1167 ;---- DETERMINE 1ST BYTE IN INDICATED ROW BY MULTIPLYING ROW VALUE BY 40
1168 ;---- ( LOW BIT OF ROW DETERMINES EVEN/ODD, 80 BYTES/ROW )
1169
1170 XCHG SI,AX
1171 MOV AL,40
1172 MUL DL
1173 TEST AL,00BH
1174 JZ R4
1175 ADD AX,2000H-40
1176
1177 XCHG SI,AX
1178 MOV DX,CX
1179
1180 ;---- DETERMINE GRAPHICS MODE CURRENTLY IN EFFECT
1181
1182 ; SET UP THE REGISTERS ACCORDING TO THE MODE
1183 CH = MASK FOR LOW OF COLUMN ADDRESS ( 7/3 FOR HIGH/MED RES )
1184 CL = # OF ADDRESS BITS IN COLUMN VALUE ( 3/2 FOR H/M )
1185 BL = MASK TO SELECT BITS FROM POINTED BYTE ( 80H/COH FOR H/M )
1186 BH = NUMBER OF VALID BITS IN POINTED BYTE ( 1/2 FOR H/M )
1187
1188 MOV BX,2C0H
1189 MOV CX,302H
1190 CMP %CRT_MODE,6
1191 JC R5
1192 MOV BX,180H
1193 MOV CX,703H
1194
1195 ;---- DETERMINE BIT OFFSET IN BYTE FROM COLUMN MASK
1196 R5: AND CH,DL
1197 ; ADDRESS OF PEL WITHIN BYTE TO CH
1198
1199 ;---- DETERMINE BYTE OFFSET FOR THIS LOCATION IN COLUMN
1200
1201 SHR DX,CL
1202 ADD SI,DX
1203 MOV DH,BH
1204
1205 ;---- MULTIPLY BH (VALID BITS IN BYTE) BY CH (BIT OFFSET)
1206
1207 SUB CL,CL
1208
1209 ROR AL,1
1210 ADD CL,CH
1211 DEC BH
1212 JNZ R6
1213 MOV AH,BL
1214 SHR AH,CL
1215 RET
1216 ENDP
1217
1218 R6:
1219 ; LEFT JUSTIFY VALUE IN AL (FOR WRITE)
1220 ; ADD IN THE BIT OFFSET VALUE
1221 ; LOOP CONTROL
1222 ; ON EXIT, CL HAS COUNT TO RESTORE BITS
1223 ; GET MASK TO AH
1224 ; MOVE THE MASK TO CORRECT LOCATION
1225 ; RETURN WITH EVERYTHING SET UP

```

```

1217 ;-----
1218 ; SCROLL UP
1219 ; THIS ROUTINE SCROLLS UP THE INFORMATION ON THE CRT
1220 ; ENTRY --
1221 ; CH,CL = UPPER LEFT CORNER OF REGION TO SCROLL
1222 ; DH,DL = LOWER RIGHT CORNER OF REGION TO SCROLL
1223 ; BOTH OF THE ABOVE ARE IN CHARACTER POSITIONS
1224 ; BH = FILL VALUE FOR BLANKED LINES
1225 ; AL = # LINES TO SCROLL (AL=0 MEANS BLANK THE ENTIRE FIELD)
1226 ; DS = DATA SEGMENT
1227 ; ES = REGEN SEGMENT
1228 ; EXIT --
1229 ; NOTHING, THE SCREEN IS SCROLLED
1230 ;-----
1231 04AC GRAPHICS_UP PROC NEAR
1232 04AC 8A D8 MOV BL,AL ; SAVE LINE COUNT IN BL
1233 04AE 8B C1 MOV AX,CX ; GET UPPER LEFT POSITION INTO AX REG
1234 ;-----
1235 ;----- USE CHARACTER SUBROUTINE FOR POSITIONING
1236 ;----- ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE
1237
1238 04B0 E8 06EC R CALL GRAPH_POSN
1239 04B3 8B F8 MOV DI,AX ; SAVE RESULT AS DESTINATION ADDRESS
1240
1241 ;----- DETERMINE SIZE OF WINDOW
1242
1243 04B5 2B D1 SUB DX,CX
1244 04B7 81 C2 0101 ADD DX,101H ; ADJUST VALUES
1245 04BB D0 E6 SAL DH,1 ; MULTIPLY ROWS BY 4 AT 8 VERT DOTS/CHAR
1246 04BD D0 E6 SAL DH,1 ; AND EVEN/ODD ROWS
1247
1248 ;----- DETERMINE CRT MODE
1249
1250 04BF 80 3E 0049 R 06 CMP #CRT_MODE,6 ; TEST FOR MEDIUM RES
1251 04C4 73 04 JNC R7 ; FIND_SOURCE
1252
1253 ;----- MEDIUM RES UP
1254 04C6 D0 E2 SAL DL,1 ; # COLUMNS * 2, SINCE 2 BYTES/CHAR
1255 04C8 D1 E7 SAL DI,1 ; OFFSET *2 SINCE 2 BYTES/CHAR
1256
1257 ;----- DETERMINE THE SOURCE ADDRESS IN THE BUFFER
1258 04CA RT; ; FIND_SOURCE
1259 04CA 06 PUSH ES ; GET SEGMENTS BOTH POINTING TO REGEN
1260 04CB 1F POP DS
1261 04CC 2A ED SUB CH,CH ; ZERO TO HIGH OF COUNT REGISTER
1262 04CE D0 E3 SAL BL,1 ; MULTIPLY NUMBER OF LINES BY 4
1263 04D0 D0 E3 SAL BL,1
1264 04D2 74 2B JZ R11 ; IF ZERO, THEN BLANK ENTIRE FIELD
1265 04D4 B0 50 MOV AL,80 ; 80 BYTES/ROW
1266 04D6 F6 E3 MUL BL,AL ; DETERMINE OFFSET TO SOURCE
1267 04D8 8B F7 MOV SI,DI ; SET UP SOURCE
1268 04DA 03 F0 ADD SI,AX ; ADD IN OFFSET TO IT
1269 04DC 8A E6 MOV AH,DH ; NUMBER OF ROWS IN FIELD
1270 04DE 2A E3 SUB AH,BL ; DETERMINE NUMBER TO MOVE
1271
1272 ;----- LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD FIELDS
1273 04E0 RB; ; ROW LOOP
1274 04E0 E8 0560 R CALL R17 ; MOVE ONE ROW
1275 04E3 81 EE 1FB0 SUB SI,2000H-80 ; MOVE TO NEXT ROW
1276 04E7 81 EF 1FB0 SUB DI,2000H-80
1277 04EB FE CC DEC AH ; NUMBER OF ROWS TO MOVE
1278 04ED 75 F1 JNZ R8 ; CONTINUE TILL ALL MOVED
1279
1280 ;----- FILL IN THE VACATED LINE(S)
1281 04EF R9; ; CLEAR_ENTRY
1282 04EF 8A C7 MOV AL,BH ; ATTRIBUTE TO FILL WITH
1283 04F1 R10;
1284 04F1 E8 0579 R CALL R18 ; CLEAR THAT ROW
1285 04F4 81 EF 1FB0 SUB DI,2000H-80 ; POINT TO NEXT LINE
1286 04F8 FE CB DEC BL ; NUMBER OF LINES TO FILL
1287 04FA 75 F5 JNZ R10 ; CLEAR_LOOP
1288 04FC E9 013D R JMP VIDEO_RETURN ; EVERYTHING DONE
1289
1290 04FF R11; ; BLANK_FIELD
1291 04FF 8A DE MOV BL,DH ; SET BLANK COUNT TO EVERYTHING IN FIELD
1292 0501 EB EC JMP R9 ; CLEAR THE FIELD
1293 0503 GRAPHICS_UP ENDP
1294 ;-----
1295 ; SCROLL DOWN
1296 ; THIS ROUTINE SCROLLS DOWN THE INFORMATION ON THE CRT
1297 ; ENTRY --
1298 ; CH,CL = UPPER LEFT CORNER OF REGION TO SCROLL
1299 ; DH,DL = LOWER RIGHT CORNER OF REGION TO SCROLL
1300 ; BOTH OF THE ABOVE ARE IN CHARACTER POSITIONS
1301 ; BH = FILL VALUE FOR BLANKED LINES
1302 ; AL = # LINES TO SCROLL (AL=0 MEANS BLANK THE ENTIRE FIELD)
1303 ; DS = DATA SEGMENT
1304 ; ES = REGEN SEGMENT
1305 ; EXIT --
1306 ; NOTHING, THE SCREEN IS SCROLLED
1307 ;-----
1308
1309 0503 GRAPHICS_DOWN PROC NEAR
1310 0503 FD STD ; SET DIRECTION
1311 0504 8A D8 MOV BL,AL ; SAVE LINE COUNT IN BL
1312 0506 8B C2 MOV AX,DX ; GET LOWER RIGHT POSITION INTO AX REG
1313 ;-----
1314 ;----- USE CHARACTER SUBROUTINE FOR POSITIONING
1315 ;----- ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE
1316
1317 0508 E8 06EC R CALL GRAPH_POSN
1318 050B 8B F8 MOV DI,AX ; SAVE RESULT AS DESTINATION ADDRESS
1319
1320 ;----- DETERMINE SIZE OF WINDOW
1321
1322 050D 2B D1 SUB DX,CX
1323 050F 81 C2 0101 ADD DX,101H ; ADJUST VALUES
1324 0513 D0 E6 SAL DH,1 ; MULTIPLY ROWS BY 4 AT 8 VERT DOTS/CHAR
1325 0515 D0 E6 SAL DH,1 ; AND EVEN/ODD ROWS
1326
1327 ;----- DETERMINE CRT MODE
1328
1329 0517 80 3E 0049 R 06 CMP #CRT_MODE,6 ; TEST FOR MEDIUM RES
1330 051C 73 05 JNC R12 ; FIND_SOURCE_DOWN
    
```

```

1331
1332
1333 051E 00 E2          I----- MEDIUM RES DOWN
1334 0520 D1 E7          SAL DL,1          ; # COLUMNS * 2, SINCE 2 BYTES/CHAR
1335 0522 47            SAL DI,1          ; OFFSET *2 SINCE 2 BYTES/CHAR
1336                                INC DI            ; POINT TO LAST BYTE
1337
1338                                I----- DETERMINE THE SOURCE ADDRESS IN THE BUFFER
1339 0523 06          R12:  PUSH E5          ; FIND SOURCE DOWN
1340 0524 1F          POP D5           ; BOTH SEGMENTS TO REGEN
1341 0525 2A ED          SUB CH,CH        ; ZERO TO HIGH OF COUNT REGISTER
1342 0527 81 C7 00F0   ADD DI,240       ; POINT TO LAST ROW OF PIXELS
1343 0528 00 E3          SAL BL,1         ; MULTIPLY NUMBER OF LINES BY 4
1344 052D 00 E3          SAL BL,1
1345 052F 74 2B          JZ R16           ; IF ZERO, THEN BLANK ENTIRE FIELD
1346 0531 80 50          MOV AL,80        ; 80 BYTES/ROW
1347 0533 F6 E3          MUL BL           ; DETERMINE OFFSET TO SOURCE
1348 0535 8B 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 E9          SUB AH,BL        ; DETERMINE NUMBER TO MOVE
1352
1353                                I----- 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 0544 81 EF 2050   DEC DI           ; NUMBER OF ROWS TO MOVE
1358 0548 FE CC          DEC R13          ; CONTINUE TILL ALL MOVED
1359 054A 75 F1          JNZ R13
1360
1361                                I----- FILL IN THE VACATED LINE(S)
1362 054C          R14:  MOV AL,BH        ; CLEAR ENTRY DOWN
1363 054C 8A C7          MOV R15,R14     ; ATTRIBUTE TO FILL WITH
1364 054E          R15:  CALL R18         ; CLEAR LOOP_DOWN
1365 054E E8 0579 R      SUB DI,2000H+80 ; CLEAR A ROW
1366 0551 81 EF 2050   DEC BL           ; 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          GRAPHICS_DOWN  ; CLEAR THE FIELD
1375 0560          ENDP
1376
1377                                I----- ROUTINE TO MOVE ONE ROW OF INFORMATION
1378
1379 0560          R17:  PROC NEAR
1380 0560 8A CA          MOV CL,DL        ; NUMBER OF BYTES IN THE ROW
1381 0562 56            PUSH SI          ; SAVE POINTERS
1382 0563 57            PUSH DI          ; MOVE THE EVEN FIELD
1383 0564 F3/ A4        REP MOVSB        ; SAVE THE EVEN FIELD
1384 0566 5F            POP DI           ; POINT TO THE ODD FIELD
1385 0567 5E            POP SI           ; SAVE THE POINTERS
1386 0568 81 C6 2000   ADD SI,2000H    ; COUNT BACK
1387 056C 81 C7 2000   ADD DI,2000H    ; MOVE THE ODD FIELD
1388 0570 56            PUSH SI          ; POINTERS BACK
1389 0571 57            PUSH DI          ; RETURN TO CALLER
1390 0572 8A CA          MOV CL,DL
1391 0574 F3/ A4        REP MOVSB
1392 0576 5F            POP DI           ; POINTERS BACK
1393 0577 5E            POP SI           ; RETURN TO CALLER
1394 0578 C3          RET
1395 0579          ENDP
1396
1397                                I----- CLEAR A SINGLE ROW
1398
1399 0579          R18:  PROC NEAR
1400 0579 8A CA          MOV CL,DL        ; NUMBER OF BYTES IN FIELD
1401 057B 57            PUSH DI          ; SAVE POINTER
1402 057C F3/ AA        REP STOSB        ; STORE THE NEW VALUE
1403 057E 5F            POP DI           ; POINTER BACK
1404 057F 81 C7 2000   ADD DI,2000H    ; POINT TO ODD FIELD
1405 0583 57            PUSH DI          ; FILL THE ODD FIELD
1406 0584 8A CA          MOV CL,DL
1407 0586 F3/ AA        REP STOSB
1408 0588 5F            POP DI           ; RETURN TO CALLER
1409 0589 C3          RET
1410 058A          ENDP
1411
1412                                -----
1413                                ; GRAPHICS WRITE
1414                                ; THIS ROUTINE WRITES THE ASCII CHARACTER TO THE CURRENT
1415                                ; POSITION ON THE SCREEN.
1416                                ; ENTRY --
1417                                ; AL = CHARACTER TO WRITE
1418                                ; BL = COLOR ATTRIBUTE TO BE USED FOR FOREGROUND COLOR
1419                                ; IF BIT 7 IS SET, THE CHAR IS XOR'D INTO THE REGEN BUFFER
1420                                ; (0 IS USED FOR THE BACKGROUND COLOR)
1421                                ; CX = NUMBER OF CHARS TO WRITE
1422                                ; DS = DATA SEGMENT
1423                                ; ES = REGEN SEGMENT
1424                                ; EXIT --
1425                                ; NOTHING IS RETURNED
1426
1427                                ; GRAPHICS READ
1428                                ; THIS ROUTINE READS THE ASCII CHARACTER AT THE CURRENT CURSOR
1429                                ; POSITION ON THE SCREEN BY MATCHING THE DOTS ON THE SCREEN TO THE
1430                                ; CHARACTER GENERATOR CODE POINTS
1431                                ; ENTRY --
1432                                ; NONE (0 IS ASSUMED AS THE BACKGROUND COLOR)
1433                                ; EXIT --
1434                                ; AL = CHARACTER READ AT THAT POSITION (0 RETURNED IF NONE FOUND)
1435                                ;
1436                                ; FOR BOTH ROUTINES, THE IMAGES USED TO FORM CHARS ARE CONTAINED IN ROM
1437                                ; FOR THE 1ST 128 CHARS. TO ACCESS CHARS IN THE SECOND HALF, THE USER
1438                                ; MUST INITIALIZE THE VECTOR AT INTERRUPT 1FH (LOCATION 0070H) TO
1439                                ; POINT TO THE USER SUPPLIED TABLE OF GRAPHIC IMAGES (8x8 BOXES).
1440                                ; FAILURE TO DO SO WILL CAUSE IN STRANGE RESULTS
1441                                -----
1442 058A          ASSUME DS:DATA,ES:DATA
1443 058A B4 00          GRAPHICS_WRITE PROC NEAR
1444 058C 50            MOV AH,0         ; ZERO TO HIGH OF CODE POINT
1445                                PUSH AX          ; SAVE CODE POINT VALUE

```

```

1445
1446
1447
1448 058D E8 06E9 R
1449 0590 8B F8
1450
1451
1452
1453 0592 58
1454 0593 3C 80
1455 0595 73 06
1456
1457
1458
1459 0597 BE 0000 E
1460 059A 0E
1461 059B EB 18
1462
1463
1464
1465 059D
S1:
1466 059D 2C 80
1467 059F 1E
1468 05A0 2B F6
1469 05A2 8E DE
1470
1471 05A4 C5 36 007C R
1472 05A8 8C DA
1473
1474 05AA 1F
1475 05AB 52
1476 05AC 0B D6
1477 05AE 75 05
1478
1479 05B0 58
1480 05B1 BE 0000 E
1481 05B4 0E
1482
1483
1484
1485 05B5
S2:
1486 05B5 D1 E0
1487 05B7 D1 E0
1488 05B9 D1 E0
1489 05BB 03 F0
1490 05BD 80 3E 0049 R 06
1491 05C2 1F
1492 05C3 72 2C
1493
1494
1495 05C5
S3:
1496 05C5 57
1497 05C6 56
1498 05C7 B6 04
1499 05C9
S4:
1500 05C9 AC
1501 05CA F6 C3 80
1502 05CB 75 16
1503 05CC AA
1504 05D0 AC
1505 05D1
1506 05D1 26: 88 85 IFFF
1507 05D6 83 C7 4F
1508 05D9 FE CE
1509 05DB 75 EC
1510 05DD 5E
1511 05DE 5F
1512 05DF 47
1513 05E0 E2 E3
1514 05E2 E9 013D R
1515
1516 05E5
S6:
1517 05E5 26: 32 05
1518 05E8 AA
1519 05E9 AC
1520 05EA 26: 32 85 IFFF
1521 05EF EB 0E
1522
1523
1524 05F1
S7:
1525 05F1 8A D3
1526 05F3 D1 E7
1527
1528 05F5 80 E3 03
1529 05F8 B0 55
1530 05FA F6 E3
1531 05FC 8A D8
1532 05FE 8A F8
1533 0600
S8:
1534 0600 57
1535 0601 56
1536 0602 B6 04
1537 0604
S9:
1538 0604 AC
1539 0605 EB 06C0 R
1540 0608 23 C3
1541 060A 86 E0
1542 060C F6 C2 80
1543 060F 74 03
1544 0611 26: 33 05
1545 0614
S10:
1546 0614 26: 89 05
1547 0617 AC
1548 0618 EB 06C0 R
1549 061B 23 C3
1550 061D 86 E0
1551 061F F6 C2 80
1552 0622 74 05
1553 0624 26: 33 85 2000
1554 0629
S11:
1555 0629 26: 89 85 2000
1556 062E 83 C7 50
1557 0631 FE CE
1558 0633 75 CF

```

SECTION 5

```

1559 0635 5E          POP     SI          I RECOVER CODE POINTER
1560 0636 5F          POP     DI          I RECOVER REGEN POINTER
1561 0637 47          INC     DI          I POINT TO NEXT CHAR POSITION
1562 0638 47          INC     DI
1563 0639 E2 C5      LOOP   SB          I MORE TO WRITE
1564 063B E9 013D R    JMP     VIDEO_RETURN
1565 063E          GRAPHICS_WRITE   ENDP
-----
1566          I GRAPHICS_READ
-----
1567          GRAPHICS_READ
-----
1568          GRAPHICS_READ PROC NEAR
1569 063E          CALL   S26        I CONVERTED TO OFFSET IN REGEN
1570 063E E8 06E9 R    MOV     SI,AX      I SAVED IN SI
1571 0641 BB F0          SUB     SP,8       I ALLOCATE SPACE FOR THE READ CODE POINT
1572 0643 83 EC 08      MOV     BP,SP      I POINTER TO SAVE AREA
1573 0646 BB EC
1574
1575
1576
1577 0648 80 3E 0049 R 06  CMP     #CRT_MODE,6
1578 064D 06          PUSH   ES
1579 064E 1F          POP     DS         I POINT TO REGEN SEGMENT
1580 064F 72 19          JC      S13        I MEDIUM RESOLUTION
1581
1582
1583
1584
1585 0651 B6 04          I----- DETERMINE GRAPHICS MODES
1586 0653          CMP     #CRT_MODE,6
1587 0653 8A 04          PUSH   ES
1588 0655 88 46 00      POP     DS         I POINT TO REGEN SEGMENT
1589 0658 45          JC      S13        I MEDIUM RESOLUTION
1590 0659 8A 84 2000    MOV     AL,[SI]    I GET FIRST BYTE
1591 065D 88 46 00      MOV     [BP],AL   I SAVE IN STORAGE AREA
1592 0660 45          INC     BP         I NEXT LOCATION
1593 0661 83 C6 50      MOV     AL,[SI+2000H] I GET LOWER REGION BYTE
1594 0664 FE CE        MOV     [BP],AL   I ADJUST AND STORE
1595 0666 75 EB          INC     BP
1596 0668 EB 16          ADD     SI,80     I POINTER INTO REGEN
1597 0669          DEC     DH        I LOOP CONTROL
1598 066A          JNZ    S12        I DO IT SOME MORE
1599 066A          JMP    SHORT S15  I GO MATCH THE SAVED CODE POINTS
1600 066A D1 E6          I----- MEDIUM RESOLUTION READ
1601 066C B6 04          SAL     SI,1      I MED RES READ
1602 066E          MOV     DH,4      I OFFSET*2 SINCE 2 BYTES/CHAR
1603 066E          MOV     DH,4      I NUMBER OF PASSES
1604 0671 81 C6 1FFE    CALL   S23        I GET BYTES FROM REGEN INTO SINGLE SAVE
1605 0675 E8 06CF R    ADD     SI,2000H-2 I GO TO LOWER REGION
1606 0678 81 EE 1FB2    CALL   S23        I GET THIS PAIR INTO SAVE
1607 067C FE CE        SUB     SI,2000H-80+2 I ADJUST POINTER BACK INTO UPPER
1608 067E 75 EE          DEC     DX
1609 0680          JNZ    S14        I KEEP GOING UNTIL ALL 8 DONE
1610
1611 0680          I----- SAVE AREA HAS CHARACTER IN IT, MATCH IT
1612 0680 BF 0000 E    MOV     DI,OFFSET CRT_CHAR_GEN I FIND CHAR
1613 0683 0E          PUSH   CS         I ESTABLISH ADDRESSING
1614 0684 07          POP     ES
1615 0685 83 ED 08      SUB     BP,8       I CODE POINTS IN CS
1616 0688 BB F5          MOV     SI,BP     I ADJUST POINTER TO START OF SAVE AREA
1617 068A B0 00          MOV     AL,0      I CURRENT CODE POINT BEING MATCHED
1618 068C
1619 068C 16          PUSH   SS         I ESTABLISH ADDRESSING TO STACK
1620 068D 1F          POP     DS        I FOR THE STRING COMPARE
1621 068E BA 0080      MOV     DX,128    I NUMBER TO TEST AGAINST
1622 0691
1623 0691 56          PUSH   SI         I SAVE SAVE AREA POINTER
1624 0692 57          PUSH   DI         I SAVE CODE POINTER
1625 0693 B9 0004      MOV     CX,4      I NUMBER OF WORDS TO MATCH
1626 0696 F3/ A7      REPE   CMPSW     I COMPARE THE 8 BYTES AS WORDS
1627 0698 5F          POP     DI        I RECOVER THE POINTERS
1628 0699 5E          POP     SI
1629 069A 74 1E          JZ     S18        I IF ZERO FLAG SET, THEN MATCH OCCURRED
1630 069C FE C0          JC     AL,0       I NO MATCH, MOVE ON TO NEXT
1631 069E 83 C7 08      ADD     DI,8      I NEXT CODE POINT
1632 06A1 4A          DEC     DX        I LOOP CONTROL
1633 06A2 75 ED          JNZ    S17        I DO ALL OF THEM
1634
1635
1636
1637 06A4 3C 00          I----- CHAR NOT MATCHED, MIGHT BE IN USER SUPPLIED SECOND HALF
1638 06A6 74 12          CMP     AL,0      I AL<= 0 IF ONLY 1ST HALF SCANNED
1639 06A8 2B C0          JE     S18        I IF = 0, THEN ALL HAS BEEN SCANNED
1640 06AA BE D8          SUB     AX,AX     I ESTABLISH ADDRESSING TO VECTOR
1641 06AC          ASSUME DS:ABS0
1642 06AC C4 3E 007C R  LES     DI,0EXT_PTR I GET POINTER
1643 06BD 8C C0          MOV     AX,ES     I SEE IF THE POINTER REALLY EXISTS
1644 06BE 0B C7          OR     AX,DI      I IF ALL 0, THEN DOESN'T EXIST
1645 06B4 74 04          JZ     S18        I NO SENSE LOOKING
1646 06B6 B0 80          MOV     AL,128   I ORIGIN FOR SECOND HALF
1647 06BB EB D2          JMP     S16       I GO BACK AND TRY FOR IT
1648
1649
1650
1651 06BA          I----- CHARACTER IS FOUND (AL=0 IF NOT FOUND)
1652 06BA 83 C4 08      ADD     SP,8       I READJUST THE STACK, THROW AWAY SAVE
1653 06BD E9 013D R    JMP     VIDEO_RETURN I ALL DONE
1654 06C0          GRAPHICS_READ   ENDP
-----
1655          I EXPAND BYTE
1656
1657          I THIS ROUTINE TAKES THE BYTE IN AL AND DOUBLES ALL
1658          I OF THE BITS, TURNING THE 8 BITS INTO 16 BITS.
1659          I THE RESULT IS LEFT IN AX
1660
1661 06C0          S21 PROC NEAR
1662 06C0 51          PUSH   CX         I SAVE REGISTER
1663 06C1 B9 0008      MOV     CX,8      I SHIFT COUNT REGISTER FOR ONE BYTE
1664 06C4          S22:
1665 06C4 D0 CB          ROR     AL,1      I SHIFT BITS, LOW BIT INTO CARRY FLAG
1666 06C6 D1 DD          ROR     BP,1      I MOVE CARRY FLAG (LOW BIT) INTO RESULT
1667 06C8 D1 FD          SAR     BP,1      I SIGN EXTEND HIGH BIT (DOUBLE IT)
1668 06CA E2 F8          LOOP   S22       I REPEAT FOR ALL 8 BITS
1669
1670 06CC 95          XCHG   AX,BP     I MOVE RESULTS TO PARAMETER REGISTER
1671 06CD 59          POP     CX        I RECOVER REGISTER
1672 06CE C3          RET          I ALL DONE
    
```



```

1673 06CF          S21  ENDP
1674
1675          |-----|
1676          | MED READ BYTE
1677          | THIS ROUTINE WILL TAKE 2 BYTES FROM THE REGEN BUFFER,
1678          | COMPARE AGAINST THE CURRENT FOREGROUND COLOR, AND PLACE
1679          | THE CORRESPONDING ON/OFF BIT PATTERN INTO THE CURRENT
1680          | POSITION IN THE SAVE AREA
1681          | ENTRY --
1682          | SI,DS = POINTER TO REGEN AREA OF INTEREST
1683          | BX = EXPANDED FOREGROUND COLOR
1684          | BP = POINTER TO SAVE AREA
1685          | EXIT --
1686          | SI AND BP ARE INCREMENTED
1687          |-----|
1688 06CF          S23  PROC  NEAR
1689 06D0 86 C4      XCHG  AL,AH          | GET FIRST BYTE AND SECOND BYTES
1690 06D2 B9 C0D0    MOV   CX,0C000H        | SWAP FOR COMPARE
1691 06D5 B2 00      MOV   DL,0             | 2 BIT MASK TO TEST THE ENTRIES
1692 06D7          | RESULT REGISTER
1693 06D7 85 C1     TEST  AX,CX            | IS THIS SECTION BACKGROUND?
1694 06D9 74 01     JZ    S25              | IF ZERO, IT IS BACKGROUND (CARRY=0)
1695 06DB F9        STC                     | WASN'T SO SET CARRY
1696 06DC          |
1697 06DC D0 D2     RCL   DL,1             | MOVE THAT BIT INTO THE RESULT
1698 06DE D1 E9     SHR   CX,1             |
1699 06E0 D1 E9     SHR   CX,1             | MOVE THE MASK TO THE RIGHT BY 2 BITS
1700 06E2 73 F3     JNC  S24              | DO IT AGAIN IF MASK DIDN'T FALL OUT
1701 06E4 88 56 00  MOV   [BP],DL          | STORE RESULT IN SAVE AREA
1702 06E7 45        INC   BP               | ADJUST POINTER
1703 06E8 C3        RET                     | ALL DONE
1704 06E9          |
1705          S23  ENDP
1706          |-----|
1707          | V4 POSITION
1708          | THIS ROUTINE TAKES THE CURSOR POSITION CONTAINED IN
1709          | THE MEMORY LOCATION, AND CONVERTS IT INTO AN OFFSET
1710          | INTO THE REGEN BUFFER, ASSUMING ONE BYTE/CHAR.
1711          | FOR MEDIUM RESOLUTION GRAPHICS, THE NUMBER MUST
1712          | BE DOUBLED.
1713          | ENTRY -- NO REGISTERS, MEMORY LOCATION @CURSOR_POSN IS USED
1714          | EXIT --
1715          | AX CONTAINS OFFSET INTO REGEN BUFFER
1716          |-----|
1717 06E9          S26  PROC  NEAR
1718 06E9 A1 0050 R  MOV   AX,@CURSOR_POSN | GET CURRENT CURSOR
1719 06EC          GRAPH_POSN LABEL NEAR
1720 06EC          PUSH  BX          | SAVE REGISTER
1721 06ED 8B D8      MOV   BX,AX            | SAVE A COPY OF CURRENT CURSOR
1722 06EF A0 004A R  MOV   AL,BYTE PTR @CRT_COLS | GET BYTES PER COLUMN
1723 06F0 F6 E4     MUL  AH                | MULTIPLY BY ROWS
1724 06F4 D1 E0     SHL  AX,1              |
1725 06F8 2A FF     SUB  BH,BH             | MULTIPLY * 4 SINCE 4 ROWS/BYTE
1726 06FA 03 C3     ADD  AX,BX             | ISOLATE COLUMN VALUE
1727 06FC 5B        POP  BX                | DETERMINE OFFSET
1728 06FD C3        RET                     | RECOVER POINTER
1729 06FE          | ALL DONE
1730          S26  ENDP
1731          |-----|
1732          | WRITE_TTY
1733          | THIS INTERFACE PROVIDES A TELETYPE LIKE INTERFACE TO THE
1734          | VIDEO CARDS. THE INPUT CHARACTER IS WRITTEN TO THE CURRENT
1735          | CURSOR POSITION, AND THE CURSOR IS MOVED TO THE NEXT POSITION.
1736          | IF THE CURSOR LEAVES THE LAST COLUMN OF THE FIELD, THE COLUMN
1737          | IS SET TO ZERO, AND THE ROW VALUE IS INCREMENTED. IF THE ROW
1738          | VALUE LEAVES THE FIELD, THE CURSOR IS PLACED ON THE LAST ROW,
1739          | FIRST COLUMN, AND THE ENTIRE SCREEN IS SCROLLED UP ONE LINE.
1740          | WHEN THE SCREEN IS SCROLLED UP, THE ATTRIBUTE FOR FILLING THE
1741          | NEWLY BLANKED LINE IS READ FROM THE CURSOR POSITION ON THE PREVIOUS
1742          | LINE BEFORE THE SCROLL. IN GRAPHICS MODE,
1743          | THE 0 COLOR IS USED.
1744          | ENTRY --
1745          | (AH) = CURRENT CRT MODE
1746          | (AL) = CHARACTER TO BE WRITTEN
1747          | NOTE THAT BACK SPACE, CARRIAGE RETURN, BELL AND LINE FEED ARE
1748          | HANDLED AS COMMANDS RATHER THAN AS DISPLAY GRAPHICS CHARACTERS
1749          | (BL) = FOREGROUND COLOR FOR CHAR WRITE IF CURRENTLY IN A GRAPHICS MODE
1750          | EXIT --
1751          | ALL REGISTERS SAVED THROUGH VIDEO_EXIT (INCLUDING (AX))
1752          |-----|
1753 06FE          ASSUME DS:DATA
1754 06FE          WRITE_TTY PROC  NEAR
1755 06FF 97         XCHG  DI,AX          | SAVE (AX) REGISTER IN (DI) FOR EXIT
1756 0701 8A 3E 0062 R MOV   BH,@ACTIVE_PAGE | READ CURSOR POSITION
1757 0705 CD 10     INT  10H              | GET CURRENT PAGE SETTING
1758 0707 8B C7     MOV   AX,DI            | READ THE CURRENT CURSOR POSITION
1759          | RECOVER CHARACTER FROM (DI) REGISTER
1760          |-----|
1761          | DX NOW HAS THE CURRENT CURSOR POSITION
1762 0709 3C D0     CMP   AL,CR           | IS IT CARRIAGE RETURN OR CONTROL
1763 070B 76 46     JBE  U0               | GO TO CONTROL CHECKS IF IT IS
1764          |
1765          |-----|
1766          | WRITE THE CHAR TO THE SCREEN
1767 070D          U0:
1768 070D B4 0A     MOV   AH,0AH          | WRITE CHARACTER ONLY COMMAND
1769 070F B9 0001  MOV   CX,1            | ONLY ONE CHARACTER
1770 0712 CD 10     INT  10H              | WRITE THE CHARACTER
1771          |-----|
1772          | POSITION THE CURSOR FOR NEXT CHAR
1773 0714 FE C2     INC   DL              |
1774 0716 3A 16 004A R CMP   DL,BYTE PTR @CRT_COLS | TEST FOR COLUMN OVERFLOW
1775 071A 75 33     JNZ  U7               | SET CURSOR
1776 071C B2 00     MOV   DL,0            | COLUMN FOR CURSOR
1777 071E 80 FE 18  CMP   DH,25-1         | CHECK FOR LAST ROW
1778 0721 75 2A     JNZ  U6               | SET_CURSOR_INC
1779          |
1780          |-----|
1781 0723          | SCROLL REQUIRED
1782 0723 B4 02     MOV   AH,02H          |
1783 0725 CD 10     INT  10H              | SET THE CURSOR
1784          |
1785          |-----|
1786          | DETERMINE VALUE TO FILL WITH DURING SCROLL

```

SECTION 5

```

1787 0727 A0 0049 R      MOV     AL,®CRT_MODE      ; GET THE CURRENT MODE
1788 072A 3C 04          CMP     AL,4              ;
1789 072C 72 06          JC      U2                ; READ-CURSOR
1790 072E 3C 07          CMP     AL,7              ;
1791 0730 B7 00          MOV     BH,0              ; FILL WITH BACKGROUND
1792 0732 75 06          JNE     U3                ; SCROLL-UP
1793 0734              ; READ-CURSOR
1794 0736 B4 08          MOV     AH,0BH           ; GET READ CURSOR COMMAND
1795 0738 CD 10          INT     10H              ; READ CHAR/ATTR AT CURRENT CURSOR
1796 073A 8A FC          MOV     BH,AH             ; STORE IN BH
1797 073A              ; SCROLL-UP
1798 073A B8 0601        MOV     AX,0601H          ; SCROLL ONE LINE
1799 073D 2B C9          SUB     CX,CX             ; UPPER LEFT CORNER
1800 073F B6 18          MOV     DH,25-1           ; LOWER RIGHT ROW
1801 0741 8A 16 004A R    MOV     DL,BYTE PTR ®CRT_COLS ; LOWER RIGHT COLUMN
1802 0745 FE CA          DEC     DL                ;
1803 0747              ; VIDEO-CALL-RETURN
1804 0747 CD 10          XCHG   INT 10H           ; SCROLL UP THE SCREEN
1805 0749              ; TTY-RETURN
1806 0749 97              XCHG   AX,DI             ; RESTORE THE ENTRY CHARACTER FROM (DI)
1807 074A E9 013D R      JMP     VIDEO_RETURN      ; RETURN TO CALLER
1808
1809 074D              ; SET-CURSOR-INC
1810 074D FE C6          INC     DH                ; NEXT ROW
1811 074F              ; SET-CURSOR
1812 074F B4 02          MOV     AH,02H           ;
1813 0751 EB F4          JMP     U4                ; ESTABLISH THE NEW CURSOR
1814
1815
1816 0753              ; ---- CHECK FOR CONTROL CHARACTERS
1817 0753 74 13          JB     U9                 ; WAS IT A CARRIAGE RETURN
1818 0755 3C 0A          CMP     AL,LF             ; IS IT A LINE FEED
1819 0757 74 13          JB     U10                ; GO TO LINE FEED
1820 0759 3C 07          CMP     AL,07H           ; IS IT A BELL
1821 075B 74 16          JB     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
1826
1827 0761 0A D2          OR      DL,DL             ; IS IT ALREADY AT START OF LINE
1828 0763 74 EA          JB     U7                 ; SET_CURSOR
1829 0765 4A            DEC     DX                ; NO -- JUST MOVE IT BACK
1830 0766 EB E7          JMP     U7                 ; SET_CURSOR
1831
1832
1833
1834 0768              ; ---- CARRIAGE RETURN FOUND
1835 0768 B2 00          MOV     DL,0              ; MOVE TO FIRST COLUMN
1836 076A EB E3          JMP     U7                 ; SET_CURSOR
1837
1838
1839
1840 076C              ; ---- LINE FEED FOUND
1841 076C 80 FE 18        CMP     DH,25-1           ; BOTTOM OF SCREEN
1842 076F 75 DC          JNE     U6                ; YES, SCROLL THE SCREEN
1843 0771 EB B0          JMP     U1                ; NO, JUST SET THE CURSOR
1844
1845
1846
1847 0773              ; ---- BELL FOUND
1848 0773 B9 0533        MOV     CX,1331           ; DIVISOR FOR 896 HZ TONE
1849 0776 B3 1F          MOV     BL,31             ; SET COUNT FOR 31/64 SECOND FOR BEEP
1850 0778 EB 0000 E      CALL    BEEP              ; SOUND THE POD BELL
1851 077B EB CC          JMP     U5                 ; TTY_RETURN
1852 077D
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868 077D 03 03 05 05 03 03 ; ASSUME DS:DATA
1869 03 04              DB     3,3,5,5,3,3,3,4 ; SUBTRACT_TABLE
1870
1871
1872 0785              ; ---- WAIT FOR LIGHT PEN TO BE DEPRESSED
1873 0785 B4 00          MOV     AH,0              ; SET NO LIGHT PEN RETURN CODE
1874 0787 BB 16 0063 R    MOV     DX,®ADDR_6845     ; GET BASE ADDRESS OF 6845
1875 078B 83 C2 06          ADD     DX,6              ; POINT TO STATUS REGISTER
1876 078E EC            IN      AL,DX             ; GET STATUS REGISTER
1877 078F AB 04          TEST   AL,®004H          ; TEST LIGHT PEN SWITCH
1878 0791 74 03          JZ      V6_A              ; GO IF YES
1879 0793 E9 0816 R      JMP     V6                ; NOT SET, RETURN
1880
1881
1882
1883 0796 A8 02          ; ---- NOW TEST FOR LIGHT PEN TRIGGER
1884 0796 75 03          V6_A: TEST   AL,2              ; TEST LIGHT PEN TRIGGER
1885 079A E9 0820 R      JNZ    V7A               ; RETURN WITHOUT RESETTNG TRIGGER
1886
1887
1888
1889 079D              ; TRIGGER HAS BEEN SET, READ THE VALUE IN
1890 079D B4 10          V7A: MOV     AH,16            ; LIGHT PEN REGISTERS ON 6845
1891
1892
1893
1894 079F BB 16 0063 R    ; ---- INPUT REGISTERS POINTED TO BY AH, AND CONVERT TO ROW COLUMN IN (DX)
1895 07A3 8A C4          MOV     DX,®ADDR_6845     ; ADDRESS REGISTER FOR 6845
1896 07A5 EE            MOV     AL,AH             ; REGISTER TO READ
1897 07A6 90            OUT     DX,AL             ; SET IT UP
1898 07A7 42            NOP                      ; I/O DELAY
1899 07A8 EC            INC     DX                ; DATA REGISTER
1900 07A9 8A E8          IN      AL,DX             ; GET THE VALUE
                          MOV     CH,AL             ; SAVE IN CH

```

```

1901 07AB 4A          DEC    DX          | ADDRESS REGISTER
1902 07AC FE C4     INC    AH
1903 07AE 8A C4     MOV    AL,AH      | SECOND DATA REGISTER
1904 07B0 EE        OUT    DX,AL
1905 07B1 42       INC    DX          | POINT TO DATA REGISTER
1906 07B2 90       NOP
1907 07B3 EC       IN     AL,DX      | GET SECOND DATA VALUE
1908 07B4 8A E5     MOV    AH,CH      | AX HAS INPUT VALUE
1909
1910                |----- AX HAS THE VALUE READ IN FROM THE 6845
1911
1912 07B6 8A 1E 0049 R MOV    BL,*CRT_MODE
1913 07BA 2A FF     SUB    BH,BH      | MODE VALUE TO BX
1914 07BC 2E: 8A 9F 07DD R MOV    BL,CS:VI[BX]
1915 07C1 2B C3     SUB    AX,BX      | DETERMINE AMOUNT TO SUBTRACT
1916 07C3 6B 1E 004E R MOV    BX,*CRT_START
1917 07C7 D1 EB     SHR    BX,1       | TAKE IT AWAY
1918 07C9 2B C3     SUB    AX,BX      | CONVERT TO CORRECT PAGE ORIGIN
1919 07CB 19 02     JNB   V2          | IF POSITIVE, DETERMINE MODE
1920 07CD 2B C0     SUB    AX,AX      | <0 PLAYS AS 0
1921
1922                |----- DETERMINE MODE OF OPERATION
1923
1924 07CF                V2:
1925 07CF B1 03     MOV    CL,3       | DETERMINE MODE
1926 07D1 80 3E 0049 R 04 CMP    *CRT_MODE,4 | SET *8 SHFT COUNT
1927 07D6 75 2A     JB    V4          | DETERMINE IF GRAPHICS OR ALPHA
1928 07DB 80 3E 0049 R 07 CMP    *CRT_MODE,7 | ALPHA_PEN
1929 07DD 74 23     JE    V4          | ALPHA_PEN
1930
1931                |----- GRAPHICS MODE
1932
1933 07DF B2 28     MOV    DL,40      | DIVISOR FOR GRAPHICS
1934 07E1 F6 F2     DIV    DL          | DETERMINE ROW(AL) AND COLUMN(AH)
1935
1936                |----- DETERMINE GRAPHIC ROW POSITION
1937
1938 07E3 8A E8     MOV    CH,AL      | SAVE ROW VALUE IN CH
1939 07E5 02 ED     ADD    CH,CH      | *2 FOR EVEN/ODD FIELD
1940 07E7 8A DC     MOV    BL,AH      | COLUMN VALUE TO BX
1941 07E9 2A FF     SUB    BH,BH      | MULTIPLY BY 8 FOR MEDIUM RES
1942 07EB 80 3E 0049 R 06 CMP    *CRT_MODE,6 | DETERMINE MEDIUM OR HIGH RES
1943 07F0 75 04     JNB   V3          | NOT HIGH RES
1944 07F2 B1 04     MOV    CL,4       | SHIFT VALUE FOR HIGH RES
1945 07F4 D0 E4     SAL    AH,1       | COLUMN VALUE TIMES 2 FOR HIGH RES
1946 07F6                V3:
1947 07F6 D3 E3     SHL    BX,CL      | MULTIPLY *16 FOR HIGH RES
1948
1949                |----- DETERMINE ALPHA CHAR POSITION
1950
1951 07F8 8A D4     MOV    DL,AH      | COLUMN VALUE FOR RETURN
1952 07FA 8A F0     MOV    DH,AL      | ROW VALUE
1953 07FC DD EE     SHR    DH,1       | DIVIDE BY 4
1954 07FE DD EE     SHR    DH,1       | FOR VALUE IN 0-24 RANGE
1955 0800 EB 12     JMP    SHORT V5   | LIGHT_PEN_RETURN_SET
1956
1957                |----- ALPHA MODE ON LIGHT PEN
1958
1959 0802                V4:
1960 0802 F6 36 004A R DIV    BYTE PTR *CRT_COLS | ALPHA PEN
1961 0806 8A F0     MOV    DH,AL      | DETERMINE ROW,COLUMN VALUE
1962 0808 8A D4     MOV    DL,AH      | ROWS TO DH
1963 080A D2 E0     SAL    AL,CL      | COLS TO DL
1964 080C 8A E8     MOV    CH,AL      | MULTIPLY ROWS * 8
1965 080E 8A DC     MOV    BL,AH      | GET RASTER VALUE TO RETURN REGISTER
1966 0810 32 FF     XOR    BH,BH      | COLUMN VALUE
1967 0812 D3 E3     SAL    BX,CL      | TO BX
1968 0814                V5:
1969 0814 B4 01     MOV    AH,1       | LIGHT PEN RETURN SET
1970 0816                V6:
1971 0816 52       PUSH   DX          | INDICATE EVERYTHING SET
1972 0817 8B 16 0063 R MOV    DX,*ADDR_6845 | LIGHT PEN RETURN
1973 081B 83 C2 07 ADD    DX,7        | GET BASE ADDRESS
1974 081E EE     OUT    DX,AL      | POINT TO RESET PARM
1975 081F 5A     POP    DX          | ADDRESS, NOT DATA, IS IMPORTANT
1976 0820                V7:
1977 0820 5D     POP    BP          | RECOVER VALUE
1978 0821 5F     POP    DI          | RETURN_NO_RESET
1979 0822 5E     POP    SI
1980 0823 1F     POP    DS
1981 0824 1F     POP    DS          | DISCARD SAVED BX,CX,DX
1982 0825 1F     POP    DS
1983 0826 1F     POP    DS
1984 0827 07     POP    ES
1985 0828 CF     IRET
1986 0829 READ_LPEN ENDP
1987 0829 CODE ENDS
1988                END

```

```

1      PAGE 110,121
2      TITLE BIOS1 ---- 01/10/86 INTERRUPT 15H BIOS ROUTINES
3      .LIST
4      0000      SEGMENT BYTE PUBLIC
5      CODE
6      PUBLIC CASSETTE_IO_1
7
8      EXTRN  CONF_TBL:NEAR          ; SYSTEM/BIOS CONFIGURATION TABLE
9      EXTRN  DOS:NEAR              ; LOAD (DS) WITH DATA SEGMENT SELECTOR
10
11     -----
12     INT 15 H
13     INPUT - CASSETTE I/O FUNCTIONS
14     ;
15     ; (AH) = 00H
16     ; (AH) = 01H
17     ; (AH) = 02H
18     ; (AH) = 03H
19     ; RETURNS FOR THESE FUNCTIONS ALWAYS (AH) = 86H, CY = 1)
20     ; IF CASSETTE PORT NOT PRESENT
21     -----
22     INPUT - UNUSED FUNCTIONS
23     ; (AH) = 04H THROUGH 7FH
24     ; RETURNS FOR THESE FUNCTIONS ALWAYS (AH) = 86H, CY = 1)
25     ; (UNLESS INTERCEPTED BY SYSTEM HANDLERS)
26     ; NOTE: THE KEYBOARD INTERRUPT HANDLER INTERRUPTS WITH AH=4FH
27     -----
28     EXTENSIONS
29     ; (AH) = 80H  DEVICE OPEN (NULL)
30     ; (BX) = DEVICE ID
31     ; (CX) = PROCESS ID
32     ;
33     ; (AH) = 81H  DEVICE CLOSE (NULL)
34     ; (BX) = DEVICE ID
35     ; (CX) = PROCESS ID
36     ;
37     ; (AH) = 82H  PROGRAM TERMINATION (NULL)
38     ; (BX) = DEVICE ID
39     ;
40     ; (AH) = 83H  EVENT WAIT (NULL)
41     ;
42     ; (AH) = 84H  JOYSTICK SUPPORT
43     ; (DX) = 00H - READ THE CURRENT SWITCH SETTINGS
44     ; RETURNS AL = SWITCH SETTINGS (BITS 7-4)
45     ; (DX) = 01H - READ THE RESISTIVE INPUTS
46     ; RETURNS AX = A(x) VALUE
47     ; BX = A(y) VALUE
48     ; CX = B(x) VALUE
49     ; DX = B(y) VALUE
50     ;
51     ; (AH) = 88H  EXTENDED MEMORY SIZE DETERMINE
52     ;
53     ; (AH) = 91H  INTERRUPT COMPLETE FLAG SET
54     ; (AL) TYPE CODE
55     ; 00H -> 7FH
56     ; SERIALIY REUSABLE DEVICES
57     ; OPERATING SYSTEM MUST SERIALIZE ACCESS
58     ; 80H -> BFH
59     ; REENRANT DEVICES; ES:BX IS USED TO
60     ; DISTINGUISH DIFFERENT CALLS (MULTIPLE I/O
61     ; CALLS ARE ALLOWED SIMULTANEOUSLY)
62     ; COH -> FFH
63     ; WAIT ONLY CALLS -- THERE IS NO
64     ; COMPLEMENTARY 'POST' FOR THESE WAITS.
65     ; THESE ARE TIMEOUT ONLY. TIMES ARE
66     ; FUNCTION NUMBER DEPENDENT.
67     ;
68     ; TYPE DESCRIPTION TIMEOUT
69     ;
70     ; 00H = DISK YES
71     ; 01H = DISKETTE YES
72     ; 02H = KEYBOARD NO
73     ; 80H = NETWORK NO
74     ; ES:BX --> NCB
75     ; FDH = DISKETTE MOTOR START YES
76     ; FEH = PRINTER YES
77     ;
78     ; (AH) = COH  RETURN CONFIGURATION PARAMETERS POINTER
79     ; RETURNS
80     ; (AH) = 00H AND CY= 0 (IF PRESENT ELSE 86 AND CY= 1)
81     ; (ES:BX) = PARAMETER TABLE ADDRESS POINTER
82     ; WHERE:
83     ;
84     ; DW 8 LENGTH OF FOLLOWING TABLE
85     ; DB MODEL_BYTE SYSTEM MODEL BYTE
86     ; DB TYPE_BYTE SYSTEM MODEL TYPE BYTE
87     ; DB BIOS_LEVEL BIOS REVISION LEVEL
88     ; 10000000 = DMA CHANNEL 3 USE BY BIOS
89     ; 01000000 = CASCADED INTERRUPT LEVEL 2
90     ; 00100000 = REAL TIME CLOCK AVAILABLE
91     ; 00010000 = KEYBOARD SCAN CODE HOOK 1AH
92     ;
93     ; DB 0 RESERVED
94     ; DB 0 RESERVED
95     ; DB 0 RESERVED
96     ;
97     -----
98     ASSUME CS:CODE
99
100    CASSETTE_IO_1 PROC FAR
101    0000 FB STI
102    0001 80 FC 80 CMP
103    0004 73 06 JAE
104
105    C1:
106    0006 B4 86 MOV AH,86H
107    0008 F9 STC
108
109    C1_F:
110    0009 CA 0002 RET 2
111
112    C1_G:
113    000C 80 FC C0 CMP AH,0C0H
114    000F 74 2E JE CONF_PARMS
115
116    ; ENABLE INTERRUPTS
117    ; CHECK FOR RANGE OF 00-7FH
118    ; SKIP AND HANDLE, ELSE RETURN ERROR
119
120    ; ERROR
121    ; SET BAD COMMAND
122    ; SET CARRY FLAG ON (CY=1)
123
124    ; COMMON EXIT
125    ; FAR RETURN EXIT FROM ROUTINES
126
127    ; CONTINUE CHECKING FOR FUNCTION
128    ; CHECK FOR CONFIGURATION PARAMETERS

```

```

115 0011 80 EC 80      SUB     AH,080H      ; BASE ON 0
116 0014 74 25        JZ     DEV_OPEN     ; DEVICE OPEN      (80H)
117 0016 FE CC        DEC     AH           ;
118 0018 74 21        JZ     DEV_CLOSE    ; DEVICE CLOSE     (81H)
119 001A FE CC        DEC     AH           ;
120 001C 74 1D        JZ     PROG_TERM    ; PROGRAM TERMINATION (82H)
121 001E FE CC        DEC     AH           ; IGNORE EVENT WAIT (83H)
122 0020 FE CC        DEC     AH           ;
123 0022 74 27        JZ     JOY_STICK    ; JOYSTICK BIOS    (84H)
124 0024 FE CC        DEC     AH           ;
125 0026 74 13        JZ     SYS_REQ      ; SYSTEM REQUEST KEY (85H)
126 0028 FE CC        DEC     AH           ; IGNORE WAIT       (86H)
127 002A FE CC        DEC     AH           ; IGNORE BLOCK MOVE (87H)
128 002C FE CC        DEC     AH           ;
129 002E 74 18        JZ     EXT_MEMORY   ; EXTENDED MEMORY SIZE (88H)
130
131 0030 80 EC 08      SUB     AH,8         ; CHECK FOR FUNCTION (90H)
132 0033 74 06        JZ     DEVICE_BUSY  ; DEVICE_BUSY
133 0035 FE CC        DEC     AH           ; CHECK FOR FUNCTION (91H)
134 0037 74 05        JZ     INT_COMPLETE ; GO TO INTERRUPT COMPLETE RETURN
135 0039 EB CB        JMP     C1           ; 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              DEV_BUSY:
142 003B FB          _CLC
143 003C EB CB        JMP     C1_F        ; TURN CARRY OFF
144                                     ; RETURN WITH (AH= 00) AND CY=0
145 003E              CASSETTE_10_1  ENDP
146
147
148 ;----- INTERRUPT COMPLETE -----;
149 ; THIS ROUTINE IS A TEMPORARY HANDLER ;
150 ; FOR INTERRUPT COMPLETE           ;
151 ; INPUT - SEE PROLOGUE             ;
152 ;-----;
153
154
155 003E              INT_COMPLETE  PROC  NEAR
156 003E CF              IRET
157 003F              INT_COMPLETE  ENDP
158                                     ; RETURN
159
160 003F 0E              CONF_PARMS  PROC  NEAR
161 0040 07              PUSH     CS
162 0041 BB 0000 E      MOV     BX,OFFSET CONF_TBL ; FUNCTION (C0H)
163 0044 32 E4          XOR     AH,AH           ; GET CODE SEGMENT
164 0046 EB C1          JMP     C1_F           ; PLACE IN SELECTOR POINTER
165 0048              CONF_PARMS  ENDP
166                                     ; GET OFFSET OF PARAMETER TABLE
167                                     ; CLEAR AH AND SET CARRY OFF
168                                     ; EXIT THROUGH COMMON RETURN
169
170 ;----- INT 15 H -- ( FUNCTION 86 H - I/O MEMORY SIZE DETERMINE ) -----;
171 ; EXT_MEMORY ;
172 ; THIS ROUTINE RETURNS THE AMOUNT OF MEMORY IN THE SYSTEM THAT IS ;
173 ; LOCATED STARTING AT THE 1024K ADDRESSING RANGE, AS DETERMINED BY ;
174 ; THE POST ROUTINES. ;
175 ; INPUT ;
176 ; AH = 88H ;
177 ; OUTPUT ;
178 ; (AX) = 0 ;
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
185                                     ; RETURN TO USER
186 004B              EXT_MEMORY   ENDP

```

```

187 PAGE
188 |----- JOY STICK -----|
189 | THIS ROUTINE WILL READ THE JOYSTICK PORT |
190 | |
191 | INPUT |
192 | (DX)=0 READ THE CURRENT SWITCHES |
193 | RETURNS (AL)= SWITCH SETTINGS IN BITS 7-4 |
194 | |
195 | (DX)=1 READ THE RESISTIVE INPUTS |
196 | RETURNS (AX)=A(X) VALUE |
197 | (BX)=B(X) VALUE |
198 | (CX)=B(A) VALUE |
199 | (DX)=B(Y) VALUE |
200 | |
201 |----- CY FLAG ON IF NO ADAPTER CARD OR INVALID CALL -----|
202 |
203 |
204 004B JOY_STICK PROC NEAR
205 004B FB STI ; INTERRUPTS BACK ON
206 004C 8B C2 MOV AX,DX ; GET SUB FUNCTION CODE
207 004E BA 0201 MOV DX,201H ; ADDRESS OF PORT
208 0051 0A C0 OR AL,AL
209 0053 74 09 JZ JOY_2 ; READ SWITCHES
210 0055 FE C8 DEC AL
211 0057 74 0A JZ JOY_3 ; READ RESISTIVE INPUTS
212 0059 EB AB JMP C1 ; GO TO ERROR RETURN
213 005B JOY_1:
214 005B FB STI ; GO TO COMMON RETURN
215 005C EB AB JMP C1_F
216
217 005E JOY_2:
218 005E EC IN AL,DX ; STRIP UNWANTED BITS OFF
219 005F 24 F0 AND AL,0F0H ; FINISHED
220 0061 EB F8 JMP JOY_1
221
222 0063 JOY_3:
223 0063 B3 01 MOV BL,1
224 0065 EB 0081 R CALL TEST_CORD
225 0068 51 PUSH CX ; SAVE A(X) VALUE
226 0069 B3 02 MOV BL,2
227 006B EB 0081 R CALL TEST_CORD
228 006E 51 PUSH CX ; SAVE A(Y) VALUE
229 006F B3 04 MOV BL,4
230 0071 EB 0081 R CALL TEST_CORD
231 0074 51 PUSH CX ; SAVE B(X) VALUE
232 0075 B3 08 MOV BL,8
233 0077 EB 0081 R CALL TEST_CORD
234 007A 6B D1 MOV CX,CX ; SAVE B(Y) VALUE
235 007C 59 POP CX ; GET B(X) VALUE
236 007D 5B POP BX ; GET A(Y) VALUE
237 007E 58 POP AX ; GET A(X) VALUE
238 007F EB DA JMP JOY_1 ; FINISHED - RETURN
239
240 0081 TEST_CORD PROC NEAR
241 0081 52 PUSH DX ; SAVE
242 0082 FA CLI ; BLOCK INTERRUPTS WHILE READING
243 0083 B0 00 MOV AL,0 ; SET UP TO LATCH TIMER 0
244 0085 E6 43 OUT TIMER+3,AL
245 0087 EB 00 JMP $+2
246 0089 E4 40 IN AL,TIMER ; READ LOW BYTE OF TIMER 0
247 008B EB 00 JMP $+2
248 008D 8A 00 MOV AH,AL ; READ HIGH BYTE OF TIMER 0
249 008F E4 40 IN AL,TIMER ; REARRANGE TO HIGH,LOW
250 0091 86 E0 XCHG AX ; SAVE
251 0093 50 PUSH AX ; SET COUNT
252 0094 B9 04FF MOV CX,4FFH ; FIRE TIMER
253 0097 EE OUT DX,AL
254 0098 EB 00 JMP $+2
255 009A TEST_CORD_1:
256 009A EC IN AL,DX ; READ VALUES
257 009B 84 C3 TEST AL,BL ; HAS PULSE ENDED?
258 009D ED FB LOOPNZ TEST_CORD_1
259 009F 83 F9 00 CMP CX,0
260 00A2 59 D1 POP CX ; ORIGINAL COUNT
261 00A3 75 04 JNZ SHORT TEST_CORD_2
262 00A5 2B C9 SUB CX,CX ; SET 0 COUNT FOR RETURN
263 00A7 EB 2D JMP SHORT TEST_CORD_3 ; EXIT WITH COUNT = 0
264 00A9 TEST_CORD_2:
265 00A9 B0 00 MOV AL,0 ; SET UP TO LATCH TIMER 0
266 00AB E6 43 OUT TIMER+3,AL
267 00AD EB 00 JMP $+2
268 00AF E4 40 IN AL,TIMER ; READ LOW BYTE OF TIMER 0
269 00B1 8A E0 MOV AH,AL ; READ HIGH BYTE OF TIMER 0
270 00B3 EB 00 JMP $+2 ; REARRANGE TO HIGH,LOW
271 00B5 E4 40 IN AL,TIMER ; READ HIGH BYTE OF TIMER 0
272 00B7 86 E0 XCHG AX ; REARRANGE TO HIGH,LOW
273
274 00B9 3B C8 CMP CX,AX ; CHECK FOR COUNTER WRAP
275 00BB 73 0B JAE TEST_CORD_4 ; GO IF NO
276 00BD 52 JAE DX
277 00BE BA FFFF MOV DX,-1
278
279 00C1 2B D0 SUB DX,AX ; ADJUST FOR WRAP
280 00C3 03 CA ADD CX,DX
281 00C5 5A POP DX
282 00C6 EB 02 JMP SHORT TEST_CORD_5
283
284 00C8 TEST_CORD_4:
285 00C8 2B C8 SUB CX,AX
286 00CA TEST_CORD_5:
287 00CA 81 E1 1FF0 AND CX,1FF0H ; ADJUST
288 00CC D1 E9 SHR CX,1
289 00DD D1 E9 SHR CX,1
290 00DE D1 E9 SHR CX,1
291 00DF D1 E9 SHR CX,1
292
293 00D6 TEST_CORD_3:
294 00D6 FB STI ; INTERRUPTS BACK ON
295 00D7 BA 0201 MOV DX,201H ; FLUSH OTHER INPUTS
296 00DA 51 PUSH CX
297 00DB 50 PUSH AX
298 00DD B9 04FF MOV CX,4FFH ; COUNT
299 00DF TEST_CORD_6:
300 00DF EC IN AL,DX

```

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

```

1 PAGE 118,121
2 TITLE POST ----- 01/10/86 SYSTEM POST AND BIOS PROCEDURES
3
4 PUBLIC A1
5 PUBLIC BEEP
6 PUBLIC CONF_TBL
7 PUBLIC CRT_CHAR_GEN
8 PUBLIC DDS
9 PUBLIC DISK_BASE
10 PUBLIC M5
11 PUBLIC M6
12 PUBLIC M7
13 PUBLIC MD_TBL1
14 PUBLIC MD_TBL2
15 PUBLIC MD_TBL3
16 PUBLIC MD_TBL4
17 PUBLIC MD_TBL5
18 PUBLIC MD_TBL6
19 PUBLIC P_O_R
20 PUBLIC RESET
21 PUBLIC VIDEO_PARMS
22 PUBLIC WAITF
23
24 EXTRN CASSETTE_IO_1:NEAR
25 EXTRN DISKETTE_IO_1:NEAR
26 EXTRN DISK_INT_1:NEAR
27 EXTRN DSKETTE_SETUP:NEAR
28 EXTRN KB_INT_1:NEAR
29 EXTRN KEYBOARD_IO_1:NEAR
30 EXTRN NEC_OUTPUT:NEAR
31 EXTRN PRINTER_IO_1:NEAR
32 EXTRN RESULTS:NEAR
33 EXTRN RS232_IO_1:NEAR
34 EXTRN SEEK:NEAR
35 EXTRN VIDEO_IO_1:NEAR
36
37 EXTRN SET_MODE:NEAR
38 EXTRN SET_CTYPE:NEAR
39 EXTRN SET_CPOS:NEAR
40 EXTRN READ_CURSOR:NEAR
41 EXTRN READ_LFEN:NEAR
42 EXTRN ACT_DISP_PAGE:NEAR
43 EXTRN SCROLL_UP:NEAR
44 EXTRN SCROLL_DOWN:NEAR
45 EXTRN READ_AC_CURRENT:NEAR
46 EXTRN WRITE_AC_CURRENT:NEAR
47 EXTRN WRITE_C_CURRENT:NEAR
48 EXTRN SET_COLOR:NEAR
49 EXTRN WRITE_DOT:NEAR
50 EXTRN READ_DOT:NEAR
51 EXTRN WRITE_TTY:NEAR
52 EXTRN VIDEO_STATE:NEAR
53
54 .LIST
55 -----
56 | THE BIOS ROUTINES ARE MEANT TO BE ACCESSED THROUGH |
57 | SOFTWARE INTERRUPTS ONLY. ANY ADDRESSES PRESENT IN |
58 | THE LISTINGS ARE INCLUDED ONLY FOR COMPLETENESS, |
59 | NOT FOR REFERENCE. APPLICATIONS WHICH REFERENCE |
60 | ABSOLUTE ADDRESSES WITHIN THE CODE SEGMENT |
61 | VIOLATE THE STRUCTURE AND DESIGN OF BIOS. |
62 -----
63
64 | ROM RESIDENT CODE |
65 -----
66
67 0000 CODE SEGMENT BYTE PUBLIC
68
69 0000 1FFF [ DB 01FFFH DUP (0CCH) ; FILL UNUSED LOCATIONS WITH INTERRUPT 3
70 ] CC
71
72 0000 ORG 0E000H
73 0000 36 32 58 30 38 35 ORG 0
74 31 20 43 4F 50 52 DB '62X0851 COPR. IBM 1986' ; COPYRIGHT NOTICE
75 2E 20 49 42 4D 20
76 31 39 38 36
77
78 | INITIAL RELIABILITY TESTS -- PHASE I |
79 -----
80
81 ASSUME CS:CODE,SS:CODE,ES:ABS0,DS:DATA
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 | LOAD A BLOCK OF TEST CODE THROUGH THE KEYBOARD PORT |
90 | FOR MANUFACTURING TEST |
91 | THIS ROUTINE WILL LOAD A TEST (MAX LENGTH=FAFFH) THROUGH |
92 | THE KEYBOARD PORT. CODE WILL BE LOADED AT LOCATION |
93 | 0000:0500. AFTER LOADING, CONTROL WILL BE TRANSFERRED |
94 | TO LOCATION 0000:0500. STACK WILL BE LOCATED JUST BELOW |
95 | THE TEST CODE. THIS ROUTINE ASSUMES THAT THE FIRST 2 |
96 | BYTES TRANSFERRED CONTAIN THE COUNT OF BYTES TO BE LOADED |
97 | (BYTE 1=COUNT LOW, BYTE 2=COUNT HI.) |
98 -----
99
100 |----- FIRST, GET THE COUNT
101
102 0021 MFG_BOOT:
103 0021 E8 19F0 R CALL SP_TEST ; GET COUNT LOW
104 0024 8A FB MOV ; SAVE IT
105 0026 E8 19F0 R CALL SP_TEST ; GET COUNT HI
106 0029 8A EB MOV CB,BL ; CX NOW HAS COUNT
107 002D FC MOV CL,BH ; SET DIR. FLAG TO INCRIMENT
108 002E FA CLI
109 002F BF 0500 MOV DI,0500H ; SET TARGET OFFSET (DS=0000)
110 0032 B0 FD MOV AL,0FDH ; UNMASK K/B INTERRUPT
111 0034 E6 21 OUT INTA01,AL
112 0036 B0 0A MOV AL,0AH ; SEND READ INT. REQUEST REG. CMD
113 0038 E6 20 OUT INTA00,AL
114 003A BA 0061 MOV DX,PORT_B ; SET UP PORT B ADDRESS
    
```



```

338 PAGE
339 -----
340 : BASE 64K READ/WRITE STORAGE TEST :
341 : DESCRIPTION :
342 : WRITE/READ/VERIFY DATA PATTERNS :
343 : AA,55,FF,01, AND 00 TO 1ST 64K OF :
344 : STORAGE. VERIFY STORAGE ADDRESSABILITY. :
345 -----
346
347 0166 AD LODSW ; ALLOW RAM CHARGE TIME.
348 0167 AD LODSW
349 0168 AD LODSW
350 0169 AD LODSW
351
352 :----- DETERMINE MEMORY SIZE AND FILL MEMORY WITH DATA
353
354 016A 8B 1E 0472 R MOV BX,DATA_WORD[RESET_FLAG-DATA40] ; SAVE 'RESET_FLAG' IN BX
355 016E 8B 2E 0496 R MOV BP,DATA_WORD[0KB_FLAG_3-DATA40] ; SAVE KEYBOARD TYPE
356 0172 B9 8000 MOV CX,08000H ; SET FOR 32K WORDS
357 0176 81 FB 1234 CMP BX,1234H ; WARM START?
358 0179 74 16 JE CLR_STG
359 017B BC 0018 R MOV SP,OFFSET C2
360 017E 59 0CCF R JMP STG1ST_CNT
361 0181 74 12 JE HOW_BIG ; STORAGE OK, DETERMINE SIZE
362 0183 8A D8 MOV BL,AL ; SAVE FAILING BIT PATTERN
363 0185 80 04 MOV AL,04H ; BP IS USED LATER AS AN ERROR INDICATOR
364 0187 E6 60 C24A: OUT PORT_A,AL ; <<<<<CHECKPOINT 4<<<<<
365 0189 2B C9 SUB CX,CX ; BASE RAM FAILURE - HANG
366 018B E2 FE C24B: LOOP C24B ; FLIPPING BETWEEN 04 AND
367 018D 86 D8 XCHG BL,AL ; FAILING BIT PATTERN
368 018F EB F6 C24A: JMP C24A
369 0191 CLR_STG:
370 0191 2B C0 SUB AX,AX ; MAKE AX=0000
371 0193 F3/ AB REP STOSW ; STORE 32K WORDS OF 0000
372 0195 HOW_BIG:
373 0195 89 1E 0472 R MOV DATA_WORD[RESET_FLAG-DATA40],BX ; RESTORE RESET FLAG
374 0199 83 FD 10 CMP BP,KBX ; IS THE KBX BIT THE ONLY ONE ON?
375 019C 74 02 JE C24C ; IF NOT THEN THIS MUST BE A P.O.R.
376 019E 2B ED SUB BP,BP ; IF P.O.R., THEN INITIALIZE THIS TO ZERO
377 01A0 C24C:
378 01A0 89 2E 0496 R MOV DATA_WORD[0KB_FLAG_3-DATA40],BP ; RESTORE RESET FLAG
379 01A4 2B ED SUB BP,BP ; BP IS USED LATER AS AN ERROR INDICATOR
380 01A6 BA 0400 MOV DX,0400H ; SET POINTER TO JUST>16KB
381 01A9 BB 0010 MOV BX,16 ; BASIC COUNT OF 16K
382 01AC FILL_LOOP:
383 01AC 8E C2 MOV ES,DX ; SET SEG. REG.
384 01AE 2B FF SUB DI,DI ; TEST PATTERN
385 01B0 B8 AA55 MOV AX,0AA55H ; SAVE PATTERN
386 01B3 8B C8 MOV CX,AX ; SEND PATTERN TO MEM.
387 01B5 26; 89 05 MOV SI,[DI],AX ; PUT SOMETHING IN AL
388 01B8 80 0F MOV AL,0FH ; GET PATTERN
389 01BA 26; 8B 05 MOV AX,ES:[DI] ; COMPARE PATTERNS
390 01BD 33 C1 XOR AX,CX ; COMPARE PATTERNS
391 01BF 75 11 JNZ HOW_BIG_END ; GO END IF NO COMPARE
392 01C1 B9 2000 MOV CX,2000H ; SET COUNT FOR 8K WORDS
393 01C4 F3/ AB REP STOSW ; FILL 8K WORDS
394 01C6 81 C2 0400 ADD DX,400H ; POINT TO NEXT 16KB BLOCK
395 01CA C3 C3 10 ADD BX,16 ; BUMP COUNT BY 16KB
396 01CD 80 FE A0 CMP DH,0A0H ; TOP OF RAM AREA YET? (A0000)
397 01D0 75 DA JNZ FILL_LOOP
398 01D2 HOW_BIG_END:
399 01D2 89 1E 0413 R MOV DATA_WORD[MEMORY_SIZE-DATA40],BX ; SAVE MEMORY SIZE
400
401 :----- SETUP STACK SEG AND SP
402
403 01D6 B8 0030 MOV AX,STACK_SS ; GET STACK VALUE
404 01D9 BE D0 MOV SS,AX ; SET THE STACK UP
405 01DB BC 0100 MOV SP,TOS ; STACK IS READY TO GO
406
407 :----- INITIALIZE THE 8259 INTERRUPT CONTROLLER CHIP
408
409 01DE B0 13 C25: MOV AL,13H ; ICW1 - EDGE, SNGL, ICW4
410 01E0 E6 20 OUT INTA00,AL ; SETUP ICW2 - INT TYPE 8 (8-F)
411 01E2 B0 08 MOV AL,8 ; SETUP ICW4 - BUFFRD,8086 MODE
412 01E4 E6 21 OUT INTA01,AL ; MASK ALL INTS. OFF
413 01E6 B0 09 MOV AL,9 ; (VIDEO ROUTINE ENABLES INTS.)
414 01E8 E6 21 OUT INTA01,AL
415 01EA B0 FF MOV AL,0FFH
416 01EC E6 21 OUT INTA01,AL
417
418 :----- SET UP THE INTERRUPT VECTORS TO TEMP INTERRUPT
419
420 01EE IE PUSH DS
421 01EF B9 0020 MOV CX,32 ; FILL ALL 32 INTERRUPTS
422 01F2 2B FF SUB DI,DI ; SET INTERRUPT LOCATION
423 01F4 BE C7 MOV ES,DI ; SET ES=0000 ALSO
424 01F6 B8 1F23 R D3: MOV AX,OFFSET D11 ; MOVE ADDR OF INTR PROC TO TBL
425 01F9 AB STOSW
426 01FA 8C C8 MOV AX,C5 ; GET ADDR OF INTR PROC SEG
427 01FC AB STOSW
428 01FD E2 F7 LOOP D3 ; VECTBL0
429
430 :----- ESTABLISH BIOS SUBROUTINE CALL INTERRUPT VECTORS
431
432 01FF BF 0040 R MOV DI,OFFSET VIDEO_INT ; SETUP ADDR TO INTR AREA
433 0202 0E PUSH CS ; SETUP ADDR OF VECTOR TABLE
434 0203 1F POP DS ; START WITH VIDEO ENTRY
435 0204 BE 1F03 R MOV SI,OFFSET VECTOR_TABLE+16
436 0207 B9 0010 MOV CX,16
437 020A A5 D3A: MOVSW ; MOVE VECTOR TABLE TO RAM
438 020B 47 INC DI ; SKIP SEGMENT POINTER
439 020C 47 INC DI
440 020D E2 FB LOOP D3A
441
442 :----- DETERMINE CONFIGURATION AND MFG. MODE
443
444
445 020F 1F POP DS ; RECOVER DATA SEG
446 0210 IE PUSH DS ; GET SWITCH INFO
447 0211 E4 62 IN AL,PORT_C ; ISOLATE SWITCHES
448 0213 24 0F AND AL,00001111B
449 0215 8A E0 MOV AH,AL ; SAVE
450 0217 BD AD MOV AL,10101010B ; ENABLE OTHER BANK OF SWS.
451 0219 E6 61 OUT PORT_B,AL

```

SECTION 5

```

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                ; ROTATE TO HIGH NIBBLE
456 0222 24 F0            AND     AL,11110000B        ; ISOLATE SWS
457 0224 0A C4            OR     AL,AH                ; COMBINE WITH OTHER BANK
458 0226 2A E4            SUB     AH,AH
459 0228 A3 0410 R        MOV     DATA_WORD[0*EQUIP_FLAG-DATA40],AX    ; SAVE SWITCH INFO
460 022B B0 99            MOV     AL,99
461 022D E6 63            OUT     CMD_PORT,AL
462 022F E8 19E3 R        CALL   KBD_RESET            ; SEE IF MFG. JUMPER IN
463 0232 80 FB EA        CMP     BL,0EAH            ; IS THIS THE EXTENDED KEYBOARD?
464 0235 75 08            JNE     MBI                 ; IF NOT THEN LEAVE THE FLAG ALONE
465 0237 C6 06 0496 R 10 MOV     DATA_AREA[0*KB_FLAG_3-DATA40],KBX    ; EXTENDED KEYBOARD
466 0239 EB 22 90            JMP     E6                 ; DONE WITH KEYBOARD HERE
467 023F
468 023F 80 FB AA        KBX1: CMP     BL,0AAH            ; KEYBOARD PRESENT?
469 0242 74 1C            JE     E6
470 0244 80 FB 65        CMP     BL,065H            ; LOAD MFG. TEST REQUEST?
471 0247 75 03            JNE     D3B
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 024E 75 10            JNZ     E6                 ; NO
476 0250 B0 38            MOV     AL,38H
477 0252 E6 61            OUT     PORT_B,AL
478 0254 90            NOP
479 0255 90            NOP
480 0256 E4 60            IN     AL,PORT_A
481 0258 2A FF            AND     AL,OFFH           ; WAS DATA LINE GROUNDED
482 025A 75 04            JNZ     E6
483 025C FE 06 0412 R    INC     DATA_AREA[0*MGF_TST-DATA40]        ; SET MANUFACTURING TEST FLAG
484
485
486 -----
487 | INITIALIZE AND START CRT CONTROLLER (6845) |
488 | TEST VIDEO READ/WRITE STORAGE. |
489 | DESCRIPTION |
490 | RESET THE VIDEO ENABLE SIGNAL. |
491 | SELECT ALPHANUMERIC MODE, 40 * 25, B & W. |
492 | READ/WRITE DATA PATTERNS TO STG. CHECK STG |
493 | ADDRESSABILITY. |
494 | ERROR = 1 LONG AND 2 SHORT BEEPS |
495 -----
496 0260
497 0260 A1 0410 R        E6:  MOV     AX,DATA_WORD[0*EQUIP_FLAG-DATA40]    ; GET SENSE SWITCH INFO
498 0263 50                PUSH   AX                    ; SAVE IT
499 0264 B0 30            MOV     AL,30H
500 0266 A3 0410 R        MOV     DATA_WORD[0*EQUIP_FLAG-DATA40],AX
501 0269 2A E4            SUB     AH,AH
502 026B CD 10            INT     10H                ; SEND INIT TO B/W CARD
503 026D B0 20            MOV     AL,20H
504 026F A3 0410 R        MOV     DATA_WORD[0*EQUIP_FLAG-DATA40],AX
505 0272 2A E4            SUB     AH,AH              ; AND INIT COLOR CARD
506 0274 CD 10            INT     10H
507 0276 58            POP     AX                  ; RECOVER REAL SWITCH INFO
508 0277 A3 0410 R        MOV     DATA_WORD[0*EQUIP_FLAG-DATA40],AX    ; RESTORE IT
509 027A 24 30            AND     AL,30H             ; AND CONTINUE
510 027C 75 0A            JNZ     E7                 ; ISOLATE VIDEO SWS
511 027E BF 0040 R        MOV     DI,OFFSET 0VIDEO INT ; SET INT 10H TO DUMMY
512 0281 C7 05 1F49 R    MOV     WORD_PTR [DI],OFFSET DUMMY_RETURN    ; RET IF NO VIDEO CARD
513 0285 E9 033B R        JMP     E1B_1              ; B/W/S VIDEO TEST
514 0288
515 0288 3C 30            E7:  CMP     AL,30H             ; TEST VIDEO1
516 028A 74 08            JNB     E7                 ; B/W CARD ATTACHED?
517 028C FE C4            INC     AH                  ; YES - SET MODE FOR B/W CARD
518 028E 3C 20            CMP     AL,20H             ; SET COLOR MODE FOR COLOR CD
519 0290 75 02            JNE     E8                 ; 80X25 MODE SELECTED?
520 0292 B4 03            MOV     AH,3                ; NO - SET MODE FOR 40X25
521 0294 86 0E            XCHG   AH,AL               ; SET MODE FOR 80X25
522 0296 50            PUSH   AX                  ; SET MODE1
523 0297 2A E4            SUB     AH,AH              ; SAVE VIDEO MODE ON STACK
524 0299 CD 10            INT     10H                ; INITIALIZE TO ALPHANUMERIC MD
525 029B 58            POP     AX                  ; CALL VIDEO 10
526 029C 50            PUSH   AX                  ; RESTORE VIDEO SENSE SWS IN AH
527 029D BB B000        MOV     BX,0B000H          ; RESAVE VALUE
528 02A0 EB 24            JMP     SHORT E8A          ; BEG VIDEO RAM ADDR B/W CD
529
530
531 |----- UNNATURAL ACT FOR ADDRESS COMPATIBILITY
532 |
533 | ORG 0E2C3H
534 | ORG 002C3H
535 | JMP NMI_INT_1
536
537 02C6
538 02C6 BA 038B        E8A: MOV     DX,3808H           ; MODE REG FOR B/W
539 02C9 B9 0800        MOV     CX,2048H          ; RAM WORD CNT FOR B/W CD
540 02CC B0 01            MOV     AL,1                ; SET MODE FOR B/W CARD
541 02CE 80 FC 30        CMP     AH,30H            ; B/W VIDEO CARD ATTACHED?
542 02D1 74 09            JE     E9                 ; YES - GO TEST VIDEO STG
543 02D3 B7 08            MOV     BH,08BH           ; BEG VIDEO RAM ADDR COLOR CD
544 02D5 BA 03DB        MOV     DX,3D8H           ; MODE REG FOR COLOR CD
545 02D8 B5 20            MOV     CH,20H            ; RAM WORD CNT FOR COLOR CD
546 02DA FE C8            DEC     AL                  ; SET MODE TO 0 FOR COLOR CD
547 02DC
548 02DC EE            E9:  OUT     DX,AL              ; TEST VIDEO STG1
549 02DD 81 3E 0472 R 1234 CMP     DATA_WORD[0*RESET_FLAG-DATA40],1234H ; DISABLE VIDEO FOR COLOR CD
550 02E3 8E C9            MOV     BX,0                ; POINT ES TO VIDEO RAM STG
551 02E5 74 07            JE     E10                 ; YES - SKIP VIDEO RAM TEST
552 02E7 8E DB            MOV     DS,BX              ; POINT DS TO VIDEO RAM STG
553
554 02E9 E8 0CFC R        ASSUME DS:NOTHING,ES:NOTHING
555 02EC 75 33            CALL   STGCTST_CNT        ; GO TEST VIDEO R/W STG
556 02EE
557 02EE 58            JNE     E17                ; R/W STG FAILURE - BEEP SPK
558
559 -----
560 | SETUP VIDEO DATA ON SCREEN FOR VIDEO |
561 | LINE TEST. |
562 | DESCRIPTION |
563 | ENABLE VIDEO SIGNAL AND SET MODE. |
564 | DISPLAY A HORIZONTAL BAR ON SCREEN. |
565 -----
566 02EE
567 02EE 58            E10: POP     AX                  ; GET VIDEO SENSE SWS (AH)
568 02EF 50            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 B8 7020    MOV     AX,7020H     ; WRT BLANKS IN REVERSE VIDEO
569
570 02F7 2B FF      SUB     DI,DI        ; SETUP STARTING LOC
571 02F9 B9 0028    MOV     CX,40        ; NO. OF BLANKS TO DISPLAY
572 02FC F3/ AB     REP     STOSW        ; WRITE VIDEO STORAGE
573
574
575 ;-----
576 ; CRT INTERFACE LINES TEST
577 ; DESCRIPTION
578 ; SENSE ON/OFF TRANSITION OF THE
579 ; VIDEO ENABLE AND HORIZONTAL
580 ; SYNC LINES.
581 ;-----
582 02FE 58        POP     AX           ; GET VIDEO SENSE SW INFO
583 02FF 50        PUSH    AX          ; SAVE IT
584 0300 80 FC 30  CMP     AH,30H      ; IF #CARD ATTACHED?
585 0303 BA 03BA    MOV     DX,03BAH   ; SETUP ADDR OF BW STATUS PORT
586 0306 74 03     JE     E11          ; YES - GO TEST LINES
587 0308 BA 03DA    MOV     DX,03DAH   ; COLOR CARD IS ATTACHED
588 030B           ; LINE_TST
589 030D           E11: MOV     AH,8        ;
590 030F           E12: SUB     CX,CX     ; OFLOOP_CNT
591 0310 EC           E13: IN     AL,DX      ; READ CRT STATUS PORT
592 0312 22 C4     AND    AL,AH       ; CHECK VIDEO/HORZ LINE
593 0314 75 04     JNZ    E14         ; ITS ON - CHECK IF IT GOES OFF
594 0316 E2 F9     LOOP  E13         ; LOOP TILL ON OR TIMEOUT
595 0318 EB 09     JMP     SHORT E17  ; GO PRINT ERROR MSG
596 0318           E14: SUB     CX,CX     ;
597 031A 2B C9     MOV     AH,8        ;
598 031A           E15: IN     AL,DX      ; READ CRT STATUS PORT
599 031B 22 C4     AND    AL,AH       ; CHECK VIDEO/HORZ LINE
600 031D 74 11     JZ     E17         ; ITS ON - CHECK NEXT LINE
601 031F E2 F9     LOOP  E15         ; LOOP IF OFF TILL IT GOES ON
602 0321           E17: POP     DS         ; CRT_ERR
603 0321           POP     DS         ;
604 0322 1F        PUSH    DS         ;
605 0322 1E        MOV     DS:#MFG_ERR_FLAG,06H ; <<<<<<CRT ERR CHKPT. 06<<<<<<
606 0323 C6 06 0015 R 06 ;
607 0328 BA 0102    MOV     DX,102H    ; GO BEEP SPEAKER
608 032B E8 19A5 R  CALL  ERR_BEEP     ;
609 032E EB 06     JMP     SHORT E18  ;
610 0330           E16:           ; NXT LINE
611 0330 B1 03     MOV     CL,3       ; GET NEXT BIT TO CHECK
612 0332 D2 EC     SHR    AH,CL       ;
613 0334 75 D7     JNZ    E18         ; GO CHECK HORIZONTAL LINE
614 0336           E18: POP     AX         ; DISPLAY CURSOR
615 0336 58        POP     AX         ; GET VIDEO SENSE SWS (AH)
616 0337 B4 00     MOV     AH,0       ; SET MODE AND DISPLAY CURSOR
617 0339 CD 10     INT    10H        ; CALL VIDEO I/O PROCEDURE
618 033B           E18_1:
619 033B BA C000    MOV     DX,0C000H ; SEE IF ADVANCED VIDEO CARD
620 033E           E18A:
621 033E 0E DA     MOV     DS,DX      ; IS PRESENT
622 0340 2B DB     SUB     BX,BX      ;
623 0342 8B 07     MOV     AX,[BX]   ; GET FIRST 2 LOCATIONS
624 0344 53        PUSH   BX         ;
625 0345 5B        POP    BX         ;
626 0346 3D AA55   CMP     AX,0AA55H ; LET BUS SETTLE
627 0349 75 05     JNZ    E18B       ; IS PRESENT?
628 034B E8 1920 R CALL  ROM_CHECK   ; NO? GO LOOK FOR OTHER MODULES
629 034E EB 04     JMP     SHORT E18C ; GO SCAN MODULE
630 0350           E18B: ADD     DX,0080H  ; POINT TO NEXT 2K BLOCK
631 0350 81 C2 0080 ;
632 0354           E18C: CMP     DX,0C800H ; TOP OF VIDEO ROM AREA YET?
633 0354 81 FA C800 ; GO SCAN FOR ANOTHER MODULE
634 0358 7C E4     JL     E18A       ;
635
636 ;-----
637 ; 8259 INTERRUPT CONTROLLER TEST
638 ; DESCRIPTION
639 ; READ/WRITE THE INTERRUPT MASK REGISTER (IMR)
640 ; WITH ALL ONES AND ZEROES. ENABLE SYSTEM
641 ; INTERRUPTS. MASK DEVICE INTERRUPTS OFF. CHECK
642 ; FOR HOT INTERRUPTS (UNEXPECTED).
643 ;-----
644 035A 1F        ASSUME DS:ABS0
645 035A           C21: POP     DS
646
647 ;----- TEST THE IMR REGISTER
648 035B C6 06 0415 R 05 ;
649
650
651 0360 B0 00     MOV     AL,0       ; SET IMR TO ZERO
652 0362 E6 21     OUT    INTA01,AL  ;
653 0364 E4 21     IN     AL,INTA01  ; READ IMR
654 0366 0A C0     OR     AL,AL       ; IMR = 0?
655 0368 75 1B     JNZ    D6         ; GO TO ERR ROUTINE IF NOT 0
656 036A B0 FF     MOV     AL,0FFH   ; DISABLE DEVICE INTERRUPTS
657 036C E6 21     OUT    INTA01,AL  ; WRITE TO IMR
658 036E E4 21     IN     AL,INTA01  ; READ IMR
659 0370 04 01     ADD    AL,1       ; ALL IMR BIT ON?
660 0372 75 11     JNZ    D6         ; NO - GO TO ERR ROUTINE
661
662 ;----- CHECK FOR HOT INTERRUPTS
663
664 ;----- INTERRUPTS ARE MASKED OFF. CHECK THAT NO INTERRUPTS OCCUR.
665
666 0374 A2 046B R  MOV     DATA_AREA[#INTR_FLAG-DATA40],AL ; CLEAR INTERRUPT FLAG
667 0377 FB      STI     ; ENABLE EXTERNAL INTERRUPTS
668 0378 2B C9    SUB     CX,CX      ; WAIT 1 SEC FOR ANY INTRTS THAT
669 037A           D4: LOOP  D4         ; MIGHT OCCUR
670 037A E2 FE    LOOP  D4         ;
671 037C           D5: LOOP  D5         ;
672 037C E2 FE    CMP     DATA_AREA[#INTR_FLAG-DATA40],00H ; ANY INTERRUPTS OCCUR?
673 037E 80 3E 046B R 00 ; NO - GO TO NEXT TEST
674 0383 74 08     JZ     D6         ;
675 0385           D6: MOV     SI,OFFSET E0 ; DISPLAY 101 ERROR
676 0385 BE 18CC R  CALL  E_MSG       ;
677 0388 E8 1976 R  MOV     CL,1       ;
678 038B FA      HLT             ; HALT THE SYSTEM
679 038C F4

```

SECTION 5


```

794 0448 B0 FE      MOV     AL,0FEH      ; ENABLE TIMER INTERRUPT
795 044A E6 21      OUT     INTA0,AL
796
797
798 ; EXPANSION I/O BOX TEST
799 ; CHECK TO SEE IF EXPANSION BOX PRESENT - IF INSTALLED,
800 ; TEST DATA AND ADDRESS BUSES TO I/O BOX
801 ; ERROR='1801'
802
803
804
805
806 044C BA 0210     EXP_10:  ; (CARD WAS ENABLED EARLIER)
807 044F B8 5555     MOV     DX,0210H    ; CONTROL PORT ADDRESS
808 0452 EE          OUT     DX,AL       ; SET DATA PATTERN
809 0453 B0 01       MOV     AL,01H      ; MAKE AL DIFFERENT
810 0455 EC          IN      AL,DX       ; RECOVER DATA
811 0456 3A C4       CMP     AL,AH       ; REPLY?
812 0458 75 43       JNE    E19          ; NO RESPONSE, GO TO NEXT TEST
813 045A F7 D0       NOT     AX           ; MAKE DATA=AAAA
814 045C EE          OUT     DX,AL
815 045D B0 01       MOV     AL,01H      ; RECOVER DATA
816 045F EC          IN      AL,DX
817 0460 3A C4       CMP     AL,AH
818 0462 75 39       JNE    E19
819
820
821
822 0464
823 0464 BB 0001     EXP2:    MOV     BX,0001H    ; LOAD HI ADDR. REG ADDRESS
824 0467 BA 0215     MOV     DX,0215H    ; GO ACROSS 16 BITS
825 046A B9 0010     MOV     CX,0016
826 046D
827 046D 2E: 88 07   EXP3:    MOV     CS:[BX],AL  ; WRITE ADDRESS F0000+BX
828 0470 90         NOP
829 0471 EC          IN      AL,DX       ; READ ADDR. HIGH
830 0472 3A C7       CMP     AL,BH
831 0474 75 21       JNE    EXP_ERR      ; GO ERROR IF MISCOMPARE
832 0476 42         INC     DX           ; DX=216H (ADDR. LOW REG)
833 0477 EC          IN      AL,DX
834 0478 3A C3       CMP     AL,BL
835 047A 75 1B       JNE    EXP_ERR      ; COMPARE TO LOW ADDRESS
836 047C 4A         DEC     DX           ; DX BACK TO 215H
837 047D D1 E3       SHL     BX,1
838 047F E2 EC       LOOP   EXP3         ; LOOP TILL '1' WALKS ACROSS BX
839
840
841
842 0481 B9 0008     ; CHECK DATA BUS
843 0484 B0 01       MOV     CX,0008H    ; DO 8 TIMES
844 0486 4A         MOV     AL,01
845 0487             DEC     DX           ; MAKE DX=214H (DATA BUS REG)
846 0487 BA 8A E0     EXP4:    MOV     AH,AL       ; SAVE DATA BUS VALUE
847 0489 EE          OUT     DX,AL       ; SEND VALUE TO REG
848 048A B0 01       MOV     AL,01H
849 048C EC          IN      AL,DX
850 048D 3A C4       CMP     AL,AH       ; RETRIEVE VALUE FROM REG
851 048F 75 06       JNE    SHORT EXP_ERR ; = TO SAVED VALUE
852 0491 D0 E0       SHL     AL,1
853 0493 E2 F2       LOOP   EXP4         ; FORM NEW DATA PATTERN
854 0495 EB 06       JMP     SHORT E19    ; LOOP TILL BIT WALKS ACROSS AL
855 0497
856 0497 BE 18DC R   EXP_ERR: MOV     SI,OFFSET F3C ; GO ON TO NEXT TEST
857 049A EB 1976 R   CALL    E_MSG
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
    
```

SECTION 5


```

1022 05A3 E2 FE          F12:  LOOP    F11          ; WAIT FOR 1 SECOND
1023 05A5                ; MOTOR_WAIT1:
1024 05A5 E2 FE          LOOP    F12
1025 05A7 33 D2          XOR     DX,DX          ; SELECT DRIVE 0
1026 05A9 B5 22          MOV     MOV             ; SELECT TRACK 34
1027 05AB 88 16 003E R  #SEEK   #SEEK_STATUS,DL
1028 05AF E8 0000 E      CALL   SEEK           ; RECALIBRATE DISKETTE AND SEEK TO 34
1029 05B2 73 05          JNC    F14            ; OK--> GO TURN OF MOTOR
1030 05B4                ; DISKETTE ERROR
1031 05B4 BE 0990 R      MOV     SI,OFFSET F3  ; GET ADDR OF MSG
1032 05B7 EB 02          JMP     SHORT F14A     ; DISPLAY MESSAGE AFTER DISKETTE SETUP
1033
1034                ;----- TURN DRIVE 0 MOTOR OFF
1035
1036 05B9                F14:   XOR     SI,SI          ; SEQUENCE END ENTRY IF NO ERROR
1037 05B9 33 F6          XOR     SI,SI          ; ZERO SI IF NO ERROR
1038 05BB                F14A:  MOV     AL,0CH         ; SEQUENCE END ENTRY IF ERROR
1039 05BB B0 0C          MOV     AL,0CH         ; TURN DRIVE 0 MOTOR OFF
1040 05BD BA 03F2        MOV     DX,03F2H      ; FDC CTL ADDRESS
1041 05C0 EE            OUT     DX,AL
1042
1043                ;-----SETUP DISKETTE STATES
1044
1045 05C1 E8 0000 E      CALL   DSKETTE_SETUP ; INITIALIZE DISKETTE PARMS
1046 05C4 72 04          JC     F14B           ; CY-->DISKETTE SETUP ERROR
1047 05C6 0B F6          OR     SI,SI           ; PREVIOUS DISKETTE ERROR
1048 05C8 74 06          JZ     F15            ; NZ-->DISKETTE ERROR BEFORE SETUP
1049 05CA                F14B:  MOV     SI,OFFSET F3  ; GET ADDR OF MSG
1050 05CA BE 0990 R      CALL   E_MSG          ; GO PRINT ERROR MSG
1051 05CD EC 1976 R
1052
1053                ;----- SETUP PRINTER AND RS232 BASE ADDRESSES IF DEVICE ATTACHED
1054
1055 05D0                F15:   MOV     #INTR_FLAG,00H ; SET SETUP INTERRUPT FLAG = 00
1056 05D0 C6 06 006B R 00 ; SI OFFSET #KB BUFFER
1057 05D5 BE 001E R      MOV     #BUFFER_HEAD,SI ; SETUP KEYBOARD PARAMETERS
1058 05D8 89 36 001A R  MOV     #BUFFER_TAIL,SI
1059 05DC 89 36 001C R  MOV     #BUFFER_START,SI
1060 05E0 89 36 00B0 R  ADD     SI,32          ;DEFAULT BUFFER OF 32 BYTES
1061 05E4 83 C6 20      ADD     #BUFFER_END,SI
1062 05E7 89 36 0082 R  MOV     DS,OFFSET #PRT_TIM_OUT ;SET DEFAULT PRINTER TIMEOUT
1063 05E8 BF 0078 R      PUSH   DS
1064 05EE 1E            POP     ES
1065 05EF 07            MOV     AX,1414H      ; DEFAULT=20
1066 05F0 B8 1414      MOV     AX,0101H      ;RS232 DEFAULT=01
1067 05F3 AB            STOSW
1068 05F4 AB            STOSW
1069 05F5 B8 0101      MOV     AX,0101H
1070 05F8 AB            STOSW
1071 05F9 AB            STOSW
1072 05FA E4 21        IN     AL,INTA01
1073 05FC 24 FC        AND    AL,0FCH        ; ENABLE TIMER AND KB INTS
1074 05FE E6 21        OUT   INTA01,AL
1075
1076 0600 83 FD 00      CMP    BP,0000        ; CHECK FOR BP= NON ZERO
1077                ; (ERROR HAPPENED)
1078 0603 74 18          JE     F15A_0         ; CONTINUE IF NO ERROR
1079 0605 BA 0002      MOV     DX,2           ; 2 SHORT BEEPS (ERROR)
1080 0608 EB 19A5 R      CALL   ERR_BEEP
1081 060B BE 0769 R      MOV     SI,OFFSET F3D ; LOAD ERROR MSG
1082 060E EC 1997 R      CALL   F_MSG
1083 0611                ERR_WAIT:
1084 0611 B4 00          MOV     AH,00
1085 0613 CD 16          INT    16H            ; WAIT FOR 'F1' KEY
1086 0615 80 FC 3B      CMP    AH,3BH         ; BYPASS ERROR
1087 0618 75 F1          JNE   ERR_WAIT
1088 061A EB 0E 90      JMP    F15A_0
1089 061D                F15A_0:
1090 061D 80 3E 0012 R 01 ; MFC MODE
1091 0622 74 06          JE     F15A           ; BYPASS BEEP
1092 0624 BA 0001      MOV     DX,1          ; 1 SHORT BEEP (NO ERRORS)
1093 0627 EB 19A5 R      CALL   ERR_BEEP
1094 062A A0 0010 R      MOV     AL,BYTE PTR #EQUIP_FLAG ; GET SWITCHES
1095 062D 24 01        AND    AL,00000001B  ; 'LOOP POST' SWITCH ON
1096 062F 75 03          JNZ   F15B           ; CONTINUE WITH BRING-UP
1097 0631 E9 005B R      JMP    START
1098 0634 2A E4          MOV     SLB,AH,AH
1099 0636 A0 0049 R      MOV     AL,#CRT_MODE
1100 0639 CD 10          INT    10H           ; CLEAR SCREEN
1101 063B                F15B:
1102 063B BD 1970 R      MOV     BP,OFFSET F4
1103 063E BE 0000      MOV     SI,0
1104 0641                F16:   MOV     DX,CS:[BP]    ; PRT_BASE
1105 0641 2E1 8B 56 00    MOV     AL,0AAH       ; GET PRINTER BASE ADDR
1106 0645 B0 AA          OUT   DX,AL           ; WRITE DATA TO PORT A
1107 0647 EE            OUT   DX,AL
1108 0648 1E            PUSH  DS              ; BUS SETTLEING
1109 0649 EC            IN    AL,DX           ; READ PORT A
1110 064A 1F            POP   DS
1111 064B 3C AA          CMP    AL,0AAH       ; DATA PATTERN SAME
1112 064D 75 05          JNE   F17            ; NO - CHECK NEXT PRT CD
1113 064F 89 54 08      MOV     [PRINTER_BASE+DATA40][SI],DX ; STORE PRT BASE ADDR
1114 0652 46            INC   SI              ; INCREMENT TO NEXT WORD
1115 0653 46            INC   SI
1116 0654                F17:   INC   BP              ; POINT TO NEXT BASE ADDR
1117 0654 45            INC   BP
1118 0655 45            INC   BP
1119 0656 81 FD 1976 R  CMP    BP,OFFSET F4E ; ALL POSSIBLE ADDRS CHECKED?
1120 065A 75 05          JNE   F16            ; PRT BASE
1121 065C BB 0000      MOV     BX,0          ; POINTER TO RS232 TABLE
1122 065F BA 03FA      MOV     DX,3FAH       ; CHECK IF RS232 CD 1 ATTCH?
1123 0662 EC            IN    AL,DX           ; READ INTR ID REG
1124 0663 A8 F8          TEST  AL,0F8H
1125 0665 75 06          JNZ   F18            ; READ INTR ID REG
1126 0667 C7 07 03F8  MOV     [RS232_BASE+DATA40][BX],3F8H ; SETUP RS232 CD #1 ADDR
1127 066B 43            INC   BX
1128 066C 43            INC   BX
1129 066D                F18:   INC   BX
1130 066D BA 02FA      MOV     DX,2FAH       ; CHECK IF RS232 CD 2 ATTCH
1131 0670 EC            IN    AL,DX           ; READ INTERRUPT ID REG
1132 0671 A8 F8          TEST  AL,0F8H
1133 0673 75 06          JNZ   F19            ; BASE_END
1134 0675 C7 07 02F8  MOV     [RS232_BASE+DATA40][BX],2F8H ; SETUP RS232 CD #2
1135 0679 43            INC   BX

```

SECTION 5

```

1136 067A 43          INC    BX
1137
1138          ;----- SET UP EQUIP FLAG TO INDICATE NUMBER OF PRINTERS AND RS232 CARDS
1139
1140 067B          F19:          ; BASE END;
1141 067B BB C6      MOV    AX,51          ; S1 HAS 25 NUMBER OF RS232
1142 067D B1 03      MOV    CL,3          ; SHIFT COUNT
1143 067F D2 C8      ROR    AL,CL         ; ROTATE RIGHT 3 POSITIONS
1144 0681 DA C3      OR     AL,BL         ; OR IN THE PRINTER COUNT
1145 0683 A2 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 AB 0F      TEST   AL,0FH
1152 068F 75 05     JNZ   F20            ; NO_GAME_CARD
1153 0691 80 0E 0011 R 10  OR     BYTE PTR #EQUIP_FLAG+1,16 ; NO_GAME_CARD;
1154 0696
1155
1156          ;----- ENABLE NMI INTERRUPTS
1157
1158 0696 E4 61      IN     AL,PORT_B    ; RESET CHECK ENABLES
1159 0698 DC 30      OR     AL,30H
1160 069A E6 61      OUT    PORT_B,AL
1161 069C 24 CF      AND    PORT_B,AL
1162 069E E6 61      OUT    PORT_B,AL
1163 06A0 B0 80      MOV    AL,B0H      ; ENABLE NMI INTERRUPTS
1164 06A2 E6 A0      OUT    A0H,AL
1165 06A4
1166 06A4 CD 19      F21:          INT    19H         ; LOAD_BOOT_STRAP;
1167                                     ; GO TO THE BOOT LOADER
1168
1169          ;----- INT 19 -----
1170          ; BOOT STRAP LOADER
1171          ; TRACK 0, SECTOR 1 IS READ INTO THE
1172          ; BOOT LOCATION (SEGMENT 0, OFFSET 7C00)
1173          ; AND CONTROL IS TRANSFERRED THERE.
1174          ; IF THERE IS A HARDWARE ERROR CONTROL IS
1175          ; TRANSFERRED TO THE ROM BASIC ENTRY POINT.
1176          ;-----
1177          ASSUME  CS:CODE,DS:ABS0
1178          ORG    0E6F2H
1179 06F2          ORG    006F2H
1180
1181 06F2          BOOT_STRAP  PROC  NEAR
1182 06F2 FB        STI
1183 06F3 2B C0     SUB    AX,AX        ; ENABLE INTERRUPTS
1184 06F5 9E D8     MOV    DS,AX       ; ESTABLISH ADDRESSING
1185
1186          ;----- RESET THE DISK PARAMETER TABLE VECTOR
1187
1188 06F7 C7 06 0078 R OFC7 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:          ; IPL SYSTEM
1195 0704 51        PUSH   CX          ; SAVE RETRY COUNT
1196 0705 B4 00    MOV    AH,0       ; RESET THE DISKETTE SYSTEM
1197 0707 CD 13    INT    13H       ; DISKETTE_IO
1198 0709 72 0F    JRC    H2        ; IF ERROR, TRY AGAIN
1199 070B B8 0201  MOV    AX,201H   ; READ IN THE SINGLE SECTOR
1200 070E 2B D2    SUB    DX,DX     ; TO THE BOOT LOCATION
1201 0710 BE C2    MOV    ES,DX
1202 0712 BB 7C00 R  MOV    BX,OFFSET #BOOT_LOCN
1203
1204 0715 B9 0001   MOV    CX,1        ; DRIVE 0, HEAD 0
1205 0718 CD 13    INT    13H       ; SECTOR 1, TRACK 0
1206 071A          H2:          ; DISKETTE_IO
1207 071A 59        POP    CX         ; RECOVER RETRY COUNT
1208 071B 73 04    JNC   H4         ; CF SET BY UNSUCCESSFUL READ
1209 071D E2 E5    LOOP  H1        ; DO IT FOR RETRY TIMES
1210
1211          ;----- UNABLE TO IPL FROM THE DISKETTE
1212
1213 071F          H3:          INT    18H       ; GO TO RESIDENT BASIC
1214 071F CD 18
1215          ;----- IPL WAS SUCCESSFUL
1216
1217 0721          H4:          JMP    #BOOT_LOCN
1218 0721          BOOT_STRAP  ENDP
1219 0721 EA 7C00 ---- R
1220 0726
1221
1222          ;-- ORG 0E729H
1223 0729          ORG 00729H
1224 0729 0417     DW    0417
1225 072B 0300     DW    768
1226 072D 0180     DW    384
1227 072F 00C0     DW    192
1228 0731 0060     DW    96
1229 0733 0030     DW    48
1230 0735 0018     DW    24
1231 0737 000C     DW    12
1232
1233 0739          RS232_IO:
1234 0739 E9 0000 E  JMP    RS232_IO_1
1235
1236 073C          CONF_TBL:
1237 073C 0008     DW    CONF_E-CONF_TBL-2
1238 073E FB      DB    MODEL_BYTE
1239 073F 00      DB    SUB_MODEL_BYTE
1240 0740 01      DB    BIOS_LEVEL
1241 0741 50      DB    01010000B
1242
1243
1244
1245 0742 00      DB    0
1246 0743 00      DB    0
1247 0744 00      DB    0
1248 0745 00      DB    0
1249 = 0746      CONF_E  EQU    $
1250                                     ; RESERVED FOR EXPANSION
    
```

```

1250
1251
1252 ; PRINT ADDRESS AND ERROR MESSAGE FOR ROM CHECKSUM ERRORS
1253 -----
ROM_ERR PROC NEAR
1254 0746 PUSH DX ; SAVE POINTER
1255 0746 52 PUSH AX
1256 0747 50 MOV DX,DS ; GET ADDRESS POINTER
1257 0748 BC DA MOV ES:#MFG_ERR_FLAG,DH ;
1258 074A 26: 88 36 0015 R ES:←MFG_ERR_FLAG,DH ;
1259 ; ←←←CHECKPOINTS CO→F4←←←
1260 074F 81 FA C800 CMP DX,0C800H ; CRT CARD IN ERROR?
1261 0753 7C 0C JL ROM_ERR_BEEP ; GIVE CRT CARD FAIL BEEP
1262 0755 EB 0CBA R CALL PRT_SEG ; PRINT SEGEMENT IN ERROR
1263 075B BE 18D7 R MOV SI,OFFSET F3A ; DISPLAY ERROR MSG
1264 075B E8 1976 R CALL E_M5G
ROM_ERR ENDP
1265 075E POP AX
1266 075E 58 POP DX
1267 075F 5A RET
1268 0760 C3
ROM_ERR_BEEP:
1269 0761 MOV DX,0102H ; BEEP 1 LONG, 2 SHORT
1270 0761 BA 0102 CALL ERR_BEEP
1271 0764 E8 19A5 R JMP SHORT ROM_ERR_END
1272 0767 EB F5
1273 0769
ROM_ERR ENDP
1274
1275 0769 45 52 52 4F 52 2E F3D DB 'ERROR. (RESUME = *F1* KEY)',CR,LF ; ERROR PROMPT
1276 20 28 52 45 53 55
1277 4D 45 20 3D 20 22
1278 46 31 22 20 4B 45
1279 59 29 0D 0A
1280
1281 ; ORG 0E82EH
1282 082E ORG 0082EH
1283 082E KEYBOARD_IO:
1284 082E E9 0000 E JMP KEYBOARD_IO_1
1285
1286 ; ORG 0E987H
1287 0987 ORG 00987H
1288 0987 KB_INT:
1289 0987 E9 0000 E JMP KB_INT_1
1290
1291 098A 20 33 30 31 0D 0A F1 DB '301',CR,LF ; KEYBOARD ERROR
1292 0990 36 30 31 0D 0A F3 DB '601',CR,LF ; DISKETTE ERROR
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 ; 1 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_1 PROC FAR
1325 0995 TIME_OF_DAY_1:
1326 0995 FB STI ; INTERRUPTS BACK ON
1327 0996 80 FC 0C CMP AH,(RTC_TBE-RTC_TB)/2 ; CHECK IF COMMAND IN VALID RANGE
1328 0999 F5 CMC ; COMPLEMENT CARRY FOR ERROR EXIT
1329 099A 72 17 JC TIME_9 ; EXIT WITH CARRY = 1 IF NOT VALID
1330
1331 099C 1E PUSH DS ; SAVE USERS (DS) SEGMENT
1332 099D E8 1A12 R CALL D05 ; GET DATA SEGMENT SELECTOR
1333 09A0 54 PUSH SI ; SAVE WORK REGISTER
1334 09A1 8A C4 MOV AL,AH ; MOVE FUNCTION TO (AL) REGISTER
1335 09A3 98 CBW ; CONVERT FUNCTION TO BYTE OFFSET
1336 09A4 03 C0 ADD AX,AX ; CONVERT FUNCTION TO WORD OFFSET (CY=0)
1337 09A6 8B F0 MOV SI,AX ; PLACE INTO ADDRESSING REGISTER
1338 09A8 FA CLI ; NO INTERRUPTS DURING TIME FUNCTIONS
1339 09A9 2E: 1F FF 94 09B6 R CALL CS:[SI]+OFFSET RTC_TB ; RETURN WITH CARRY FLAG SET FOR RESULT
1340
1341 09AE FB STI ; INTERRUPTS BACK ON
1342 09AF B4 00 MOV AH,0 ; CLEAR (AH) TO ZERO
1343 09B1 5E POP SI ; RECOVER USERS REGISTER
1344 09B2 1F POP DS ; RECOVER USERS SEGMENT SELECTOR
1345 09B3 RET ; RETURN WITH CY= 0 IF NO ERROR
1346 09B3 CA 0002 TIME_9:
1347
1348 09B6 09CE R RTC_TB DW RTC_00 ; ROUTINE VECTOR TABLE (AH)=
1349 09B8 09DF R DW RTC_I0 ; 00H = READ CURRENT CLOCK COUNT
1350 09BA 09ED R DW RTC_N5 ; 01H = SET CLOCK COUNT
1351 09BC 09ED R DW RTC_NS ; 02H INVALID
1352 09BE 09ED R DW RTC_NS ; 03H INVALID
1353 09C0 09ED R DW RTC_NS ; 04H INVALID
1354 09C2 09ED R DW RTC_NS ; 05H INVALID
1355 09C4 09ED R DW RTC_NS ; 06H INVALID
1356 09C6 09ED R DW RTC_NS ; 07H INVALID
1357 09C8 09ED R DW RTC_NS ; 08H INVALID
1358 09CA 09EF R DW RTC_AD ; 09H INVALID
1359 09CC 09F4 R DW RTC_AD ; 0AH = READ SYSTEM DAY COUNTER
1360 = 09CE DW RTC_B0 ; 0BH = WRITE SYSTEM DAY COUNTER
1361 RTC_TBE EQU $
1362 09CE TIME_OF_DAY_1 ENDP
1363
    
```

SECTION 5

```

1364 09CE          RTC_00 PROC    NEAR          ; READ TIME COUNT
1365 09CE A0 0070 R      MOV      AL,®TIMER_OFL ; GET THE OVERFLOW FLAG
1366 09D1 C6 06 0070 R 00 MOV      ®TIMER_OFL,0 ; AND THEN RESET THE OVERFLOW FLAG
1367 09D6 B8 0E 006E R 00 MOV      CX,®TIMER_HIGH ; GET COUNT OF TIME HIGH WORD
1368 09DA BB 16 006C R      MOV      DX,®TIMER_LOW  ; GET COUNT OF TIME LOW WORD
1369 09DE C3            RET                          ; RETURN WITH NO CARRY
1370
1371 09DF          RTC_10:      ; SET TIME COUNT
1372 09DF B9 16 006C R      MOV      ®TIMER_LOW,DX ; SET TIME COUNT LOW WORD
1373 09E3 B9 0E 006E R 00 MOV      ®TIMER_HIGH,CX ; SET THE TIME COUNT HIGH WORD
1374 09E7 C6 06 0070 R 00 MOV      ®TIMER_OFL,0  ; RESET OVERFLOW FLAG
1375 09EC C3            RET                          ; RETURN WITH NO CARRY
1376
1377 09ED          RTC_NS:      ; INVALID FUNCTION (NOT SUPPORTED)
1378 09ED F9            STC                          ; SET CARRY FLAG FOR ERROR (CY=1)
1379 09EE C3            RET                          ; EXIT THROUGH COMMON RETURN
1380
1381 09EF          RTC_A0:      ; READ SYSTEM DAY COUNTER
1382 09EF BB 0E 00CE R      MOV      CX,®DAY_COUNT ; GET COUNT OF DAYS
1383 09F3 C3            RET                          ; EXIT THROUGH COMMON RETURN WITH CY=0
1384
1385 09F4          RTC_B0:      ; SET SYSTEM DAY COUNTER
1386 09F4 B9 0E 00CE R      MOV      ®DAY_COUNT,CX ; SET COUNT OF DAYS
1387 09F8 C3            RET                          ; EXIT THROUGH COMMON RETURN WITH CY=0
1388
1389 09F9          RTC_00 ENDP
1390
1391          ; ORG 00C59H
1392 0C59          ORG 00C59H
1393 0C59 E9 0000 E      DISKETTE_IO: JMP    DISKETTE_IO_1
1394
1395          ;--- BEEP
1396          ;-----
1397          ; ENTRY:
1398          ; (BL) = DURATION COUNTER ( 1 FOR 1/64 SECOND )
1399          ; (CX) = FREQUENCY DIVISOR (1193180/FREQUENCY) (1331 FOR 886 HZ)
1400          ; EXIT:
1401          ; (AX),(BL),(CX) MODIFIED.
1402          ;-----
1403
1404 0C5C          BEEP PROC    NEAR          ; SETUP TIMER 2
1405 0C5C 9C          PUSHF      ; SAVE INTERRUPT STATUS
1406 0C5D FA          CLI          ; BLOCK INTERRUPTS DURING UPDATE
1407 0C5E B0 B6      MOV      AL,10110110B ; SELECT TIMER 2,LSB,MSB,BINARY
1408 0C60 E6 43      OUT      TIMER+3,AL ; WRITE THE TIMER MODE REGISTER
1409 0C62 90          NOP          ; I/O DELAY
1410 0C63 8A C1      MOV      AL,CL ; DIVISOR FOR HZ (LOW)
1411 0C65 E6 42      OUT      TIMER+2,AL ; WRITE TIMER 2 COUNT - LSB
1412 0C67 90          NOP          ; I/O DELAY
1413 0C68 8A C5      MOV      AL,CH ; DIVISOR FOR HZ (HIGH)
1414 0C6A E6 42      OUT      TIMER+2,AL ; WRITE TIMER 2 COUNT - MSB
1415 0C6C E4 61      IN      AL,PORT_B ; GET CURRENT SETTING OF PORT
1416 0C6E 8A E0      MOV      AH,AL ; SAVE THAT SETTING
1417 0C70 0C 03      OR      AL,GATE2+SPK2 ; GATE TIMER 2 AND TURN SPEAKER ON
1418 0C72 E6 61      OUT      PORT_B,AL ; AND RESTORE INTERRUPT STATUS
1419 0C74 9D          POPF
1420 0C75
1421 0C75 B9 040B      GT:      MOV      CX,1035 ; 1/64 SECOND PER COUNT (BL)
1422 0C78 B8 0CA0 R ; DELAY COUNT FOR 1/64 OF A SECOND
1423 0C7B FE CB      CALL    WAITF ; GO TO BEEP DELAY 1/64 COUNT
1424 0C7D 75 F6      DEC     BL ; (BL) LENGTH COUNT EXPIRED?
1425          JNZ     GT ; NO - CONTINUE BEEPING SPEAKER
1426 0C7F 9C          ; SAVE INTERRUPT STATUS
1427 0C80 FA          ; BLOCK INTERRUPTS DURING UPDATE
1428 0C81 E4 61      IN      AL,PORT_B ; GET CURRENT PORT VALUE
1429 0C83 0C FC      OR      AL,NOT (GATE2+SPK2) ; ILOCK CURRENT SPEAKER BITS IN CASE
1430 0C85 22 E0      AND     AH,AL ; SOMEONE TURNED THEM OFF DURING BEEP
1431 0C87 8A C4      MOV      AL,AH ; RECOVER VALUE OF PORT
1432 0C89 24 FC      AND     AL,NOT (GATE2+SPK2) ; FORCE SPEAKER DATA OFF
1433 0C8B E6 61      OUT      PORT_B,AL ; AND STOP SPEAKER TIMER
1434 0C8D 9D          ; RESTORE INTERRUPT FLAG STATE
1435 0C8E B9 040B      MOV      CX,1035 ; FORCE 1/64 SECOND DELAY (SHORT)
1436 0C91 E8 0CA0 R ; MINIMUM DELAY BETWEEN ALL BEEPS
1437 0C94 9C          ; SAVE INTERRUPT STATUS
1438 0C95 FA          ; BLOCK INTERRUPTS DURING UPDATE
1439 0C96 E4 61      IN      AL,PORT_B ; GET CURRENT PORT VALUE IN CASE
1440 0C98 24 03      AND     AL,GATE2+SPK2 ; SOMEONE TURNED THEM ON
1441 0C9A 0A C4      OR      AL,AH ; RECOVER VALUE OF PORT B
1442 0C9C E6 61      OUT      PORT_B,AL ; RESTORE SPEAKER STATUS
1443 0C9E 9D          ; RESTORE INTERRUPT FLAG STATE
1444 0C9F C3            RET
1445
1446 0CA0          BEEP ENDP
1447
1448          ;--- WAITF
1449          ;-----
1450          ; FIXED TIME WAIT ROUTINE (HARDWARE CONTROLLED - NOT PROCESSOR)
1451          ; ENTRY:
1452          ; (CX) = COUNT OF 15.085737 MICROSECOND INTERVALS TO WAIT
1453          ; MEMORY REFRESH TIMER I OUTPUT AT THE DMA CHANNEL 0
1454          ; ADDRESS REGISTER USED AS REFERENCE.
1455          ; EXIT:
1456          ; AFTER (CX) TIME COUNT (PLUS OR MINUS 31 MICROSECONDS)
1457          ; (CX) = 0
1458          ;-----
1459
1460 0CA0          WAITF PROC NEAR          ; DELAY FOR (CX)*15.085737 US
1461 0CA0 50          PUSH     SHR ; SAVE WORK REGISTER (AH)
1462 0CA1 D1 E9      SHR     CX,1 ; DIVIDE 15us COUNT DOWN TO 30us COUNT
1463 0CA3 E3 13      JXCX   WAITF9 ; EXIT IF COUNT WAS ZERO OR ONE
1464
1465 0CA5 E6 0C      OUT     DMA+12,AL ; CLEAR THE DMA BYTE POINTER FLIP/FLOP
1466 0CA7
1467 0CA7 C3      WAITF1: PUSHF ; SAVE INTERRUPT STATE
1468 0CA8 FA          CLI ; BLOCK INTERRUPTS TILL NEXT CHANGE
1469 0CA9          WAITF3: ; WAIT FOR REFRESH ADDRESS CHANGE
1470 0CA9 E4 00      IN      AL,DMA ; READ CURRENT ADDRESS LOW BYTE
1471 0CAB 24 FE      AND     AL,11111110B ; DISCARD LOW BIT (30us)
1472 0CAD 3A E0      CMP     AH,AL ; DID VALUE JUST CHANGE
1473 0CAF BA E0      MOV     AH,AL ; SAVE NEW/OLD VALUE INCASE IT DID
1474 0CB1 E4 00      IN      AL,DMA ; READ HIGH BYTE (AND IGNORE)
1475 0CB3 74 F4      JE      WAITF3 ; WAIT FOR A CHANGE IN ADDRESS BITS
1476
1477 0CB5 9D          POPF ; RESTORE INTERRUPTS
    
```

```

1478 OCB6 E2 EF          LOOP      WAITF1          ; DECREMENT CYCLES COUNT TILL COUNT END
1479 OCB8                WAITF9:      POP      AX          ; RESTORE (AH)
1480 OCB8 58            RET                ; RETURN (CX)= 0
1481 OCB9 C3
1482
1483 OCB8                WAITF  ENDP
1484
-----
1485 ;
1486 ; PRINT A SEGMENT VALUE TO LOOK LIKE A 20 BIT ADDRESS
1487 ; DX MUST CONTAIN SEGMENT VALUE TO BE PRINTED
1488 ;
-----
1489 OCB8                PRG_SEG PROC      NEAR
1490 OCB8 BA C6          MOV      AL,DX          ;GET MSB
1491 OCB8 E8 1958 R     CALL    XPC_BYTE
1492 OCBF 8A C2          MOV      AL,DL          ;LSB
1493 OCC1 E8 1958 R     CALL    XPC_BYTE
1494 OCC4 80 30          MOV      AL,'0'        ; PRINT A '0 '
1495 OCC6 E8 1969 R     CALL    PRT_HEX
1496 OCC9 80 20          MOV      AL,' '        ;SPACE
1497 OCCB E8 1969 R     CALL    PRT_HEX
1498 OCC8 C3            RET
1499 OCCF                PRG_SEG ENDP
1500
-----
1501 ; THIS SUBROUTINE PERFORMS A READ/WRITE STORAGE TEST ON A BLOCK
1502 ; OF STORAGE.
1503 ; ENTRY REQUIREMENTS:
1504 ; ES = ADDRESS OF STORAGE SEGMENT BEING TESTED
1505 ; DS = ADDRESS OF STORAGE SEGMENT
1506 ; CX = WORD COUNT OF STORAGE BLOCK TO BE TESTED
1507 ; EXIT PARAMETERS:
1508 ; ZERO FLAG = 0 IF STORAGE ERROR (DATA COMPARE OR PARITY)
1509 ; CHECK. AL=0 DENOTES A PARITY CHECK. ELSE AL=XOR'ED
1510 ; BIT PATTERN OF THE EXPECTED DATA PATTERN VS THE ACTUAL
1511 ; DATA READ.
1512 ; AX,BX,CX,DX,DI, AND SI ARE ALL DESTROYED.
1513 ;
1514 ;
1515 ;
1516 OCCF                STGTST_CNT PROC      NEAR
1517 OCFB BB D9          MOV      BX,CX
1518 OCD1 FC            CLD
1519 OCD2 2B FF          SUB      DI,DI
1520 OCD4 2B C0          SUB      AX,AX
1521 OCB6                C2_1:
1522 OCB6 86 05          MOV     [DI],AL        ; ON FIRST BYTE
1523 OCD8 8A 05          MOV     AL,[DI]
1524 OCB8 32 C4          XOR     AL,AH
1525 OCD8 75 F2          JNZ    CT             ; O.K.?
1526 OCDE FE C4          INC     AH            ; GO ERROR IF NOT
1527 OCED 8A C4          MOV     AL,AH
1528 OCCE 75 F2          JNZ    C2_1
1529 OCE4 8B 55AA       MOV     AX,'055AAH'   ; LOOP TILL WRAP THROUGH FF
1530 OCE7 8B D0          MOV     DX,AX         ; SET INITIAL COMPARE PATTERN.
1531 OCE9 F3 AB          REP     STOSW         ; FILL STORAGE LOCATIONS IN BLOCK
1532 OCEB E4 61         IN     AL,PORT_B     ;
1533 OCED 0C 30         OR     AL,'030H'     ; TOGGLE PARITY CHECK LATCHES
1534 OCEF E6 61         OUT    PORT_B,AL
1535 OCF1 90            NOP
1536 OCF2 24 CF        AND    AL,OCFH
1537 OCF4 E6 61         OUT    PORT_B,AL
1538
1539 OCF6 4F            I      DEC     DI          ; POINT TO LAST WORD JUST WRITTEN
1540 OCF7 4F            DEC     DI
1541 OCF8 FD            STD
1542 OCF9 8B F7         MOV     SI,DI         ; SET DIR FLAG TO GO BACKWARDS
1543 OCFB 8B CB         MOV     CX,BX        ; INITIALIZE DESTINATION POINTER
1544 OCFD                C3:
1545 OCFD AD            LODSW
1546 OCFE 33 C2         XOR     AX,DX         ; SETUP WORD COUNT FOR LOOP
1547 OCD0 75 57         JNE    CTX           ; INNER TEST LOOP
1548 ODD2 8B AA55       MOV     AX,'0AA55H'  ; READ OLD TEST WORD FROM STORAGE
1549 ODD5 AB            STOSW                ; DATA READ AS EXPECTED ?
1550 ODD6 E2 F5         LOOP   C3            ; NO - GO TO ERROR ROUTINE
1551 ;
1552 ODD8 FC            I      CLD                ; GET NEXT DATA PATTERN TO WRITE
1553 ODD9 47            INC     DI            ; WRITE INTO LOCATION JUST READ
1554 ODDA 47            INC     DI            ; DECREMENT WORD COUNT AND LOOP
1555 ODD8 8B F7         MOV     SI,DI         ; SET DIR FLAG TO GO FORWARD
1556 ODD9 8B CB         MOV     CX,BX        ; SET POINTER TO BEG LOCATION
1557 ODDF 8B D0         MOV     DX,AX        ; INITIALIZE DESTINATION POINTER
1558 ODD1                C4:
1559 ODD1 AD            LODSW                ; SETUP COMPARE PATTERN OF "0AA55H".
1560 ODD2 33 C2         XOR     AX,DX         ; INNER TEST LOOP
1561 ODD4 75 43         JNE    CTX           ; READ OLD TEST WORD FROM STORAGE
1562 ODD6 8B FF        MOV     AX,'0FFFFH'  ; DATA READ AS EXPECTED ?
1563 ODD9 AB            STOSW                ; NO - GO TO ERROR ROUTINE
1564 ODDA E2 F5         LOOP   C4            ; GET NEXT DATA PATTERN TO WRITE
1565 ;
1566 ODDC 4F            I      DEC     DI          ; WRITE INTO LOCATION JUST READ
1567 ODDD 4F            DEC     DI            ; DECREMENT WORD COUNT AND LOOP
1568 ODD8 FD            STD
1569 ODDF 8B F7         MOV     SI,DI         ; POINT TO LAST WORD JUST WRITTEN
1570 ODD1 8B CB         MOV     CX,BX        ; SET DIR FLAG TO GO BACKWARDS
1571 ODD2 8B D0         MOV     DX,AX        ; INITIALIZE DESTINATION POINTER
1572 ODD5                C5:
1573 ODD5 AD            LODSW                ; SETUP WORD COUNT FOR LOOP
1574 ODD6 33 C2         XOR     AX,DX         ; SETUP COMPARE PATTERN "0FFFFH".
1575 ODD8 75 2F         JNE    CTX           ; INNER TEST LOOP
1576 ODDA 8B 0101      MOV     AX,'00101H'  ; READ OLD TEST WORD FROM STORAGE
1577 ODD2 AB            STOSW                ; DATA READ AS EXPECTED ?
1578 ODD5 E2 F5         LOOP   C5            ; NO - GO TO ERROR ROUTINE
1579 ;
1580 ODD3 FC            I      CLD                ; GET NEXT DATA PATTERN TO WRITE
1581 ODD3 47            INC     DI            ; WRITE INTO LOCATION JUST READ
1582 ODD3 47            INC     DI            ; DECREMENT WORD COUNT AND LOOP
1583 ODD3 8B F7         MOV     SI,DI         ; SET DIR FLAG TO GO FORWARD
1584 ODD5 8B CB         MOV     CX,BX        ; SET POINTER TO BEG LOCATION
1585 ODD7 8B D0         MOV     DX,AX        ; INITIALIZE DESTINATION POINTER
1586 ODD9                C6:
1587 ODD9 AD            LODSW                ; SETUP WORD COUNT FOR LOOP
1588 ODDA 33 C2         XOR     AX,DX         ; SETUP COMPARE PATTERN "00101H".
1589 ODDC 75 1B         JNE    CTX           ; INNER TEST LOOP
1590 ODD3 AB            STOSW                ; READ OLD TEST WORD FROM STORAGE
1591 ODD5 E2 F8         LOOP   C6            ; DATA READ AS EXPECTED ?
1592 ;
1593 ODD3 8B F7         MOV     SI,DI         ; NO - GO TO ERROR ROUTINE
1594 ODD5 8B CB         MOV     CX,BX        ; GET NEXT DATA PATTERN TO WRITE
1595 ODD7 8B D0         MOV     DX,AX        ; WRITE ZERO INTO LOCATION READ
1596 ODD9                C6:
1597 ODD9 AD            LODSW                ; DECREMENT WORD COUNT AND LOOP
1598 ODDA 33 C2         XOR     AX,DX
1599 ODDC 75 1B         JNE    CTX
1600 ODD3 AB            STOSW
1601 ODD5 E2 F8         LOOP   C6
    
```

SECTION 5

```

1592          ;
1593 0D41 4F          DEC     DI          ; POINT TO LAST WORD JUST WRITTEN
1594 0D42 4F          DEC     DI
1595 0D43 FD          STD
1596 0D44 BB F7      MOV     SI,DI       ; SET DIR FLAG TO GO BACKWARDS
1597 0D46 BB CB      MOV     CX,BX       ; INITIALIZE DESTINATION POINTER
1598 0D48 BB D0      MOV     DX,AX       ; SETUP WORD COUNT FOR LOOP
1599 0D4A             MOV     DX,AX       ; SETUP COMPARE PATTERN "00000H"
1600 0D4A AD          C6X:  LODSW
1601 0D4B 33 C2      XOR     AX,DX       ; VERIFY MEMORY IS ZERO.
1602 0D4D 75 0A      JNE    C7X         ; DATA READ AS EXPECTED ?
1603 0D4F E2 F0      LOOP  C6X         ; NO - GO TO ERROR ROUTINE
1604             ; DECREMENT WORD COUNT AND LOOP
1605 0D51 E4 62      IN     AL,PORT_C   ; DID A PARITY ERROR OCCUR ?
1606 0D53 24 C0      AND    AL,0C0H     ; ZERO FLAG WILL BE OFF, IF PARITY ERROR
1607 0D55 B0 D0      MOV    AL,0        ; AL=0 DATA COMPARE OK
1608 0D57
1609 0D57 FC          C7:   CLD             ; SET DIRECTION FLAG TO INC
1610 0D58 C3          RET
1611 0D59
1612 0D59 3C 00      C7X:  CMP     AL,0       ; FIND BYTE THAT FAILED.
1613 0D5B 75 FA      JNZ    C7
1614 0D5D BA C4      MOV    AL,AH
1615 0D5F EB F6      JMP    SHORT C7
1616 0D61
1617             STGTST_CNT      ENDP
1618
1619 0F57             ; ORG     0EF57H
1620 0F57 E9 0000 E   DISK_INT:  JMP     DISK_INT_1
1621
1622             ; ORG     0EF79H
1623 0F79             ORG     0EF79H
1624
1625 -----
1626             MEDIA/DRIVE PARAMETER TABLES
1627 -----
1628             ;
1629             ; 40 TRACK LOW DATA RATE MEDIA IN 40 TRACK LOW DATA RATE DRIVE :
1630 -----
1631 0F79             MD_TBL1 LABEL BYTE
1632 0F7A DF          DB     11011111B   ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
1633 0F7B 25          DB     2           ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
1634 0F7C 02          DB     MOTOR_WAIT  ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
1635 0F7D 09          DB     2           ; 512 BYTES/SECTOR
1636 0F7E 2A          DB     09          ; EOT ( LAST SECTOR ON TRACK)
1637 0F7F FF          DB     02AH       ; GAP LENGTH
1638 0F80 50          DB     0FFH       ; DTL
1639 0F81 F6          DB     050H       ; GAP LENGTH FOR FORMAT
1640 0F82 0F          DB     0F6H       ; FILL BYTE FOR FORMAT
1641 0F83 08          DB     15          ; HEAD SETTLE TIME (MILLISECONDS)
1642 0F84 27          DB     8           ; MOTOR START TIME (1/8 SECONDS)
1643 0F85 80          DB     39          ; MAX. TRACK NUMBER
1644             DB     RATE_250 ; DATA TRANSFER RATE
1645 -----
1646             ; 40 TRACK LOW DATA RATE MEDIA IN 80 TRACK HI DATA RATE DRIVE :
1647 -----
1648 0F86             MD_TBL2 LABEL BYTE
1649 0F87 02          DB     11011111B   ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
1650 0F88 25          DB     2           ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
1651 0F89 02          DB     MOTOR_WAIT  ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
1652 0F8A 09          DB     2           ; 512 BYTES/SECTOR
1653 0F8B 2A          DB     09          ; EOT ( LAST SECTOR ON TRACK)
1654 0F8C FF          DB     02AH       ; GAP LENGTH
1655 0F8D 50          DB     0FFH       ; DTL
1656 0F8E F6          DB     054H       ; GAP LENGTH FOR FORMAT
1657 0F8F 0F          DB     0F6H       ; FILL BYTE FOR FORMAT
1658 0F90 08          DB     15          ; HEAD SETTLE TIME (MILLISECONDS)
1659 0F91 27          DB     8           ; MOTOR START TIME (1/8 SECONDS)
1660 0F92 40          DB     39          ; MAX. TRACK NUMBER
1661             DB     RATE_300 ; DATA TRANSFER RATE
1662 -----
1663             ; 80 TRACK HI DATA RATE MEDIA IN 80 TRACK HI DATA RATE DRIVE :
1664 -----
1665 0F93             MD_TBL3 LABEL BYTE
1666 0F94 DF          DB     11011111B   ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
1667 0F95 25          DB     2           ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
1668 0F96 02          DB     MOTOR_WAIT  ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
1669 0F97 0F          DB     2           ; 512 BYTES/SECTOR
1670 0F98 1B          DB     15          ; EOT ( LAST SECTOR ON TRACK)
1671 0F99 FF          DB     01BH       ; GAP LENGTH
1672 0F9A 54          DB     0FFH       ; DTL
1673 0F9B F6          DB     054H       ; GAP LENGTH FOR FORMAT
1674 0F9C 0F          DB     0F6H       ; FILL BYTE FOR FORMAT
1675 0F9D 08          DB     15          ; HEAD SETTLE TIME (MILLISECONDS)
1676 0F9E 4F          DB     8           ; MOTOR START TIME (1/8 SECONDS)
1677 0F9F 00          DB     79          ; MAX. TRACK NUMBER
1678             DB     RATE_500 ; DATA TRANSFER RATE
1679 -----
1680             ; 80 TRACK LOW DATA RATE MEDIA IN 80 TRACK LOW DATA RATE DRIVE :
1681 -----
1682 0FA0             MD_TBL4 LABEL BYTE
1683 0FA1 DF          DB     11011111B   ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
1684 0FA2 25          DB     2           ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
1685 0FA3 02          DB     MOTOR_WAIT  ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
1686 0FA4 09          DB     2           ; 512 BYTES/SECTOR
1687 0FA5 2A          DB     09          ; EOT ( LAST SECTOR ON TRACK)
1688 0FA6 FF          DB     02AH       ; GAP LENGTH
1689 0FA7 50          DB     0FFH       ; DTL
1690 0FA8 F6          DB     050H       ; GAP LENGTH FOR FORMAT
1691 0FA9 0F          DB     0F6H       ; FILL BYTE FOR FORMAT
1692 0FAB 08          DB     15          ; HEAD SETTLE TIME (MILLISECONDS)
1693 0FAC 4F          DB     8           ; MOTOR START TIME (1/8 SECONDS)
1694 0FAC 80          DB     79          ; MAX. TRACK NUMBER
1695             DB     RATE_250 ; DATA TRANSFER RATE
1696 -----
1697             ; 80 TRACK LOW DATA RATE MEDIA IN 80 TRACK HI DATA RATE DRIVE :
1698 -----
1699 0FAD             MD_TBL5 LABEL BYTE
1700 0FAE DF          DB     11011111B   ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
1701 0FAF 25          DB     2           ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
1702 0FB0 02          DB     MOTOR_WAIT  ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
1703 0FB1 09          DB     2           ; 512 BYTES/SECTOR
1704 0FB2 2A          DB     09          ; EOT ( LAST SECTOR ON TRACK)
1705 0FB3 FF          DB     02AH       ; GAP LENGTH
1706             DB     0FFH   ; DTL
    
```

```

1706 0FB4 50          DB      050H          ; GAP LENGTH FOR FORMAT
1707 0FB5 F6          DB      0F6H          ; FILL BYTE FOR FORMAT
1708 0FB6 0F          DB      15           ; HEAD SETTLE TIME (MILLISECONDS)
1709 0FB7 08          DB      8            ; MOTOR START TIME (1/8 SECONDS)
1710 0FB8 4F          DB      79           ; MAX. TRACK NUMBER
1711 0FB9 80          DB      RATE_250     ; DATA TRANSFER RATE
1712
1713                ; -----
1713                ; 80 TRACK HI DATA RATE MEDIA IN 80 TRACK HI DATA RATE DRIVE ; 1
1714                ; -----
1715 0FBA          MD_TBL6 LABEL BYTE
1716 0FBA AF          DB      10101111B    ; SRT=A, HD UNLOAD=0F - 1ST SPECIFY BYTE
1717 0FBB 02          DB      2            ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
1718 0FBC 25          DB      MOTOR_WAIT   ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
1719 0FBD 02          DB      2            ; 512 BYTES/SECTOR
1720 0FBE 12          DB      18           ; EOT (LAST SECTOR ON TRACK)
1721 0FBF 18          DB      01BH        ; GAP LENGTH
1722 0FC0 FF          DB      0FFH        ; DTL
1723 0FC1 6C          DB      06CH        ; GAP LENGTH FOR FORMAT
1724 0FC2 F6          DB      0F6H        ; FILL BYTE FOR FORMAT
1725 0FC3 0F          DB      8            ; 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                ; -----
1730                ; DISK BASE ;
1731                ; THIS IS THE SET OF PARAMETERS REQUIRED FOR DISKETTE OPERATION. ;
1732                ; THEY ARE POINTED AT BY THE DATA VARIABLE DISK_POINTER, TO ;
1733                ; MODIFY THE PARAMETERS, BUILD ANOTHER PARAMETER BLOCK AND POINT ;
1734                ; DISK_POINTER TO IT. ;
1735                ; -----
1736                ;
1737 0FC7          ORG      0EFC7H
1738 0FC7          DISK_BASE LABEL BYTE
1739 0FC7 CF          DB      10011111B    ; SRT=C, HD UNLOAD=0F - 1ST SPECIFY BYTE
1740 0FC8 02          DB      2            ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
1741 0FC9 25          DB      MOTOR_WAIT   ; WAIT AFTER OPN TIL MOTOR OFF
1742 0FCA 02          DB      2            ; 512 BYTES/SECTOR
1743 0FCB 08          DB      8            ; EOT (LAST SECTOR ON TRACK)
1744 0FCC 2A          DB      02AH        ; GAP LENGTH
1745 0FCD FF          DB      0FFH        ; DTL
1746 0FCE 50          DB      050H        ; GAP LENGTH FOR FORMAT
1747 0FCF F6          DB      0F6H        ; FILL BYTE FOR FORMAT
1748 0FD0 19          DB      25           ; HEAD SETTLE TIME (MILLISECONDS)
1749 0FD1 04          DB      4            ; MOTOR START TIME (1/8 SECONDS)
1750
1751                ;
1752 0FD2          ORG      0EFD2H
1753 0FD2          PRINTER_IO; ORG      00FD2H
1754 0FD2 E9 0000 E   JMP      PRINTER_IO_1
1755
1756                ;
1757 1045          ORG      0F045H
1758 1045 0000 E   M1      DW      OFFSET SET_MODE ; TABLE OF ROUTINES WITHIN VIDEO I/O
1759 1047 0000 E   DW      OFFSET SET_CTYPE
1760 1049 0000 E   DW      OFFSET SET_CPOS
1761 104B 0000 E   DW      OFFSET READ_CURSOR
1762 104D 0000 E   DW      OFFSET READ_LEN
1763 104F 0000 E   DW      OFFSET ACT_DISP_PAGE
1764 1051 0000 E   DW      OFFSET SCROLL_UP
1765 1053 0000 E   DW      OFFSET SCROLL_DOWN
1766 1055 0000 E   DW      OFFSET READ_AC_CURRENT
1767 1057 0000 E   DW      OFFSET WRITE_AC_CURRENT
1768 1059 0000 E   DW      OFFSET WRITE_C_CURRENT
1769 105B 0000 E   DW      OFFSET SET_CLR
1770 105D 0000 E   DW      OFFSET WRITE_DOT
1771 105F 0000 E   DW      OFFSET READ_DOT
1772 1061 0000 E   DW      OFFSET WRITE_TTY
1773 1063 0000 E   DW      OFFSET VIDEO_STATE
1774 = 0020       MIL      EQU      $-M1
1775
1776                ;
1777 1065          ORG      0F065H
1778 1065          VIDEO_IO; ORG      01065H
1779 1065 E9 0000 E   JMP      VIDEO_IO_1
1780
1781                ; -----
1781                ; VIDEO PARAMETERS --- INIT_TABLE
1782                ; -----
1783                ;
1784 10A4          ORG      0F0A4H
1785                ORG      010A4H
1786 10A4          VIDEO_PARAMS LABEL BYTE
1787 10A4 38 28 2D 0A 1F 06 DB      38H,28H,2DH,0AH,1FH,6,19H ; SET UP FOR 40X25
1788 19 64
1789 10AB 1C 02 07 06 07 DB      1CH,2,7,6,7
1790 10B0 00 00 00 00 00 DB      0,0,0,0
1791 0010 = 0010       M4      EQU      $-VIDEO_PARAMS
1792
1793 10B4 71 50 5A 0A 1F 06 DB      71H,50H,5AH,0AH,1FH,6,19H ; SET UP FOR 80X25
1794 19 64
1795 10BB 1C 02 07 06 07 DB      1CH,2,7,6,7
1796 10C0 00 00 00 00 00 DB      0,0,0,0
1797
1798 10C4 38 28 2D 0A 7F 06 DB      38H,28H,2DH,0AH,7FH,6,64H ; SET UP FOR GRAPHICS
1799 64
1800 10CB 70 02 01 06 07 DB      70H,2,1,6,7
1801 10D0 00 00 00 00 00 DB      0,0,0,0
1802
1803 10D4 61 50 52 0F 19 06 DB      61H,50H,52H,0FH,19H,6,19H ; SET UP FOR 80X25 B&W CARD
1804 19 64
1805 10DB 19 02 0D 0B 0C DB      19H,2,0DH,0BH,0CH
1806 10E0 00 00 00 00 00 DB      0,0,0,0
1807
1808 10E4 0800       M5      DW      2048 ; TABLE OF REGEN LENGTHS
1809 10E6 1000       DW      4096 ; 40X25
1810 10E8 4000       DW      16384 ; 80X25
1811 10EA 4000       DW      16384 ; GRAPHICS
1812
1813                ; -----
1813                ; COLUMNS
1814 10EC 28 28 50 50 28 28 M6      DB      40,40,80,80,40,40,80,80
1815 50 50
1816                ; -----
1816                ; C_REG_TAB
1817 10F4 2C 28 2D 29 2A 2E M7      DB      2CH,28H,2DH,29H,2AH,2EH,1EH,29H ; TABLE OF MODE SETS
1818 1E 29

```

```

1819 PAGE
1820 ----- INT 12 -----
1821 | MEMORY SIZE DET
1822 | THIS ROUTINE DETERMINES THE AMOUNT OF MEMORY IN THE SYSTEM
1823 | AS REPRESENTED BY THE SWITCHES ON THE PLANAR. NOTE THAT THE
1824 | SYSTEM MAY NOT BE ABLE TO USE I/O MEMORY UNLESS THERE IS A FULL
1825 | COMPLEMENT OF 64K BYTES ON THE PLANAR.
1826 | INPUT
1827 | NO REGISTERS
1828 | THE MEMORY SIZE VARIABLE IS SET DURING POWER ON DIAGNOSTICS
1829 | ACCORDING TO THE FOLLOWING HARDWARE ASSUMPTIONS:
1830 | PORT 60 BITS 3,2 = 00 - 256K BASE RAM
1831 | 01 - 512K BASE RAM
1832 | 10 - 576K BASE RAM
1833 | 11 - 640K BASE RAM
1834 | PORT 62 BITS 3-0 INDICATE AMOUNT OF I/O RAM IN 32K INCREMENTS
1835 | E.G., 0000 - NO RAM IN I/O CHANNEL
1836 | 0010 - 64K RAM IN I/O CHANNEL, ETC.
1837 | OUTPUT
1838 | (AX) = NUMBER OF CONTIGUOUS 1K BLOCKS OF MEMORY
1839 -----
1840 | ASSUME CS:CODE,DS:DATA
1841 | ORG 0F841H
1842 1841 | ORG 01841H
1843 1841 |
1844 1841 FB | MEMORY_SIZE_DET PROC FAR
1845 1842 1E | STI | INTERRUPTS BACK ON
1846 1843 E8 1A12 R | PUSH DS | SAVE SEGMENT
1847 1846 A1 0D13 R | CALL DDS
1848 1849 1F | MOV AX,0MEMORY_SIZE | GET VALUE
1849 184A CF | POP DS | RECOVER SEGMENT
1850 184B | IRET | RETURN TO CALLER
1851 | MEMORY_SIZE_DET ENDP
1852 -----
1853 | ----- INT 11 -----
1854 | EQUIPMENT DETERMINATION
1855 | THIS ROUTINE ATTEMPTS TO DETERMINE WHAT OPTIONAL
1856 | DEVICES ARE ATTACHED TO THE SYSTEM.
1857 | INPUT
1858 | NO REGISTERS
1859 | THE EQUIP_FLAG VARIABLE IS SET DURING THE POWER ON
1860 | DIAGNOSTICS USING THE FOLLOWING HARDWARE ASSUMPTIONS:
1861 | PORT 40 = LOW ORDER BYTE OF EQUIPMENT
1862 | PORT 3FA = INTERRUPT ID REGISTER OF 8250
1863 | BITS 7-3 ARE ALWAYS 0
1864 | PORT 378 = OUTPUT PORT OF PRINTER -- 8255 PORT THAT
1865 | CAN BE READ AS WELL AS WRITTEN
1866 | OUTPUT
1867 | (AX) IS SET, BIT SIGNIFICANT, TO INDICATE ATTACHED I/O
1868 | BIT 15,14 = NUMBER OF PRINTERS ATTACHED
1869 | BIT 13 NOT USED
1870 | BIT 12 = GAME I/O ATTACHED
1871 | BIT 11,10,9 = NUMBER OF RS232 CARDS ATTACHED
1872 | BIT 8 UNUSED
1873 | BIT 7,6 = NUMBER OF DISKETTE DRIVES
1874 | 00=1, 01=2, 10=3, 11=4 ONLY IF BIT 0 = 1
1875 | BIT 5,4 = INITIAL VIDEO MODE
1876 | 00 - UNUSED
1877 | 01 - 40X25 BW USING COLOR CARD
1878 | 10 - 80X25 BW USING COLOR CARD
1879 | 11 - 80X25 BW USING BW CARD
1880 | BIT 3,2 = PLANAR RAM SIZE (00=256K,01=512K,10=576K,11=640K)
1881 | BIT 1 = MATH COPROCESSOR
1882 | BIT 0 = IPL FROM DISKETTE -- THIS BIT INDICATES THAT
1883 | THERE ARE DISKETTE DRIVES ON THE SYSTEM
1884 | NO OTHER REGISTERS AFFECTED
1885 -----
1886 | ASSUME CS:CODE,DS:DATA
1887 | ORG 0F84DH
1888 184D | ORG 0184DH
1889 184D |
1890 184D FB | EQUIPMENT PROC FAR
1891 184E 1E | STI | INTERRUPTS BACK ON
1892 184F E8 1A12 R | PUSH DS | SAVE SEGMENT REGISTER
1893 1852 A1 0D10 R | CALL DDS
1894 1855 1F | MOV AX,0EQUIP_FLAG | GET THE CURRENT SETTINGS
1895 1856 CF | POP DS | RECOVER SEGMENT
1896 1857 | IRET | RETURN TO CALLER
1897 | EQUIPMENT ENDP
1898 -----
1899 | ----- INT 15 -----
1900 |
1901 1859 | ORG 0F859H
1902 1859 | ORG 01859H
1903 1859 E9 0000 E | CASSETTE_10:
1904 | JMP CASSETTE_10_1
1905 -----
1906 | NON-MASKABLE INTERRUPT ROUTINE:
1907 | THIS ROUTINE WILL PRINT A "PARITY CHECK 1 OR 2" MESSAGE ;
1908 | AND ATTEMPT TO FIND THE STORAGE LOCATION CONTAINING THE ;
1909 | BAD PARITY. IF FOUND, THE SEGMENT ADDRESS WILL BE ;
1910 | PRINTED. IF NO PARITY ERROR CAN BE FOUND (INTERMITTANT ;
1911 | READ PROBLEM) ?????<-WILL BE PRINTED WHERE THE ADDRESS ;
1912 | WOULD NORMALLY GO. ;
1913 -----
1914 185C |
1915 | NM1_INT_1 PROC NEAR
1916 185C 50 | ASSUME DS:DATA
1917 185D E4 62 | IN AX | SAVE ORIG CONTENTS OF AX
1918 185F A8 C0 | IN AL,PORT_C
1919 1861 75 03 | TEST AL,0C0H | PARITY CHECK?
1920 1863 EB 58 90 | JNZ NM1_1 | NO, EXIT FROM ROUTINE
1921 1866 | JMP D14
1922 1866 BA ---- R | NM1_1:
1923 1869 8E DA | MOV DX,DATA
1924 186B BE 18E2 R | MOV SI,OFFSET D1 | ADDR OF ERROR MSG
1925 186E A8 40 | TEST AL,40H | I/O PARITY CHECK
1926 1870 75 03 | JNZ D13 | I/O PARITY ERROR MSG
1927 1872 BE 18F2 R | MOV SI,OFFSET D2 | MUST BE PLANAR
1928 1875 |
1929 1875 B4 00 | MOV AH,0
1930 1877 A0 0049 R | MOV AL,#CRT_MODE
1931 187A CD 10 | INT 10H
1932 187C E8 1997 R | CALL VIDEO_IO PROCEDURE
1933 | | PRINT ERROR MSG

```



```

1933
1934
1935
1936 187F B0 00      MOV     AL,00H          ; DISABLE TRAP
1937 1881 E6 A0      OUT    0AH,AL
1938 1883 E4 61      IN     AL,PORT B
1939 1885 OC 30      OR     AL,00100000B    ; TOGGLE PARITY CHECK ENABLES
1940 1887 E6 61      OUT    PORT_B,AL
1941 1889 24 CF      AND    AL,10011111B
1942 188B E6 61      OUT    PORT_B,AL
1943 188D BB 1E 0013 R  MOV    BX,#MEMORY_SIZE ; GET MEMORY SIZE WORD
1944 1891 FC          CLD
1945 1892 2B D2      SUB    DX,DX           ; SET DIR FLAG TO INCREMENT
1946 1894          NMI_LOOP:           ; POINT DX AT START OF MEM
1947 1894 8E DA      MOV    DS,DX
1948 1896 8E C2      MOV    ES,DX
1949 1898 B9 4000    MOV    CX,4000H
1950 189B 2B F6      SUB    SI,SI           ; SET FOR 16KB SCAN
1951          ; SET SI TO BE RELATIVE TO
1952 189D F3/ AC      REP    LODSB          ; START OF ES
1953 189F E4 62      IN     AL,PORT C      ; READ 16KB OF MEMORY
1954 18A1 24 C0      AND    AL,11000000B    ; SEE IF PARITY CHECK HAPPENED
1955 18A3 75 11      JNZ    PRT_NMI        ; GO PRINT ADDRESS IF IT DID
1956 18A5 91 C2 0400  ADD    DX,0400H       ; POINT TO NEXT 16K BLOCK
1957 18A9 83 EB 10   SUB    BX,16D
1958 18AC 75 E6      JNZ    NMI_LOOP
1959 18AE BE 1902 R  MOV    SI,(OFFSET D2A) ; PRINT ROW OF ?????? IF PARITY
1960 18B1 E8 1997 R  CALL  P_MSG          ; CHECK COULD NOT BE RE-CREATED
1961 18B4 FA          CLI
1962 18B5 F4          HLT                  ; HALT SYSTEM
1963 18B6          PRT_NMI:
1964 18B6 8C DA      MOV    DX,DS
1965 18B8 EB 0CBA R  CALL  PRT_SEG        ; PRINT SEGMENT VALUE
1966 18BB FA          CLI
1967 18BC F4          HLT                  ; HALT SYSTEM
1968 18BD          D14:
1969 18BD 58          POP    AX             ; RESTORE ORIG CONTENTS OF AX
1970 18BE CF          IRET
1971 18BF          NMI_INT_1:         ENDP
1972
1973
1974          ;-----
1975          ; ROS CHECKSUM SUBROUTINE
1976 18BF          ;
1977 18BF B9 0000    ROS_CHECKSUM PROC NEAR ; NEXT ROS MODULE
1978 18C2          MOV    CX,0           ; NUMBER OF BYTES TO ADD
1979 18C2 32 C0      ROS_CHECKSUM_CNT:     ; ENTRY FOR OPTIONAL ROS TEST
1980 18C4          XOR    AL,AL
1981 18C4 02 07      C26: ADD    AL,DS:[BX]
1982 18C6 43 C0      INC    BX             ; POINT TO NEXT BYTE
1983 18C7 E2 FB      LOOP  C26            ; ADD ALL BYTES IN ROS MODULE
1984 18C9 0A C0      OR     AL,AL         ; SUM = 0?
1985 18CB C3          RET
1986 18CC          ROS_CHECKSUM ENDP
1987
1988          ;-----
1989          ; MESSAGE AREA FOR POST
1990 18CC 31 30 31 0D 0A E0 DB '101',CR,LF ; SYSTEM BOARD ERROR
1991 18D1 20 32 30 31 0D 0A E1 DB ' 201',CR,LF ; MEMORY ERROR
1992 18D7 52 4F 4D 0D 0A E2 DB 'ROM',CR,LF ; ROM CHECKSUM ERROR
1993 18DC 31 38 30 31 0D 0A F3C DB '1801',CR,LF ; EXPANSION IO BOX ERROR
1994 18E2 50 41 52 49 54 59 D1 DB 'PARITY CHECK 2',CR,LF
1995          20 43 48 45 43 4B
1996          20 32 0D 0A
1997 18F2 50 41 52 49 54 59 D2 DB 'PARITY CHECK 1',CR,LF
1998          20 43 48 45 43 4B
1999          20 31 0D 0A
2000 1902 3F 3F 3F 3F 3F 0D D2A DB '?????',CR,LF
2001          0A
2002
2003
2004          ;-----
2005          ; BLINK LED PROCEDURE FOR MFG RUN-IN TESTS
2006          ; IF LED IS ON, TURN IT OFF. IF OFF, TURN ON.
2007          ;-----
2008          ASSUME DS:DATA
2009 1909          BLINK_INT PROC NEAR
2010 190A 50          STI
2011 190B E4 61      PUSH  AX             ; SAVE AX REG CONTENTS
2012 190D 8A E0      MOV    AH,AL         ; READ CURRENT VAL OF PORT B
2013 190F F6 D0      NOT    AL
2014 1911 24 40      AND    AL,01000000B    ; FLIP ALL BITS
2015 1913 80 E4 BF  AND    AH,10111111B    ; ISOLATE CONTROL BIT
2016 1916 0A C4      OR     AL,AH         ; MASK OUT OF ORIGINAL VAL
2017 1918 E6 61      OUT    PORT_B,AL     ; OR NEW CONTROL BIT IN
2018 191A B0 20      MOV    AL,E0I
2019 191C E6 20      MOV    INTA00,AL
2020 191E 58          POP    AX
2021 191F CF          IRET                ; RESTORE AX REG
2022 1920          BLINK_INT ENDP
2023
2024
2025          ;-----
2026          ; THIS ROUTINE CHECKSUMS OPTIONAL ROM MODULES AND
2027          ; IF CHECKSUM IS OK, CALLS INIT/TEST CODE IN MODULE
2028 1920          ;-----
2029 1920 B8 ---- R  ROM_CHECK PROC NEAR
2030 1923 8E C0      MOV    ES,AX         ; POINT ES TO DATA AREA
2031 1925 2A E4      SUB    AH,AH         ; ZERO OUT AH
2032 1927 8A 47 02  MOV    AL,[BX+2]     ; GET LENGTH INDICATOR
2033 192A B1 09      MOV    CL,09H        ; MULTIPLY BY 512
2034 192C D3 E0      SHL    AX,CL
2035 192E 8B C8      MOV    CX,AX
2036 1930 51          ; SET COUNT
2037 1931 B9 0004    PUSH  CX             ; SAVE COUNT
2038 1934 D3 E8      SHR    AX,CL         ; ADJUST
2039 1936 03 D0      ADD    DX,AX         ; SET POINTER TO NEXT MODULE
2040 1938 59          POP    CX            ; RETRIEVE COUNT
2041 1939 E8 18C2 R  CALL  ROM_CHECKSUM_CNT ; DO CHECKSUM
2042 193C 74 06      JZ     ROM_CHECK_1
2043 193E E8 0746 R  CALL  ROM_ERR        ; POST CHECKSUM ERROR
2044 1941 EB 14 90   JMP    ROM_CHECK_END ; AND EXIT
2045 1944          ROM_CHECK_1:
2046 1944 52          PUSH  DX             ; SAVE POINTER
    
```

SECTION 5

```

2047 1945 26: CT 06 0067 R 0003      MOV     ES:10_ROM_INIT,0003H  ; LOAD OFFSET
2048 194C 26: BC 1E 0069 R          MOV     ES:10_ROM_SEG,DS    ; LOAD SEGMENT
2049 1951 26: FF 1E 0067 R          CALL   DWORD PTR ES:10_ROM_INIT ; CALL INIT./TEST ROUTINE
2050 1956 5A                          POP     DX
ROM_CHECK_END:
2051 1957                          RET                                     ; RETURN TO CALLER
2052 1957 C3
2053 1958
2054
2055
2056
2057
2058
2059
-----
; CONVERT AND PRINT ASCII CODE
; AL MUST CONTAIN NUMBER TO BE CONVERTED.
; AX AND BX DESTROYED.
-----
XPC_BYTE PROC NEAR
        PUSH  AX
        MOV   CL,4
        SHR  AL,CL
        CALL XLAT_PR
        POP  AX
        AND  AL,0FH
        PROC NEAR
XLAT_PR  ADD  AL,090H
        DAA
        DAA  AL,040H
        PRT_HEX PROC NEAR
            MOV  AH,14
            MOV  BH,0
            INT  10H
            RET
        PRT_HEX ENDP
        XLAT_PR ENDP
        XPC_BYTE ENDP
F4 LABEL WORD ; PRINTER SOURCE TABLE
        DW 3BCH
        DW 378H
        DW 278H
F4E LABEL WORD
-----
; THIS SUBROUTINE WILL PRINT A MESSAGE ON THE DISPLAY
;
; ENTRY REQUIREMENTS:
; S1 = OFFSET(ADDRESS) OF MESSAGE BUFFER
; CX = MESSAGE BYTE COUNT
; MAXIMUM MESSAGE LENGTH IS 36 CHARACTERS
-----
E_MSG PROC NEAR
        MOV  BP,S1
        CALL P_MSG
        PUSH DS
        CALL DDS
        MOV  AL,BYTE PTR 0EQUIP_FLAG ; LOOP/HALT ON ERROR
        AND  AL,01H
        JNZ  G12
        MOV  CL,1
        MOV  AL,89H
        OUT  CMD_PORT,AL
        MOV  AL,T0000101B
        OUT  PORT_B,AL
        MOV  AL,0MFG_ERR_FLAG
        OUT  PORT_A,AL
        HLT
        MFG_HALT:
        CL,1
        MOV  AL,89H
        OUT  CMD_PORT,AL
        MOV  AL,T0000101B
        OUT  PORT_B,AL
        MOV  AL,0MFG_ERR_FLAG
        OUT  PORT_A,AL
        HLT
        G12:
        POP  DS
        WRITE_MSG:
        RET
        E_MSG ENDP
P_MSG PROC NEAR
        GT2A:
        MOV  AL,CS:[S1]
        INC  SI
        PUSH AX
        CALL PRT_HEX
        POP  AX
        CMP  AL,10
        JNE  G12A
        RET
        P_MSG ENDP
        ASSUME CS:CODE,DS:DATA
-----
; THIS PROCEDURE WILL ISSUE LONG TONES (1-3/4 SECONDS) AND ONE OR
; MORE SHORT TONES (9/32 SECOND) TO INDICATE A FAILURE ON THE
; PLANAR BOARD, A BAD MEMORY MODULE, OR A PROBLEM WITH THE CRT.
;
; ENTRY PARAMETERS:
; DL = NUMBER OF LONG TONES TO BEEP.
; DL = NUMBER OF SHORT TONES TO BEEP.
-----
ERR_BEEP PROC NEAR
        PUSHF
        CL,1
        OR  DH,DH
        JZ  G3
        MOV  BL,112
        MOV  CX,1280
        CALL BEEP
        MOV  CX,49715
        CALL WAITF
        DEC  DI
        JNZ  G1
        PUSH DS
        CALL DDS
        CMP  MFG_TST,01H
        POP  DS
        JE   MFG_HALT
        G3:
        MOV  BL,18
        MOV  CX,1208
        CALL BEEP
        ; SAVE FLAGS
        ; DISABLE SYSTEM INTERRUPTS
        ; ANY LONG ONES TO BEEP
        ; NO, DO THE SHORT ONES
        ; LONG BEEPS
        ; COUNTER FOR LONG BEEPS (1-3/4 SECONDS)
        ; DIVISOR FOR 932 HZ
        ; DO THE BEEP
        ; 2/3 SECOND DELAY AFTER LONG BEEP
        ; DELAY BETWEEN BEEPS
        ; ANY MORE LONG BEEPS TO DO
        ; LOOP TILL DONE
        ; SAVE DS REGISTER CONTENTS
        ; MANUFACTURING TEST MODE?
        ; RESTORE ORIGINAL CONTENTS OF (DS)
        ; YES - STOP BLINKING LED
        ; SHORT BEEPS
        ; COUNTER FOR A SHORT BEEP (9/32)
        ; DIVISOR FOR 987 HZ
        ; DO THE SOUND
    
```

```

2161 19D1 B9 8178      MOV     CX,33144      ; 1/2 SECOND DELAY AFTER SHORT BEEP
2162 19D4 E8 0CA0 R    CALL   WAITF        ; DELAY BETWEEN BEEPS
2163 19D7 FE CA       DEC     DL           ; DONE WITH SHORT BEEPS COUNT
2164 19D9 75 EE       JNZ    G3           ; LOOP TILL DONE
2165 19DB B9 8178      MOV     CX,33144      ; 1/2 SECOND DELAY AFTER LAST BEEP
2166 19DE E8 0CA0 R    CALL   WAITF        ; MAKE IT ONE SECOND DELAY BEFORE RETURN
2167 19E1 9D          POPF   RET           ; RESTORE FLAGS TO ORIGINAL SETTINGS
2168 19E2 C3          RET                 ; RETURN TO CALLER
2169 19E3
ERR_BEOP             ENDP
2170
2171
2172 ;-----
2173 ; THIS PROCEDURE WILL SEND A SOFTWARE RESET TO THE KEYBOARD.
2174 ; SCAN CODE 'AA' SHOULD BE RETURNED TO THE CPU.
2175 ;-----
2176 KBD_RESET          PROC    NEAR
2177         ASSUME  DS:IBS0
2178         MOV     AL,08H
2179         OUT    PORT_B,AL
2180         MOV     CX,10582
G8:        LOOP     G8           ; SET KBD CLK LINE LOW
                ; WRITE 8255 PORT B
                ; HOLD KBD CLK LOW FOR 20 MS
2181 19EA E2 FE       LOOP   G8           ; LOOP FOR 20 MS
2182 19EC B0 C8       MOV     AL,0C8H
2183 19EE E6 61       OUT    PORT_B,AL
                ; SET CLK, ENABLE LINES HIGH
2184 19F0
SP_TEST1:         MOV     AL,48H
                ; ENTRY FOR MANUFACTURING TEST 2
                ; SET KBD CLK HIGH, ENABLE LOW
2185 19F0 B0 48       OUT    PORT_B,AL
2186 19F2 E6 61       MOV     AL,0FDH
                ; ENABLE KEYBOARD INTERRUPTS
2187 19F4 B0 FD       OUT    INTA01,AL
                ; WRITE 8259 IMR
2188 19F6 E6 21       MOV     DATA_AREA[0*INTR_FLAG-DATA0],0
                ; RESET INTERRUPT INDICATOR
2189 19F8 C6 06 046B R 00  MOV     STI
                ; ENABLE INTERRUPTS
2190 19FD FB          STI
                ; SETUP INTERRUPT TIMEOUT CNT
2191 19FE 2B C9       SUB     CX,CX
G9:        TEST    DATA_AREA[0*INTR_FLAG-DATA0],02H
                ; DID A KEYBOARD INTR OCCUR?
2193 1A00 F6 06 046B R 02  JNZ    G10
                ; YES - READ SCAN CODE RETURNED
2194 1A05 75 02       LOOP   G9
                ; NO - LOOP TILL TIMEOUT
2195 1A07 E2 F7
2196 1A09
G10:       IN      AL,PORT_A
                ; READ KEYBOARD SCAN CODE
2197 1A09 E4 60       MOV     BL,AL
                ; SAVE SCAN CODE JUST READ
2198 1A0B BA D8       MOV     AL,0C8H
                ; CLEAR KEYBOARD
2199 1A0D B0 C8       OUT    PORT_B,AL
2200 1A0F E6 61       RET
                ; RETURN TO CALLER
KBD_RESET          ENDP
2201 1A11 C3
2202 1A12
2203
2204 1A12             DDS    PROC    NEAR
                ; LOAD (DS) TO DATA AREA
2205 1A12 2E: 8E IE 1A18 R  MOV     DS,CS:DDSDATA
                ; PUT SEGMENT VALUE OF DATA AREA INTO DS
2206 1A17 C3          RET
                ; RETURN TO USER WITH (DS)= DATA
2207
2208 1A18 ---- R      DDSDATA DW    DATA
                ; SEGMENT SELECTOR VALUE FOR DATA AREA
2209
2210 DDS          ENDP
2211
2212 ;--- HARDWARE INT 08 H --- ( IRQ LEVEL 0 )
2213 ;-----
2214 ; THIS ROUTINE HANDLES THE TIMER INTERRUPT FROM FROM CHANNEL 0 OF
2215 ; THE 8254 TIMER. INPUT FREQUENCY IS 1.19318 MHZ AND THE DIVISOR
2216 ; IS 65536, RESULTING IN APPROXIMATELY 18.2 INTERRUPTS EVERY SECOND.
2217 ;-----
2218 ; THE INTERRUPT HANDLER MAINTAINS A COUNT (4016C) OF INTERRUPTS SINCE
2219 ; POWER ON TIME, WHICH MAY BE USED TO ESTABLISH TIME OF DAY.
2220 ; THE INTERRUPT HANDLER ALSO DECREASES THE MOTOR CONTROL COUNT (40140)
2221 ; OF THE DISKETTE, AND WHEN IT EXPIRES, WILL TURN OFF THE
2222 ; DISKETTE MOTOR(=), AND RESET THE MOTOR RUNNING FLAGS.
2223 ; THE INTERRUPT HANDLER WILL ALSO INVOKE A USER ROUTINE THROUGH
2224 ; INTERRUPT ICH AT EVERY TIME TICK. THE USER MUST CODE A
2225 ; ROUTINE AND PLACE THE CORRECT ADDRESS IN THE VECTOR TABLE.
2226 ;-----
2227         ASSUME  CS:CODE,DS:DATA
2228
2229 1A1A             TIMER_INT_1 PROC    NEAR
2230 1A1A FB          STI
                ; INTERRUPTS BACK ON
2231 1A1B IE          PUSH  DS
2232 1A1C 50          PUSH  AX
2233 1A1D 52          PUSH  DX
                ; SAVE MACHINE STATE
2234 1A1E 8B ---- R  MOV     AX,DATA
                ; GET ADDRESS OF DATA SEGMENT
2235 1A21 BE D8       MOV     DS,AX
                ; ESTABLISH ADDRESSABILITY
2236 1A23 FF 06 006C R INC     @TIMER_LOW
                ; INCREMENT TIME
2237 1A27 75 04       JNZ    T4
                ; GO TO TEST DAY
2238 1A29 FF 06 006E R INC     @TIMER_HIGH
                ; INCREMENT HIGH WORD OF TIME
2239 1A2D             T4:    CMP     @TIMER_HIGH,018H
                ; TEST FOR COUNT EQUALING 24 HOURS
2240 1A32 75 19       JNZ    T5
                ; GO TO DISKETTE_CTL
2241 1A34 81 3E 006C R 00B0 CMP     @TIMER_LOW,0B0H
                ; GO TO DISKETTE_CTL
2242 1A3A 75 11       JNZ    T5
2243
2244 ;----- TIMER HAS GONE 24 HOURS
2245
2246
2247 1A3C 2B C0       SUB     AX,AX
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,1
                ; SET TIMER ELAPSED 24 HOURS FLAG
2251 1A49 FF 06 00CE R INC     @DAY_COUNT
                ; INCREMENT ELAPSED DAY COUNTER
2252
2253 ;----- TEST FOR DISKETTE TIME OUT
2254
2255 T5:
2256 1A4D             DEC     @MOTOR_COUNT
                ; DECREMENT DISKETTE MOTOR CONTROL
2257 1A51 75 08       JNZ    T6
                ; RETURN IF COUNT NOT OUT
2258 1A53 B0 26 003F R F0 AND     @MOTOR_STATUS,0F0H
                ; TURN OFF MOTOR RUNNING BITS
2259 1A58 B0 0C       MOV     AL,0CH
2260 1A5A BA 03F2     MOV     DX,03F2H
                ; FDC CTL PORT
2261 1A5D EE          OUT    DX,AL
                ; TURN OFF THE MOTOR
2262
2263 T6:
2264 1A5E             INT    ICH
                ; TIMER TICK INTERRUPT
                ; TRANSFER CONTROL TO A USER ROUTINE
2265
2266 1A60 FA          CLI
                ; DISABLE INTERRUPTS TILL STACK CLEARED
2267 1A61 B0 20       MOV     AL,E01
                ; GET END OF INTERRUPT MASK
2268 1A63 E6 20       OUT    INTA00,AL
                ; END OF INTERRUPT TO 8259 - 1
2269 1A65 5A          POP    DX
                ; RESTORE (DX)
2270 1A66 58          POP    AX
2271 1A67 1F          POP    DS
                ; RESET MACHINE STATE
2272 1A68 CF          IRET
                ; RETURN FROM INTERRUPT
2273
2274 1A69             TIMER_INT_1 ENDP

```

PAGE		CHARACTER GENERATOR GRAPHICS FOR 320X200 AND 640X200 GRAPHICS		
ADDRESS	ORG	ORG	OF	FACEH
ADDRESS	IA6E	ORG	OF	FACEH
ADDRESS	IA6E	ORG	OF	FACEH
ADDRESS	IA6E	ORG	OF	FACEH
2275	IA6E	00	00	00 00 00 00
2276	IA6E	00	00	00 00 00 00
2277	IA6E	00	00	00 00 00 00
2278	IA6E	00	00	00 00 00 00
2279	IA6E	00	00	00 00 00 00
2280	IA6E	00	00	00 00 00 00
2281	IA6E	00	00	00 00 00 00
2282	IA6E	00	00	00 00 00 00
2283	IA6E	00	00	00 00 00 00
2284	IA7E	81	A5	81 BD 90
2285	81			
2286	IA7E	7E	FF	DB FF C3 E7
2287	7E			
2288	IA8E	6C	FE	FE FC 78
2289	6C			
2290	IA8E	10	38	7C FE 7C 38
2291	10			
2292	IA9E	38	7C	38 FE FC 7C
2293	38			
2294	IA9E	10	10	38 FC FE FC 7C
2295	10			
2296	IAA6	00	00	18 3C 3C 18
2297	00			
2298	IAAE	FF	FF	ET C3 C3 E7
2299	FF			
2300	IA8E	00	3C	66 42 42 66
2301	00			
2302	IA8E	FF	C3	99 BD BD 99
2303	FF			
2304	IAC6	0F	07	0F 7D CC CC
2305	0F			
2306	IACE	3C	66	66 66 3C 18
2307	3C			
2308	IAD6	3F	33	3F 30 30 70
2309	3F			
2310	IAD6	7F	63	7F 63 63 67
2311	7F			
2312	IAE6	99	5A	3C E7 E7 3C
2313	99			
2314	5A			
2315	IAEE	80	E0	F8 FE F8 E0
2316	80			
2317	IAF6	02	0E	3E FE 3E 0E
2318	02			
2319	IAFE	18	3C	7E 18 18 7E
2320	18			
2321	IB06	66	66	66 66 66 00
2322	66			
2323	IB0E	7F	DB	DB 7B 1B 1B
2324	7F			
2325	IB16	3E	63	38 6C 6C 38
2326	3E			
2327	IB1E	00	00	00 00 7E 7E
2328	00			
2329	IB26	18	3C	7E 18 7E 3C
2330	18			
2331	IB2E	18	3C	7E 18 18 18
2332	18			
2333	IB36	18	18	18 18 7E 3C
2334	18			
2335	IB3E	18	00	0C FE 0C 18
2336	18			
2337	IB46	00	30	60 FE 60 30
2338	00			
2339	IB4E	00	C0	C0 C0 FE
2340	00			
2341	IB56	00	24	66 FF 66 24
2342	00			
2343	IB5E	00	18	3C FE FF FF
2344	00			
2345	IB66	00	FF	FF FE 3C 18
2346	00			
2347	00			
2348	IB6E	00	00	00 00 00 00
2349	00			
2350	IB76	30	78	30 30 00 00
2351	30			
2352	IB7E	6C	6C	6C 00 00 00
2353	6C			
2354	IB86	6C	FE	6C FE 6C
2355	6C			
2356	IB8E	30	7C	C0 78 0C 6C
2357	30			
2358	IB96	00	C6	CC 18 30 66
2359	00			
2360	IB9E	38	6C	38 7C 6C CC
2361	38			
2362	IBA6	60	C0	00 00 00 00
2363	60			
2364	IBAE	18	30	60 60 60 30
2365	18			
2366	IBB6	60	30	18 18 30 60
2367	60			
2368	IBBE	00	66	3C FF 3C 66
2369	00			
2370	IBC6	00	30	30 FC 30 30
2371	00			
2372	IBCE	00	00	00 00 00 00
2373	00			
2374	IBD6	00	00	00 FC 00 00
2375	00			
2376	IBDE	00	00	00 00 00 00
2377	00			
2378	IBE6	06	0C	18 30 60 C0
2379	06			
2380	0C			
2381	IBEE	7C	C6	CE DE F6 E6
2382	7C			
2383	IBF6	30	70	30 30 30 30
2384	30			
2385	IBFE	78	CC	0C 38 60 CC
2386	78			
2387	IC06	78	CC	0C 38 0C CC
2388	78			



2389	1C0E	1C	3C	6C	CC	FE	0C	DB	01CH,03CH,06CH,0CCH,0FEH,00CH,01EH,000H D_34	4
2390		1E	00							
2391	1C16	FC	00	F8	0C	0C	CC	DB	0FCH,0C0H,0F8H,00CH,00CH,0CCH,078H,000H D_35	5
2392		78	00							
2393	1C1E	38	60	C0	F8	CC	CC	DB	038H,060H,0C0H,0F8H,00CH,0CCH,078H,000H D_36	6
2394		78	00							
2395	1C26	FC	00	0C	18	30	30	DB	0FCH,0CCH,00CH,018H,030H,030H,030H,000H D_37	7
2396		30	00							
2397	1C2E	78	CC	CC	78	CC	CC	DB	078H,0CCH,0CCH,078H,0CCH,0CCH,078H,000H D_38	8
2398		78	00							
2399	1C36	78	00	CC	7C	0C	18	DB	078H,0CCH,0CCH,07CH,00CH,018H,070H,000H D_39	9
2400		70	00							
2401	1C3E	00	30	30	00	00	30	DB	000H,030H,030H,000H,000H,030H,030H,000H D_3A	10
2402		30	00							COLON
2403	1C46	00	30	30	00	00	30	DB	000H,030H,030H,000H,000H,030H,030H,060H D_3B	11
2404		30	60							SEMICOLON
2405	1C4E	18	30	60	C0	60	30	DB	018H,030H,060H,0C0H,060H,030H,018H,000H D_3C	12
2406		18	00							LESS THAN
2407	1C56	00	00	FC	00	00	FC	DB	000H,000H,0FCH,000H,000H,0FCH,000H,000H D_3D	13
2408		00	00							EQUAL
2409	1C5E	60	30	18	0C	18	30	DB	060H,030H,018H,00CH,018H,030H,060H,000H D_3E	14
2410		60	00							GREATER THAN
2411	1C66	78	CC	0C	18	30	00	DB	078H,0CCH,00CH,018H,030H,000H,030H,000H D_3F	15
2412		30	00							QUESTION MARK
2413										
2414	1C6E	7C	C6	DE	DE	DE	C0	DB	07CH,0C6H,0DEH,0DEH,0DEH,0C0H,078H,000H D_40	16
2415		78	00							AT
2416	1C76	30	78	CC	CC	FC	CC	DB	030H,078H,0CCH,0CCH,0FCH,0CCH,0CCH,000H D_41	17
2417		CC	00							A
2418	1C7E	F8	60	66	7C	66	66	DB	0FCH,066H,066H,07CH,066H,066H,0FCH,000H D_42	18
2419		FC	00							B
2420	1C86	3C	66	C0	C0	C0	66	DB	03CH,066H,0C0H,0C0H,0C0H,066H,03CH,000H D_43	19
2421		3C	66							C
2422	1C8E	F8	6C	66	66	66	6C	DB	0F8H,06CH,066H,066H,066H,06CH,0F8H,000H D_44	20
2423		F8	00							D
2424	1C96	FE	62	68	78	68	62	DB	0FEH,062H,068H,078H,068H,062H,0FEH,000H D_45	21
2425		FE	60							E
2426	1C9E	FE	62	68	78	68	60	DB	0FEH,062H,068H,078H,068H,060H,0F0H,000H D_46	22
2427		F0	00							F
2428	1CA6	3C	66	C0	C0	CE	66	DB	03CH,066H,0C0H,0C0H,0CEH,066H,03EH,000H D_47	23
2429		3E	00							G
2430	1CAE	CC	CC	CC	FC	CC	CC	DB	0CCH,0CCH,0CCH,0FCH,0CCH,0CCH,0CCH,000H D_48	24
2431		CC	00							H
2432	1CB6	78	30	30	30	30	30	DB	078H,030H,030H,030H,030H,030H,078H,000H D_49	25
2433		78	00							I
2434	1CBE	1E	0C	0C	0C	CC	CC	DB	01EH,00CH,00CH,00CH,0CCH,0CCH,078H,000H D_4A	26
2435		78	00							J
2436	1CC6	E6	66	66	78	6C	66	DB	0E6H,066H,06CH,078H,06CH,066H,0E6H,000H D_4B	27
2437		E6	00							K
2438	1CCE	F0	60	60	60	62	66	DB	0F0H,060H,060H,060H,062H,066H,0FEH,000H D_4C	28
2439		FE	00							L
2440	1CD6	C6	E6	FE	FE	D6	C6	DB	0C6H,0EEH,0FEH,0FEH,0D6H,0C6H,0C6H,000H D_4D	29
2441		C6	00							M
2442	1CDE	C6	E6	FE	DE	CE	C6	DB	0C6H,0EEH,0FEH,0FEH,0D6H,0C6H,0C6H,000H D_4E	30
2443		C6	00							N
2444	1CE6	38	00	C6	C6	C6	6C	DB	038H,06CH,0C6H,0C6H,0C6H,06CH,038H,000H D_4F	31
2445		38	00							O
2446										
2447	1CEE	FC	66	66	7C	60	60	DB	0FCH,066H,066H,07CH,060H,060H,0F0H,000H D_50	32
2448		F0	00							P
2449	1CF6	78	CC	CC	CC	DC	78	DB	078H,0CCH,0CCH,0CCH,0DCH,078H,01CH,000H D_51	33
2450		1C	00							Q
2451	1CFE	FC	66	66	7C	6C	78	DB	0FCH,066H,066H,07CH,06CH,066H,0E6H,000H D_52	34
2452		E6	00							R
2453	1D06	78	CC	E0	70	1C	CC	DB	078H,0CCH,0E0H,070H,01CH,0CCH,078H,000H D_53	35
2454		78	00							S
2455	1D0E	FC	00	30	30	30	30	DB	0FCH,0B4H,030H,030H,030H,030H,078H,000H D_54	36
2456		78	00							T
2457	1D16	CC	CC	CC	CC	CC	CC	DB	0CCH,0CCH,0CCH,0CCH,0CCH,0CCH,0FCH,000H D_55	37
2458		FC	00							U
2459	1D1E	CC	CC	CC	CC	CC	78	DB	0CCH,0CCH,0CCH,0CCH,0CCH,078H,030H,000H D_56	38
2460		30	00							V
2461	1D26	C6	C6	D6	D6	FE	EE	DB	0C6H,0C6H,0C6H,0D6H,0FEH,0EEH,0C6H,000H D_57	39
2462		C6	00							W
2463	1D2E	C6	C6	6C	38	38	6C	DB	0C6H,0C6H,06CH,038H,038H,06CH,0C6H,000H D_58	40
2464		C6	00							X
2465	1D36	CC	CC	CC	78	30	30	DB	0CCH,0CCH,0CCH,078H,030H,030H,078H,000H D_59	41
2466		78	00							Y
2467	1D3E	FC	6C	8C	18	32	66	DB	0FEH,0C6H,08CH,018H,032H,066H,0FEH,000H D_5A	42
2468		FE	00							Z
2469	1D46	78	60	60	60	60	60	DB	078H,060H,060H,060H,060H,060H,078H,000H D_5B	43
2470		78	00							[LEFT BRACKET
2471	1D4E	C0	60	30	18	0C	06	DB	0C0H,060H,030H,018H,00CH,066H,002H,000H D_5C	44
2472		02	00							BACKSLASH
2473	1D56	78	18	18	18	18	18	DB	078H,018H,018H,018H,018H,018H,078H,000H D_5D	45
2474		78	00							RIGHT BRACKET
2475	1D5E	10	38	6C	C6	00	00	DB	010H,038H,06CH,0C6H,000H,000H,000H,000H D_5E	46
2476		00	00							CIRCUMFLEX
2477	1D66	00	00	00	00	00	00	DB	000H,000H,000H,000H,000H,000H,000H,0FFH D_5F	47
2478		00	FF							UNDERScore
2479										
2480	1D6E	30	30	18	00	00	00	DB	030H,030H,018H,000H,000H,000H,000H,000H D_60	48
2481		00	00							APOSTROPHE REV
2482	1D76	00	00	78	0C	7C	CC	DB	000H,000H,078H,00CH,07CH,0CCH,076H,000H D_61	49
2483		76	00							a
2484	1D7E	E0	60	7C	66	66	66	DB	0E0H,060H,060H,07CH,066H,066H,0DCH,000H D_62	50
2485		DC	00							b
2486	1D86	00	00	78	CC	C0	CC	DB	000H,000H,078H,0CCH,0C0H,0CCH,0CCH,078H,000H D_63	51
2487		78	00							c
2488	1D8E	1C	0C	0C	7C	CC	CC	DB	01CH,00CH,00CH,07CH,0CCH,0CCH,076H,000H D_64	52
2489		76	00							d
2490	1D96	00	00	78	CC	FC	C0	DB	000H,000H,078H,0CCH,0FCH,0C0H,078H,000H D_65	53
2491		78	00							e
2492	1D9E	38	C6	F0	F0	60	60	DB	038H,06CH,060H,0F0H,060H,060H,0F0H,000H D_66	54
2493		F0	00							f
2494	1DA6	00	00	76	CC	CC	7C	DB	000H,000H,076H,0CCH,0CCH,07CH,0CCH,078H,000H D_67	55
2495		0C	F8							g
2496	1DAE	E0	60	6C	76	66	66	DB	0E0H,060H,06CH,076H,066H,066H,0E6H,000H D_68	56
2497		E6	00							h
2498	1DB6	30	00	70	30	30	30	DB	030H,000H,070H,030H,030H,030H,078H,000H D_69	57
2499		78	00							i
2500	1DBE	0C	00	0C	0C	0C	0C	DB	00CH,000H,00CH,00CH,0CCH,0CCH,0CCH,078H D_6A	58
2501		CC	78							j
2502	1DC6	E0	60	66	6C	78	6C	DB	0E0H,060H,066H,06CH,078H,06CH,0E6H,000H D_6B	59
										k

```

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

```

2545 PAGE
2546 ;--- INT IA -----
2547 ; TIME OF DAY
2548 ; THIS ROUTINE ALLOWS THE CLOCK TO BE SET/READ
2549 ;
2550 ; INPUT
2551 ; (AH) = 0 READ THE CURRENT CLOCK SETTING
2552 ; RETURNS CX = HIGH PORTION OF COUNT
2553 ; DX = LOW PORTION OF COUNT
2554 ; AL = 0 IF TIMER HAS NOT PASSED
2555 ; 24 HOURS SINCE LAST READ
2556 ; <=0 IF ON ANOTHER DAY
2557 ; (AH) = 1 SET THE CURRENT CLOCK
2558 ; CX = HIGH PORTION OF COUNT
2559 ; DX = LOW PORTION OF COUNT
2560 ; NOTE: COUNTS OCCUR AT THE RATE OF
2561 ; 119180/65536 COUNTS/SEC
2562 ; (OR ABOUT 18.2 PER SECOND -- SEE EQUATES BELOW)
2563 ;-----
2564 ; ASSUME CS:CODE,DS:DATA
2565 ; ORG 0FE6H
2566 IE6E ORG 01E6H
2567 IE6E TIME_OF_DAY: TIME_OF_DAY_11
2568 IE6E E9 0995 R JMP TIME_OF_DAY_11
2569
2570 ; ORG 0FEA5H
2571 IEA5 ORG 01EA5H
2572 IEA5 TIMER_INT:
2573 IEA5 E9 1A1A R JMP TIMER_INT_1
2574
2575 ;-----
2576 ; THESE ARE THE VECTORS WHICH ARE MOVED INTO
2577 ; THE 8086 INTERRUPT AREA DURING POWER ON.
2578 ; ONLY THE OFFSETS ARE DISPLAYED HERE, CODE
2579 ; SEGMENT WILL BE ADDED FOR ALL OF THEM, EXCEPT
2580 ; WHERE NOTED.
2581 ;-----
2582 ; ASSUME CS:CODE
2583 ; ORG 01EF3H
2584 IEF3 ORG 01EF3H
2585 IEF3 VECTOR_TABLE LABEL WORD ; VECTOR TABLE VALUES FOR POST TESTS
2586 IEF3 IEA5 R DW OFFSET TIMER_INT ; INT 08H - HARDWARE TIMER 0 IRQ 0
2587 IEF5 09B7 R DW OFFSET KB_INT ; INT 09H - KEYBOARD IRQ 1
2588 IEF7 1F23 R DW OFFSET DT1 ; INT 0AH - IRQ 2
2589 IEF9 1F23 R DW OFFSET D11 ; INT 0BH - IRQ 3
2590 IEFB 1F23 R DW OFFSET D11 ; INT 0CH - IRQ 4
2591 IEFD 1F23 R DW OFFSET D11 ; INT 0DH - IRQ 5
2592 IEFF 0F57 R DW OFFSET DISK_INT ; INT 0EH - DISKETTE IRQ 6
2593 IF01 1F23 R DW OFFSET D11 ; INT 0FH - IRQ 7
2594
2595 ;----- SOFTWARE INTERRUPTS ( BIOS CALLS AND POINTERS )
2596
2597 IF03 1065 R DW OFFSET VIDEO_IO ; INT 10H -- VIDEO DISPLAY
2598 IF05 1840 R DW OFFSET EQUIPMENT ; INT 11H -- GET EQUIPMENT FLAG WORD
2599 IF07 1841 R DW OFFSET MEMORY_SIZE_DET ; INT 12H -- GET REAL MODE MEMORY SIZE
2600 IF09 0C59 R DW OFFSET DISKETTE_IO ; INT 13H -- DISKETTE
2601 IF0B 0739 R DW OFFSET RS232_IO ; INT 14H -- COMMUNICATION ADAPTER
2602 IF0D 1859 R DW OFFSET CASSETTE_IO ; INT 15H -- EXPANDED BIOS FUNCTION CALL
2603 IF0F 082E R DW OFFSET KEYBOARD_IO ; INT 16H -- KEYBOARD INPUT
2604 IF11 0FD2 R DW OFFSET PRINTER_IO ; INT 17H -- PRINTER OUTPUT
2605 IF13 0000 DW 00000H ; INT 18H -- 0F60H INSERTED FOR BASIC
2606 IF15 08F2 R DW OFFSET BOOT_STRAP ; INT 19H -- BOOT FROM SYSTEM MEDIA
2607 IF17 1E6E R DW OFFSET TIME_OF_DAY ; INT 1AH -- TIME OF DAY
2608 IF19 1F49 R DW OFFSET DUMMY_RETURN ; INT 1BH -- KEYBOARD BREAK ADDRESS
2609 IF1B 1F49 R DW OFFSET DUMMY_RETURN ; INT 1CH -- TIMER BREAK ADDRESS
2610 IF1D 10A4 R DW OFFSET VIDEO_PARAMS ; INT 1DH -- VIDEO PARAMETERS
2611 IF1F 0FC7 R DW OFFSET DISK_BASE ; INT 1EH -- DISKETTE PARAMETERS
2612 IF21 0000 DW 00000H ; INT 1FH -- POINTER TO VIDEO EXTENSION
2613
2614 ;-----
2615 ; TEMPORARY INTERRUPT SERVICE ROUTINE
2616 ; 1. THIS ROUTINE IS ALSO LEFT IN PLACE AFTER THE
2617 ; POWER ON DIAGNOSTICS TO SERVICE UNUSED
2618 ; INTERRUPT VECTORS. LOCATION 'INTR_FLAG' WILL
2619 ; CONTAIN EITHER: 1. LEVEL OF HARDWARE INT. THAT
2620 ; CAUSED CODE TO BE EXEC.
2621 ; 2. 'FF' FOR NON-HARDWARE INTERRUPTS THAT WAS
2622 ; EXECUTED ACCIDENTLY.
2623 ;-----
2624 IF23 D11 PROC NEAR DS:DATA
2625 ; ASSUME DS:DATA
2626 IF23 IE PUSH DS
2627 IF24 E8 IA12 R CALL DDS ; SAVE REG AX CONTENTS
2628 IF27 50 R PUSH AX ; READ IN-SERVICE REG
2629 IF28 B0 0B MOV AL,0BH ; IFIND OUT WHAT LEVEL BEING
2630 IF2A E6 20 OUT INTA00,AL ; SERVICED
2631 IF2C 90 NOP ; GET LEVEL
2632 IF2D EA 20 IN AL,INTA00 ; SAVE IT
2633 IF2F 8A E0 MOV AH,AL ;
2634 IF31 0A C4 OR AL,AH ; 00? (NO HARDWARE ISR ACTIVE)
2635 IF33 75 04 JNZ INT ;
2636 IF35 B4 FF MOV AH,0FFH ;
2637 IF37 EB 0A JMP SHORT SET_INTR_FLAG ; SET FLAG TO FF IF NON-HOWARE
2638 IF39
2639 IF39 E4 21 MOV IN,AL,INTA01 ; GET MASK VALUE
2640 IF3B 0A C4 OR AL,AH ; MASK OFF LVL BEING SERVICED
2641 IF3D E6 21 OUT INTA01,AL ;
2642 IF3F B0 20 MOV AL,ED ;
2643 IF41 E6 20 OUT INTA00,AL ;
2644 IF43
2645 IF43 88 26 006B R SET_INTR_FLAG: MOV @INTR_FLAG,AH ; SET FLAG
2646 IF47 58 POP AX ; RESTORE REG AX CONTENTS
2647 IF48 1F POP DS ;
2648 IF49 DUMMY_RETURN: ;
2649 IF49 CF IRET ; NEED IRET FOR VECTOR TABLE
2650 IF4A ENDP D11
2651
2652 ;-----
2653 ; DUMMY RETURN FOR ADDRESS COMPATIBILITY
2654 ;
2655 ; ORG 0FF53H
2656 IF53 CF ORG 01F53H
2657 IF53 CF IRET
    
```

```

2658      PAGE
2659      ---- INT 05H -----
2660      | PRINT_SCREEN |
2661      | THIS LOGIC WILL BE INVOKED BY INTERRUPT 05H TO PRINT THE SCREEN. |
2662      | THE CURSOR POSITION AT THE TIME THIS ROUTINE IS INVOKED WILL BE |
2663      | SAVED AND RESTORED UPON COMPLETION. THE ROUTINE IS INTENDED TO |
2664      | RUN WITH INTERRUPTS ENABLED IF A SUBSEQUENT PRINT_SCREEN KEY |
2665      | IS DEPRESSED WHILE THIS ROUTINE IS PRINTING IT WILL BE IGNORED. |
2666      | THE BASE PRINTERS STATUS IS CHECKED FOR NOT BUSY AND NOT OUT OF |
2667      | PAPER. AN INITIAL STATUS ERROR WILL ABEND THE PRINT REQUEST. |
2668      | ADDRESS 0050:0000 CONTAINS THE STATUS OF THE PRINT_SCREEN |
2669      |
2670      | 50:0 = 0 PRINT_SCREEN HAS NOT BEEN CALLED OR UPON RETURN |
2671      | = 1 FROM A CALL THIS INDICATES A SUCCESSFUL OPERATION. |
2672      | = I PRINT_SCREEN IS IN PROGRESS - IGNORE THIS REQUEST. |
2673      | = 255 ERROR ENCOUNTERED DURING PRINTING. |
2674      |-----|
2675      | ORG OFF54H |
2676      | ORG 01F54H |
2677      |
2678      IF54      PRINT_SCREEN_1 PROC FAR
2679      | DELAY INTERRUPT ENABLE TILL FLAG SET |
2680      IF54 IE    PUSH DS          | USE 0040:0100 FOR STATUS AREA STORAGE |
2681      IF55 E8 1A12 R CALL DDS          | GET STATUS_BYTE DATA SEGMENT |
2682      IF56 30 3E 0100 R 01 CMP #STATUS_BYTE, I | SEE IF PRINT_ALREADY IN PROGRESS |
2683      IF5D 74 7C    JE PR190       | EXIT IF PRINT_ALREADY IN PROGRESS |
2684      IF5F C6 06 0100 R 01 MOV #STATUS_BYTE, I | INDICATE PRINT_NOW IN PROGRESS |
2685      IF64 FB      STI             | MUST RUN WITH INTERRUPTS ENABLED |
2686      IF65 50      PUSH AX         | SAVE WORK REGISTERS |
2687      IF66 53      PUSH BX         |
2688      IF67 51      PUSH CX         |
2689      IF68 52      PUSH DX         |
2690      IF69 B4 0F  MOV AH, 0FH      | WILL REQUEST THE CURRENT SCREEN MODE |
2691      IF6B CD 10  INT 10H          | (AL)= MODE |
2692      | (AH)= NUMBER COLUMNS/LINE |
2693      | (BH)= VISUAL PAGE |
2694      IF6D 8A CC  MOV CL, AH          | WILL MAKE USE OF (CX) REGISTER TO |
2695      IF6F 8A 2E 0084 R MOV CH, #ROWS      | CONTROL ROWS ON SCREEN & COLUMNS |
2696      IF73 FE C5  INC CH           | ADJUST ROWS ON DISPLAY COUNT |
2697      | (CL)= NUMBER COLUMNS/LINE |
2698      | (CH)= NUMBER OF ROWS ON DISPLAY |
2699      |-----|
2700      | AT THIS POINT WE KNOW THE COLUMNS/LINE COUNT IS IN (CL) |
2701      | AND THE NUMBER OF ROWS ON THE DISPLAY IS IN (CH) |
2702      | THE PAGE IF APPLICABLE IS IN (BH) - THE STACK HAS |
2703      | (DS), (AX), (BX), (CX), (DX) PUSHED. |
2704      |-----|
2705      IF75 33 02  XOR DX, DX          | FIRST PRINTER |
2706      IF77 B4 02  MOV AH, 02H        | SET PRINTER STATUS REQUEST COMMAND |
2707      IF79 CD 17  INT 17H           | REQUEST CURRENT PRINTER STATUS |
2708      IF7B 84 00  XOR AH, 080H      | CHECK FOR PRINTER BUSY (NOT CONNECTED) |
2709      IF7E F6 C4 A0 OR AX, 0         | OR OUT OF PAPER |
2710      IF81 75 4E  JNZ PR170       | ERROR EXIT IF PRINTER STATUS ERROR |
2711      |
2712      IF83 EB 1FDD R CALL CRLF         | CARRIAGE RETURN LINE FEED TO PRINTER |
2713      IF86 51      PUSH CX           | SAVE SCREEN BOUNDS |
2714      IF87 B4 03  MOV AH, 03H        | NOW READ THE CURRENT CURSOR POSITION |
2715      IF89 CD 10  INT 10H           | AND RESTORE AT END OF ROUTINE |
2716      IF8B 59      POP CX           | RECALL SCREEN BOUNDS |
2717      IF8C 52      PUSH DX           | PRESERVE THE ORIGINAL POSITION |
2718      IF8D 33 02  XOR DX, DX          | INITIAL CURSOR (0,0) AND FIRST PRINTER |
2719      |-----|
2720      | THIS LOOP IS TO READ EACH CURSOR POSITION FROM THE |
2721      | SCREEN AND PRINT IT. (BH)= VISUAL PAGE (CH)= ROWS |
2722      |-----|
2723      IF8F      PR110:
2724      IF8F B4 02  MOV AH, 02H        | INDICATE CURSOR SET REQUEST |
2725      IF91 CD 10  INT 10H           | NEW CURSOR POSITION ESTABLISHED |
2726      IF93 B4 08  MOV AH, 08H        | INDICATE READ CHARACTER FROM DISPLAY |
2727      IF95 CD 10  INT 10H           | CHARACTER NOW IN (AL) |
2728      IF97 0A C0  OR AL, AL          | SEE IF VALID CHAR |
2729      IF99 75 02  JNZ PR120       | JUMP IF VALID CHAR |
2730      IF9B B0 20  MOV AL, ' '        | ELSE MAKE IT A BLANK |
2731      |
2732      IF9D      PR120:
2733      IF9D 52      PUSH DX           | SAVE CURSOR POSITION |
2734      IF9E 33 02  XOR DX, DX          | INDICATE FIRST PRINTER (DX= 0) |
2735      IFA0 32 E4  XOR AH, AH        | INDICATE PRINT CHARACTER IN (AL) |
2736      IFA2 CD 17  INT 17H           | PRINT THE CHARACTER |
2737      IFA4 5A      POP DX           | RECALL CURSOR POSITION |
2738      IFA5 F6 C4 29 TEST AH, 29H      | TEST FOR PRINTER ERROR |
2739      IFA7 FE C2  JNZ PR160       | EXIT IF ERROR DETECTED |
2740      IFAA FE C2  INC DL           | ADVANCE TO NEXT COLUMN |
2741      IFAC 3A CA  CMP CL, DL        | SEE IF AT END OF LINE |
2742      IFAE 75 0F  JNZ PR110       | IF NOT LOOP FOR NEXT COLUMN |
2743      IFB0 32 02  XOR DL, DL        | BACK TO COLUMN 0 |
2744      IFB2 8A E2  MOV AH, DL        | (AH)=0 |
2745      IFB4 52      PUSH DX           | SAVE NEW CURSOR POSITION |
2746      IFB5 E8 1FDD R CALL CRLF         | LINE FEED CARRIAGE RETURN |
2747      IFB8 5A      POP DX           | RECALL CURSOR POSITION |
2748      IFB9 FE C6  INC DH           | ADVANCE TO NEXT LINE |
2749      IFBB 3A EE  CMP CH, DH        | FINISHED? |
2750      IFBD 75 02  JNZ PR110       | IF NOT LOOP FOR NEXT LINE |
2751      |
2752      IFBF      PR110:
2753      IFBF 5A      POP DX           | GET CURSOR POSITION |
2754      IFC0 B4 02  MOV AH, 02H        | INDICATE REQUEST CURSOR SET |
2755      IFC2 CD 10  INT 10H           | CURSOR POSITION NOW RESTORED |
2756      IFC4 FA      CLI             | BLOCK INTERRUPTS TILL STACK CLEARED |
2757      IFC5 C6 06 0100 R 00 MOV #STATUS_BYTE, 0 | MOVE OK RESULTS FLAG TO STATUS_BYTE |
2758      IFCA EB 08  JMP SHORT PR180   | EXIT PRINTER ROUTINE |
    
```



```

2757                                     PAGE
2758 IFCC                                PR160:                                |
2759 IFCC 5A                              POP     DX                                | ERROR EXIT
2760 IFCD 84 02                            MOV     AH,02H                           | GET CURSOR POSITION
2761 IFCF CD 10                            INT     10H                               | INDICATE REQUEST CURSOR SET
2762 IFD1                                    PR170:                                | CURSOR POSITION RESTORED
2763 IFD1 FA                              CLI
2764 IFD2 C6 06 0100 R FF                 MOV     *STATUS_BYTE,0FFH                | BLOCK INTERRUPTS TILL STACK CLEARED
2765 IFD7                                    PR180:                                | SET ERROR FLAG
2766 IFD7 5A                              POP     DX                                | EXIT ROUTINE
2767 IFD8 59                              POP     CX                                | RESTORE ALL THE REGISTERS USED
2768 IFD9 58                              POP     BX
2769 IFDA 58                              POP     AX
2770 IFDB 58                              PR190:                                | ROUTINE BUSY EXIT
2771 IFDB 1F                              POP     DS
2772 IFDC CF                              IRET                                     | RETURN WITH INITIAL INTERRUPT MASK
2773 IFDD                                    PRINT_SCREEN_1 ENDP
2774
2775 ;----- CARRIAGE RETURN, LINE FEED SUBROUTINE
2776
2777 IFDD                                    CRLF PROC NEAR
2778                                     |
2779 IFDD 33 D2                            XOR     DX,DX                              | SEND CR,LF TO FIRST PRINTER
2780 IFDE B8 000D                           MOV     AX,CR                              | ASSUME FIRST PRINTER (DX=0)
2781 IFDF 17                                INT     17H                               | GET THE PRINT CHARACTER COMMAND AND
2782 IFE0 B8 000A                           MOV     AX,LF                              | THE CARRIAGE RETURN CHARACTER
2783 IFE1 CD 17                            INT     17H                               | NOW GET THE LINE FEED AND
2784 IFE9 C3                                RET                                         | SEND IT TO THE BIOS PRINTER ROUTINE
2785 IFEA                                    CRLF ENDP
2786
2787 ;-----
2788 ; POWER ON RESET VECTOR ;
2789 ;-----
2790 ORG 0FFF0H
2791 IF F0 ORG 01FF0H
2792
2793 ;----- POWER ON RESET
2794 P_OR LABEL FAR
2795 IF F0 DB 0EAH
2796 IF F1 E05B DW 0E05BH ; LOW WORD OF RESET
2797 IF F3 F000 DW 0F000H ; SEGMENT
2798
2799 IF F5 30 31 2F 31 30 2F DB '01/10/86' ; RELEASE MARKER
2800 38 36
2801
2802 ; ORG 0FFFEH
2803 IF FE ORG 01FFFEH
2804 IF FE MODEL: DB MODEL_BYTE
2805 IF FE FB
2806
2807 IF FF CODE ENDS
2808 END
    
```

Notes:



System BIOS Listing - 11/8/82

Quick Reference - 64/256K Board

Equates	5-113
8088 Interrupt Locations	5-113
Stack	5-113
Data Areas	5-113
Power-On-Self-Test	5-115
Boot Strap Loader	5-127
I/O Support	
RS-232C	5-128
Keyboard	5-131
Diskette	5-138
Printer	5-146
Display	5-148
System Configuration Analysis	
Memory Size Determine	5-167
Equipment Determination	5-167
Graphics Character Generator	5-171
Time of Day	5-172
Print Screen	5-175

Notes:



```

1 $TITLE(BIOS FOR THE IBM PERSONAL COMPUTER XT)
2
3
4
5 THE BIOS ROUTINES ARE MEANT TO BE ACCESSED THROUGH
6 SOFTWARE INTERRUPTS ONLY. ANY ADDRESSES PRESENT IN
7 THE LISTINGS ARE INCLUDED ONLY FOR COMPLETENESS.
8 NOT FOR REFERENCE. APPLICATIONS WHICH REFER TO
9 ABSOLUTE ADDRESSES WITHIN THE CODE SEGMENT
10 VIOLATE THE STRUCTURE AND DESIGN OF BIOS.
11
12
13
14 EQUATES
15
16 PORT_A EQU 60H ; 8255 PORT A ADDR
17 PORT_B EQU 61H ; 8255 PORT B ADDR
18 PORT_C EQU 62H ; 8255 PORT C ADDR
19 CMD_PORT EQU 63H
20 INTA00 EQU 20H ; 8259 PORT
21 INTA01 EQU 21H ; 8259 PORT
22 EOI 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 01 ; TIMER 0 INTR RECVD MASK
27 DMA08 EQU 08 ; DMA STATUS REG PORT ADDR
28 DMA EQU 00 ; 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 KBDINT EQU 02 ; KEYBOARD INTR MASK
33 KB_DATA EQU 60H ; KEYBOARD SCAN CODE PORT
34 KB_CTL EQU 61H ; CONTROL BITS FOR KEYBOARD SENSE DATA
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105

```

```

LOC OBJECT          LINE  SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82
0018 ??            106  KB_FLAG_1      DB      ?           ; SECOND BYTE OF KEYBOARD STATUS
107
0080              108  INS_SHIFT      EQU    80H          ; INSERT KEY IS DEPRESSED
0040              109  CAPS_SHIFT     EQU    40H          ; CAPS LOCK KEY IS DEPRESSED
0020              110  NUM_SHIFT     EQU    20H          ; NUM LOCK KEY IS DEPRESSED
0010              111  SCROLL_SHIFT  EQU    10H          ; SCROLL LOCK KEY IS DEPRESSED
0008              112  HOLD_STATE    EQU    08H          ; SUSPEND KEY HAS BEEN TOGGLED
113
0019 ???           114  ALT_INPUT     DB      ?           ; STORAGE FOR ALTERNATE KEYPAD ENTRY
001A ?????        115  BUFFER_HEAD   DW      ?           ; POINTER TO HEAD OF KEYBOARD BUFFER
001C ?????        116  BUFFER_TAIL   DW      ?           ; POINTER TO TAIL OF KEYBOARD BUFFER
001E 116          117  KB_BUFFER     DW      16 DUP(?)  ; ROOM FOR 15 ENTRIES
)
)

003E              118  KB_BUFFER_END LABEL  WORD
119
120  ;----- HEAD = TAIL INDICATES THAT THE BUFFER IS EMPTY
121
0045              122  NUM_KEY       EQU    69           ; SCAN CODE FOR NUMBER LOCK
0046              123  SCROLL_KEY    EQU    70           ; SCROLL LOCK KEY
0038              124  ALT_KEY      EQU    56           ; ALTERNATE SHIFT KEY SCAN CODE
001D              125  CTL_KEY      EQU    29           ; SCAN CODE FOR CONTROL KEY
003A              126  CAPS_KEY    EQU    58           ; SCAN CODE FOR SHIFT LOCK
002A              127  LEFT_KEY     EQU    42           ; SCAN CODE FOR LEFT SHIFT
0036              128  RIGHT_KEY    EQU    54           ; SCAN CODE FOR RIGHT SHIFT
0052              129  INS_KEY      EQU    82           ; SCAN CODE FOR INSERT KEY
0053              130  DEL_KEY      EQU    83           ; SCAN CODE FOR DELETE KEY
131
132  ;-----
133  ; DISKETTE DATA AREAS
134  ;-----
003E ??           135  SEEK_STATUS  DB      ?           ; DRIVE RECALIBRATION STATUS
136  ; BIT 3-0 = DRIVE 3-0 NEEDS RECAL
137  ; BEFORE NEXT SEEK IF BIT 15 = 0
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    DB      ?           ; ATTACHMENT FAILED TO RESPOND
0040              151  BAD_SEEK    EQU    40H          ; SEEK OPERATION FAILED
0020              152  BAD_NEC     EQU    20H          ; NEC CONTROLLER HAS FAILED
0010              153  BAD_CRC     EQU    10H          ; BAD CRC ON DISKETTE READ
0009              154  DMA_BOUNDARY EQU    09H          ; ATTEMPT TO DMA ACROSS 64K BOUNDARY
0008              155  BAD_DMA     EQU    08H          ; DMA OVERRUN ON OPERATION
0004              156  RECORD_NOT_FND EQU    04H          ; REQUESTED SECTOR NOT FOUND
0003              157  WRITE_PROTECT EQU    03H          ; WRITE ATTEMPTED ON WRITE PROT DISK
0002              158  BAD_ADDR_MARK EQU    02H          ; ADDRESS MARK NOT FOUND
0001              159  BAD_CMD     EQU    01H          ; BAD COMMAND PASSED TO DISKETTE I/O
160
0042 1?           161  NEC_STATUS  DB      7 DUP(?)   ; STATUS BYTES FROM NEC
??
)
)

162  ;-----
163  ; VIDEO DISPLAY DATA AREA
164  ;-----
0049 ??           165  ;-----
004A ?????        166  CRT_MODE     DB      ?           ; CURRENT CRT MODE
004C ?????        167  CRT_COLS    DW      ?           ; NUMBER OF COLUMNS ON SCREEN
004E ?????        168  CRT_LEN     DW      ?           ; LENGTH OF REGEN IN BYTES
0050 18          169  CRT_START   DW      ?           ; STARTING ADDRESS IN REGEN BUFFER
0050 18          170  CURSOR_POSN DW      8 DUP(?)   ; CURSOR FOR EACH OF UP TO 8 PAGES
)
)

0060 ?????        171  CURSOR_MODE  DW      ?           ; CURRENT CURSOR MODE SETTING
0062 ??           172  ACTIVE_PAGE DB      ?           ; CURRENT PAGE BEING DISPLAYED
0063 ?????        173  ADDR_6845   DW      ?           ; BASE ADDRESS FOR ACTIVE DISPLAY CARD
0065 ??           174  CRT_MODE_SET DB      ?           ; CURRENT SETTING OF THE 3X8 REGISTER
0066 ??           175  CRT_PALETTE DB      ?           ; CURRENT PALETTE SETTING COLOR CARD
176
177  ;-----
178  ; POST DATA AREA
179  ;-----
0067 ?????        180  IO_ROM_INIT DW      ?           ; PNTR TO OPTIONAL I/O ROM INIT ROUTINE
0069 ?????        181  IO_ROM_SEG  DW      ?           ; POINTER TO IO ROM SEGMENT
006B ??           182  INTR_FLAG   DB      ?           ; FLAG TO INDICATE AN INTERRUPT HAPPEND
183
184  ;-----
185  ; TIMER DATA AREA
186  ;-----
006C ?????        187  TIMER_LOW   DW      ?           ; LOW WORD OF TIMER COUNT
006E ?????        188  TIMER_HIGH  DW      ?           ; HIGH WORD OF TIMER COUNT
0070 ??           189  TIMER_OFL  DB      ?           ; TIMER HAS ROLLED OVER SINCE LAST READ
190  ; COUNTS_SEC EQU    18
191  ; COUNTS_MIN EQU    1092
192  ; COUNTS_HOUR EQU    65543
193  ; COUNTS_DAY EQU    1573040 = 1800B0H
194
195  ;-----
196  ; SYSTEM DATA AREA
197  ;-----
0071 ??           198  BIOS_BREAK  DB      ?           ; BIT 7=1 IF BREAK KEY HAS BEEN HIT
0072 ?????        199  RESET_FLAG  DW      ?           ; WORD=1234H IF KEYBOARD RESET UNDERWAY
200
201  ;-----
202  ; FIXED DISK DATA AREAS
203  ;-----
0074 ?????        203  ;-----
0076 ?????        204  ;-----
205  ;-----
206  ; PRINTER AND RS232 TIME-OUT VARIABLES
207  ;-----
0078 14          208  PRINT_TIM_OUT DB      4 DUP(?)
??
)
)

007C 14          209  RS232_TIM_OUT DB      4 DUP(?)
??
)
)

```

```

LOC OBJECT                LINE SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82
-----
210 ;-----
211 ; ADDITIONAL KEYBOARD DATA AREA ;
212 ;-----
213 BUFFER_START DW ?
0080 7777
214 BUFFER_END DW ?
0082 7777
215 DATA ENDS
216 ;-----
217 ; EXTRA DATA AREA ;
218 ;-----
219 XDATA SEGMENT AT 50H
0000 ??
220 STATUS_BYTE DB ?
221 XDATA ENDS
222 ;-----
223 ; VIDEO DISPLAY BUFFER ;
224 ;-----
225 VIDEO_RAM SEGMENT AT 0B800H
0000
226 REGEN LABEL BYTE
0000 16384
227 REGENW LABEL WORD
??
228 DB 16384 DUP(?)
-----
229 VIDEO_RAM ENDS
230 ;-----
231 ; ROM RESIDENT CODE ;
232 ;-----
233 CODE SEGMENT AT 0F000H
0000 (57344
??
234 DB 57344 DUP(?) ; FILL LOWEST 56K
-----
235
236 DB '1501512 COPR. IBM 1982' ; COPYRIGHT NOTICE
-----
237
238
239 ;-----
240 ; INITIAL RELIABILITY TESTS -- PHASE I ;
241 ;-----
242
243 ASSUME CS:CODE,SS:CODE,ES:ABS0,DS:DATA
244 ;-----
245 ; DATA DEFINITIONS ;
246 ;-----
247
248
249 C1 DW C11 ; RETURN ADDRESS
E016 D7E0
250 C2 DW C24 ; RETURN ADDRESS FOR DUMMY STACK
E018 7EE1
251
252 F3B DB ' KB OK',13 ; KB FOR MEMORY SIZE
E01A 204B42204F4B
E020 0D
-----
253
254 ;-----
255 ; LOAD A BLOCK OF TEST CODE THROUGH THE KEYBOARD PORT ;
256 ; FOR MANUFACTURING TEST. ;
257 ; THIS ROUTINE WILL LOAD A TEST (MAX LENGTH=FAFFH) THROUGH ;
258 ; THE KEYBOARD PORT. CODE WILL BE LOADED AT LOCATION ;
259 ; 0000:0500. AFTER LOADING, CONTROL WILL BE TRANSFERRED ;
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
E021
E021 E8131A
270 MOV BH,BL ; SAVE IT
E024 8AFB
271 CALL SP_TEST ; GET COUNT HI
E026 E80E1A
E028 8AEB
272 MOV CH,BL
E02B 8ACF
273 MOV CL,BH ; CX NOW HAS COUNT
E02D FC
274 CLD ; SET DIR. FLAG TO INCREMENT
E02E FA
275 CLI
E02F BF0005
276 MOV DI,0500H ; SET TARGET OFFSET (DS=0000)
E032 B0FD
277 MOV AL,0FDH ; UNMASK K/B INTERRUPT
E034 E621
278 OUT INTA01,AL ; SEND READ INT. REQUEST REG. CMD
E036 B09A
279 MOV AL,0AH ;
E038 E620
280 OUT INTA00,AL ;
E03A B8100
281 MOV DX,61H ; SET UP PORT B ADDRESS
E03D B8C4C
282 MOV BX,4CCCH ; CONTROL BITS FOR PORT B
E040 B402
283 MOV AH,02H ; K/B REQUEST PENDING MASK
E042
284
285 TST:
286 MOV AL,BL
E042 8AC3
287 OUT DX,AL ; TOGGLE K/B CLOCK
E044 EE
288 MOV AL,BH
E045 8AC7
289 OUT DX,AL
E047 EE
288 MOV AL,BH
E048 4A
289 DEC DX ; POINT DX AT ADDR. 60 (KB DATA)
E049
290
291 TST1:
292 IN AL,INTA00
E049 E420
293 AND AL,AH ; GET IRR REG
E04B 22C4
294 JZ TST1 ; KB REQUEST PENDING?
E04D 74FA
295 IN AL,DX ; LOOP TILL DATA PRESENT
E04F EC
296 STOSB ; GET DATA
E050 AA ; STORE IT
E051 42
297 INC DX ; POINT DX BACK AT PORT B (61)
E052 E2EE
298 LOOP TST ; LOOP TILL ALL BYTES READ
E054 EA00050000
299 JMP MFG_TEST_RTN ; FAR JUMP TO CODE THAT WAS JUST
300 ; LOADED
301

```



```

LOC OBJECT          LINE SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82
E18E                531 CLR_STG:
E18E 2BC0            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,0040H ; SET POINTER TO JUST>16KB
E199 BB1000         537 MOV BX,16 ; BASIC COUNT OF 16K
E19C                538 FILL_LOOP:
E19C 8EC2           539 MOV ES,DX ; SET SEG. REG. ---
E19E 2BFF          540 SUB DI,DI
E1A0 B855AA        541 MOV AX,0AA55H ; TEST PATTERN
E1A3 88C6          542 MOV CX,AX ; SAVE PATTERN
E1A5 268905        543 MOV ES:[DI],AX ; SEND PATTERN TO MEM.
E1A8 B00F          544 MOV AL,0FH ; PUT SOMETHING IN AL
E1AA 26B05         545 MOV AX,ES:[DI] ; GET PATTERN
E1AD 33C1          546 XOR AX,CX ; COMPARE PATTERNS
E1AF 7511          547 JNZ HOW_BIG.END ; GO END IF NO COMPARE
E1B1 B90020        548 MOV CX,2000H ; SET COUNT FOR 8K WORDS
E1B4 F3            549 REP STOSW ; FILL 8K WORDS
E1B5 AB
E1B6 81C20004      550 ADD DX,400H ; POINT TO NEXT 16KB BLOCK
E1BA 83C310        551 ADD BX,16 ; BUMP COUNT BY 16KB
E1BD 80FEA0        552 JNC DH,0A0H ; TOP OF RAM AREA YET? (A0000)
E1C0 75DA          553 JNZ FILL_LOOP
E1C2                554 HOW_BIG.END:
E1C2 891E1304      555 MOV DATA_WORD[OFFSET MEMORY_SIZE],BX ; SAVE MEMORY SIZE
E1C3                556
E1C3                557 ;-----SETUP STACK SEG AND SP
E1C3                558
E1C6 B83000        559 MOV AX,STACK ; GET STACK VALUE
E1C9 8ED0          560 MOV SS,AX ; SET THE STACK UP
E1CB BC0001        561 MOV SP,OFFSET TOS ; STACK IS READY TO GO
E1C3                562 ;-----INITIALIZE THE 8259 INTERRUPT CONTROLLER CHIP ;
E1C3                563
E1C3                564 ;-----
E1C3                565 C25: MOV AL,13H ; ICW1 - EDGE, SNGL, ICW4
E1D0 E620          566 OUT INTA0,AL ; SETUP ICW2 - INT TYPE 8 (8-F)
E1D2 B008          567 MOV AL,8 ;
E1D4 E621          568 OUT INTA0,AL ; SETUP ICW4 - BUFFRD,8086 MODE
E1D6 B009          569 MOV AL,9 ;
E1D8 E621          570 OUT INTA0,AL ; MASK ALL INTS. OFF
E1DA B0FF          571 MOV AL,0FFH ; (VIDEO ROUTINE ENABLES INTS.)
E1DC E621          572 OUT INTA0,AL ;
E1D3                573 ;-----SET UP THE INTERRUPT VECTORS TO TEMP INTERRUPT
E1D3                574
E1D3                575
E1DE IE            576 PUSH DS
E1DF B92000        577 MOV CX,32 ; FILL ALL 32 INTERRUPTS
E1E2 2BFF          578 SUB DI,DI ; FIRST INTERRUPT LOCATION
E1E4 8EC7          579 MOV ES,DI ; SET ES=0000 ALSO
E1E6 B823FF        580 MOV AX,OFFSET D11 ; MOVE ADDR OF INTR PROC TO TBL
E1E9 AB            581 STOSW
E1EA 8CC8          582 MOV AX,CS ; GET ADDR OF INTR PROC SEG
E1EC AB            583 STOSW
E1ED E2F7          584 LOOP D3 ; VECTBLO
E1E3                585
E1E3                586 ;-----ESTABLISH BIOS SUBROUTINE CALL INTERRUPT VECTORS
E1E3                587
E1E3                588 MOV DI,OFFSET VIDEO_INT ; SETUP ADDR TO INTR AREA
E1EF BF4000        589 PUSH CS
E1F2 0E            590 POP DS ; SETUP ADDR OF VECTOR TABLE
E1F3 1F            591 MOV AX,DS ; SET AX=SEGMENT
E1F4 8CD8          592 MOV SI,OFFSET VECTOR_TABLE+16 ; START WITH VIDEO ENTRY
E1F6 BE03FF90      593 MOV CX,16 ;
E1FA B91000        594 D3A: MOVSW ; MOVE VECTOR TABLE TO RAM
E1FD A5            595 INC DI ; SKIP SEGMENT POINTER
E1FE 47            596 INC DI ;
E1FF 47            597 LOOP D3A ;
E200 E2FB          598 ;-----DETERMINE CONFIGURATION AND MFG. MODE ;
E200                599
E200                600 ;-----
E200                601
E202 1F            602 POP DS
E203 1E            603 PUSH DS ; RECOVER DATA SEG
E204 E462          604 IN AL,PORT_C ; GET SWITCH INFO
E206 240F          605 AND AL,00001111B ; ISOLATE SWITCHES
E208 8AE0          606 MOV AH,AL ; SAVE
E20A B0AD          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 ; ROTATE TO HIGH NIBBLE
E211 B104          611 MOV CL,AL ; ISOLATE
E213 D2C0          612 ROL AL,CL ; COMBINE WITH OTHER BANK
E215 24F0          613 AND AL,11110000B ;
E217 8AC4          614 OR AL,AH ;
E219 24E4          615 SUB AH,AH ;
E21B A31004        616 MOV DATA_WORD[OFFSET EQUIP_FLAG],AX ; SAVE SWITCH INFO
E21E B099          617 MOV AL,99H ;
E220 E663          618 CMOV PORT,AL ;
E222 E80518        619 CALL KBD_RESET ; SEE IF MFG. JUMPER IN
E225 80FBAA        620 CMP BL,0AAH ; KEYBOARD PRESENT?
E228 7418          621 JE E6 ;
E22A 80FB65        622 CMP BL,065H ; LOAD MFG. TEST REQUEST?
E22D 7503          623 JNE D3B ; GO TO BOOTSTRAP IF 50
E22F E9EFD        624 JMP MFG_BOOT ;
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 ;
E23A 24FF          630 AND AL,0FFH ; WAS DATA LINE GROUNDED
E23C 7504          631 JNZ E6 ;
E23E FE061204      632 INC DATA_AREA[OFFSET MFG_TST] ; SET MANUFACTURING TEST FLAG
E23E                633

```

```

634 ;-----
635 ; INITIALIZE AND START CRT CONTROLLER (6845) ;
636 ; TEST VIDEO READ/WRITE STORAGE. ;
637 ; DESCRIPTION ;
638 ; RESET THE VIDEO ENABLE SIGNAL. ;
639 ; SELECT ALPHANUMERIC MODE, # 25, B & W. ;
640 ; READ/WRITE DATA PATTERNS TO STG. CHECK STG ;
641 ; ADDRESSABILITY. ;
642 ; ERROR = 1 LONG AND 2 SHORT BEEPS ;
643 ;-----
E242 E6:
E242 A11004 644 MOV AX,DATA_WORD[OFFSET EQUIP_FLAG] ; GET SENSE SWITCH INFO
E245 50 645 PUSH AX ; SAVE IT
E246 B030 647 MOV AL,30H ;
E248 A31004 648 MOV DATA_WORD[OFFSET EQUIP_FLAG],AX
E24B 2AE4 649 SUB AH,AH ;
E24D CD10 650 INT 10H ; SEND INIT TO B/W CARD
E24F B020 651 MOV AL,20H ;
E251 A31004 652 MOV DATA_WORD[OFFSET EQUIP_FLAG],AX
E254 2AE4 653 SUB AH,AH ; AND INIT COLOR CARD
E256 CD10 654 INT 10H ;
E258 58 655 POP AX ; RECOVER REAL SWITCH INFO
E259 A31004 656 MOV DATA_WORD[OFFSET EQUIP_FLAG],AX ; RESTORE IT
E25C 2430 658 AND AL,30H ; AND CONTINUE
E25E 750A 659 JNZ E7 ; ISOLATE VIDEO SWS
E260 BF4000 660 MOV DI,OFFSET VIDEO_INT ; SET INT 10H TO DUMMY
E263 C7054BFF 661 MOV [DI],OFFSET DUMMY_RETURN ; RETURN IF NO VIDEO CARD
E267 E9A000 662 JMP E10_1 ; BYPASS VIDEO TEST
E26A 663 E7: ; TEST VIDEO:
E26A 3C30 664 CMP AL,30H ; B/W CARD ATTACHED?
E26C 7408 665 JE EB ; YES - SET MODE FOR B/W CARD
E26E FEC4 666 INC AH ; SET COLOR MODE FOR COLOR CD
E270 3C20 667 CMP AL,20H ; 80X25 MODE SELECTED?
E272 7502 668 JNE EB ; NO - SET MODE FOR 40X25
E274 B403 669 MOV AH,3 ; SET MODE FOR 80X25
E276 86E0 670 EB ; SET MODE:
E278 50 671 PUSH AX ; SAVE VIDEO MODE ON STACK
E279 2AE4 672 SUB AH,AH ; INITIALIZE TO ALPHANUMERIC MD
E27B CD10 673 INT 10H ; CALL VIDEO IO
E27D 58 674 POP AX ; RESTORE VIDEO SENSE SWS IN AH
E27E 50 675 PUSH AX ; RESAVE VALUE
E27F BF00B0 676 MOV BX,0B000H ; BEG VIDEO RAM ADDR B/W CD
E282 BA803 677 MOV DX,38BH ; MODE REG FOR B/W
E285 B90008 678 MOV CX,2048 ; RAM WORD CNT FOR B/W CD
E288 B001 679 MOV AH,1 ; SET MODE FOR BW CARD
E28A 80FC30 680 CMP AH,30H ; B/W VIDEO CARD ATTACHED?
E28D 7409 681 JE E9 ; YES - GO TEST VIDEO STG
E28F B7B8 682 MOV BH,0B8H ; BEG VIDEO RAM ADDR COLOR CD
E291 BAD803 683 MOV DX,3D8H ; MODE REG FOR COLOR CD
E294 B520 684 MOV CH,20H ; RAM WORD CNT FOR COLOR CD
E296 FEC8 685 DEC AL ; SET MODE TO 0 FOR COLOR CD
E298 686 E9: ; TEST VIDEO STG:
E298 EE 687 OUT DX,AL ; DISABLE VIDEO FOR COLOR CD
E299 813E72043412 688 CMP DATA_WORD[OFFSET RESET_FLAG],1234H ; POD INIT BY KBD RESET?
E29F 8EC3 689 MOV ES,BX ; POINT ES TO VIDEO RAM STG
E2A1 7407 690 JE E10 ; YES - SKIP VIDEO RAM TEST
E2A3 8EDB 691 MOV DS,BX ; POINT DS TO VIDEO RAM STG
E2A5 E8C703 692 CALL ASSUME DS:NOTHING,ES:NOTHING
E2A8 7546 693 CALL STGTST_CNT ; GO TEST VIDEO R/W STG
694 JNE E17 ; R/W STG FAILURE - BEEP SKP
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 ;-----
E2AA E10:
E2AA 58 702 POP AX ; GET VIDEO SENSE SWS (AH)
E2AB 50 703 PUSH AX ; SAVE IT
E2AC B400 704 MOV AH,0 ; ENABLE VIDEO AND SET MODE
E2AE CD10 705 INT 10H ; VIDEO
E2B0 B82070 706 MOV AX,T020H ; WRT BLANKS IN REVERSE VIDEO
707
708
709 ;----- UNNATURAL ACT FOR ADDRESS COMPATIBILITY
710
E2B3 EB11 711 JMP SHORT E10A
E2C3 712 ORG 0E2C3H
E2C3 E99915 713 JMP NMI_INT
714
E2C6 E10A:
E2C6 2BFF 715 SUB D1,D1 ; SETUP STARTING LOC
E2C8 B92800 716 MOV CX,40 ; NO. OF BLANKS TO DISPLAY
E2CB F3 717 REP ; WRITE VIDEO STORAGE
E2CC AB 718
719 ;-----
720 ; CRT INTERFACE LINES TEST ;
721 ; DESCRIPTION ;
722 ; SENSE ON/OFF TRANSITION OF THE ;
723 ; VIDEO ENABLE AND HORIZONTAL ;
724 ; SYNC LINES. ;
725 ;-----
E2CD 58 726 POP AX ; GET VIDEO SENSE SW INFO
E2CE 50 727 PUSH AX ; SAVE IT
E2CF 80FC30 728 CMP AH,30H ; B/W CARD ATTACHED?
E2D2 BAB0A3 729 MOV DX,03BAH ; SETUP ADDR OF BW STATUS PORT
E2D5 7403 730 JE E11 ; YES - GO TEST LINES
E2D7 BADA03 731 MOV DX,03DAH ; COLOR CARD IS ATTACHED
E2DA B408 732 E11: MOV AH,8 ; LINE_TST;
E2DC 733 ;
E2DD 734 E12: SUB CX,CX ; OFLOOP_CNT;
E2DE 2BC9 735 SUB CX,CX ;
E2E0 736 E13: ;
E2DE EC 737 IN AL,DX ; READ CRT STATUS PORT
E2DF 22C4 738 AND AL,AH ; CHECK VIDEO/HORZ LINE
E2E1 7504 739 JNZ E14 ; ITS ON - CHECK IF IT GOES OFF
E2E3 E2F9 740 LOOP E13 ; LOOP TILL ON OR TIMEOUT
E2E5 EB09 741 JMP SHORT E17 ; GO PRINT ERROR MSG
E2E7 742 E14: SUB CX,CX ;
E2E7 2BC9 743 ;
E2E9 744 E15: ;
E2E9 EC 745 IN AL,DX ; READ CRT STATUS PORT
E2EA 22C4 746 AND AL,AH ; CHECK VIDEO/HORZ LINE
E2EC 747 JZ E16 ; ITS ON - CHECK NEXT LINE
E2EE E2F9 748 LOOP E15 ; LOOP IF OFF TILL IT GOES ON

```



```

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

```

```

LOC OBJECT          LINE SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82
940 ;-----
941 ; EXPANSION I/O BOX TEST ;
942 ; CHECK TO SEE IF EXPANSION BOX PRESENT - IF INSTALLED, ;
943 ; TEST DATA AND ADDRESS BUSES TO I/O BOX ;
944 ; ERROR='1801' ;
945 ;-----
946
947 ;----- DETERMINE IF BOX IS PRESENT
948
E418 949 EXP_10: ; (CARD WAS ENABLED EARLIER)
E418 BA1002 950 MOV DX,0210H ; CONTROL PORT ADDRESS
E418 B85555 951 MOV TEST DATA 5555H ; SET DATA PATTERN
E41E EE 952 OUT DX,AL
E41F B001 953 MOV AL,01H ; MAKE AL DIFFERENT
E421 EC 954 IN AL,DX ; RECOVER DATA
E422 3AC4 955 CMP AL,AH ; REPLY?
E424 7544 956 JNE E19 ; NO RESPONSE, GO TO NEXT TEST
E426 F7D0 957 NOT AX ; MAKE DATA=AAAA
E428 EE 958 OUT DX,AL
E429 B001 959 MOV AL,01H
E42B EC 960 IN AL,DX ; RECOVER DATA
E42C 3AC4 961 CMP AL,AH
E42E 753A 962 JNE E19
963
964 ;----- CHECK ADDRESS BUS
965
E430 966 EXP2:
E430 BB0100 967 MOV BX,0001H
E433 BA1502 968 MOV DX,0215H ; LOAD HI ADDR, REG ADDRESS
E436 B91000 969 MOV CX,0016 ; GO ACROSS 16 BITS
E439 970 EXP3:
E439 2E8807 971 MOV CS:[BX],AL ; WRITE ADDRESS F0000+BX
E43C 90 972 NOP
E43D EC 973 IN AL,DX ; READ ADDR. HIGH
E43E 3AC7 974 CMP AL,BH
E440 7521 975 JNE EXP_ERR ; GO ERROR IF MISCOMPARE
E442 42 976 INC DX ; DX=216H (ADDR. LOW REG)
E443 EC 977 IN AL,DX
E444 3AC3 978 CMP AL,BL ; COMPARE TO LOW ADDRESS
E446 751B 979 JNE EXP_ERR
E448 4A 980 DEC DX ; DX BACK TO 215H
E449 D1E3 981 SHL BX,1
E44B E2EC 982 LOOP EXP3 ; LOOP TILL '1' WALKS ACROSS BX
983
984 ;----- CHECK DATA BUS
985
E44D B90800 986 MOV CX,0008 ; DO 8 TIMES
E450 B001 987 MOV AL,01
E452 4A 988 DEC DX ; MAKE DX=214H (DATA BUS REG)
E453 989 EXP4:
E453 8AE0 990 MOV AH,AL
E455 EE 991 OUT DX,AL ; SEND VALUE TO REG
E456 B001 992 MOV AL,01H ; SAVE DATA BUS VALUE
E458 EC 993 IN AL,DX ; RETRIEVE VALUE FROM REG
E459 3AC4 994 CMP AL,AH ; = TO SAVED VALUE
E45B 7506 995 JNE SHORT EXP_ERR
E45D D0E0 996 SHL AL,1 ; FORM NEW DATA PATTERN
E45F E2F2 997 LOOP EXP4 ; LOOP TILL BIT WALKS ACROSS AL
E461 EB07 998 JMP SHORT E19 ; GO ON TO NEXT TEST
E463 999 EXP_ERR:
E463 BE0FF990 1000 MOV SI,OFFSET F3C
E467 E83F15 1001 CALL E_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
E46A 1010
E46A E8EC15 1011
E46D 1E 1012
E46E 1013
E46E 813E72003412 1014
E474 7503 1015
E476 E99F00 1016
E479 1017
E479 B81000 1018
E47C EB28 1019
E47E 1020
E47E 8B1E1300 1021
E482 83EB10 1022
E485 B104 1023
E487 D3EB 1024
E489 8BCB 1025
E48B BB0004 1026
E48E 1027
E48E 8EDB 1028
E490 8EC3 1029
E492 81C30004 1030
E496 52 1031
E497 51 1032
E498 53 1033
E499 50 1034
E49A B90020 1035
E49D E8CF01 1036
E4A0 754C 1037
E4A2 58 1038
E4A3 051000 1039
E4A6 1040
E4A6 50 1041
E4A7 B80A00 1042
E4AA B90300 1043
E4AD 1044
E4AD 33D2 1045
E4AF F7F3 1046
E4B1 80CA30 1047
E4B4 52 1048
E4B5 E2F6 1049
E4B7 B90300 1050
E4BA 1051
E4BA 58 1052
E4BB E8DE14 1053
E4BE E2FA 1054
E4C0 B90700 1055
E4C3 BE1AE0 1056
E4C6 1057
E4C6 2E8A04 1058
E4C9 46 1059
E4CA E8CF14 1060
E4CD E2F7 1061
E4CF 58 1062
E4D0 3D1000 1063
E4D3 74A9 1064
E4D5 5B 1065
E4D6 59 1066
E4D7 5A 1067
E4D8 E2B4 1068
E4DA B00A 1069
E4DC E8BD14 1070
1071
1072 ;----- DMA TC0 SHOULD BE ON BY NOW - SEE IF IT IS
1073
E4DF E408 1074
E4E1 2401 1075
E4E3 7533 1076
E4E5 1F 1077
E4E6 C06150003 1078
E4EB E966FE 1079
1080
1081 ;----- PRINT FAILING ADDRESS AND XOR'ED PATTERN IF DATA COMPARE ERROR
1082
E4EE 8AE8 1083
E4F0 B00D 1084
E4F2 E8A714 1085
E4F5 B00A 1086
E4F7 E8A214 1087
E4FA 58 1088
E4FB 83C406 1089
E4FE 8CDA 1090
E500 1F 1091
E501 1E 1092
E502 A31300 1093
1094
E505 88361500 1095
1096
E509 E8CE1A 1097
E50C 8AC5 1098
E50E E87A14 1099
E511 BE04F990 1100
E515 E89114 1101

```

```

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 ;-----
1108 ROM_SCAN:
E518      MOV     DX,0C800H          ; SET BEGINNING ADDRESS
E51B      ROM_SCAN !:
1110      MOV     DS,DX
1111      SUB     BX,BX          ; SET BX=0000
E51D      MOV     AX,[BX]      ; GET 1ST WORD FROM MODULE
E51F      PUSH  BX
E521      POP     BX          ; BUS SETTLING
E522      CMP     AX,0AA55H    ; = TO ID WORD?
E523      JNZ     NEXT_ROM    ; PROCEED TO NEXT ROM IF NOT
E524      CALL   ROM_CHECK   ; GO CHECK OUT MODULE
E525      JMP     ARE_WE_DONE  ; CHECK FOR END OF ROM SPACE
E52E      NEXT_ROM:
1120      ADD     DX,0080H      ; POINT TO NEXT 2K ADDRESS
E532      ARE_WE_DONE:
1122      CMP     DX,0F600H    ; AT F6000 YET?
E532      JNC     ARE_WE_DONE  ; GO CHECK ANOTHER ADD. IF NOT
E533      JMP     BASE_ROM_CHK ; GO CHECK BASIC ROM.
1126 ;-----
1127 ; A CHECKSUM IS DONE FOR THE 4 ROS MODULES CONTAINING BASIC CODE ;
1128 ;-----
E53B      BASE_ROM_CHK:
1129      MOV     AH,4          ; NO. OF ROS MODULES TO CHECK
E53B      E4:
1130      SUB     BX,BX          ; SETUP STARTING ROS ADDR
E53D      MOV     DS,DX
1133      ; CHECK ROS
E541      CALL   ROS_CHECKSUM ; CONTINUE IF OK
E544      JE     E545
E546      CALL   ROM_ERR      ; POINT ERROR
E549      E5:
1139      ADD     DX,0200H    ; POINT TO NEXT 8K MODULE
E54D      DEC     AH          ; ANY MORE TO DO?
E54F      JNZ     E54E        ; YES - CONTINUE
1142 ;-----
1143 ; DISKETTE ATTACHMENT TEST ;
1144 ; DESCRIPTION ;
1145 ; CHECK IF IPL DISKETTE DRIVE IS ATTACHED TO SYSTEM. IF ;
1146 ; ATTACHED, VERIFY STATUS OF NEC FDC AFTER A RESET. ISSUE ;
1147 ; A RECALL AND SEEK CMD TO FDC AND CHECK STATUS. COMPLETE ;
1148 ; SYSTEM INITIALIZATION THEN PASS CONTROL TO THE BOOT ;
1149 ; LOADER PROGRAM. ;
1150 ;-----
E551      F9:
1151      POP     DS
E551      MOV     AL,BYTE PTR EQUIP_FLAG ; DISKETTE PRESENT?
E552      AND     AL,01H        ; NO - BYPASS DISKETTE TEST
E555      JZ     F15
1155      F10:
E559      IN     AL,INTA01      ; DISK_TEST:
E559      AND     AL,0BFH
E55B      OUT    INTA01,AL      ; ENABLE DISKETTE INTERRUPTS
E55D      MOV     AH,0          ; RESET NEC FDC
E55F      MOV     DL,AH
E561      INT    13H           ; SET FOR DRIVE 0
E563      TEST   AH,OFFH      ; VERIFY STATUS AFTER RESET
E565      JNZ     F13          ; STATUS OK?
E568      JNZ     F13          ; NO - FDC FAILED
1165 ;-----
1166 ;----- TURN DRIVE 0 MOTOR ON
1167 ;-----
E56A      MOV     DX,03F2H      ; GET ADDR OF FDC CARD
E56D      MOV     AL,1CH        ; TURN MOTOR ON, EN DMA/INT
E56F      OUT    DX,AL         ; WRITE FDC CONTROL REG
E570      SUB     CX,CX
1172      F11:
E572      LOOP  F11            ; MOTOR WAIT:
E574      F12:
1174      LOOP  F12            ; WAIT FOR 1 SECOND
E574      XOR     DX,DX        ; MOTOR WAIT1:
E576      MOV     CH,1          ; SELECT DRIVE 0
E578      MOV     SEEK_STATUS,DL ; SELECT TRACK 1
E57A      CALL   SEEK
E57E      JC     F13           ; RECALIBRATE DISKETTE
E581      MOV     CH,34        ; GO TO ERR SUBROUTINE IF ERR
E583      CALL   SEEK_TRACK_34 ; SELECT TRACK 34
E585      MOV     CH,34        ; SEEK TO TRACK 34
E588      JNC     F14          ; OK, TURN MOTOR OFF
E58A      MOV     SI,OFFSET F3 ; DSK_ERR:
E58A      MOV     SI,OFFSET F3 ; GET ADDR OF MSG
E58E      CALL   E_MSG         ; GO PRINT ERROR MSG
1187 ;-----
1188 ;----- TURN DRIVE 0 MOTOR OFF
1189 ;-----
E591      F14:
1190      MOV     AL,0CH        ; DR0 OFF:
E591      MOV     DX,03F2H    ; TURN DRIVE 0 MOTOR OFF
E593      OUT    DX,AL         ; FDC CTL ADDRESS
E596      E4:
1192      OUT    DX,AL
1194 ;-----
1195 ;----- SETUP PRINTER AND R5232 BASE ADDRESSES IF DEVICE ATTACHED
1196 ;-----
E597      F15:
E597      MOV     INTR_FLAG,00H ; SET STRAY INTERRUPT FLAG = 00
E59C      MOV     SI,OFFSET KB_BUFFER ; SETUP KEYBOARD PARAMETERS
E59F      MOV     BUFFER_HEAD,SI
E5A3      MOV     BUFFER_TAIL,SI
E5A7      MOV     BUFFER_START,SI
E5AB      ADD     SI,32        ; DEFAULT BUFFER OF 32 BYTES
E5AE      MOV     BUFFER_END,SI
E5B2      MOV     DI,OFFSET PRINT_TIM_OUT ; SET DEFAULT PRINTER TIMEOUT
E5B5      PUSH  DS
E5B6      POP   ES
E5B7      MOV     AX,1414H     ; DEFAULT=20
E5BA      STOSW
E5BB      STOSW
E5BC      MOV     AX,0101H     ;R5232 DEFAULT=01
E5BF      STOSW
E5C0      MOV     AX,0101H
E5C1      IN     AL,INTA01
E5C3      AND     AL,0FH
E5C5      OUT    INTA01,AL      ; ENABLE TIMER AND KB INTS

```



```

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
                                           1218 ; (ERROR HAPPENED)
E5CA 7419                                 1219 JZ F15A_0 ; (ERROR HAPPENED)
E5CC BA0200                               1220 MOVB DX,2 ; CONTINUE IF NO ERROR
E5CF E80614                               1221 CALL ERR_BEEP ; 2 SHORT BEEPS (ERROR)
E5D2 BD9E890                             1222 MOV SI,OFFSET F3D ; LOAD ERROR MSG
E5D6 E8F113                               1223 CALL P_MSG
E5D9                                       1224 ERR_WAIT:
E5D9 B400                                 1225 MOV AH,00
E5DB CD16                                 1226 INT 16H ; WAIT FOR 'F1' KEY
E5DD 80FC3B                               1227 CMP AH,3BH
E5E0 75F7                                 1228 JNE ERR_WAIT
E5E2 EB0E90                               1229 JMP F15A ; BYPASS ERROR
E5E5                                       1230 F15A_0:
E5E5 803C120001                           1231 CMP MFC_TST,1 ; MFG MODE
E5EA 7406                                 1232 JE F15A ; BYPASS BEEP
E5EC BA0100                               1233 MOV DX,1 ; 1 SHORT BEEP (NO ERRORS)
E5EF EB0613                               1234 CALL ERR_BEEP
E5F2 A01000                               1235 F15A: MOV AL,BYTE PTR EQUIP_FLAG ; GET SWITCHES
E5F5 2401                                 1236 AND AL,00000001B ; 'LOOP POST' SWITCH ON
E5F7 7503                                 1237 JNZ F15B ; CONTINUE WITH BRING-UP
E5F9 E95FFA                               1238 JMP START
E5FC 2AE4                                 1239 F15B: SUB AH,AH
E5FE A04900                               1240 MOV AL,CRT_MODE
E601 CD10                                 1241 INT 10H ; CLEAR SCREEN
E603                                       1242 F15C:
E603 BDA3F990                             1243 MOV BP,OFFSET F4 ; PRT_SRC_TBL
E607 BE0000                               1244 MOV SI,0
E60A                                       1245 F16:
E60A 2E8B5600                             1246 MOV DX,CS:[BP] ; PRT_BASE1
E60E B0AA                                 1247 MOV AL,0AAH ; GET PRINTER BASE ADDR
E610 EE                                  1248 OUT DX,AL ; WRITE DATA TO PORT A
E611 1E                                  1249 DS 16 ; YES - STORE PRT BASE ADDR
E612 EC                                  1250 IN AL,DX ; BUS SETTLEING
E613 1F                                  1251 POP DS ; READ PORT A
E614 3CAA                                 1252 CMP AL,0AAH ; DATA PATTERN SAME
E616 7505                                 1253 JNE F17 ; NO - CHECK NEXT PRT CD
E618 895408                               1254 MOV PRINTER_BASE[SI],DX ; YES - STORE PRT BASE ADDR
E61B 46                                  1255 INC SI ; INCREMENT TO NEXT WORD
E61C 46                                  1256 INC SI
E61D                                       1257 F17:
E61D 45                                  1258 INC BP ; POINT TO NEXT BASE ADDR
E61E 45                                  1259 INC BP
E61F 81FDA9F9                             1260 CMP BP,OFFSET F4E ; ALL POSSIBLE ADDRS CHECKED?
E623 78E5                                 1261 JNE F16 ; PRT_BASE
E625 BB0000                               1262 MOV BX,0 ; POINTER TO RS232 TABLE
E628 BAF403                               1263 MOV DX,3FAH ; CHECK IF RS232 CD 1 ATTCH?
E62B EC                                  1264 IN AL,DX ; READ INTR ID REG
E62C 8BF8                                 1265 TEST AL,0F8H
E62E 7506                                 1266 JNZ F18
E630 C707F803                             1267 MOV RS232_BASE[BX],3F8H ; SETUP RS232 CD #1 ADDR
E634 42                                  1268 INC BX
E635 43                                  1269 INC BX
E636                                       1270 F18:
E636 BAF402                               1271 MOV DX,2FAH ; CHECK IF RS232 CD 2 ATTCH
E639 EC                                  1272 IN AL,DX ; READ INTERRUPT ID REG
E63A 8BF8                                 1273 TEST AL,0F8H
E63C 7506                                 1274 JNZ F19 ; BASE END
E63E C707FB02                             1275 MOV RS232_BASE[BX],2FBH ; SETUP RS232 CD #2
E642 43                                  1276 INC BX
E643 43                                  1277 INC BX
E644                                       1278
E644                                       1279 ;----- SET UP EQUIP FLAG TO INDICATE NUMBER OF PRINTERS AND RS232 CARDS
E644                                       1280
E644                                       1281 F19:
E644 8BC6                                 1282 MOV AX,SI ; BASE END:
E646 B103                                 1283 MOV CL,3 ; SI HAS 2* NUMBER OF RS232
E648 D2C8                                 1284 ROR AL,CL ; SHIFT COUNT
E64A 0AC3                                 1285 OR AL,BL ; ROTATE RIGHT 3 POSITIONS
E64C A21100                               1286 MOV BYTE PTR EQUIP_FLAG+1,AL ; STORE AS SECOND BYTE
E64F BA0102                               1287 MOV DX,201H
E652 EC                                  1288 IN AL,DX
E653 90                                  1289 NOP
E654 90                                  1290 NOP
E655 90                                  1291 NOP
E656 A80F                               1292 TEST AL,0FH
E658 7505                                 1293 JNZ F20 ; NO_GAME_CARD
E65A 800E110010                           1294 OR BYTE PTR EQUIP_FLAG+1,16 ; NO_GAME_CARD:
E65F                                       1295 F20:
E65F                                       1296
E65F                                       1297 ;----- ENABLE NMI INTERRUPTS
E65F                                       1298
E65F E461                                 1299 IN AL,PORT_B ; RESET CHECK ENABLES
E661 0C30                                 1300 OR AL,30H
E663 E661                                 1301 OUT PORT_B,AL
E665 24CF                                 1302 AND AL,0CFH
E667 E661                                 1303 OUT PORT_B,AL
E669 B080                                 1304 MOV AL,80H ; ENABLE NMI INTERRUPTS
E66B E6A0                                 1305 OUT 0A0H,AL
E66D                                       1306 F21:
E66D CD19                                 1307 INT 19H ; LOAD BOOT_STRAP:
                                           1308 ; GO TO THE BOOT LOADER

```



```

LOC OBJECT          LINE SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82
1408 ;--- INT 19 -----
1409 ; BOOT STRAP LOADER
1410 ; TRACK 0, SECTOR 1 IS READ INTO THE
1411 ; BOOT LOCATION (SEGMENT 0, OFFSET 7C00)
1412 ; AND CONTROL IS TRANSFERRED THERE.
1413 ;
1414 ; IF THERE IS A HARDWARE ERROR CONTROL IS
1415 ; TRANSFERRED TO THE ROM BASIC ENTRY POINT.
1416 ;-----
1417 ; ASSUME CS:CODE,DS:ABS0
1418 ; ORG 0E6F2H
1419
E6F2 1420 BOOT_STRAP PROC NEAR
E6F2 FB 1421 STI
E6F3 2B00 1422 SUB AX,AX ; ENABLE INTERRUPTS
E6F5 8E08 1423 MOV DS,AX ; ESTABLISH ADDRESSING
1424
1425 ;----- RESET THE DISK PARAMETER TABLE VECTOR
E6F7 C7067800CTEF 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 ; IPL SYSTEM
E704 51 1434 PUSH CX ; SAVE RETRY COUNT
E705 B400 1435 MOV AH,0 ; RESET THE DISKETTE SYSTEM
E707 CD13 1436 INT 13H ; DISKETTE_IO
E709 720F 1437 JC H2 ; IF ERROR, TRY AGAIN
E70B B80102 1438 MOV AX,201H ; READ IN THE SINGLE SECTOR
E70E 2B02 1439 SUB DX,DX ; TO THE BOOT LOCATION
E710 8E2C 1440 MOV ES,DX
E712 BB007C 1441 MOV BX,OFFSET BOOT_LOCN
1442 ; DRIVE 0, HEAD 0
E715 B90100 1443 MOV CX,1 ; SECTOR 1, TRACK 0
E718 CD13 1444 INT 13H ; DISKETTE_IO
E71A 1445
E71A 59 1446 H2: POP CX ; RECOVER RETRY COUNT
E71B 7304 1447 JNC H4 ; CF SET BY UNSUCCESSFUL READ
E71D E2E5 1448 LOOP H1 ; DO IT FOR RETRY TIMES
1449
1450 ;----- UNABLE TO IPL FROM THE DISKETTE
1451
E71F 1452 H3: INT 18H ; GO TO RESIDENT BASIC
E71F CD18 1453
1454
1455 ;----- IPL WAS SUCCESSFUL
1456
E721 1457 H4:
E721 EA007C0000 1458 JMP BOOT_LOCN
1459 BOOT_STRAP ENDP
1460

```

```

1461:-----INT 14-----
1462: RS232_10
1463: THIS ROUTINE PROVIDES BYTE STREAM I/O TO THE COMMUNICATIONS
1464: PORT ACCORDING TO THE PARAMETERS:
1465: (AH)=0 INITIALIZE THE COMMUNICATIONS PORT
1466: (AL) HAS PARAMETERS FOR INITIALIZATION
1467:
1468: 7 6 5 4 3 2 1 0
1469: ----- BAUD RATE -- -PARITY-- STOPB1 --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: 100 - 600
1474: 101 - 1200
1475: 110 - 2400
1476: 110 - 4800
1477: 111 - 9600
1478:
1479: ON RETURN, CONDITIONS SET AS IN CALL TO COMMO STATUS (AH=3)
1480: (AH)=1 SEND THE CHARACTER IN (AL) OVER THE COMMO LINE
1481: (AL) REGISTER IS PRESERVED
1482: ON EXIT, BIT 7 OF AH IS SET IF THE ROUTINE WAS UNABLE
1483: TO TRANSMIT THE BYTE OF DATA OVER THE LINE.
1484: IF BIT 7 OF AH IS NOT SET, THE REMAINDER OF AH
1485: IS SET AS IN A STATUS REQUEST, REFLECTING THE
1486: CURRENT STATUS OF THE LINE.
1487: (AH)=2 RECEIVE A CHARACTER IN (AL) FROM COMMO LINE BEFORE
1488: RETURNING TO CALLER
1489: ON EXIT, AH HAS THE CURRENT LINE STATUS, AS SET BY THE
1490: THE STATUS ROUTINE, EXCEPT THAT THE ONLY BITS
1491: LEFT ON ARE THE ERROR BITS (7,4,3,2,1)
1492: IF AH HAS BIT 7 ON (TIME OUT) THE REMAINING
1493: BITS ARE NOT PREDICTABLE,
1494: THUS, AH IS NON ZERO ONLY WHEN AN ERROR
1495: OCCURRED.
1496: (AH)=3 RETURN THE COMMO PORT STATUS IN (AX)
1497: AH CONTAINS THE LINE STATUS
1498: BIT 7 = TIME OUT
1499: BIT 6 = TRANS SHIFT REGISTER EMPTY
1500: BIT 5 = TRAN HOLDING REGISTER EMPTY
1501: BIT 4 = BREAK DETECT
1502: BIT 3 = FRAMING ERROR
1503: BIT 2 = PARITY ERROR
1504: BIT 1 = OVERRUN ERROR
1505: BIT 0 = DATA READY
1506: AL CONTAINS THE MODEM STATUS
1507: BIT 7 = RECEIVED LINE SIGNAL DETECT
1508: BIT 6 = RING INDICATOR
1509: BIT 5 = DATA SET READY
1510: BIT 4 = CLEAR TO SEND
1511: BIT 3 = DELTA RECEIVE LINE SIGNAL DETECT
1512: BIT 2 = TRAILING EDGE RING DETECTOR
1513: BIT 1 = DELTA DATA SET READY
1514: BIT 0 = DELTA CLEAR TO SEND
1515:
1516: (IDX) = 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 PARMS OF CALL
1524: ALL OTHERS UNCHANGED
1525:-----
1526: ASSUME CS:CODE,DS:DATA
1527: ORG 0E729H
1528: A1 LABEL WORD ; TABLE OF INIT VALUES
1529: DW 1047 ; 110 BAUD
1530: DW 768 ; 150
1531: DW 384 ; 300
1532: DW 192 ; 600
1533: DW 96 ; 1200
1534: DW 48 ; 2400
1535: DW 24 ; 4800
1536: DW 12 ; 9600
1537:
1538: E739 RS232_10 PROC FAR
1539:
1540: ;----- VECTOR TO APPROPRIATE ROUTINE
1541:
1542: E739 FB STI ; INTERRUPTS BACK ON
1543: E73A 1E PUSH DS ; SAVE SEGMENT
1544: E73B 52 PUSH DX
1545: E73C 56 PUSH SI
1546: E73D 57 PUSH DI
1547: E73E 51 PUSH CX
1548: E73F 53 PUSH BX
1549: E740 8BF2 MOV SI,DX ; RS232 VALUE TO SI
1550: E742 8BFA OR D1,DX
1551: E744 D1E6 SHL SI,1 ; WORD OFFSET
1552: E746 E81013 CALL DDS
1553: E7 49 8B14 MOV DX,RS232_BASE[SI] ; GET BASE ADDRESS
1554: E749 0B02 OR DX,DX ; TEST FOR 0 BASE ADDRESS
1555: E74D 7413 JZ A3 RETURN
1556: E74F 0AE4 OR AH,AH ; TEST FOR (AH)=0
1557: E751 7416 JZ A4 ; COMMUN INIT
1558: E753 FECC DEC AH ; TEST FOR (AH)=1
1559: E755 7445 JZ A5 ; SEND AL
1560: E757 FECC DEC AH ; TEST FOR (AH)=2
1561: E759 746A JZ A12 ; RECEIVE INTO AL
1562: E75B DEC AH ; TEST FOR (AH)=3
1563: E75D 7503 JNZ A3
1564: E75F E98300 JMP A18 ; COMMUNICATION STATUS
1565: E762 JZ A3 ; RETURN FROM RS232
1566: E762 5B POP BX
1567: E763 59 POP CX
1568: E764 5F POP DI
1569: E765 5E POP SI
1570: E766 5A POP DX
1571: E767 1F POP DS
1572: E768 CF IRET ; RETURN TO CALLER, NO ACTION
1573:
1574:

```

```

1575 ;----- INITIALIZE THE COMMUNICATIONS PORT
1576
1577
E769 E769 8AE0 1577 A4: MOV AH,AL ; SAVE INIT PARMS IN AH
E76B E76B 83C203 1578 ADD DX,3 ; POINT TO 8250 CONTROL REGISTER
E76E E76E B080 1580 MOV AL,80H
E770 E770 EE 1581 OUT DX,AL ; SET DLAB=1
1582
1583 ;----- DETERMINE BAUD RATE DIVISOR
1584
E771 E771 8AD4 1585 MOV DL,AH ; GET PARMS TO DL
E773 E773 B104 1586 MOV CL,4
E775 E775 D2C2 1587 ROL DL,CL
E777 E777 81E20E00 1588 AND DX,0EH ; ISOLATE THEM
E77B E77B BF29E7 1589 MOV DI,OFFSET A1 ; BASE OF TABLE
E77E E77E 03FA 1590 D1,DX ; PUT INTO INDEX REGISTER
E780 E780 8B14 1591 MOV DX,RS232_BASE[SI] ; POINT TO HIGH ORDER OF DIVISOR
E782 E782 42 1592 INC DX
E783 E783 2E8A4501 1593 MOV AL,CS:[DI]+1 ; GET HIGH ORDER OF DIVISOR
E787 E787 EE 1594 OUT DX,AL ; SET M5 OF DIV TO 0
E788 E788 4A 1595 DEC DX
E789 E789 2E8A05 1596 MOV AL,SC:[DI] ; GET LOW ORDER OF DIVISOR
E78C E78C EE 1597 OUT DX,AL ; SET L6 OF DIVISOR
E78D E78D 83C203 1598 ADD
E790 E790 8AC4 1599 MOV AL,AH ; GET PARMS BACK
E792 E792 241F 1600 AND AL,01FH ; STRIP OFF THE BAUD BITS
E794 E794 EE 1601 OUT DX,AL ; LINE CONTROL TO 8 BITS
E795 E795 4A 1602 DEC DX
E796 E796 4A 1603 DEC DX
E797 E797 B000 1604 MOV AL,0
E799 E799 EE 1605 OUT DX,AL ; INTERRUPT ENABLES ALL OFF
E79A E79A EB49 1606 JMP SHORT A18 ; COM_STATUS
1607
1608 ;----- SEND CHARACTER IN (AL) OVER COMMO LINE
1609
E79C E79C 50 1610 A5:
E79E E79E 83C204 1611 PUSH AX ; SAVE CHAR TO SEND
E7A0 E7A0 B003 1612 ADD DX,4 ; MODEM CONTROL REGISTER
E7A2 E7A2 EE 1613 MOV AL,3 ; DTR AND RTS
E7A4 E7A4 42 1614 OUT DX,AL ; DATA TERMINAL READY, REQUEST TO SEND
E7A5 E7A5 42 1615 INC DX ; MODEM STATUS REGISTER
E7A7 E7A7 B730 1616 INC DX
E7A8 E7A8 44 1617 MOV BH,20H ; DATA SET READY & CLEAR TO SEND
E7AA E7AA 7408 1618 CALL WAIT_FOR_STATUS ; ARE BOTH TRUE
E7AB E7AB 7408 1619 JE A9 ; YES, READY TO TRANSMIT CHAR
1620
E7AC E7AC 59 1621 A7:
E7AD E7AD 8AC1 1622 POP CX
E7AE E7AE 59 1623 MOV AL,CL ; RELOAD DATA BYTE
E7AF E7AF 80CC80 1624 A8: OR AH,80H ; INDICATE TIME OUT
E7B2 E7B2 EBAA 1625 JMP A3 ; RETURN
E7B4 E7B4 4A 1626 A9: DEC DX ; CLEAR TO SEND
E7B5 E7B5 B720 1627 MOV BH,20H ; LINE STATUS REGISTER
E7B7 E7B7 E83800 1628 CALL WAIT_FOR_STATUS ; WAIT SEND
E7B8 E7B8 75F0 1629 MOV AL,CL ; IS TRANSMITTER READY
E7BC E7BC 83EA05 1630 CALL WAIT_FOR_STATUS ; TEST FOR TRANSMITTER READY
E7BD E7BD 59 1631 JNZ A7 ; RETURN WITH TIME OUT SET
E7BE E7BE 59 1632 A11: SUB DX,5 ; OUT CHAR
E7BF E7BF 59 1633 POP CX ; DATA PORT
E7C0 E7C0 8AC1 1634 MOV AL,CL ; RECOVER IN CX TEMPORARILY
E7C2 E7C2 EE 1635 OUT DX,AL ; MOVE CHAR TO AL FOR OUT, STATUS IN AH
E7C3 E7C3 EB9D 1636 JMP A3 ; OUTPUT CHARACTER
1637 ; RETURN
1638
1639 ;----- RECEIVE CHARACTER FROM COMMO LINE
1640
E7C5 E7C5 83C204 1641 A12:
E7C6 E7C6 B001 1642 ADD DX,4 ; MODEM CONTROL REGISTER
E7CA E7CA EE 1643 OUT DX,AL ; DATA TERMINAL READY
E7CB E7CB 42 1644 INC DX
E7CC E7CC 42 1645 INC DX ; MODEM STATUS REGISTER
E7CD E7CD 42 1646 INC DX
1647
E7CD E7CD B720 1648 A13: MOV BH,20H
E7CF E7CF E82000 1649 CALL WAIT_FOR_STATUS ; WAIT_DSR
E7D2 E7D2 75D8 1650 JNZ A8 ; DATA SET READY
E7D4 E7D4 4A 1651 A15: DEC DX ; TEST FOR DSR
E7D5 E7D5 B701 1652 A16: MOV BH,1 ; RETURN WITH ERROR
E7D7 E7D7 E81800 1653 CALL WAIT_FOR_STATUS ; WAIT_DSR END
E7DA E7DA 75D3 1654 JNZ A8 ; LINE STATUS REGISTER
E7DC E7DC 80E41E 1655 A17: AND AH,0001110B ; WAIT RCVR
E7DD E7DD 80E41E 1656 MOV DX,RS232_BASE[SI] ; RECEIVE BUFFER FULL
E7DE E7DE 80E41E 1657 INC DX ; TEST FOR REC. BUFF. FULL
E7DF E7DF 8B14 1658 MOV AL,DX ; SET TIME OUT ERROR
E7E1 E7E1 EC 1659 IN AL,DX ; GET CHAR
E7E2 E7E2 910FF 1660 JMP A3 ; TEST FOR ERR CONDITIONS ON RCVR CHAR
1661 ; DATA PORT
1662 ; GET CHARACTER FROM LINE
1663 ; RETURN
1664
E7E5 E7E5 8B14 1665 A18: MOV DX,RS232_BASE[SI] ; CONTROL PORT
E7E6 E7E6 83C205 1666 ADD DX,5 ; GET LINE CONTROL STATUS
E7EA E7EA EC 1667 IN AL,DX ; GET LINE CONTROL STATUS
E7EB E7EB 8AE0 1668 MOV AH,AL ; PUT IN AH FOR RETURN
E7ED E7ED 42 1669 INC DX ; POINT TO MODEM STATUS REGISTER
E7EE E7EE EC 1670 IN AL,DX ; GET MODEM CONTROL STATUS
E7EF E7EF 970FF 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 1684 WAIT_FOR_STATUS PROC NEAR
E7F2 8A5D7C 1685 MOV BL,RS232_TIM_OUT[D1] ; LOAD OUTER LOOP COUNT
E7F5 1686 WFS0: SUB CX,CX
E7F5 2BC9 1687 WFS1:
E7F7 1688 IN AL,DX ; GET STATUS
E7F7 EC 1689 MOV AH,AL ; MOVE TO AH
E7F8 8AE0 1690 AND AL,BH ; ISOLATE BITS TO TEST
E7FA 22C7 1691 CMP AL,BH ; EXACTLY = TO MASK
E7FC 3AC7 1692 JE WFS_END ; RETURN WITH ZERO FLAG ON
E7FE 7408 1693 LDOP WFS1 ; TRY AGAIN
E800 E2F5 1694 DEC BL
E802 FECB 1695 JNZ WFS0
E804 75EF 1696
E806 0AFF 1697 OR BH,BH ; SET ZERO FLAG OFF
E808 1698 WFS_END:
E808 C3 1699 RET
1700 WAIT_FOR_STATUS ENDP
1702 RS232_ID ENDP
1703
E809 4552524F522E20 1704 F3D DB 'ERROR. (RESUME = F1 KEY)',13,10 ; ERROR PROMPT
2852455354045
203D2022463122
204B455929
E823 0D
E824 0A
1705

```

LOC OBJECT

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

```

1706 ;----- INT 16 -----
1707 ; KEYBOARD I/O
1708 ; THESE ROUTINES PROVIDE KEYBOARD SUPPORT
1709 ; INPUT
1710 ; (AH)=0 READ THE NEXT ASCII CHARACTER STRUCK FROM THE KEYBOARD
1711 ; RETURN THE RESULT IN (AL), SCAN CODE IN (AH)
1712 ; (AH)=1 SET THE Z FLAG TO INDICATE IF AN ASCII CHARACTER IS
1713 ; AVAILABLE TO BE READ.
1714 ; (ZF)=1 -- NO CODE AVAILABLE
1715 ; (ZF)=0 -- CODE IS AVAILABLE
1716 ; IF ZF = 0, THE NEXT CHARACTER IN THE BUFFER TO BE READ
1717 ; IS IN AX, AND THE ENTRY REMAINS IN THE BUFFER
1718 ; (AH)=2 RETURN THE CURRENT SHIFT STATUS IN AL REGISTER
1719 ; THE BIT SETTINGS FOR THIS CODE ARE INDICATED IN THE
1720 ; THE EQUATES FOR KB_FLAG
1721 ; OUTPUT
1722 ; AS NOTED ABOVE, ONLY AX AND FLAGS CHANGED
1723 ; ALL REGISTERS PRESERVED
1724 ;-----
1725 ; ASSUME CS:CODE,DS:DATA
1726 ; ORG 0E82EH
E82E ; KEYBOARD_IO PROC FAR
E82E ; STI
E82E ; INTERRUPTS BACK ON
E82F ; PUSH DS ; SAVE CURRENT DS
E830 ; PUSH BX ; SAVE BX TEMPORARILY
E831 ; CALL DDS
E834 ; OR AH,AH ; AH=0
E836 ; JZ K1 ; ASCII_READ
E838 ; DEC AH ; AH=1
E83A ; JZ K2 ; ASCII_STATUS
E83C ; DEC AH ; AH=2
E83E ; JZ K3 ; SHIFT_STATUS
E840 ; JMP SHORT INT10_END ; EXIT
1740 ;----- READ THE KEY TO FIGURE OUT WHAT TO DO
E842 ; K1:
E842 ; STI ; ASCII_READ
E843 ; NOP ; INTERRUPTS BACK ON DURING LOOP
E844 ; CLI ; ALLOW AN INTERRUPT TO OCCUR
E845 ; MOV BX,BUFFER_HEAD ; INTERRUPTS BACK OFF
E849 ; CMP BX,BUFFER_TAIL ; GET POINTER TO HEAD OF BUFFER
E84D ; JZ K1 ; TEST END OF BUFFER
E84F ; MOV AX,[BX] ; LOOP UNTIL SOMETHING IN BUFFER
E851 ; CALL K4 ; GET SCAN CODE AND ASCII CODE
E854 ; MOV BUFFER_HEAD,BX ; MOVE POINTER TO NEXT POSITION
E858 ; JMP SHORT INT10_END ; STORE VALUE IN VARIABLE
1753 ; RETURN
1754 ;----- ASCII STATUS
E85A ; K2:
E85A ; CLI ; INTERRUPTS OFF
E85B ; MOV BX,BUFFER_HEAD ; GET HEAD POINTER
E85F ; CMP BX,BUFFER_TAIL ; GET HEAD POINTER
E863 ; MOV AX,[BX] ; IF EQUAL (Z=1) THEN NOTHING THERE
E865 ; STI ; INTERRUPTS BACK ON
E866 ; POP BX ; RECOVER REGISTER
E867 ; POP DS ; RECOVER SEGMENT
E868 ; RET 2 ; THROW AWAY FLAGS
1765 ;----- SHIFT STATUS
E86B ; K3:
E86B ; MOV AL,KB_FLAG ; GET THE SHIFT STATUS FLAGS
E86E ; INT10_END
E86E ; POP BX ; RECOVER REGISTER
E86F ; POP DS ; RECOVER REGISTERS
E870 ; IRET ; RETURN TO CALLER
1774 ; KEYBOARD_IO ENDP
1775
1776 ;----- INCREMENT A BUFFER POINTER
1777
E871 ; K4
E871 ; PROC NEAR
E871 ; INC BX ; MOVE TO NEXT WORD IN LIST
E872 ; INC BX
E873 ; CMP BX,BUFFER_END ; AT END OF BUFFER?
E877 ; JNE K5 ; NO, CONTINUE
E879 ; MOV BX,BUFFER_START ; YES, RESET TO BUFFER BEGINNING
E87D ; K5:
E87D ; RET
E87D ; K4 ENDP
1787 ;----- TABLE OF SHIFT KEYS AND MASK VALUES
1788
E87E ; K6
E87E ; LABEL BYTE
E87F ; DB INS_KEY ; INSERT KEY
E880 ; DB CAPS_KEY,NUM_KEY,SCROLL_KEY,ALT_KEY,CTL_KEY
E881 ; 46
E882 ; 38
E883 ; 1D
E884 ; 2A
E885 ; 36
0008 ; 1793 DB LEFT_KEY,RIGHT_KEY
1794 ; K6L EQU $-K6
1795
1796 ;----- SHIFT_MASK_TABLE
1797
E886 ; K7
E886 ; LABEL BYTE
E887 ; DB INS_SHIFT ; INSERT MODE SHIFT
E888 ; DB CAPS_SHIFT,NUM_SHIFT,SCROLL_SHIFT,ALT_SHIFT,CTL_SHIFT
E889 ; 1D
E88A ; 08
E88B ; 04
E88C ; 02
E88D ; 01
1801 ; DB LEFT_SHIFT,RIGHT_SHIFT
1802
1803 ;----- SCAN CODE TABLES
1804
E88E ; K8
E88E ; DB 27,-1,0,-1,-1,-1,30,-1
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				
E896 FF				
E897 FF	1806		DB	-1,-1,-1,31,-1,127,-1,17
E898 FF				
E899 1F				
E89A FF				
E89B 7F				
E89C FF				
E89D 11				
E89E 17				
E89F 05	1807		DB	23,5,18,20,25,21,9,15
E8A0 12				
E8A1 14				
E8A2 19				
E8A3 15				
E8A4 09				
E8A5 0F				
E8A6 10	1808		DB	16,27,29,10,-1,1,19
E8A7 1B				
E8A8 1D				
E8A9 0A				
E8AA FF				
E8AB 01				
E8AC 13				
E8AD 04	1809		DB	4,6,7,8,10,11,12,-1,-1
E8AE 06				
E8AF 07				
E8B0 08				
E8B1 0A				
E8B2 0B				
E8B3 0C				
E8B4 FF				
E8B5 FF				
E8B6 FF	1810		DB	-1,-1,28,26,24,3,22,2
E8B7 FF				
E8B8 1C				
E8B9 1A				
E8BA 18				
E8BB 03				
E8BC 16				
E8BD 02				
E8BE 0E	1811		DB	14,13,-1,-1,-1,-1,-1
E8BF 0D				
E8C0 FF				
E8C1 FF				
E8C2 FF				
E8C3 FF				
E8C4 FF				
E8C5 FF				
E8C6 20	1812		DB	' ',-1
E8C7 FF				
E8C8	1813	1-----	CTL TABLE SCAN	
E8C8 5E	1814	K9	LABEL BYTE	
E8C9 5F	1815		DB	94,95,96,97,98,99,100,101
E8CA 60				
E8CB 61				
E8CC 62				
E8CD 63				
E8CE 64				
E8CF 65				
E8D0 66	1816		DB	102,103,-1,-1,119,-1,132,-1
E8D1 67				
E8D2 FF				
E8D3 FF				
E8D4 77				
E8D5 FF				
E8D6 84				
E8D7 FF				
E8D8 73	1817		DB	115,-1,116,-1,117,-1,118,-1
E8D9 FF				
E8DA 74				
E8DB FF				
E8DC 75				
E8DD FF				
E8DE 76				
E8DF FF				
E8E0 FF	1818		DB	-1
E8E1	1819	1-----	LC TABLE	
E8E1 1B	1820	K10	LABEL BYTE	
E8E2 31323334353637	1821		DB	01BH,'1234567890='',08H,09H
3839302D3D				
E8E2 08				
E8E3 09				
E8E4 0F				
E8E5 71776572747975	1822		DB	'qwertyuiop[]',0DH,-1,'asdfghjkl;',027H
696F705B5D				
E8E6 0D				
E8E7 FF				
E8E8 6173646667686A				
6B6C3B				
E908 21	1823		DB	60H,-1,5CH,'zxcvbnm,./',-1,'*',-1,' '
E909 60				
E90A FF				
E90B 5C				
E90C 7A786376626E6D				
2C2E2F				
E916 FF				
E917 2A				
E918 FF				
E919 20				
E91A FF	1824		DB	-1
E91B	1825	1-----	UC TABLE	
E91B 1B	1826	K11	LABEL BYTE	
E91C 21402324	1827		DB	27,'!@#',37,05EH,'&'()_+',08H,0
E920 25				
E921 5E				
E922 262A28295F2B				
E928 08				
E929 00				
E92A 51574552545955	1828		DB	'QWERTYUIOP[]',0DH,-1,'ASDFGHJKL:'
494F507B7D				
E936 0D				
E937 FF				
E938 4153444647484A				


```

LOC OBJECT          LINE  SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82
484C3A22
E943 7E
E944 FF
E945 7C5A584356424E
      4D3C3E3F
E950 FF
E951 00
E952 FF
E953 20
E954 FF

      1830  ;----- UC TABLE SCAN
      1831  K12 LABEL BYTE
      1832  DB 84,85,86,87,88,89,90

E955
E955 54
E956 55
E957 56
E958 57
E959 58
E95A 59
E95B 5A
E95C 5B
E95D 5C
E95E 5D

      1833  ;----- DB 91,92,93

E95F
      1834  ;----- ALT TABLE SCAN
      1835  K13 LABEL BYTE
      1836  DB 104,105,106,107,108

E960 69
E961 6A
E962 6B
E963 6C
E964 6D
E965 6E
E966 6F
E967 70
E968 71

      1837  ;----- DB 109,110,111,112,113

E969
E969 3738392D343536
      2B313233302E

      1838  ;----- NUM STATE TABLE
      1839  K14 LABEL BYTE
      1840  DB '789-456+123.0'

E976
E976 47
E977 48
E978 49
E979 FF
E97A 4B
E97B FF
E97C 4D
E97D FF
E97E 4F
E97F 50
E980 51
E981 52
E982 53

      1841  ;----- BASE CASE TABLE
      1842  K15 LABEL BYTE
      1843  DB 71,72,73,-1,75,-1,77

      1844  ;----- DB -1,79,80,81,82,83

      1845  ;----- KEYBOARD INTERRUPT ROUTINE
      1846  ;-----
      1847
      1848  ORG 0E987H
      1849  KB_INT PROC FAR
      1850  STI ; ALLOW FURTHER INTERRUPTS
      1851  PUSH AX
      1852  PUSH BX
      1853  PUSH CX
      1854  PUSH DX
      1855  PUSH SI
      1856  PUSH DI
      1857  PUSH DS
      1858  PUSH ES
      1859  CLD ; FORWARD DIRECTION
      1860  CALL AL,KB_DATA
      1861  IN AL,KB_CTL
      1862  PUSH AX
      1863  IN AL,KB_CTL
      1864  MOV AH,AL
      1865  OR AL,80H
      1866  OUT KB_CTL,AL
      1867  XCHG AH,AL
      1868  OUT KB_CTL,AL
      1869  POP AX
      1870  MOV AH,AL
      1871  ; GET BACK ORIGINAL CONTROL
      1872  ; KB HAS BEEN RESET
      1873  ; RECOVER SCAN CODE
      1874  ; SAVE SCAN CODE IN AH ALSO
      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  ;

E987
E987 FB
E988 50
E989 53
E98A 51
E98B 52
E98C 56
E98D 57
E98E 1E
E98F 06
E990 FC
E991 E8C510
E994 E460
E996 50
E997 E461
E999 8AE0
E99B 0C80
E99D E661
E99F 86E0
E9A1 E661
E9A3 58
E9A4 8AE0

      1845  ;----- TEST FOR OVERRUN SCAN CODE FROM KEYBOARD
      1874  CMP AL,OFFH ; IS THIS AN OVERRUN CHAR
      1875  JNZ K16 ; NO, TEST FOR SHIFT KEY
      1876  JMP K62 ; BUFFER_FULL_BEEP

      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  ;

E9A6 3CFF
E9A8 7503
E9AA E97A02
E877
E878
E879
E880 K16:
E881 AND AL,07FH ; TEST_SHIFT
E882 PUSH CS ; TURN OFF THE BREAK BIT
E883 POP ES ;
E884 MOV DI,OFFSET K6 ; ESTABLISH ADDRESS OF SHIFT TABLE
E885 MOV CX,K6L ; SHIFT KEY TABLE
E886 REPNE SCASB ; LENGTH
; LOOK THROUGH THE TABLE FOR A MATCH

E887 MOV AL,AH ; RECOVER SCAN CODE
E888 JE K17 ; JUMP IF MATCH FOUND
E889 JMP K25 ; IF NO MATCH, THEN SHIFT NOT FOUND
E890
E891 ;----- SHIFT KEY FOUND
E892
E893 K17: SUB DI,OFFSET K6+1 ; ADJUST PTR TO SCAN CODE MTC
E894 MOV AH,CS:K7[DI] ; GET MASK INTO AH
E895 TEST AL,80H ; TEST FOR BREAK KEY
E896 JNZ K23 ; BREAK_SHIFT_FOUND
E897
E898 ;----- SHIFT MAKE FOUND, DETERMINE SET OR TOGGLE
E899
E899 CMP AH,SCROLL_SHIFT ; IF SCROLL SHIFT OR ABOVE, TOGGLE KEY
E900 JAE K18
E901
E902
E903 ;----- PLAIN SHIFT KEY, SET SHIFT ON
E904
E905 OR KB_FLAG,AH ; TURN ON SHIFT BIT
E906 JMP K26 ; INTERRUPT_RETURN

E9C0 81EF7FE8
E9C4 2E8AA586E8
E9C9 A880
E9CB 7551

E9C0 80FC10
E900 7307

E9D2 08261700
E9D6 E98000

```

```

LOC OBJECT          LINE SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82
1907
1908 ;----- TOGGLED SHIFT KEY, TEST FOR 1ST MAKE OR NOT
1909
E909
E909 F606170004    1911 TEST    KB_FLAG,CTL_SHIFT    ; SHIFT-TOGGLE
E90E 7565         1912 JNZ     K25                   ; CHECK CTL_SHIFT STATE
E90D 3C52         1913 CMP     AL,INS_KEY            ; JUMP IF CTL STATE
E9E2 7522         1914 JNZ     K22                   ; CHECK FOR INSERT KEY
E9E4 F606170008    1915 TEST    KB_FLAG,ALT_SHIFT    ; JUMP IF NOT INSERT KEY
E9E9 755A         1916 JNZ     K25                   ; CHECK FOR ALTERNATE SHIFT
E9EB F606170020    1917 K19:  TEST    KB_FLAG,NUM_STATE ; CHECK FOR BASE STATE
E9F0 75D0         1918 JNZ     K21                   ; JUMP IF NUM LOCK IS ON
E9F2 F606170003    1919 TEST    KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT ; CHECK FOR NUM LOCK IS ON
E9F7 740D         1920 JZ      K22                   ; JUMP IF BASE STATE
1921
E9F9
E9F9 B83052        1922 K20:  MOV     AX,5230H           ; NUMERIC ZERO, NOT INSERT KEY
E9FC E9D601        1923 JMP     K57                   ; PUT OUT AN ASCII ZERO
E9FF
E9FF F606170003    1924 K21:  TEST    KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT ; BUFFER FILL
EA04 74F3         1925 JZ      K20                   ; MIGHT BE NUMERIC
1926
EA06
EA06 84261800      1927 K22:  TEST    AH,KB_FLAG_1        ; SHIFT TOGGLE KEY HIT; PROCESS IT
EA0A 754D         1928 JNZ     K26                   ; IS KEY ALREADY DEPRESSED
EA0C 08261800      1929 OR     KB_FLAG_1,AH          ; JUMP IF KEY ALREADY DEPRESSED
EA10 30261700      1930 XOR     KB_FLAG,AH           ; INDICATE THAT THE KEY IS DEPRESSED
EA14 3C52         1931 CMP     AL,INS_KEY            ; TOGGLE THE SHIFT STATE
EA16 7541         1932 JNE     K25                   ; TEST FOR 1ST MAKE OF INSERT KEY
EA18 B80052        1933 MOV     AX,INS_KEY*256        ; CHECK IF NOT INSERT KEY
EA1B E9B701        1934 JMP     K57                   ; SET SCAN CODE INTO AH, 0 INTO AL
1935
1936
1937
1938
1939 ;----- BREAK SHIFT FOUND
1940
EA1E
EA1E 80FC10        1941 K23:  CMP     AH,SCROLL_SHIFT      ; BREAK-SHIFT-FOUND
EA21 731A         1942 JAE     K24                   ; IS THIS A TOGGLE KEY
EA23 F6D4         1943 NOT    AH                     ; YES, HANDLE BREAK TOGGLE
EA25 20261700      1944 AND    KB_FLAG,AH           ; INVERT MASK
EA29 3CB8         1945 CMP     AL,ALT_KEY+80H       ; TURN OFF SHIFT BIT
EA2B 752C         1946 JNE     K24                   ; IS THIS ALTERNATE SHIFT RELEASE
1947
1948 ;----- ALTERNATE SHIFT KEY RELEASED, GET THE VALUE INTO BUFFER
1949
EA2D A01900        1950
EA2D A01900        1951 MOV     AL,ALT_INPUT          ; ALTERNATE SHIFT KEY RELEASED, GET THE VALUE INTO BUFFER
EA30 B400          1952 MOV     AH,0                 ; SCAN CODE OF 0
EA32 83261900      1953 MOV     AL,INPUT,AH          ; ZERO OUT THE FIELD
EA36 3C00          1954 CMP     AL,0                 ; WAS THE INPUT=0
EA38 741F         1955 JE      K26                   ; INTERRUPT_RETURN
EA3A E9A101        1956 JMP     K58                   ; IT WASN'T, SO PUT IN BUFFER
EA3D
EA3D F6D4          1957 K24:  NOT    AH                     ; BREAK-TOGGLE
EA3F 20261800      1958 AND    KB_FLAG_1,AH          ; INVERT MASK
EA43 EB14          1959 JMP     SHORT K26             ; INDICATE NO LONGER DEPRESSED
1960
1961
1962 ;----- TEST FOR HOLD STATE
1963
EA45
EA45 3C80          1964 K25:  CMP     AL,80H               ; NO-SHIFT-FOUND
EA47 7310          1965 JAE     K26                   ; TEST FOR BREAK KEY
EA49 F606180008    1966 TEST    KB_FLAG_1,HOLD_STATE ; NOTHING FOR BREAK CHARS FROM HERE ON
EA4E 7417          1967 JZ      K28                   ; ARE WE IN HOLD STATE
EA50 3C45          1968 CMP     AL,NUM_KEY           ; BRANCH AROUND TEST IF NOT
EA52 7405          1969 JE      K26                   ; CAN'T END HOLD ON NUM_LOCK
EA54 80261800F7    1970 AND    KB_FLAG_1,NOT_HOLD_STATE ; TURN OFF THE HOLD STATE BIT
EA59
EA59 FA           1971 K26:  CLI     INT0                 ; INTERRUPT_RETURN
EA5A B020          1972 MOV     AL,E01               ; TURN OFF INTERRUPTS
EA5C E620          1973 OUT    020H,AL              ; END OF INTERRUPT COMMAND
EA5E 07            1974
EA5F 1F           1975 K27:  INT    020H                 ; SEND COMMAND TO INT CONTROL PORT
EA60 5F           1976
EA61 5E           1977
EA62 5A           1978 POP     ES                   ; INTERRUPT_RETURN=NO-E01
EA63 59           1979 POP     DS
EA64 5B           1980 POP     DI
EA65 58           1981 POP     DX
EA66 CF           1982 POP     CX
1983 POP     BX
1984 POP     AX
1985 IRET
1986
1987
1988 ;----- NOT IN HOLD STATE, TEST FOR SPECIAL CHARS
1989
EA67
EA67 F606170008    1990 K28:  TEST    KB_FLAG,ALT_SHIFT    ; NO-HOLD-STATE
EA6C 7503         1991 JNZ     K29                   ; ARE WE IN ALTERNATE SHIFT
EA6E E99100        1992 JMP     K38                   ; JUMP IF ALTERNATE SHIFT
1993
1994
1995 ;----- TEST FOR RESET KEY SEQUENCE (CTL ALT DEL)
1996
EA71
EA71 F606170004    1997 K29:  TEST    KB_FLAG,CTL_SHIFT    ; TEST-RESET
EA76 7433         1998 JZ      K31                   ; ARE WE IN CONTROL SHIFT ALSO
EA78 3C53         1999 CMP     AL,DEL_KEY           ; NO RESET
EA7A 752F         2000 JNE     K31                   ; SHIFT STATE IS THERE, TEST KEY
2001
2002
2003 ;----- CTL-ALT-DEL HAS BEEN FOUND, DO I/O CLEANUP
2004
EA7C C70672003412  2005 MOV     RESET_FLAG,1234H     ; SET FLAG FOR RESET FUNCTION
EA82 EA5BE000F0     2006 JMP     RESET                 ; JUMP TO POWER ON DIAGNOSTICS
2007
EA87
EA87 52            2008 ;----- ALT-INPUT-TABLE
EA88 4F           2009 LABEL BYTE
EA89 50           2010 DB     82,19,80,81,75,76,77
EA8A 51
EA8B 4B
EA8C 4C
EA8D 4D
EA8E 47
EA8F 48
EA90 49
2011
EA91 10
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 12
EA94 13
EA95 14
EA96 15
EA97 16
EA98 17
EA99 18
EA9A 19
EA9B 1E
EA9C 1F
EA9D 20
EA9E 21
EA9F 22
EAA0 23
EAA1 24
EAA2 25
EAA3 26
EAA4 2C
EAA5 2D
EAA6 2E
EAA7 2F
EAA8 30
EAA9 31
EAAA 32

2014 DB 24,25,30,31,32,33,34,35

2015 DB 36,37,38,44,45,46,47,48

2016 DB 49,50

2017
2018 ;----- IN ALTERNATE SHIFT, RESET NOT FOUND
2019
EAAB K31:
EAAB 3C39 ; NO-RESET
EAAD 7505 ; TEST FOR SPACE KEY
EAFB B020 ; NOT THERE
EAB1 E92101 ; SET SPACE CHAR
; BUFFER_FILL

2025 ;----- LOOK FOR KEY PAD ENTRY
2026
EA84 K32:
EA84 BF87EA ; ALT-KEY-PAD
EAB7 B90A00 ; ALT-INPUT-TABLE
EABA F2 ; LOOK FOR ENTRY USING KEYPAD
EABR AE ; LOOK FOR MATCH
EABC 7512 ; NO ALT KEYPAD
EABE 81EF88EA ; DI 'NOW HAS ENTRY VALUE
EAC2 A01900 ; GET THE CURRENT BYTE
EACS B40A ; MULTIPLY BY 10
EAC7 F6E4 ; MUL AH
EAC9 03C7 ; ADD AX,DI
EACB A21900 ; MOV ALT_INPUT,AL
EACE EB89 ; JMP K26
; THROW AWAY THAT KEYSTROKE
2040
;----- LOOK FOR SUPERSHIFT ENTRY
2041
EAD0 K33:
EAD0 C606190000 ; NO-ALT-KEYPAD
EAD5 B91A00 ; MOV ALT_INPUT,0
EAD8 F2 ; DI,ES ALREADY POINTING
EAD9 AE ; LOOK FOR MATCH IN ALPHABET
EADA 7505 ; NOT FOUND, FUNCTION KEY OR OTHER
EADC B000 ; ASCII CODE OF ZERO
EADE E9F400 ; PUT IT IN THE BUFFER
2050
;----- LOOK FOR TOP ROW OF ALTERNATE SHIFT
2051
EAE1 K34:
EAE1 3C02 ; ALT-TOP-ROW
EAE3 720C ; CMP AL,2
EAE5 3C0E ; JB K35 ; KEY WITH '1' ON IT
EAE7 7308 ; CMP AL,14 ; NOT ONE OF INTERESTING KEYS
EAE9 80C476 ; JAE K35 ; IS IT IN THE REGION
EAEF B000 ; ADD AH,118 ; ALT-FUNCTION
EAE8 E9E400 ; MOV AL,0 ; CONVERT PSEUDO SCAN CODE TO RANGE
EAE9 80C476 ; JMP K57 ; INDICATE AS SUCH
EAE8 E9E400 ; BUFFER_FILL
2061
;----- TRANSLATE ALTERNATE SHIFT PSEUDO SCAN CODES
2062
EAF1 K35:
EAF1 3C3B ; ALT-FUNCTION
EAF3 7303 ; CMP AL,59 ; TEST FOR IN TABLE
EAF5 ; JAE K37 ; ALT-CONTINUE
EAF5 E961FF ; JMP K26 ; CLOSE-RETURN
EAF8 ; K37: ; IGNORE THE KEY
EAF8 3C47 ; CMP AL,71 ; ALT-CONTINUE
EAFB 73F9 ; JAE K36 ; IN KEYPAD REGION
EAFB 73F9 ; MOV BX,OFFSET K13 ; IF SO, IGNORE
EAFD B95F9 ; JMP K63 ; ALT SHIFT PSEUDO SCAN TABLE
EAFD E91B01 ; TRANSLATE THAT
2074
;----- NOT IN ALTERNATE SHIFT
2075
EB02 K38:
EB02 F606170004 ; TEST KB_FLAG,CTL_SHIFT ; ARE WE IN CONTROL SHIFT
EB07 7458 ; JZ K44 ; NOT-CTL-SHIFT
2080
;----- CONTROL SHIFT, TEST SPECIAL CHARACTERS
2081
;----- TEST FOR BREAK AND PAUSE KEYS
2082
EB09 3C46 ; CMP AL,SCROLL_KEY ; TEST FOR BREAK
EB0B 7518 ; MOV K39 ; NO-BREAK
EB0D BB1E8000 ; MOV BX,BUFFER_START ; RESET BUFFER TO EMPTY
EB11 891E1A00 ; MOV JNE BUFFER_HEAD,BX
EB15 891E1C00 ; MOV JNE BUFFER_TAIL,BX
EB19 C606110080 ; INT BIOS_BREAK,80H ; TURN ON BIOS BREAK BIT
EB1E CD1B ; INT IBH ; BREAK INTERRUPT VECTOR
EB20 28C0 ; SUB AX,AX ; PUT OUT DUMMY CHARACTER
EB22 E9B000 ; JMP K57 ; BUFFER_FILL
EB25 ; K39: ; NO-BREAK
EB25 3C45 ; CMP AL,NUM_KEY ; LOOK FOR PAUSE KEY
EB27 7521 ; JNE K41 ; NO-PAUSE
EB29 800E180080 ; OR KB_FLAG_1,HOLD_STATE ; TURN ON THE HOLD FLAG
EB2E B020 ; MOV AL,1 ; END OF INTERRUPT TO CONTROL PORT
EB30 E620 ; OUT 020H,AL ; ALLOW FURTHER KEYSTROKE INTS
2099
;----- DURING PAUSE INTERVAL, TURN CRT BACK ON
2100
EB32 803E490007 ; CMP CRT_MODE,7 ; IS THIS BLACK AND WHITE CARD
EB37 7407 ; JE K40 ; YES, NOTHING TO DO
EB39 B40803 ; MOV DX,03DBH ; PORT FOR COLOR CARD
EB3C A06500 ; MOV AL,CRT_MODE_SET ; GET THE VALUE OF THE CURRENT MODE
EB3F EE ; OUT DX,AL ; SET THE CRT MODE, SO THAT CRT IS ON

```

```

LOC OBJECT          LINE SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82
EB40                2107 K40:                ; PAUSE-LOOP
EB40 F606180008    2108                TEST   KB_FLAG_1,HOLD_STATE
EB45 75F9           2109                JNZ   K40                ; LOOP UNTIL FLAG TURNED OFF
EB47 E914FF        2110                JMP   K27                ; INTERRUPT_RETURN_NO_EOI
EB4A                2111 K41:                ; NO-PAUSE
                2112
                2113 ;----- TEST SPECIAL CASE KEY 55
                2114
EB4A 3C37           2115                CMP   AL,55
EB4C 7506           2116                JNE   K42                ; NOT-KEY-55
EB4E B80072        2117                MOV   AX,114*256        ; START/STOP PRINTING SWITCH
EB51 E98100        2118                JMP   K57                ; BUFFER_FILL
                2119
                2120 ;----- SET UP TO TRANSLATE CONTROL SHIFT
                2121
EB54                2122 K42:                ; NOT-KEY-55
EB54 B8EE8         2123                MOV   BX,OFFSET K8     ; SET UP TO TRANSLATE CTL
EB57 3C3B          2124                CMP   AL,59            ; IS IT IN TABLE
                2125                ; CTL-TABLE-TRANSLATE
EB59 7276           2126                JB    K56                ; YES, GO TRANSLATE CHAR
EB5B                2127 K43:                ; CTL-TABLE-TRANSLATE
EB5B B8CE8         2128                MOV   BX,OFFSET K9     ; CTL TABLE SCAN
EB5E E9BC00        2129                JMP   K63                ; TRANSLATE_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 732C          2135                JAE   K48                ; HANDLE KEYPAD REGION
EB65 F606170003    2136                TEST  KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT
EB6A 745A          2137                JZ    K54                ; TEST FOR SHIFT STATE
                2138
                2139 ;----- UPPER CASE, HANDLE SPECIAL CASES
                2140
EB6C 3C0F          2141                CMP   AL,15            ; BACK TAB KEY
EB6E 7505          2142                JNE   K45                ; NOT-BACK-TAB
EB70 B8000F        2143                MOV   AX,15*256        ; SET PSEUDO SCAN CODE
EB73 EB06          2144                JMP   SHORT K57         ; BUFFER_FILL
EB75                2145 K45:                ; NOT-BACK-TAB
EB75 3C37           2146                CMP   AL,55            ; PRINT SCREEN KEY
EB77 7509          2147                JNE   K46                ; NOT-PRINT-SCREEN
                2148
                2149 ;----- 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   $H                ; ISSUE PRINT SCREEN INTERRUPT
EB7F E9DCFE        2154                JMP   K27                ; GO BACK WITHOUT EOI OCCURRING
EB82                2155 K46:                ; NOT-PRINT-SCREEN
EB82 3C3B          2156                CMP   AL,59            ; FUNCTION KEYS
EB84 7206           2157                JB    K47                ; NOT-UPPER-FUNCTION
EB86 B85E9         2158                MOV   BX,OFFSET K12    ; UPPER CASE PSEUDO SCAN CODES
EB89 E99100        2159                JMP   K63                ; TRANSLATE_SCAN
EB8C                2160 K47:                ; NOT-UPPER-FUNCTION
EB8C B81BE9        2161                MOV   BX,OFFSET K11    ; POINT TO UPPER CASE TABLE
EB8F EB40           2162                JMP   SHORT K56         ; OK, TRANSLATE THE CHAR
                2163
                2164 ;----- KEYPAD KEYS, MUST TEST NUM LOCK FOR DETERMINATION
                2165
EB91                2166 K48:                ; KEYPAD-REGION
EB91 F606170020    2167                TEST  KB_FLAG,NUM_STATE ; ARE WE IN NUM_LOCK
EB96 7520           2168                JNZ   K52                ; TEST FOR SURE
EB98 F606170003    2169                TEST  KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT ; ARE WE IN SHIFT STATE
EB9D 7520           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 740B           2176                JE    K50                ; MINUS
EBA3 3C4E          2177                CMP   AL,78            ;
EBA5 740C          2178                JE    K51                ;
EBA7 2C47          2179                SUB   AL,71            ; CONVERT ORIGIN
EBA9 B876E9        2180                MOV   BX,OFFSET K15    ; BASE CASE TABLE
EBAE EB71           2181                JMP   SHORT K64         ; CONVERT TO PSEUDO SCAN
EBAE                2182 K50:                ;
EBAE B82D4A        2183                MOV   AX,74*256+''''   ; MINUS
EBB1 EB22           2184                JMP   SHORT K57         ; BUFFER_FILL
EBB3                2185 K51:                ;
EBB3 B82B4E        2186                MOV   AX,78*256+''''   ; PLUS
EBB6 EB1D           2187                JMP   SHORT K57         ; BUFFER_FILL
                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
EBBD 75E0           2193                JNZ   K49                ; SHIFTED TEMP OUT OF NUM STATE
EBBF                2194 K53:                ; REALLY-NUM STATE
EBBF 2C46           2195                SUB   AL,70            ; CONVERT ORIGIN
EBC1 B869E9        2196                MOV   BX,OFFSET K14    ; NUM STATE TABLE
EBC4 EB0B           2197                JMP   SHORT K56         ; TRANSLATE_CHAR
                2198
                2199 ;----- PLAIN OLD LOWER CASE
                2200
EBC6                2201 K54:                ; NOT-SHIFT
EBC6 3C3B          2202                CMP   AL,59            ; TEST FOR FUNCTION KEYS
EBC8 7204           2203                JB    K55                ; NOT-LOWER-FUNCTION
EBCA B000           2204                MOV   AL,0              ; SCAN CODE IN AH ALREADY
EBCD EB07           2205                JMP   SHORT K57         ; BUFFER_FILL
EBCD                2206 K55:                ; NOT-LOWER-FUNCTION
EBCD BBE1E8        2207                MOV   BX,OFFSET K10    ; LC TABLE
                2208
                2209 ;----- TRANSLATE THE CHARACTER
                2210
EBD1                2211 K56:                ; TRANSLATE-CHAR
EBD1 FEC8          2212                DEC   AL                ; CONVERT ORIGIN
EBD3 2ED7          2213                XLAT  CS:K11            ; CONVERT THE SCAN CODE TO ASCII
                2214
                2215 ;----- PUT CHARACTER INTO BUFFER
                2216
EBD5                2217 K57:                ; BUFFER-FILL
EBD5 3CFF          2218                CMP   AL,-1            ; IS THIS AN IGNORE CHAR
EBD7 741F          2219                JE    K59                ; YES, DO NOTHING WITH IT
EBD9 80FCFF        2220                CMP   AH,-1            ; LOOK FOR -1 PSEUDO SCAN
EBDC 741A          2221                JE    K59                ; NEAR_INTERRUPT_RETURN
                2222

```

```

LOC OBJECT                LINE SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82
2223 ;----- HANDLE THE CAPS LOCK PROBLEM
2224
EBDE F606170040          K58:                ; BUFFER-FILL-NOTEST
EBDE 7420                2226     TEST    KB_FLAG,CAPS_STATE ; ARE WE IN CAPS LOCK STATE
2227     JZ      K6T          ; SKIP IF NOT
2228
;----- IN CAPS LOCK STATE
2229
EBE5 F606170003          TEST    KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT ; TEST FOR SHIFT STATE
EBEA 740F                JZ      K60          ; IF NOT SHIFT, CONVERT LOWER TO UPPER
2230
2231 ;----- CONVERT ANY UPPER CASE TO LOWER CASE
2232
EBEC 3C41                CMP     AL,'A'        ; FIND OUT IF ALPHABETIC
EBEE 7215                JB      K61          ; NOT_CAPS_STATE
EBF0 3C5A                CMP     AL,'Z'        ;
EBF2 7711                JA      K61          ; NOT_CAPS_STATE
EBF4 0420                ADD     AL,'A'-'A'    ; CONVERT TO LOWER CASE
EBF6 EB0D                JMP     SHORT K61     ; NOT_CAPS_STATE
EBF8                    K59:                2242     JMP     K26          ; NEAR-INTERRUPT-RETURN
EBF8 E95EFE                2243     JMP     K26          ; INTERRUPT_RETURN
2244
;----- CONVERT ANY LOWER CASE TO UPPER CASE
2245
EBFB                    K60:                2246     ; LOWER-TO-UPPER
EBFB 3C61                CMP     AL,'a'        ; FIND OUT IF ALPHABETIC
EBFD 7206                JB      K61          ; NOT_CAPS_STATE
EBFF 3C7A                CMP     AL,'z'        ;
EC01 7702                JA      K61          ; NOT_CAPS_STATE
EC03 2C20                SUB     AL,'a'-'A'    ; CONVERT TO UPPER CASE
EC05                    K61:                2253     ; NOT_CAPS_STATE
EC05 8B1E1C00            2254     MOV     BX,BUFFER_TAIL ; GET THE END POINTER TO THE BUFFER
EC09 8BF3                2255     MOV     SI,BX         ; SAVE THE VALUE
EC0B EB63FC                2256     CALL   K4           ; ADVANCE THE TAIL
EC0E 3B1E1A00            2257     CMP     BX,BUFFER_HEAD ; HAS THE BUFFER WRAPPED AROUND
EC12 7413                JE      K62          ; BUFFER_FULL_BEEP
EC14 8904                2259     MOV     [SI],AX       ; STORE THE VALUE
EC16 891E1C00            2260     MOV     BUFFER_TAIL,BX ; MOVE THE POINTER UP
EC1A E93CFE                2261     JMP     K26          ; INTERRUPT_RETURN
2262
;----- TRANSLATE SCAN FOR PSEUDO SCAN CODES
2263
EC1D                    K63:                2265     ; TRANSLATE-SCAN
EC1D 2C3B                K64:                2267     SUB     AL,59         ; CONVERT ORIGIN TO FUNCTION KEYS
EC1F                    K64:                2268     ; TRANSLATE-SCAN-ORG
EC1F 2ED7                XLAT   C5:K9         ; CTL TABLE SCAN
EC21 8AE0                2269     MOV     AH,AL        ; PUT VALUE INTO AH
EC23 B000                2270     MOV     AL,0         ; ZERO ASCII CODE
EC25 EBAE                2271     JMP     K5T          ; PUT IT INTO THE BUFFER
2272
KB_INT ENDP
2273
;----- BUFFER IS FULL, SOUND THE BEEPER
2274
EC27                    K62:                2276     ; BUFFER-FULL-BEEP
EC27 B020                2277     ; END OF INTERRUPT COMMAND
EC29 E620                2279     OUT    20H,AL       ; SEND COMMAND TO INT CONTROL PORT
EC2B BB8000            2280     MOV     BX,080H     ; NUMBER OF CYCLES FOR 1/12 SECOND TONE
EC2E E461                2281     IN     AL,KB_CTL    ; GET CONTROL INFORMATION
EC30 50                2282     PUSH  AX           ; SAVE
EC31                    K65:                2283     ; BEEP-CYCLE
EC31 24FC                2284     AND    AL,0FCH     ; TURN OFF TIMER GATE AND SPEAKER DATA
EC33 E661                2285     OUT    KB_CTL,AL    ; OUTPUT TO CONTROL
EC35 B94800            2286     MOV     CX,48H     ; HALF CYCLE TIME FOR TONE
EC38                    K66:                2287     ; SPEAKER OFF
EC38 E2FE                2288     LOOP  K66         ;
EC3A 0C02                2289     OR     AL,2        ; TURN ON SPEAKER BIT
EC3C E661                2290     OUT    KB_CTL,AL    ; OUTPUT TO CONTROL
EC3E B94800            2291     MOV     CX,48H     ; SET UP COUNT
EC41                    K67:                2292     ; ANOTHER HALF CYCLE
EC41 E2FE                2293     LOOP  K67         ;
EC43 4B                2294     DEC   BX           ; TOTAL TIME COUNT
EC44 75EB                2295     JNZ   K65         ; DO ANOTHER CYCLE
EC46 58                2296     POP   AX           ; RECOVER CONTROL
EC47 E661                2297     OUT    KB_CTL,AL    ; OUTPUT THE CONTROL
EC49 E912FE                2298     JMP   K2T          ;
2299
EC4C 20333031          2300 F1    DB      ' 301',13,10 ; KEYBOARD ERROR
EC50 0D
EC51 0A
EC52 363031          2301 F3    DB      '601',13,10 ; DISKETTE ERROR
EC55 0D
EC56 0A

```

```

2302
2303
2304 ; DISKETTE 1/0
2305 ; THIS INTERFACE PROVIDES ACCESS TO THE 5 1/4 DISKETTE DRIVES
2306 ; INPUT
2307 (AH)=0 RESET DISKETTE SYSTEM
2308 ; HARD RESET TO NEC, PREPARE COMMAND, RECAL REQUIRED
2309 ; ON ALL DRIVES
2310 (AH)=1 READ THE STATUS OF THE SYSTEM INTO (AL)
2311 ; DISKETTE_STATUS FROM LAST OPERATION IS USED
2312 ;
2313 ; REGISTERS FOR READ/WRITE/VERIFY/FORMAT
2314 ; (DL) - DRIVE NUMBER (0-3 ALLOWED, VALUE CHECKED)
2315 ; (DH) - HEAD NUMBER (0-1 ALLOWED, NOT VALUE CHECKED)
2316 ; (CH) - TRACK NUMBER (0-39, NOT VALUE CHECKED)
2317 ; (CL) - SECTOR NUMBER (1-8, NOT VALUE CHECKED,
2318 ; NOT USED FOR FORMAT)
2319 ; (AL) - NUMBER OF SECTORS ( MAX = 8, NOT VALUE CHECKED, NOT USED )
2320 ;
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 ; (C,H,R,N), WHERE C = TRACK NUMBER, H=HEAD NUMBER,
2331 ; R = SECTOR NUMBER, N= NUMBER OF BYTES PER SECTOR
2332 ; (00=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 ;
2340 ; OUTPUT
2341 ; AH = STATUS OF OPERATION
2342 ; STATUS BITS ARE DEFINED IN THE EQUATES FOR
2343 ; DISKETTE_STATUS VARIABLE IN THE DATA SEGMENT OF THIS
2344 ; MODULE.
2345 ;
2346 ; CY = 0 SUCCESSFUL OPERATION (AH=0 ON RETURN)
2347 ; CY = 1 FAILED OPERATION (AH HAS ERROR REASON)
2348 ; FOR READ/WRITE/VERIFY
2349 ; DS,BX,DX,CH,CL PRESERVED
2350 ; AL = NUMBER OF SECTORS ACTUALLY READ
2351 ; ***** AL MAY NOT BE CORRECT IF TIME OUT ERROR OCCURS
2352 ; NOTE: IF AN ERROR IS REPORTED BY THE DISKETTE CODE, THE
2353 ; APPROPRIATE ACTION IS TO RESET THE DISKETTE, THEN RETRY
2354 ; THE OPERATION, ON READ ACCESSES, NO MOTOR START DELAY
2355 ; IS TAKEN, SO THAT THREE RETRIES ARE REQUIRED ON READS
2356 ; TO ENSURE THAT THE PROBLEM IS NOT DUE TO MOTOR
2357 ; START-UP.
-----
2357 ASSUME CS:CODE,DS:DATA,ES:DATA
2358 ORG 0EC59H
2359 DISKETTE_10 PROC FAR
2360 STI ; INTERRUPTS BACK ON
2361 PUSH BX ; SAVE ADDRESS
2362 PUSH CX
2363 PUSH DS ; SAVE SEGMENT REGISTER VALUE
2364 PUSH SI ; SAVE ALL REGISTERS DURING OPERATION
2365 PUSH DI
2366 PUSH BP
2367 PUSH DX
2368 MOV BP,SP ; SET UP POINTER TO HEAD PARM
2369 CALL DDS
2370 CALL JI ; CALL THE REST TO ENSURE DS RESTORED
2371 MOV BX,4 ; GET THE MOTOR WAIT PARAMETER
2372 CALL GET_PARM
2373 MOV MOTOR_COUNT,AH ; SET THE TIMER COUNT FOR THE MOTOR
2374 MOV AH,DISKETTE_STATUS ; GET STATUS OF OPERATION
2375 CMP AH,1 ; SET THE CARRY FLAG TO INDICATE
2376 CMC ; SUCCESS OR FAILURE
2377 POP DX ; RESTORE ALL REGISTERS
2378 POP BP
2379 POP DI
2380 POP SI
2381 POP DS
2382 POP CX
2383 POP BX ; RECOVER ADDRESS
2384 RET 2 ; THROW AWAY SAVED FLAGS
2385 DISKETTE_10 ENDP
2386
2387 J1 PROC NEAR
2388 MOV DH,AL ; SAVE # SECTORS IN DH
2389 AND MOTOR_STATUS,07FH ; INDICATE A READ OPERATION
2390 OR AH,AH ; AH=0
2391 JZ DISK_RESET
2392 DEC AH ; AH=1
2393 JZ DISK_STATUS ; DISK STATUS
2394 MOV DISKETTE_STATUS,0 ; RESET THE STATUS INDICATOR
2395 CMP DL,4 ; TEST FOR DRIVE IN 0-3 RANGE
2396 JAE J3 ; ERROR IF ABOVE
2397 DEC AH ; AH=2
2398 JZ DISK_READ
2399 DEC AH ; AH=3
2400 JNZ J2 ; TEST_DISK_VERF
2401 JMP DISK_WRITE ; TEST_DISK_WRITE
2402 J2: ; TEST_DISK_VERF
2403 DEC AH ; AH=4
2404 JZ DISK_VERF
2405 DEC AH ; AH=5
2406 JZ DISK_FORMAT
2407 J3: ; BAD COMMAND
2408 MOV DISKETTE_STATUS,BAD_CMD ; ERROR CODE, NO SECTORS TRANSFERRED
2409 RET ; UNDEFINED OPERATION
2410 J1 ENDP
2411

```

```

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

2412 ;----- RESET THE DISKETTE SYSTEM
2413
ECB7                                     2414 DISK_RESET PROC NEAR
ECB7 BAF203                             2415 MOV DX,03F2H ; ADAPTER CONTROL PORT
ECB8 FA                                 2416 CLI ; NO INTERRUPTS
ECBB A03F00                             2417 MOV AL,MOTOR_STATUS ; WHICH MOTOR IS ON
ECBE B104                             2418 MOV CL,4 ; SHIFT COUNT
ECC0 D2E0                             2419 SAL AL,CL ; MOVE MOTOR VALUE TO HIGH NYBBLE
ECC2 A820                             2420 TEST AL,20H ; SELECT CORRESPONDING DRIVE
ECC4 750C                             2421 JG J5 ; JUMP IF MOTOR ONE IS ON
ECC6 A840                             2422 TEST AL,40H
ECC8 7506                             2423 JNZ J4 ; JUMP IF MOTOR TWO IS ON
ECCA A880                             2424 TEST AL,80H
ECCC 7406                             2425 JZ J6 ; JUMP IF MOTOR ZERO IS ON
ECCE FEC0                             2426 INC AL
ECDD                                     2427 J4:
ECDE FEC0                             2428 J5: INC AL
ECDF                                     2429 J5: INC AL
EDA                                     2430
EDA 0C08                             2431 J6:
EDC6 EE                                 2432 OR AL,8 ; TURN ON INTERRUPT ENABLE
EDC7 C6063E0000                       2433 OUT DX,AL ; RESET THE ADAPTER
EDCC C606410000                       2434 MOV MOV SEEK_STATUS,0 ; SET RECAL REQUIRED ON ALL DRIVES
ECE1 0C04                             2435 MOV DISKETTE_STATUS,0 ; SET OK STATUS FOR DISKETTE
ECE3 EE                                 2436 OR AL,4 ; TURN OFF RESET
ECE4 FB                                 2437 OUT DX,AL ; TURN OFF THE RESET
ECES E82A02                             2438 STI ; REENABLE THE INTERRUPTS
ECF5                                     2439 CALL CHK_STAT_2 ; DD SENSE INTERRUPT STATUS
ECF5 A04200                             2440 MOV AL,NEC_STATUS ; FOLLOWING RESET
ECF8 3CC0                             2441 CMP AL,0C0H ; IGNORE ERROR RETURN AND DO OWN TEST
ECED 7406                             2442 JZ J7 ; TEST FOR DRIVE READY TRANSITION
ECF4 C3                                 2443 OR DISKETTE_STATUS,BAD_NEG ; SET ERROR CODE
2444 RET
2445
2446 ;----- SEND SPECIFY COMMAND TO NEC
2447
ECF5                                     2448
ECF5 B403                             2449 J7: MOV AH,03H ; DRIVE READY
ECF7 E84701                             2450 CALL NEC_OUTPUT ; SPECIFY COMMAND
ECFA BB0100                             2451 MOV BX,1 ; OUTPUT THE COMMAND
ECFD E86C01                             2452 CALL GET_PARAM ; FIRST BYTE PARAM IN BLOCK
ED00 BB0300                             2453 ; TO THE NEC CONTROLLER
ED03 E86601                             2454 MOV BX,3 ; SECOND BYTE PARAM IN BLOCK
ED06 C3                                 2455 CALL GET_PARAM ; TO THE NEC CONTROLLER
ED06 C3                                 2456 RET ; RESET RET
2457 ; RETURN TO CALLER
2458 DISK_RESET ENDP
2459
2460 ;----- DISKETTE STATUS ROUTINE
2461
ED07                                     2462 DISK_STATUS PROC NEAR
ED07 A04100                             2463 MOV AL,DISKETTE_STATUS
ED0A C3                                 2464 RET
2465 DISK_STATUS ENDP
2466
2467 ;----- DISKETTE READ
2468
ED0B                                     2469 DISK_READ PROC NEAR
ED0B B046                             2470 MOV AL,046H ; READ COMMAND FOR DMA
ED0D                                     2471 J9: ; DISK READ CONT
ED0D E8B801                             2472 CALL DMA_SETUP ; SET UP THE DMA
ED10 B4E6                             2473 MOV AH,0E6H ; SET UP RD COMMAND FOR NEC CONTROLLER
ED12 E836                             2474 JMP SHORT RW_OPN ; GO DO THE OPERATION
2475 DISK_READ ENDP
2476
2477 ;----- DISKETTE VERIFY
2478
ED14                                     2479 DISK_VERIFY PROC NEAR
ED14 B042                             2480 MOV AL,042H ; VERIFY COMMAND FOR DMA
ED16 E8F5                             2481 JMP J9 ; DO AS IF DISK READ
2482 DISK_VERIFY ENDP
2483
2484 ;----- DISKETTE FORMAT
2485
ED18                                     2486 DISK_FORMAT PROC NEAR
ED18 800E3F0080                       2487 OR MOTOR_STATUS,80H ; INDICATE WRITE OPERATION
ED1D B04A                             2488 MOV AL,04AH ; WILL WRITE TO THE DISKETTE
ED1F E8A601                             2489 CALL DMA_SETUP ; SET UP THE DMA
ED22 B44D                             2490 MOV AH,04DH ; ESTABLISH THE FORMAT COMMAND
ED24 E824                             2491 JMP SHORT RW_OPN ; DO THE OPERATION
ED26                                     2492 J10: ; CONTINUATION OF RW_OPN FOR FMT
ED26 BB0700                             2493 MOV BX,7 ; GET THE
ED29 E84001                             2494 CALL GET_PARAM ; BYTES/SECTOR VALUE TO NEC
ED2C BB0900                             2495 MOV BX,9 ; GET THE
ED2F E83A01                             2496 CALL GET_PARAM ; SECTORS/TRACK VALUE TO NEC
ED32 BB0F00                             2497 MOV BX,15 ; GET THE
ED35 E83401                             2498 CALL GET_PARAM ; GAP LENGTH VALUE TO NEC
ED38 BB1100                             2499 MOV BX,17 ; GET THE FILLER BYTE
ED3B E9AB00                             2500 JMP J16 ; TO THE CONTROLLER
2501 DISK_FORMAT ENDP
2502
2503 ;----- DISKETTE WRITE ROUTINE
2504
ED3E                                     2505 DISK_WRITE PROC NEAR
ED3E 800E3F0080                       2506 OR MOTOR_STATUS,80H ; INDICATE WRITE OPERATION
ED43 B04A                             2507 MOV AL,04AH ; DMA WRITE COMMAND
ED45 E88001                             2508 CALL DMA_SETUP ; SET UP THE DMA
ED48 B4C5                             2509 MOV AH,0C5H ; NEC COMMAND TO WRITE TO DISKETTE
2510 DISK_WRITE ENDP
2511
2512 ;----- ALLOW WRITE ROUTINE TO FALL INTO RW_OPN
2513

```

LOC OBJECT

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

```

2514 :-----
2515 : RW_OPN
2516 : THIS ROUTINE PERFORMS THE READ/WRITE/VERIFY OPERATION
2517 :-----
ED4A 2518 RW_OPN PROC NEAR
ED4A 7308 2519 JNC J11 ; TEST FOR DMA ERROR
ED4C C606410009 2520 MOV DISKETTE_STATUS,DMA_BOUNDARY ; SET ERROR
ED51 B000 2521 MOV AL,0 ; NO SECTORS TRANSFERRED
ED53 C3 2522 RET ; RETURN TO MAIN ROUTINE
ED54 2523 J11: ; DO RW_OPN
ED54 50 2524 PUSH AX ; SAVE THE COMMAND
2525
2526 :----- TURN ON THE MOTOR AND SELECT THE DRIVE
2527
ED55 51 2528 PUSH CX ; SAVE THE T/S PARMS
ED56 8ACA 2529 MOV CL,DL ; GET DRIVE NUMBER AS SHIFT COUNT
ED58 B001 2530 MOV AL,I ; MASK FOR DETERMINING MOTOR BIT
ED5A D2E0 2531 SHL AL,CL ; SHIFT THE MASK BIT
ED5C FA 2532 CLI ; NO INTERRUPTS WHILE DETERMINING
2533 ; MOTOR STATUS
ED5D C6064000FF 2534 MOV MOTOR_COUNT,OFFH ; SET LARGE COUNT DURING OPERATION
ED5E 84063F00 2535 TEST AL,MOTOR_STATUS ; TEST THAT MOTOR FOR OPERATING
ED66 7531 2536 JNZ J14 ; IF RUNNING, SKIP THE WAIT
ED68 80263F00F0 2537 AND MOTOR_STATUS,0F0H ; TURN OFF ALL MOTOR BITS
ED6D 08063F00 2538 OR MOTOR_STATUS,AL ; TURN ON THE CURRENT MOTOR
ED71 FB 2539 STI ; INTERRUPTS BACK ON
ED72 B010 2540 MOV AL,10H ; MASK BIT
ED74 D2E0 2541 SAL AL,CL ; DEVELOP BIT MASK FOR MOTOR ENABLE
ED76 0AC2 2542 OR AL,DL ; GET DRIVE SELECT BITS IN
ED78 0C0C 2543 OR AL,0CH ; NO RESET, ENABLE DMA/INT
ED7A 52 2544 PUSH DX ; SAVE REG
ED7B BAF203 2545 MOV DX,03F2H ; CONTROL PORT ADDRESS
ED7E EE 2546 OUT DX,AL ;
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
ED87 BB1400 2553 MOV BX,20 ; GET THE MOTOR WAIT
ED8A E8D100 2554 CALL GET_PARM ; PARAMETER
ED8D 0AE4 2555 OR AH,AH ; TEST FOR NO WAIT
ED8F 2556 J12: ; TEST WAIT TIME
ED8F 7408 2557 JZ J14 ; EXIT WITH TIME EXPIRED
ED91 2BC9 2558 SUB CX,CX ; SET UP 1/8 SECOND LOOP TIME
ED93 2559 J13: ; WAIT FOR THE REQUIRED TIME
ED93 E2FE 2560 LOOP J13 ; DECREMENT TIME VALUE
ED95 FECC 2561 DEC AH ; ARE WE DONE YET
ED97 EBF6 2562 JMP J12 ; MOTOR RUNNING
ED99 2563 J14: ; INTERRUPTS BACK ON FOR BYPASS WAIT
ED99 FB 2564 STI
ED9A 59 2565 POP CX
2566
2567 :----- DO THE SEEK OPERATION
2568
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
EDA1 B600 2572 MOV DH,0 ; SET NO SECTORS READ IN CASE OF ERROR
EDA3 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 MOV SI,SI ; SO THAT IT WILL RETURN TO MOTOR OFF
2576 ; LOCATION
2577
2578 :----- SEND OUT THE PARAMETERS TO THE CONTROLLER
2579
EDAA E89400 2580 CALL NEC_OUTPUT ; OUTPUT THE OPERATION COMMAND
EDAD 8A6401 2581 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 ;
EDB4 80E404 2584 AND AH,4 ; ISOLATE THAT BIT
EDB7 0AE2 2585 OR AH,DL ; OR IN THE DRIVE NUMBER
EDB9 E88500 2586 CALL NEC_OUTPUT ;
2587
2588 :----- TEST FOR FORMAT COMMAND
2589
EDBC 80FF4D 2590 CMP BH,04DH ; IS THIS A FORMAT OPERATION
EDBF 7503 2591 JNE J15 ; NO, CONTINUE WITH R/W/V
EDC1 E962FF 2592 JMP J10 ; IF SO, HANDLE SPECIAL
EDC4 2593 J15: ;
EDC4 8AE5 2594 MOV AH,CH ; CYLINDER NUMBER
EDC6 E87800 2595 CALL NEC_OUTPUT ;
EDC9 8A6401 2596 MOV AH,[BP+1] ; HEAD NUMBER FROM STACK
EDCC E87200 2597 CALL NEC_OUTPUT ;
EDCF 8AE1 2598 MOV AH,CL ; SECTOR NUMBER
EDD1 E86000 2599 CALL NEC_OUTPUT ;
EDD4 BB0700 2600 MOV BX,7 ; BYTES/SECTOR PARM FROM BLOCK
EDD7 E89200 2601 CALL GET_PARM ; TO THE NEC
EDDA BB0900 2602 MOV BX,9 ; EOT PARM FROM BLOCK
EDDD E88C00 2603 CALL GET_PARM ; TO THE NEC
EDE0 BB0B00 2604 MOV BX,11 ; CALL LENGTH PARM FROM BLOCK
EDE3 E88600 2605 CALL GET_PARM ; TO THE NEC
EDE6 BB0D00 2606 MOV BX,13 ; DTL PARM FROM BLOCK
EDE9 2607 J16: ; RW_OPN FINISH
EDE9 E88000 2608 CALL GET_PARM ; TO THE NEC
EDEB 5E 2609 POP SI ; CAN NOW DISCARD THAT DUMMY
2610 ; RETURN ADDRESS
2611
2612 :----- LET THE OPERATION HAPPEN
2613
EDED E84301 2614 CALL WAIT_INT ; WAIT FOR THE INTERRUPT
EDF0 2615 J17: ; MOTOR OFF
EDF0 7245 2616 JC J21 ; LOOK FOR ERROR
EDF2 E87401 2617 CALL RESULTS ; GET THE NEC STATUS
EDF5 723F 2618 JC J20 ; LOOK FOR ERROR
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 ST0
EDFC 24C0 2625 AND AL,0C0H ; TEST FOR NORMAL TERMINATION
EDFE 743B 2626 JZ J22 ; OPN OK
EE00 3C40 2627 CMP AL,040H ; TEST FOR ABNORMAL TERMINATION
EE02 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 ST1
EE05 D0E0           2633      SAL    AL,T                ; TEST FOR EOT FOUND
EE07 B404           2634      MOV    AH,RECORD_NOT_FND      ;
EE09 T224           2635      JC     J19                    ; RW_FAIL
EE0B D0E0           2636      SAL    AL,I                ;
EE0D D0E0           2637      SAL    AL,I                ; TEST FOR CRC ERROR
EE0F B410           2638      MOV    AH,BAD_CRC          ;
EE11 T21C           2639      JC     J19                    ; RW_FAIL
EE13 D0E0           2640      SAL    AL,I                ; TEST FOR DMA OVERRUN
EE15 B408           2641      MOV    AH,BAD_DMA          ;
EE17 T216           2642      JC     J19                    ; RW_FAIL
EE19 D0E0           2643      SAL    AL,I                ;
EE1B D0E0           2644      SAL    AL,I                ; TEST FOR RECORD NOT FOUND
EE1D B404           2645      MOV    AH,RECORD_NOT_FND      ;
EE1F T20E           2646      JC     J19                    ; RW_FAIL
EE21 D0E0           2647      SAL    AL,I                ;
EE23 B403           2648      MOV    AH,WRITE_PROTECT      ; TEST FOR WRITE_PROTECT
EE25 T208           2649      JC     J19                    ; RW_FAIL
EE27 D0E0           2650      SAL    AL,I                ; TEST MISSING ADDRESS MARK
EE29 B402           2651      MOV    AH,BAD_ADDR_MARK      ;
EE2B T202           2652      JC     J19                    ; RW_FAIL
2653
2654                ;----- NEC MUST HAVE FAILED
2655
EE2D                2656      J18:   MOV    AH,BAD_NEC          ; RW-NEC-FAIL
EE2E B420           2657      J19:   MOV    AH,BAD_NEC          ; RW-FAIL
EE2F                2658
EE2F 08264100       2659      OR     DISKETTE_STATUS,AH      ; HOW MANY WERE REALLY TRANSFERRED
EE33 E87801         2660      CALL  NUM_TRANS              ; RW_ERR
EE36                2661      J20:   RET                    ; RETURN TO CALLER
EE36 C3             2662
EE37                2663      J21:   CALL  RESULTS            ; RW_ERR RES
EE37 E82F01         2664      CALL  RET                   ; FLUSH THE RESULTS BUFFER
EE3A C3             2665
2666                ;----- OPERATION WAS SUCCESSFUL
2667
EE3B                2668      J22:   MOV    OPN_OK            ; OPN_OK
EE3B E87001         2670      CALL  NUM_TRANS              ; HOW MANY GOT MOVED
EE3E 32E4           2671      XOR    AH,AH                 ; NO ERRORS
EE40 C3             2672
2673                ; RW_OPN ENDP
2674
-----
2675      ; NEC_OUTPUT
2676      ; THIS ROUTINE SENDS A BYTE TO THE NEC CONTROLLER AFTER TESTING
2677      ; FOR CORRECT DIRECTION AND CONTROLLER READY THIS ROUTINE WILL
2678      ; TIME OUT IF THE BYTE IS NOT ACCEPTED WITHIN A REASONABLE
2679      ; AMOUNT OF TIME, SETTING THE DISKETTE STATUS ON COMPLETION.
2680      ; INPUT
2681      ; (AH) BYTE TO BE OUTPUT
2682      ; OUTPUT
2683      ; CY = 0 SUCCESS
2684      ; CY = 1 FAILURE -- DISKETTE STATUS UPDATED
2685      ; IF A FAILURE HAS OCCURRED, THE RETURN IS MADE ONE LEVEL
2686      ; HIGHER THAN THE CALLER OF NEC_OUTPUT.
2687      ; THIS REMOVES THE REQUIREMENT OF TESTING AFTER EVERY
2688      ; CALL OF NEC_OUTPUT.
2689      ; (AL) DESTROYED
-----
EE41                2690      NEC_OUTPUT   PROC   NEAR
EE41 52             2692      PUSH  DX                    ; SAVE REGISTERS
EE42 51             2693      PUSH  CX                    ;
EE43 BAF403         2694      MOV    DX,03F4H              ; STATUS PORT
EE46 33C9           2695      XOR    CX,CX                 ; COUNT FOR TIME OUT
EE48                2696      J23:
EE48 EC             2697      IN    AL,DX                  ; GET STATUS
EE49 A840           2698      TEST  AL,040H                ; TEST DIRECTION BIT
EE4B 740C           2699      JZ    J25                     ; DIRECTION OK
EE4D E2F9           2700      LOOP  J23                     ;
EE4F                2701      J24:
EE4F 800E410000     2702      OR     DISKETTE_STATUS,TIME_OUT ; TIME_ERROR
EE54 59             2703      POP   CX                     ;
EE55 5A             2704      POP   DX                     ;
EE56 58             2705      POP   AX                     ; SET ERROR CODE AND RESTORE REGS
EE57 F9             2706      STC                            ; DISCARD THE RETURN ADDRESS
EE58 C3             2707      RET                            ; INDICATE ERROR TO CALLER
EE59                2708
EE59 33C9           2709      XOR    CX,CX                 ; RESET THE COUNT
EE5B                2710
EE5B EC             2711      IN    AL,DX                  ; GET THE STATUS
EE5C A880           2712      TEST  AL,080H                ; IS IT READY?
EE5E 7504           2713      JNZ   J27                     ; YES, GO OUTPUT
EE60 E2F9           2714      LOOP  J26                     ; COUNT DOWN AND TRY AGAIN
EE62 EBEB           2715      JMP   J24                     ; ERROR CONDITION
EE64                2716      J27:
EE64 8AC4           2717      MOV    AL,AH                  ; OUTPUT
EE66 B2F5           2718      MOV    DL,0F5H                ; GET BYTE TO OUTPUT
EE68 EE             2719      OUT   DX,AL                   ; DATA PORT (3F5)
EE69 59             2720      POP   CX                     ; OUTPUT THE BYTE
EE6A 5A             2721      POP   DX                     ; RECOVER REGISTERS
EE6B C3             2722      RET                            ; CY = 0 FROM TEST INSTRUCTION
2723      NEC_OUTPUT   ENDP

```

```

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 * 2 :
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 :-----
EE6C GET_PARM PROC NEAR
EE6C IE 2738 : SAVE SEGMENT
EE6D 2BC0 2739 PUSH DS ; ZERO TO AX
EE6F 8ED8 2740 SUB AX,AX
EE71 C5367800 2741 ASSUME DS:ABS0
EE75 D1EB 2742 LDS SI,DISK_POINTER ; POINT TO BLOCK
EE77 8A20 2743 SHR BX,1 ; DIVIDE BX BY 2, AND SET FLAG
EE79 1F 2744 FOR EXIT
EE7A 8A20 2745 MOV AH,[SI+BX] ; GET THE WORD
EE7B 1F 2746 POP DS ; RESTORE SEGMENT
EE7C 12C5 2747 ASSUME DS:DATA
EE7D C3 2748 JC NEC_OUTPUT ; IF FLAG SET, OUTPUT TO CONTROLLER
EE7E C3 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 :-----
EE7D SEEK PROC NEAR
EE7E B001 2764 MOV AL,1 ; ESTABLISH MASK FOR RECAL TEST
EE7F 51 2765 PUSH CX ; SAVE INPUT VALUES
EE80 8ACA 2766 MOV CL,DL ; GET DRIVE VALUE INTO CL
EE82 D2C0 2767 ROL AL,CL ; SHIFT IT BY THE DRIVE VALUE
EE84 59 2768 POP CX ; RECOVER TRACK VALUE
EE85 84063E00 2769 TEST AL,SEEK_STATUS ; TEST FOR RECAL REQUIRED
EE89 7613 2770 JNZ J28 ; NO RECAL
EE8B 08063E00 2771 OR SEEK_STATUS,AL ; TURN ON THE NO RECAL BIT IN FLAG
EE8F B407 2772 MOV AH,07H ; RECALIBRATE COMMAND
EE91 E8ADFF 2773 CALL NEC_OUTPUT
EE94 8AE2 2774 MOV AH,DL
EE96 E8ABFF 2775 CALL NEC_OUTPUT ; OUTPUT THE DRIVE NUMBER
EE99 E87600 2776 CALL CHK_STAT_2 ; GET THE INTERRUPT AND SENSE INT STATUS
EE9C 7229 2777 JC J32 ; SEEK_ERROR
2778 :
2779 :-----
2780 :----- DRIVE IS IN SYNCH WITH CONTROLLER, SEEK TO TRACK
2781 :
EE9E J28: 2782
EE9E B40F 2783 MOV AH,0FH ; SEEK COMMAND TO NEC
EEA0 E89EFF 2784 CALL NEC_OUTPUT
EEA3 8AE2 2785 MOV AH,DL ; DRIVE NUMBER
EEA5 E899FF 2786 CALL NEC_OUTPUT
EEA8 8AE5 2787 MOV AH,CH ; TRACK NUMBER
EEAA E894FF 2788 CALL NEC_OUTPUT
EEAD E86200 2789 CALL CHK_STAT_2 ; GET ENDING INTERRUPT AND
; SENSE STATUS
2790 :
2791 :----- WAIT FOR HEAD SETTLE
2792 :
2793 :
2794 :
EEB0 9C 2795 PUSHF ; SAVE STATUS FLAGS
EEB1 BB1200 2796 MOV BX,18 ; GET HEAD SETTLE PARAMETER
EEB4 E8B5FF 2797 CALL GET_PARM
EEB7 51 2798 PUSH CX
EEB8 :
EEB8 B92602 2799 MOV CX,550 ; SAVE REGISTER
EEBB 8AE4 2800 OR AH,AH ; HEAD SETTLE
EEBD 7406 2801 JZ J31 ; 1 MS LOOP
EEBF : ; TEST FOR TIME EXPIRED
EEBF E2FE 2802 LOOP J30
EEC1 FECC 2803 DEC AH ; DELAY FOR 1 MS
EEC3 EBF3 2804 JMP J29 ; DECREMENT THE COUNT
EEC5 : ; DO IT SOME MORE
EEC5 59 2805 J31: POP CX ; RECOVER STATE
EEC6 9D 2806 POPF
EEC7 :
EEC7 C3 2807 J32: RET ; SEEK ERROR
2808 : ; RETURN TO CALLER
2809 :
2810 :
2811 SEEK ENDP

```

```

2812 ;-----
2813 ; DMA_SETUP
2814 ; THIS ROUTINE SETS UP THE DMA FOR READ/WRITE/VERIFY OPERATIONS.
2815 ; INPUT
2816 ; (AL) = MODE BYTE FOR THE DMA
2817 ; (ES:BX) -- ADDRESS TO READ/WRITE THE DATA
2818 ; OUTPUT
2819 ; (AX) DESTROYED
2820 ;-----
ECC8
ECC8 51
ECC9 FA
ECCA E60C
ECCB 50
ECCD 58
ECEE E60B
EED0 8CC0
EED2 B104
EED4 D3C0
EED6 8AE8
EED8 24F0
EEDA 03C3
EEDC 7302
EEDF FEC5
EEE0
EEE0 50
EEE1 E604
EEE3 8AC4
EEE5 E604
EEE7 8AC5
EEE9 240F
EEEB E681
2821 DMA_SETUP PROC NEAR
2822 PUSH CX ; SAVE THE REGISTER
2823 CL I ; NO MORE INTERRUPTS
2824 OUT DMA+12,AL ; SET THE FIRST/LAST F/F
2825 PUSH AX
2826 POP AX
2827 OUT DMA+11,AL ; OUTPUT THE MODE BYTE
2828 MOV AX,ES ; GET THE ES VALUE
2829 MOV CL,4 ; SHIFT COUNT
2830 ROL AX,CL ; ROTATE LEFT
2831 MOV CH,AL ; GET HIGHEST NYBLE OF ES TO CH
2832 AND AL,0F0H ; ZERO THE LOW NYBLE FROM SEGMENT
2833 AX,BX ; TEST FOR CARRY FROM ADDITION
2834 JNC J33
2835 INC CH ; CARRY MEANS HIGH 4 BITS MUST BE INC
2836 J33:
2837 PUSH AX ; SAVE START ADDRESS
2838 OUT DMA+4,AL ; OUTPUT LOW ADDRESS
2839 MOV AL,AH ; OUTPUT HIGH ADDRESS
2840 OUT DMA+4,AL ; GET HIGH 4 BITS
2841 AL,CH ; OUTPUT HIGH ADDRESS
2842 AND AL,0FH ; GET HIGH 4 BITS
2843 OUT 0B1H,AL ; OUTPUT THE HIGH 4 BITS TO
2844 ; THE PAGE REGISTER
2845
2846 ;---- DETERMINE COUNT
2847
2848 MOV AH,DH ; NUMBER OF SECTORS
2849 SUB AL,AL ; TIMES 256 INTO AX
2850 SHR AX,1 ; SECTORS * 128 INTO AX
2851 PUSH AX
2852 MOV BX,6 ; GET THE BYTES/SECTOR PARM
2853 CALL GET_PARM
2854 MOV CL,AH ; USE AS SHIFT COUNT (0=128, 1=256 ETC)
2855 POP AX
2856 SHL AX,CL ; MULTIPLY BY CORRECT AMOUNT
2857 DEC AX ; -1 FOR DMA VALUE
2858 PUSH AX ; SAVE COUNT VALUE
2859 OUT DMA+5,AL ; LOW BYTE OF COUNT
2860 MOV AL,AH ; HIGH BYTE OF COUNT
2861 OUT DMA+5,AL ; INTERRUPTS BACK ON
2862 STI ; RECOVER COUNT VALUE
2863 POP AX ; RECOVER ADDRESS VALUE
2864 POP AX ; ADD, TEST FOR 64K OVERFLOW
2865 ADD AX,CX ; RECOVER REGISTER
2866 POP CX ; MODE FOR 8237
2867 MOV AL,2 ; INITIALIZE THE DISKETTE CHANNEL
2868 OUT DMA+10,AL ; RETURN TO CALLER,
2869 RET ; CFL SET BY ABOVE IF ERROR
2870
2871 DMA_SETUP ENDP
2872 ;-----
2873 ; CHK_STAT_2
2874 ; THIS ROUTINE HANDLES THE INTERRUPT RECEIVED AFTER A
2875 ; RECALIBRATE, SEEK, OR RESET TO THE ADAPTER.
2876 ; THE INTERRUPT IS WAITED FOR, THE INTERRUPT STATUS SENSED,
2877 ; AND THE RESULT RETURNED TO THE CALLER.
2878 ; INPUT
2879 ; NONE
2880 ; OUTPUT
2881 ; CY = 0 SUCCESS
2882 ; CY = 1 FAILURE -- ERROR IS IN DISKETTE_STATUS
2883 ; (AX) DESTROYED
2884 ;-----
EF12
EF12 E81E0D
EF15 7214
EF17 B408
EF19 E825FF
EF1C E84A0D
EF1F 720A
EF21 A04200
EF24 2460
EF26 3C00
EF28 7402
EF2A F8
EF2B
EF2B C3
EF2C
EF2C 800E410040
EF31 F9
EF32 C3
2885 CHK_STAT_2 PROC NEAR
2886 CALL WAIT_INT ; WAIT FOR THE INTERRUPT
2887 JC J34 ; IF ERROR, RETURN IT
2888 MOV AH,08H ; SENSE INTERRUPT STATUS COMMAND
2889 CALL NEC_OUTPUT
2890 CALL RESULTS ; READ IN THE RESULTS
2891 JC J34 ; CHK2 RETURN
2892 MOV AL,NEC_STATUS ; GET THE FIRST STATUS BYTE
2893 AND AL,060H ; ISOLATE THE BITS
2894 CMP AL,060H ; TEST FOR CORRECT VALUE
2895 JZ J35 ; IF ERROR, GO MARK IT
2896 CLC ; GOOD RETURN
2897 J34:
2898 RET ; RETURN TO CALLER
2899 J35:
2900 RET ; CHK2_ERROR
2901 OR DISKETTE_STATUS,BAD_SEEK ; DISKETTE_STATUS,BAD_SEEK
2902 STC ; ERROR RETURN CODE
2903 RET
2904 CHK_STAT_2 ENDP

```

LOC OBJECT

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

```

2904 ;-----
2905 ; WAIT_INT ;
2906 ; THIS ROUTINE WAITS FOR AN INTERRUPT TO OCCUR. A TIME OUT ;
2907 ; ROUTINE TAKES PLACE DURING THE WAIT, SO THAT AN ERROR MAY BE ;
2908 ; RETURNED IF THE DRIVE IS NOT READY. ;
2909 ; INPUT ;
2910 ; NONE ;
2911 ; OUTPUT ;
2912 ; CY = 0 SUCCESS -- DISKETTE_STATUS IS SET ACCORDINGLY ;
2913 ; CY = 1 FAILURE -- DISKETTE_STATUS IS SET ACCORDINGLY ;
2914 ; (AH) DESTROYED ;
2915 ;-----
EF33 2916 WAIT_INT PROC NEAR
EF33 FB 2917 STI ; TURN ON INTERRUPTS, JUST IN CASE
EF34 53 2918 PUSH BX
EF35 51 2919 PUSH CX ; SAVE REGISTERS
EF36 B302 2920 MOV BL,2 ; CLEAR THE COUNTERS
EF38 33C9 2921 XOR CX,CX ; FOR 2 SECOND WAIT
EF3A 2922 J36:
EF3A F6063E0080 2923 TEST SEEK_STATUS,INT_FLAG ; TEST FOR INTERRUPT OCCURRING
EF3F 750C 2924 JNZ J37
EF41 E2F7 2925 LOOP J36 ; COUNT DOWN WHILE WAITING
EF43 F6CB 2926 DEC BL ; SECOND LEVEL COUNTER
EF45 75F3 2927 JNZ J36
EF47 800E410080 2928 OR DISKETTE_STATUS,TIME_OUT ; NOTHING HAPPENED
EF4C F9 2929 STC ; ERROR RETURN
EF4D 9C 2930 J37:
EF4D 9C 2931 PUSHF ; SAVE CURRENT CARRY
EF4E 80263E007F 2932 AND SEEK_STATUS,NOT_INT_FLAG ; TURN OFF INTERRUPT FLAG
EF4F 9D 2933 POPF ; RECOVER CARRY
EF54 59 2934 POP CX
EF55 5B 2935 POP BX ; RECOVER REGISTERS
EF56 C3 2936 RET ; GOOD RETURN CODE COMES FROM TEST INST
2937 ;-----
2938 WAIT_INT ENDP
2939 ;-----
2940 ; DISK_INT ;
2941 ; THIS ROUTINE HANDLES THE DISKETTE INTERRUPT ;
2942 ; INPUT ;
2943 ; NONE ;
2944 ; OUTPUT ;
2945 ; THE INTERRUPT FLAG IS SET IS SEEK_STATUS ;
2946 ;-----
EF57 2947 ORG 0EF57H
EF57 2948 DISK_INT PROC FAR ; RE ENABLE INTERRUPTS
EF57 FB 2949 STI
EF58 IE 2950 PUSH DS
EF59 50 2951 PUSH AX
EF5A E8FC0A 2952 CALL DDS
EF5D 800E3E0080 2953 OR SEEK_STATUS,INT_FLAG
EF62 B020 2954 MOV AL,20H ; END OF INTERRUPT MARKER
EF64 E620 2955 OUT 20H,AL ; INTERRUPT CONTROL PORT
EF66 58 2956 POP AX
EF67 1F 2957 POP DS ; RECOVER SYSTEM
EF68 CF 2958 IRET ; RETURN FROM INTERRUPT
2959 DISK_INT ENDP
2960 ;-----
2961 ; RESULTS ;
2962 ; THIS ROUTINE WILL READ ANYTHING THAT THE NEC CONTROLLER HAS ;
2963 ; TO SAY FOLLOWING AN INTERRUPT. ;
2964 ; INPUT ;
2965 ; NONE ;
2966 ; OUTPUT ;
2967 ; CY = 0 SUCCESSFUL TRANSFER ;
2968 ; CY = 1 FAILURE -- TIME OUT IN WAITING FOR STATUS ;
2969 ; NEC_STATUS AREA HAS STATUS BYTE LOADED INTO IT ;
2970 ; (AH) DESTROYED ;
2971 ;-----
EF69 2972 RESULTS PROC NEAR
EF69 FC 2973 CLD
EF6A B4F200 2974 MOV DI,OFFSET NEC_STATUS ; POINTER TO DATA AREA
EF6D 51 2975 PUSH CX ; SAVE COUNTER
EF6E 52 2976 DX CX
EF6F 53 2977 PUSH BX
EF70 B307 2978 MOV BL,7 ; MAX STATUS BYTES
2979 ;-----
2980 ;----- WAIT FOR REQUEST FOR MASTER
2981 J38:
EF72 33C9 2982 XOR CX,CX ; INPUT_LOOP
EF74 BAF403 2983 MOV DX,03F4H ; COUNTER
EF77 2984 J39: ; STATUS_PORT
EF77 EC 2985 IN AL,DX ; WAIT FOR MASTER
EF78 A880 2986 TEST AL,080H ; GET STATUS
EF7A 750C 2987 J40A ; MASTER READY
EFTC E2F9 2988 LOOP J39 ; TEST DIR
EFTF 800E410080 2989 OR DISKETTE_STATUS,TIME_OUT ; WAIT_MASTER
EF83 2990 ; RESULTS_ERROR
EF83 F9 2991 J40: ; SET ERROR RETURN
EF84 5B 2992 STC
EF85 5A 2993 POP BX
EF86 59 2994 POP DX
EF87 C3 2995 POP CX
2996 RET
2997 ;-----
2998 ;----- TEST THE DIRECTION BIT
2999 J40A:
EF88 3000 J40A:
EF88 EC 3001 IN AL,DX ; GET STATUS REG AGAIN
EF89 A840 3002 TEST AL,040H ; TEST DIRECTION BIT
EF8B 7507 3003 J42 ; OK TO READ STATUS
EF8D 3004 J41: ; NEC_FAIL
EF8D 800E410020 3005 OR DISKETTE_STATUS,BAD_NEC
EF92 EBEF 3006 JMP J40 ; RESULTS_ERROR
3007 ;-----
3008 ;----- READ IN THE STATUS
3009 J42:
EF94 3010 J42:
EF94 42 3011 INC DX ; INPUT_STAT
EF95 EC 3012 IN AL,DX ; POINT AT DATA PORT
EF96 8805 3013 MOV [DI],AL ; GET THE DATA
EF98 47 3014 INC DI ; STORE THE BYTE
EF99 B90A00 3015 MOV CX,10 ; INCREMENT THE POINTER
EF9C E2FE 3016 LOOP J43 ; LOOP TO KILL TIME FOR NEC
EF9E 4A 3017 DEC CX ; POINT AT STATUS PORT
EF9F EC 3018 IN AL,DX ; GET STATUS
EFA0 A810 3019 TEST AL,010H ; TEST FOR NEC STILL BUSY

```

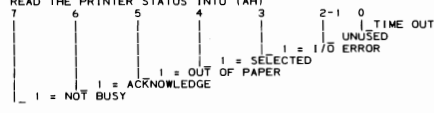
```

LOC OBJECT          LINE SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82
EFA2 7406          3020 JZ      J44          ; RESULTS DONE
EFA4 FECB          3021 DEC     BL           ; DECREMENT THE STATUS COUNTER
EFA6 75CA          3022 JNZ    J38          ; GO BACK FOR MORE
EFA8 EBE3          3023 JMP    J41          ; CHIP HAS FAILED
3024
3025 ;----- RESULT OPERATION IS DONE
3026
EFAA
3027 J44:
EFAA 5B            3028 POP    BX           ;
EFA4 5A            3029 POP    DX           ;
EFA6 59            3030 POP    CX           ; RECOVER REGISTERS
EFA4 C3            3031 RET     ; GOOD RETURN CODE FROM TEST INST
3032 ;-----
3033 ; NUM_TRANS
3034 ; THIS ROUTINE CALCULATES THE NUMBER OF SECTORS THAT
3035 ; WERE ACTUALLY TRANSFERRED TO/FROM THE DISKETTE
3036 ; INPUT
3037 ; (CH) = CYLINDER OF OPERATION
3038 ; (CL) = START SECTOR OF OPERATION
3039 ; OUTPUT
3040 ; (AL) = NUMBER ACTUALLY TRANSFERRED
3041 ; NO OTHER REGISTERS MODIFIED
3042 ;-----
EFAE              3043 NUM_TRANS PROC NEAR
EFAE A04500        3044 MOV    AL,NEC_STATUS+3 ; GET CYLINDER ENDED UP ON
EFB1 3AC5          3045 CMP    AL,CH         ; SAME AS WE STARTED
EFB3 A04700        3046 MOV    AL,NEC_STATUS+5 ; GET ENDING SECTOR
EFB6 740A          3047 JZ     J45           ; IF ON SAME CYL, THEN NO ADJUST
EFB8 B0B000        3048 MOV    BX,8          ;
EFBB EBAEFE        3049 CALL  GET_PARM      ; GET EOT VALUE
EFBE 8AC4          3050 MOV    AL,AH         ; INTO AL
EFC0 FEC0          3051 INC    AL           ; USE EOT+1 FOR CALCULATION
EFC2              3052 J45:
EFC2 2AC1          3053 SUB    AL,CL         ; SUBTRACT START FROM END
EFC4 C3            3054 RET
3055 NUM_TRANS ENDP
3056 RESULTS ENDP
3057 ;-----
3058 ; DISK_BASE
3059 ; THIS IS THE SET OF PARAMETERS REQUIRED FOR DISKETTE OPERATION.
3060 ; THEY ARE POINTED AT BY THE DATA VARIABLE DISK_POINTER. TO
3061 ; MODIFY THE PARAMETERS, BUILD ANOTHER PARAMETER BLOCK AND POINT
3062 ; DISK_POINTER TO IT.
3063 ;-----
EFC7              3064 ORG    DEFCTH
EFC7              3065 DISK_BASE LABEL BYTE
EFC7 CF            3066 DB    11001111B    ; SRT=C, HD UNLOAD=0F - 1ST SPECIFY BYTE
EFC8 02            3067 DB    2             ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
EFC9 25            3068 DB    MOTOR_WAIT   ; WAIT AFTER OPN TIL MOTOR OFF
EFCA 02            3069 DB    2             ; 512 BYTES/SECTOR
EFCB 08            3070 DB    8             ; EOT ( LAST SECTOR ON TRACK)
EFC C 2A          3071 DB    02AH         ; GAP LENGTH
EFCD FF            3072 DB    0FFH         ; DTL
EFCE 50            3073 DB    050H         ; GAP LENGTH FOR FORMAT
EFCF F6            3074 DB    0F6H         ; FILL BYTE FOR FORMAT
EFD0 19            3075 DB    25           ; HEAD SETTLE TIME (MILLI SECONDS)
EFD1 04            3076 DB    4            ; MOTOR START TIME (1/8 SECONDS)
3077

```

```

3078 ;-- INT 17 -----
3079 ; PRINTER_IO
3080 ; THIS ROUTINE PROVIDES COMMUNICATION WITH THE PRINTER
3081 ; INPUT
3082 ; (AH)=0 PRINT THE CHARACTER IN (AL)
3083 ; ON RETURN, AH=1 IF CHARACTER COULD NOT BE PRINTED
3084 ; (TIME OUT), OTHER BITS SET AS ON NORMAL STATUS CALL
3085 ; (AH)=1 INITIALIZE THE PRINTER PORT
3086 ; RETURNS WITH (AH) SET WITH PRINTER STATUS
3087 ; (AH)=2 READ THE PRINTER STATUS INTO (AH)
3088 ;
3089 ;
3090 ;
3091 ;
3092 ;
3093 ;
3094 ;
3095 ;
3096 ;
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 ; 40BH 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
    
```



```

EFD2 ; 3111 ASSUME CS:CODE,DS:DATA
EFD2 ; 3112 PRINTER_IO PROC FAR
EFD2 FB ; 3113 STI ; INTERRUPTS BACK ON
EFD3 1E ; 3114 PUSH DS ; SAVE SEGMENT
EFD4 52 ; 3115 PUSH DX
EFD5 56 ; 3116 PUSH SI
EFD6 51 ; 3117 PUSH CX
EFD7 53 ; 3118 PUSH BX
EFD8 E87E0A ; 3119 CALL DDS
EFD8 8BF2 ; 3120 MOV SI,DX ; GET PRINTER PARM
EFD8 8A5C78 ; 3121 MOV BL,PRINT_TIM_OUT[SI] ; LOAD TIME-OUT PARM
EFD8 D1E6 ; 3122 SHL SI,1 ; WORD OFFSET INTO TABLE
EFD8 8B5408 ; 3123 MOV DX,PRINTER_BASE[SI] ; GET BASE ADDRESS FOR PRINTER CARD
EFD8 0BD2 ; 3124 OR DX,DX ; TEST DX FOR ZERO,
; 3125 ; INDICATING NO PRINTER
EFD8 740C ; 3126 JZ B1 ; RETURN
EFD8 0AE4 ; 3127 OR AH,AH
EFD8 740E ; 3128 JZ B2 ; PRINT AL
EFD8 FECC ; 3129 DEC AH ; TEST FOR (AH)=1
EFD8 743F ; 3130 JZ B3 ; INIT PRN
EFD8 FECC ; 3131 DEC AH ; TEST FOR (AH)=2
EFD8 7428 ; 3132 JZ B5 ; PRINTER STATUS
EFD8 ; 3133 B1: RETURN
EFD8 ; 3134 POP BX
EFD8 ; 3135 POP CX
EFD8 ; 3136 POP SI ; RECOVER REGISTERS
EFD8 ; 3137 POP DX ; RECOVER REGISTERS
EFD8 ; 3138 POP DS
EFD8 ; 3139 IRET
EFD8 ; 3140
EFD8 ; 3141 ;----- PRINT THE CHARACTER IN (AL)
EFD8 ; 3142 B2:
EFD8 ; 3143 PUSH AX ; SAVE VALUE TO PRINT
EFD8 ; 3144 OUT DX,AL ; OUTPUT CHAR TO PORT
EFD8 ; 3145 INC DX ; POINT TO STATUS PORT
EFD8 ; 3146 B3:
EFD8 ; 3147 SUB CX,CX ; WAIT_BUSY
EFD8 ; 3148 B3_1:
EFD8 ; 3149 IN AL,DX ; GET STATUS
EFD8 ; 3150 MOV AH,AL ; STATUS TO AH ALSO
EFD8 ; 3151 TEST AL,80H ; IS THE PRINTER CURRENTLY BUSY
EFD8 ; 3152 JNZ B4 ; OUT_STROBE
EFD8 ; 3153 LOOP B3_1 ; TRY_AGAIN
EFD8 ; 3154 DEC BL ; DROP LOOP COUNT
EFD8 ; 3155 JNZ B3 ; GO TILL TIMEOUT ENDS
EFD8 ; 3156 OR AH,1 ; SET ERROR FLAG
EFD8 ; 3157 AND AH,0F9H ; TURN OFF THE OTHER BITS
EFD8 ; 3158 JMP SHORT B7 ; RETURN WITH ERROR FLAG SET
EFD8 ; 3159 B4:
EFD8 ; 3160 MOV AL,0DH ; OUT_STROBE
EFD8 ; 3161 INC DX ; SET THE STROBE HIGH
EFD8 ; 3162 OUT DX,AL ; STROBE IS BIT 0 OF PORT C OF 8255
EFD8 ; 3163 OUT DX,AL ; SET THE STROBE LOW
EFD8 ; 3164 MOV AL,0FH ;
EFD8 ; 3165 OUT DX,AL ;
EFD8 ; 3166 POP AX ; RECOVER THE OUTPUT CHAR
EFD8 ; 3167
EFD8 ; 3168 ;----- PRINTER STATUS
EFD8 ; 3169 B5:
EFD8 ; 3170 PUSH AX ; SAVE AL REG
EFD8 ; 3171 B6:
EFD8 ; 3172 MOV DX,PRINTER_BASE[SI]
EFD8 ; 3173 INC DX
EFD8 ; 3174 IN AL,DX ; GET PRINTER STATUS
EFD8 ; 3175 MOV AH,AL
EFD8 ; 3176 AND AH,0FBH ; TURN OFF UNUSED BITS
EFD8 ; 3177 B7:
EFD8 ; 3178 POP DX ; STATUS SET
EFD8 ; 3179 MOV AL,DL ; RECOVER AL REG
EFD8 ; 3180 XOR AH,48H ; GET CHARACTER INTO AL
EFD8 ; 3181 JMP B1 ; FLIP A COUPLE OF BITS
EFD8 ; 3182 ; RETURN FROM ROUTINE
    
```

LOC OBJECT	LINE	SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82
	3183	
	3184	;----- INITIALIZE THE PRINTER PORT
	3185	
F030	3186	B8:
F030 50	3187	PUSH AX ; SAVE AL
F031 42	3188	INC DX ; POINT TO OUTPUT PORT
F032 42	3189	INC DX
F033 B008	3190	MOV AL,8 ; SET INIT LINE LOW
F035 EE	3191	OUT DX,AL
F036 BBE803	3192	MOV AX,1000
F039	3193	B9:
F039 48	3194	DEC AX ; INIT_LOOP
F03A 75FD	3195	JNZ B9 ; LOOP FOR RESET TO TAKE
F03C B00C	3196	MOV AL,0CH ; INIT_LOOP
F03E EE	3197	OUT DX,AL ; NO INTERRUPTS, NON AUTO LF,
F03F EBDD	3198	OUT DX,AL ; INIT HIGH
	3199	JMP B6 ; PRT_STATUS_1
	3200	PRINTER_ID
	3201	ENDP

```

3202
3203 --- INT 10 ---
3204 | VIDEO_ID |
3205 | THESE ROUTINES PROVIDE THE CRT INTERFACE |
3206 | THE FOLLOWING FUNCTIONS ARE PROVIDED: |
3207 | (AH)=0 SET MODE (AL) CONTAINS MODE VALUE |
3208 | (AL)=0 40X25 BW (POWER ON DEFAULT) |
3209 | (AL)=1 40X25 COLOR |
3210 | (AL)=2 80X25 BW |
3211 | (AL)=3 80X25 COLOR |
3212 | GRAPHICS MODES |
3213 | (AL)=4 320X200 COLOR |
3214 | (AL)=5 320X200 BW |
3215 | (AL)=6 640X200 BW |
3216 | CRT MODE=7 80X25 B&W CARD (USED INTERNAL TO VIDEO ONLY) |
3217 | *** NOTE BW MODES OPERATE SAME AS COLOR MODES, BUT |
3218 | COLOR BURST IS NOT ENABLED |
3219 | (AH)=1 SET CURSOR TYPE |
3220 | (CH) = BITS 4-0 = START LINE FOR CURSOR |
3221 | ** HARDWARE WILL ALWAYS CAUSE BLIN |
3222 | ** SETTING BIT 5 OR 6 WILL CAUSE ERRATIC |
3223 | BLINKING OR NO CURSOR AT ALL |
3224 | (CL) = BITS 4-0 = END LINE FOR CURSOR |
3225 | (AH)=2 SET CURSOR POSITION |
3226 | (DH,DL) = ROW,COLUMN 10,01 IS UPPER LEFT |
3227 | (BH) = PAGE NUMBER (MUST BE 0 FOR GRAPHICS MODES) |
3228 | (AH)=3 READ CURSOR POSITION |
3229 | (BH) = PAGE NUMBER (MUST BE 0 FOR GRAPHICS MODES) |
3230 | ON EXIT (DH,DL) = ROW,COLUMN OF CURRENT CURSOR |
3231 | (CH,CL) CURSOR MODE CURRENTLY SET |
3232 | (AH)=4 READ LIGHT PEN POSITION |
3233 | ON EXIT: |
3234 | (AH) = 0 -- LIGHT PEN SWITCH NOT DOWN/NOT TRIGGERED |
3235 | (AH) = 1 -- VALID LIGHT PEN VALUE IN REGISTERS |
3236 | (DH,DL) = ROW,COLUMN OF CHARACTER LP POSN |
3237 | (CH) = RASTER LINE (0-199) |
3238 | (BX) = PIXEL COLUMN (0-319,639) |
3239 | (AH)=5 SELECT ACTIVE DISPLAY PAGE (VALID ONLY FOR ALPHA MODES) |
3240 | (AL)=NEW PAGE VAL (0-7 FOR MODES 0&1, 0-3 FOR MODES 2&3) |
3241 | (AH)=6 SCROLL ACTIVE PAGE UP |
3242 | (AL) = NUMBER OF LINES, INPUT LINES BLANKED AT BOTTOM |
3243 | OF WINDOW |
3244 | AL = 0 MEANS BLANK ENTIRE WINDOW |
3245 | (CH,CL) = ROW,COLUMN OF UPPER LEFT CORNER OF SCROLL |
3246 | (DH,DL) = ROW,COLUMN OF LOWER RIGHT CORNER OF SCROLL |
3247 | (BH) = ATTRIBUTE TO BE USED ON BLANK LINE |
3248 | (AH)=7 SCROLL ACTIVE PAGE DOWN |
3249 | (AL) = NUMBER OF LINES, INPUT LINES BLANKED AT TOP |
3250 | OF WINDOW |
3251 | AL = 0 MEANS BLANK ENTIRE WINDOW |
3252 | (CH,CL) = ROW,COLUMN OF UPPER LEFT CORNER OF SCROLL |
3253 | (DH,DL) = ROW,COLUMN OF LOWER RIGHT CORNER OF SCROLL |
3254 | (BH) = ATTRIBUTE TO BE USED ON BLANK LINE |
3255 |
3256 | CHARACTER HANDLING ROUTINES |
3257 |
3258 | (AH) = 8 READ ATTRIBUTE/CHARACTER AT CURRENT CURSOR POSITION |
3259 | (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY) |
3260 | ON EXIT: |
3261 | (AL) = CHAR READ |
3262 | (AH) = ATTRIBUTE OF CHARACTER READ (ALPHA MODES ONLY) |
3263 | (AH) = 9 WRITE ATTRIBUTE/CHARACTER AT CURRENT CURSOR POSITION |
3264 | (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY) |
3265 | (CX) = COUNT OF CHARACTERS TO WRITE |
3266 | (AL) = CHAR TO WRITE |
3267 | (BL) = ATTRIBUTE OF CHARACTER (ALPHA)/COLOR OF CHAR |
3268 | (GRAPHICS) |
3269 | SEE NOTE ON WRITE DOT FOR BIT 7 OF BL = 1 |
3270 | (AH) = 10 WRITE CHARACTER ONLY AT CURRENT CURSOR POSITION |
3271 | (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY) |
3272 | (CX) = COUNT OF CHARACTERS TO WRITE |
3273 | (AL) = CHAR TO WRITE |
3274 | FOR READ/WRITE CHARACTER INTERFACE WHILE IN GRAPHICS MODE, THE |
3275 | CHARACTERS ARE FORMED FROM A CHARACTER GENERATOR IMAGE |
3276 | MAINTAINED IN THE SYSTEM ROM. ONLY THE 1ST 128 CHARS |
3277 | ARE CONTAINED THERE. TO READ/WRITE THE SECOND 128 |
3278 | CHARS, THE USER MUST INITIALIZE THE POINTER AT |
3279 | INTERRUPT 1FH (LOCATION 0007CH) TO POINT TO THE 1K BYTE |
3280 | TABLE CONTAINING THE CODE POINTS FOR THE SECOND |
3281 | 128 CHARS (128-255). |
3282 | FOR WRITE CHARACTER INTERFACE IN GRAPHICS MODE, THE REPLICATION |
3283 | FACTOR CONTAINED IN (CX) ON ENTRY WILL PRODUCE VALID |
3284 | RESULTS ONLY FOR CHARACTERS CONTAINED ON THE SAME ROW. |
3285 | CONTINUATION TO SUCCEEDING LINES WILL NOT PRODUCE |
3286 | CORRECTLY. |
3287 |
3288 | GRAPHICS INTERFACE |
3289 | (AH) = 11 SET COLOR PALETTE |
3290 | (BH) = PALETTE COLOR ID BEING SET (0-127) |
3291 | (BL) = COLOR VALUE TO BE USED WITH THAT COLOR ID |
3292 | NOTE: FOR THE CURRENT COLOR CARD, THIS ENTRY POINT |
3293 | HAS MEANING ONLY FOR 320X200 GRAPHICS. |
3294 | COLOR ID = 0 SELECTS THE BACKGROUND COLOR (0-15) |
3295 | COLOR ID = 1 SELECTS THE PALETTE TO BE USED: |
3296 | 0 = GREEN(1)/RED(2)/YELLOW(3) |
3297 | 1 = CYAN(11)/MAGENTA(2)/WHITE(3) |
3298 | IN 40X25 OR 80X25 ALPHA MODES, THE VALUE SET |
3299 | FOR PALETTE COLOR 0 INDICATES THE |
3300 | BORDER COLOR TO BE USED (VALUES 0-31, |
3301 | WHERE 16-31 SELECT THE HIGH INTENSITY |
3302 | BACKGROUND SET. |
3303 |
3304 | (AH) = 12 WRITE DOT |
3305 | (DX) = ROW NUMBER |
3306 | (CX) = COLUMN NUMBER |
3307 | (AL) = COLOR VALUE |
3308 | IF BIT 7 OF AL = 1, THEN THE COLOR VALUE IS |
3309 | EXCLUSIVE OR'D WITH THE CURRENT CONTENTS OF |
3310 | THE DOT |
3311 | (AH) = 13 READ DOT |
3312 | (DX) = ROW NUMBER |
3313 | (CX) = COLUMN NUMBER |
3314 | (AL) RETURNS THE DOT READ |

```



```

3314 :
3315 : ASCII TELETYPE ROUTINE FOR OUTPUT
3316 :
3317 : (AH) = 14 WRITE TELETYPE TO ACTIVE PAGE
3318 : (AL) = CHAR TO WRITE
3319 : (BL) = FOREGROUND COLOR IN GRAPHICS MODE
3320 : NOTE -- SCREEN WIDTH IS CONTROLLED BY PREVIOUS MODE SET
3321 :
3322 : (AH) = 15 CURRENT VIDEO STATE
3323 : RETURNS THE CURRENT VIDEO STATE
3324 : (AL) = MODE CURRENTLY SET | SEE AH=0 FOR EXPLANATION|
3325 : (AH) = NUMBER OF CHARACTER COLUMNS ON SCREEN
3326 : (BH) = CURRENT ACTIVE DISPLAY PAGE
3327 :
3328 : CS,SS,DS,ES,BX,CX,DX PRESERVED DURING CALL
3329 : ALL OTHERS DESTROYED
3330 :-----
3331 ASSUME CS:CODE,DS:DATA,ES:VIDEO_RAM
3332 ORG 0F045H
3333 M1 LABEL WORD ; TABLE OF ROUTINES WITHIN VIDEO I/O
3334 DW OFFSET SET_MODE
3335 DW OFFSET SET_CTYPE
3336 DW OFFSET SET_CPOS
3337 DW OFFSET READ_CURSOR
3338 DW OFFSET READ_LPEN
3339 DW OFFSET ACT_DISP_PAGE
3340 DW OFFSET SCROLL_UP
3341 DW OFFSET SCROLL_DOWN
3342 DW OFFSET READ_AC_CURRENT
3343 DW OFFSET WRITE_AC_CURRENT
3344 DW OFFSET WRITE_C_CURRENT
3345 DW OFFSET SET_COLOR
3346 DW OFFSET WRITE_DOT
3347 DW OFFSET READ_DOT
3348 DW OFFSET WRITE_TTY
3349 DW OFFSET VIDEO_STATE
3350 MIL EQU $-M1
3351
3352 ORG 0F065H
3353 VIDEO_10 PROC NEAR
3354 STI ; INTERRUPTS BACK ON
3355 CLD ; SET DIRECTION FORWARD
3356 PUSH ES
3357 PUSH DS ; SAVE SEGMENT REGISTERS
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 CMP AX,MIL ; TEST FOR WITHIN 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
3374 MOV AX,0B800H ; SEGMENT FOR COLOR CARD
3375 MOV DI,EQUIP_FLAG ; GET EQUIPMENT SETTING
3376 AND DI,30H ; ISOLATE CRT SWITCHES
3377 CMP DI,30H ; IS SETTING FOR BW CARD?
3378 JNE M3
3379 MOV AH,0B0H ; SEGMENT FOR BW CARD
3380 M3:
3381 MOV ES,AX ; SET UP TO POINT AT VIDEO RAM AREAS
3382 POP AX ; RECOVER VALUE
3383 MOV AH,CRT_MODE ; GET CURRENT MODE INTO AH
3384 JMP WORD PRT CS:[SI+OFFSET M1]
3385 VIDEO_10 ENDP

```

LOC OBJECT

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

```

3386 ;-----
3387 ; SET_MODE ;
3388 ; THIS ROUTINE INITIALIZES THE ATTACHMENT TO ;
3389 ; THE SELECTED MODE. THE SCREEN IS BLANKED. ;
3390 ; INPUT ;
3391 ; (AL) = MODE SELECTED (RANGE 0-9) ;
3392 ; OUTPUT ;
3393 ; NONE ;
3394 ;-----
3395
3396 ;----- TABLES FOR USE IN SETTING OF MODE
3397
3398
3399
3400 VIDEO_PARMS ORG 0F0A4H LABEL BYTE
3401 ;----- INIT_TABLE
3402 DB 38H,28H,2DH,0AH,1FH,6,19H ; SET UP FOR 40X25

FOA4 38
FOA5 28
FOA6 2D
FOA7 0A
FOA8 1F
FOA9 06
FOAA 19
FOAB 1C
FOAC 02
FOAD 07
FOAE 06
FOAF 07
FOB0 00
FOB1 00
FOB2 00
FOB3 00
0010

FOB4 71
FOB5 50
FOB6 5A
FOB7 0A
FOB8 1F
FOB9 06
FOBA 19
FOBB 1C
FOBC 02
FOBD 07
FOBE 06
FOBF 07
FOC0 00
FOC1 00
FOC2 00
FOC3 00

FOC4 38
FOC5 28
FOC6 2D
FOC7 0A
FOC8 7F
FOC9 06
FOCA 64
FOCB 70
FOCC 02
FOCD 01
FOCE 06
FOCF 07
FOD0 00
FOD1 00
FOD2 00
FOD3 00

FOD4 61
FOD5 50
FOD6 52
FOD7 0F
FOD8 19
FOD9 06
FODA 19
FODB 19
FODC 02
FODD 0D
FODE 0B
FODF 0C
FOE0 00
FOE1 00
FOE2 00
FOE3 00

FOE4
FOE4 0008
FOE4 0010
FOE8 0040
FOEA 0040

FOEC
FOEC 28
FOED 28
FOEE 50
FOEF 50
FOF0 28
FOF1 28
FOF2 50
FOF3 50

FOF4
FOF4 2C
FOF5 28
FOF6 2D
FOF7 29
FOF8 2A
FOF9 2E
FOFA 1E
FOFB 29

3403 DB 0,0,0,0
3404 M4 EQU %-VIDEO_PARMS
3405
3406 DB 71H,50H,5AH,0AH,1FH,6,19H ; SET UP FOR 80X25

3407 DB 1CH,2,7,6,7

3408 DB 0,0,0,0

3409
3410 DB 38H,28H,2DH,0AH,7FH,6,64H ; SET UP FOR GRAPHICS

3411 DB 70H,2,1,6,7

3412 DB 0,0,0,0

3413
3414 DB 61H,50H,52H,0FH,19H,6,19H ; SET UP FOR 80X25 B&W CARD

3415 DB 19H,2,0DH,0BH,0CH

3416 DB 0,0,0,0

3417
3418 M5 LABEL WORD ; TABLE OF REGEN LENGTHS
3419 DW 2048 ; 40X25
3420 DW 4096 ; 80X25
3421 DW 16384 ; GRAPHICS
3422 DW 16384
3423
3424 ;----- COLUMNS
3425
3426 M6 LABEL BYTE
3427 DB 40,40,80,80,40,40,80,80

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

```

```

3433
3434 SET_MODE PROC NEAR
F0FC B4D403 3435 MOV DX,03D4H ; ADDRESS OF COLOR CARD
FOFF B300 3436 BL 0 ; MODE SET FOR COLOR CARD
F101 83FF30 3437 CMP DI,30H ; IS BW CARD INSTALLED
F104 7506 3438 JNE M8 ; OK WITH COLOR
F106 B007 3439 MOV AL,7 ; INDICATE BW CARD MODE
F108 B2B4 3440 MOV DL,0B4H ; ADDRESS OF BW CARD (3B4)
F10A FEC3 3441 INC BL ; MODE SET FOR BW CARD
F10C 3442 M8:
F10C 8AE0 3443 MOV AH,AL ; SAVE MODE IN AH
F10E A24900 3444 MOV CRT_MODE,AL ; SAVE IN GLOBAL VARIABLE
F111 89166300 3445 MOV ADDR_6845,DX ; SAVE ADDRESS OF BASE
F115 1E 3446 PUSH DS ; SAVE POINTER TO DATA SEGMENT
F116 50 3447 PUSH AX ; SAVE MODE
F117 52 3448 PUSH DX ; SAVE OUTPUT PORT VALUE
F118 83C204 3449 ADD DX,4 ; POINT TO CONTROL REGISTER
F11B 8AC3 3450 MOV AL,BL ; GET MODE SET FOR CARD
F11E EE 3451 OUT DX,AL ; RESET VIDEO
F11F 5A 3452 POP DX ; BACK TO BASE REGISTER
F121 2BC0 3453 SUB AX,AX ; SET UP FOR ABS0 SEGMENT
F121 8ED8 3454 MOV DS,AX ; ESTABLISH VECTOR TABLE ADDRESSING
3455 ASSUME DS:ABS0
F123 C51E7400 3456 LDS BX,PARAM_PTR ; GET POINTER TO VIDEO PARMS
F127 58 3457 POP AX ; RECOVER PARMS
3458 ASSUME DS:CODE
F128 B91000 3459 MOV CX,M4 ; LENGTH OF EACH ROW OF TABLE
F12B 80FC02 3460 CMP AH,2 ; DETERMINE WHICH ONE TO USE
F12E 7210 3461 JC M9 ; MODE IS 0 OR 1
F130 03D9 3462 ADD BX,CX ; MOVE TO NEXT ROW OF INIT TABLE
F132 80FC04 3463 CMP AH,4 ;
F135 7209 3464 JC M9 ; MODE IS 2 OR 3
F137 03D9 3465 ADD BX,CX ; MOVE TO GRAPHICS ROW OF INIT_TABLE
F139 80FC07 3466 CMP AH,7 ;
F13C 7202 3467 JC M7 ; MODE IS 4,5, OR 6
F13E 03D9 3468 ADD BX,CX ; MOVE TO BW CARD ROW OF INIT_TABLE
3469
3470 ;----- BX POINTS TO CORRECT ROW OF INITIALIZATION TABLE
3471
F140 3472 M9:
F140 50 3473 PUSH AX ; OUT INIT
F141 32E4 3474 XOR AH,AH ; SAVE MODE IN AH
3475 ; AH WILL SERVE AS REGISTER
3476 ; NUMBER DURING LOOP
3477 ;----- LOOP THROUGH TABLE, OUTPUTTING REG ADDRESS, THEN VALUE FROM TABLE
3478
F143 3479 M10:
F143 8AC4 3480 MOV AL,AH ; INIT LOOP
F145 EE 3481 OUT DX,AL ; GET 6845 REGISTER NUMBER
F146 42 3482 INC DX ; POINT TO DATA PORT
F147 FEC4 3483 INC AH ; NEXT REGISTER VALUE
F149 8A07 3484 MOV AL,[BX] ; GET TABLE VALUE
F14B EE 3485 OUT DX,AL ; OUT TO CHIP
F14C 43 3486 INC BX ; NEXT IN TABLE
F14D 4A 3487 DEC DX ; BACK TO POINTER REGISTER
F14E EF23 3488 LOOP M10 ; DO THE WHOLE TABLE
F150 5B 3489 POP AX ; GET MODE BACK
F151 1F 3490 POP DS ; RECOVER SEGMENT VALUE
3491 ASSUME DS:DATA
3492
3493 ;----- FILL REGEN AREA WITH BLANK
3494
F152 33FF 3495 XOR DI,D1 ; SET UP POINTER FOR REGEN
F154 8934E00 3496 MOV CRT_START,DI ; START ADDRESS SAVED IN GLOBAL
F158 C06620000 3497 MOV ACTIVE_PAGE,0 ; SET PAGE VALUE
F15D B9020 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 7404 3502 JE M11 ; BW CARD INIT
F16A 33C0 3503 XOR AX,AX ; FILL FOR GRAPHICS MODE
F16C EB05 3504 JMP SHORT M13 ; CLEAR BUFFER
F16E 3505 M11:
F16E B508 3506 MOV CH,08H ; BW CARD INIT
F170 3507 M12:
F170 B82007 3508 MOV AX,"**"*256 ; BUFFER SIZE ON BW CARD
F173 3509 M13:
F173 F3 3510 REP STOSW ; NO GRAPHICS INIT
F174 AB 3511 REP STOSW ; FILL CHAR FOR ALPHA
; CLEAR BUFFER
; FILL THE REGEN BUFFER WITH BLANKS
3512
3513 ;----- ENABLE VIDEO AND CORRECT PORT SETTING
3514
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,AH ; INTO AX REGISTER
F180 8BF0 3517 MOV SI,AX ; TABLE POINTER, INDEXED BY MODE
F182 8B166300 3518 MOV DX,ADDR_6845 ; PREPARE TO OUTPUT TO
; VIDEO ENABLE PORT
F186 83C204 3520 ADD DX,4
F189 2E8A84F40 3521 MOV AL,CS:[SI+OFFSET M7] ; SET VIDEO ENABLE PORT
F18E EE 3522 OUT DX,AL ; CLEAR BUFFER
F18F A26500 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 2E8A84ECF0 3528 MOV AL,CS:[SI+OFFSET M6]
F197 32E4 3529 XOR AH,AH
F199 A34A00 3530 MOV CRT_COLS,AX ; NUMBER OF COLUMNS IN THIS SCREEN
3531
3532 ;----- SET CURSOR POSITIONS
3533
F19C 81E60E00 3534 AND SI,0EH ; WORD OFFSET INTO CLEAR LENGTH TABLE
F1A0 2E8B8CE4F0 3535 MOV CRT_LEN,CX ; LENGTH TO CLEAR
F1A5 890E4C00 3536 MOV CX,2 ; SAVE LENGTH OF CRT -- NOT USED FOR BW
F1A9 B90800 3537 MOV DI,OFFSET CURSOR_POSN ; CLEAR ALL CURSOR POSITIONS
F1AC BF5000 3538 MOV DI,OFFSET CURSOR_POSN
F1AF 1E 3539 PUSH DS
F1B0 07 3540 POP ES ; ESTABLISH SEGMENT
F1B1 33C0 3541 XOR AX,AX ; ADDRESSING
F1B3 F3 3542 REP STOSW ; FILL WITH ZEROS
F1B4 AB

```

```

LOC 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                ; 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     M14:
F1C1 EE          3553     OUT    DX,AL          ; OUTPUT THE CORRECT VALUE TO 3D9 PORT
F1C2 A26600      3554     MOV     CRT_PALETTE,AL ; SAVE THE VALUE FOR FUTURE USE
3555
3556 |----- NORMAL RETURN FROM ALL VIDEO RETURNS
3557
F1C5             3558     VIDEO_RETURN:
F1C5 5F          3559     POP    DI             ;
F1C6 5E          3560     POP    SI             ;
F1C7 5B          3561     POP    BX             ;
F1C8             3562     M15:                 ; VIDEO_RETURN_C
F1C8 59          3563     POP    CX             ;
F1C9 5A          3564     POP    DX             ;
F1CA 1F          3565     POP    DS             ;
F1CB 07          3566     POP    ES             ; RECOVER SEGMENTS
F1CC CF          3567     IRET   ; ALL DONE
3568     SET_MODE      ENDP
-----
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
-----
F1CD             3576     SET_CTYPE      PROC    NEAR
F1CD B40A        3577     MOV     AH,10         ; 6845 REGISTER FOR CURSOR SET
F1CF 890E6000   3579     MOV     CURSOR_MODE,CX ; SAVE IN DATA AREA
F1D3 E80200     3580     CALL  M16            ; OUTPUT CX REG
F1D6 EBED       3581     JMP     VIDEO_RETURN
3582
3583 |----- THIS ROUTINE OUTPUTS THE CX REGISTER TO THE 6845 REGS NAMED IN AH
3584
F1D8             3585     M16:
F1D8 8B166300   3586     MOV     DX,ADDR_6845  ; ADDRESS REGISTER
F1DC 8AC4       3587     MOV     AL,AH         ; GET VALUE
F1DE EE         3588     OUT    DX,AL         ; REGISTER SET
F1DF 42         3589     INC    DX             ; DATA REGISTER
F1E0 8AC5       3590     MOV     AL,CH         ; DATA
F1E2 EE         3591     OUT    DX,AL         ;
F1E3 4A         3592     DEC    DX             ;
F1E4 8AC4       3593     MOV     AL,AH         ;
F1E6 FECD       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     SET_CTYPE      ENDP
-----
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
-----
F1EE             3611     SET_CPOS      PROC    NEAR
F1EE 8ACF       3612     MOV     CL,BH         ;
F1F0 32ED       3613     XOR    CH,CH         ; ESTABLISH LOOP COUNT
F1F2 D1E1       3614     SAL    CX,1          ; WORD OFFSET
F1F4 8BF1       3615     MOV     SI,CX         ; USE INDEX REGISTER
F1F6 895450     3616     MOV     [SI+OFFSET_CURSOR_POSN],DX ; SAVE THE POINTER
F1F9 383E6200   3618     CMP    ACTIVE_PAGE,BH ;
F1FD 7505       3619     JNZ    M17           ; SET_CPOS_RETURN
F1FF 8BC2       3620     MOV     AX,DX         ; GET ROW/COLUMN TO AX
F201 E80200     3621     CALL  M18            ; CURSOR SET
F204           3622     M17:             ; SET_CPOS_RETURN
F204 EBBF       3623     JMP     VIDEO_RETURN
F205           3624     SET_CPOS      ENDP
3625
3626 |----- SET CURSOR POSITION, AX HAS ROW/COLUMN FOR CURSOR
3627
F206             3628     M18     PROC    NEAR
F206 E87C00     3629     CALL  POSITION        ; DETERMINE LOCATION IN REGEN BUFFER
F209 8BC8       3630     MOV     CX,AX         ;
F20B 030E4E00   3631     ADD    CX,CRT_START  ; ADD IN THE START ADDR FOR THIS PAGE
F20F D1F9       3632     SAR    CX,1          ; DIVIDE BY 2 FOR CHAR ONLY COUNT
F211 B40E       3633     MOV     AH,14         ; REGISTER NUMBER FOR CURSOR
F213 E8C2FF     3634     CALL  M16            ; OUTPUT THE VALUE TO THE 6845
F216 C3         3635     RET
3636     M18     ENDP

```

```

3637 -----
3638 | ACT_DISP_PAGE |
3639 | THIS ROUTINE SETS THE ACTIVE DISPLAY PAGE, ALLOWING THE |
3640 | FULL USE OF THE RAM SET ASIDE FOR THE VIDEO ATTACHMENT; |
3641 | INPUT |
3642 | AL HAS THE NEW ACTIVE DISPLAY PAGE |
3643 | OUTPUT |
3644 | THE 6845 IS RESET TO DISPLAY THAT PAGE |
3645 -----
F217 ACT_DISP_PAGE PROC NEAR
F217 A26200 MOV ACTIVE_PAGE,AL ; SAVE ACTIVE PAGE VALUE
F21A 8B0E4C00 MOV CX,CRT_LEN ; GET SAVED LENGTH OF REGEN BUFFER
F21E 98 CEBW ; CONVERT AL TO WORD
F21F 50 PUSH AX ; SAVE PAGE VALUE
F220 F7E1 MUL CX ; DISPLAY PAGE TIMES REGEN LENGTH
F222 A34E00 MOV CRT_START,AX ; SAVE START ADDRESS FOR
3653 ; LATER REQUIREMENTS
F225 8BC8 MOV CX,AX ; START ADDRESS TO CX
F227 D1F9 SAR CX,1 ; DIVIDE BY 2 FOR 6845 HANDLING
F229 B40C MOV AH,12 ; 6845 REGISTER FOR START ADDRESS
F22B E8AAFF CALL M16 ;
F22E 5B POP BX ; RECOVER PAGE VALUE
F22F D1E3 SAL BX,1 ; *2 FOR WORD OFFSET
F231 8B4750 MOV AX,[BX + OFFSET CURSOR_POSN] ; GET CURSOR FOR THIS PAGE
F234 E8CFFF CALL M18 ; SET THE CURSOR POSITION
F237 E8B8 JMP SHORT VIDEO_RETURN
3663 ACT_DISP_PAGE ENDP
3664 -----
3665 | READ_CURSOR |
3666 | THIS ROUTINE READS THE CURRENT CURSOR VALUE FROM THE |
3667 | 6845, FORMATS IT, AND SENDS IT BACK TO THE CALLER |
3668 | INPUT |
3669 | BH - PAGE OF CURSOR |
3670 | OUTPUT |
3671 | DX - ROW, COLUMN OF THE CURRENT CURSOR POSITION |
3672 | CX - CURRENT CURSOR MODE |
3673 -----
F239 READ_CURSOR PROC NEAR
F239 8ADF MOV BL,BH
F23B 32FF XOR BH,BH
F23D D1E3 SAL BX,1 ; WORD OFFSET
F23F 8B5750 MOV DX,[BX+OFFSET CURSOR_POSN]
F242 8B0E6000 MOV CX,CURSOR_MODE
F246 5F POP D1
F247 5E POP SI
F248 5B POP BX
F249 58 POP AX ; DISCARD SAVED CX AND DX
F24A 58 POP AX
F24B 1F POP DS
F24C 07 POP ES
F24D CF POP IRET
3687 READ_CURSOR ENDP
3688 -----
3689 | SET COLOR |
3690 | THIS ROUTINE WILL ESTABLISH THE BACKGROUND COLOR, THE OVERSCAN |
3691 | COLOR, AND THE FOREGROUND COLOR SET FOR MEDIUM RESOLUTION |
3692 | GRAPHICS |
3693 | INPUT |
3694 | IBH1 HAS COLOR ID |
3695 | IF BH=0, THE BACKGROUND COLOR VALUE IS SET |
3696 | FROM THE LOW BITS OF BL (0-31) |
3697 | IF BH=1, THE PALETTE SELECTION IS MADE |
3698 | BASED ON THE LOW BIT OF BL: |
3699 | 0=GREEN, RED, YELLOW FOR COLORS 1,2,3 |
3700 | 1=BLUE, CYAN, MAGENTA FOR COLORS 1,2,3 |
3701 | |
3702 | (BL) HAS THE COLOR VALUE TO BE USED |
3703 | OUTPUT |
3704 | THE COLOR SELECTION IS UPDATED |
3705 -----
F24E SET_COLOR PROC NEAR
F24E 8B166300 MOV DX,ADDR_6845 ; I/O PORT FOR PALETTE
F252 83C205 ADD DX,5 ; OVERSCAN PORT
F255 A06600 MOV AL,CRT_PALETTE ; GET THE CURRENT PALETTE VALUE
F258 0AFF OR BH,BH ; IS THIS COLOR 0?
F25A 750E JNZ M20 ; OUTPUT COLOR 1
3712 |----- HANDLE COLOR 0 BY SETTING THE BACKGROUND COLOR
3713 |
3714 |
3715 | AND AL,0E0H ; TURN OFF LOW 5 BITS OF CURRENT
3716 | AND BL,01FH ; TURN OFF HIGH 3 BITS OF INPUT VALUE
3717 | OR AL,BL ; PUT VALUE INTO REGISTER
3718 | M19: ; OUTPUT THE PALETTE
3719 | OUT DX,AL ; OUTPUT COLOR SELECTION TO 309 PORT
3720 | MOV CRT_PALETTE,AL ; SAVE THE COLOR VALUE
3721 | JMP VIDEO_RETURN
3722 |
3723 |----- HANDLE COLOR 1 BY SELECTING THE PALETTE TO BE USED
3724 |
3725 | M20:
3726 | AND AL,0DFH ; TURN OFF PALETTE SELECT BIT
3727 | SHR BL,1 ; TEST THE LOW ORDER BIT OF BL
3728 | JNC M19 ; ALREADY DONE
3729 | OR AL,20H ; AL,20H
3730 | JMP M19 ; TURN ON PALETTE SELECT BIT
3731 | GO DO IT
3732 | SET_COLOR ENDP
3733 |----- VIDEO STATE
3734 | RETURNS THE CURRENT VIDEO STATE IN AX
3735 | AH = NUMBER OF COLUMNS ON THE SCREEN
3736 | AL = CURRENT VIDEO MODE
3737 | BH = CURRENT ACTIVE PAGE
3738 |-----
F274 VIDEO_STATE PROC NEAR
F274 8A264A00 MOV AH,BYTE PTR CRT_COLS ; GET NUMBER OF COLUMNS
F278 A04900 MOV AL,CRT_MODE ; CURRENT MODE
F27B 8A3E6200 MOV BH,ACTIVE_PAGE ; GET CURRENT ACTIVE PAGE
F27F 5F POP D1 ; RECOVER REGISTERS
F280 5E POP SI
F281 59 POP CX ; DISCARD SAVED BX
F282 E943FF JMP M15 ; RETURN TO CALLER
3747 VIDEO_STATE ENDP

```

```

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 3757 POSITION PROC NEAR
F286 53 3758 PUSH BX ; SAVE REGISTER
F286 8B08 3759 MOV BX,AX
F286 8A04 3760 MOV AL,AH ; ROWS TO AL
F28A F626A00 3761 MUL BYTE PTR CRT_COLS ; DETERMINE BYTES TO ROW
F286 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
3767 |-----|
3768 | POSITION | ENDP
3769 |-----|
3770 | SCROLL_UP |
3771 | THIS ROUTINE MOVES A BLOCK OF CHARACTERS UP |
3772 | ON THE SCREEN |
3773 | INPUT |
3774 | (AH) = CURRENT CRT MODE |
3775 | (AL) = NUMBER OF ROWS TO SCROLL |
3776 | (CX) = ROW/COLUMN OF UPPER LEFT CORNER |
3777 | (DX) = ROW/COLUMN OF LOWER RIGHT CORNER |
3778 | (BH) = ATTRIBUTE TO BE USED ON BLANKED LINE |
3779 | (DS) = DATA SEGMENT |
3780 | (ES) = REGEN BUFFER SEGMENT |
3781 | OUTPUT |
3782 | NONE -- THE REGEN BUFFER IS MODIFIED |
3783 |-----|
3784 | ASSUME CS:CODE,DS:DATA,ES:DATA |
F296 3784 SCROLL_UP PROC NEAR
F296 8AD8 3785 MOV BL,AL ; SAVE LINE COUNT IN BL
F296 80FC04 3786 CMP AH,4 ; TEST FOR GRAPHICS MODE
F296 7208 3787 JC NI ; HANDLE SEPARATELY
F29D 80FC07 3788 CMP AH,7 ; TEST FOR BW CARD
F2A0 7403 3789 JE NI
F2A2 E9F001 3790 JMP GRAPHICS_UP
F2A5 3791 NI: ; UP CONTINUE
F2A5 53 3792 PUSH BX ; SAVE FILL ATTRIBUTE IN BH
F2A6 8BC1 3793 MOV AX,CX ; UPPER LEFT POSITION
F2A8 E83700 3794 CALL SCROLL_POSITION ; DO SETUP FOR SCROLL
F2AB 7431 3795 JZ NT ; BLANK FIELD
F2AD 03F0 3796 ADD SI,AX ; FROM ADDRESS
F2AF 8AE6 3797 MOV AH,0H ; # ROWS IN BLOCK
F2B1 2AE3 3798 SUB AH,BL ; # ROWS TO BE MOVED
F2B3 3799 N2: ; ROW LOOP
F2B3 E87200 3800 CALL N10 ; MOVE ONE ROW
F2B6 03F5 3801 ADD SI,BP
F2B8 03FD 3802 ADD DI,BP ; POINT TO NEXT LINE IN BLOCK
F2BA FECC 3803 DEC AH ; COUNT OF LINES TO MOVE
F2BC 75F5 3804 JNZ N2 ; ROW_LOOP
F2BE 3805 N3: ; CLEAR ENTRY
F2BE 58 3806 POP AX ; RECOVER ATTRIBUTE IN AH
F2BF 8020 3807 MOV AL,' ' ; FILL WITH BLANKS
F2C1 E86D00 3808 CALL N4: ; CLEAR_LOOP
F2C4 03FD 3809 ADD DI,BP ; CLEAR THE ROW
F2C6 FECB 3810 DEC BL ; POINT TO NEXT LINE
F2C8 75F7 3811 DEC BL ; COUNTER OF LINES TO SCROLL
F2CA 3812 N5: ; CLEAR_LOOP
F2CA 3813 JMP SCROLL_END
F2CA E88C07 3814 CALL DOS
F2CD 803E490007 3815 CMP CRT_MODE,7 ; IS THIS THE BLACK AND WHITE CARD
F2D2 7407 3816 JE N6 ; IF SO, SKIP THE MUE RESET
F2D4 A06500 3817 MOV AL,CRT_MODE_SET ; GET THE VALUE OF THE MODE SET
F2D7 8AD803 3818 MOV DX,03D8H ; ALWAYS SET COLOR CARD PORT
F2DA EE 3819 OUT DX,AL
F2DB 3820 N6: ; VIDEO_RET_HERE
F2DB E9E7FE 3821 JMP VIDEO_RETURN
F2DE 3822 N7: ; BLANK FIELD
F2DE 8ADE 3823 MOV BL,0H ; GET ROW COUNT
F2E0 EB0C 3824 JMP ENDP ; GO CLEAR THAT AREA
3825 SCROLL_UP ENDP
3826
3827 |-----|
3828 | HANDLE COMMON SCROLL SET UP HERE |
F2E2 3829 SCROLL_POSITION PROC NEAR
F2E2 803E490002 3830 CMP CRT_MODE,2 ; TEST FOR SPECIAL CASE HERE
F2E7 7218 3831 JB N9 ; HAVE TO HANDLE 80X25 SEPARATELY
F2E9 803E490003 3832 CMP CRT_MODE,3
F2EE 7111 3833 JA N9
3834
3835 |-----|
3836 | 80X25 COLOR CARD SCROLL |
F2F0 52 3837 PUSH DX
F2F1 BADA03 3838 MOV DX,3DAH ; GUARANTEED TO BE COLOR CARD HERE
F2F4 50 3839 PUSH AX
F2F5 3840 N8: ; WAIT_DISP_ENABLE
F2F5 EC 3841 IN AL,DX ; GET PORT
F2F6 A808 3842 TEST AL,8 ; WAIT FOR VERTICAL RETRACE
F2F8 74F9 3843 JZ N8 ; WAIT_DISP_ENABLE
F2FA 8025 3844 MOV AL,25H
F2FC B2D8 3845 MOV DL,0D8H ; DX=3DB
F2FE EE 3846 OUT DX,AL ; TURN OFF VIDEO
F2FF 58 3847 POP AX ; DURING VERTICAL RETRACE
F300 5A 3848 POP DX
F301 3849 N9:
F301 E881FF 3850 CALL POSITION
F304 0304E00 3851 ADD AX,CRT_START ; CONVERT TO REGEN POINTER
F308 8BF8 3852 MOV DI,AX ; OFFSET OF ACTIVE PAGE
F30A 8BF0 3853 MOV SI,AX ; TO ADDRESS FOR SCROLL
F30C 2801 3854 SUB DI,CX ; FROM ADDRESS FOR SCROLL
F30E FEC6 3855 INC DH ; DX = # ROWS, #COLS IN BLOCK
F310 FEC2 3856 INC DL ; INCREMENT FOR 0 ORIGIN
F312 32ED 3857 XOR CH,CH ; SET HIGH BYTE OF COUNT TO ZERO
F314 8B2E4A00 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 F626A00 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 1F            3864      POP     DS          ; FOR BOTH POINTERS
F324 80FB00        3865      CMP     BL,0        ; 0 SCROLL MEANS BLANK FIELD
F327 C3            3866      RET     ; RETURN WITH FLAGS SET
                   3867      SCROLL_POSITION ENDP
                   3868
                   3869      ;----- MOVE_ROW
                   3870
F328              3871      N10     PROC     NEAR
F328 8ACA          3872      MOV     CL,DL       ; GET # OF COLS TO MOVE
F32A 56            3873      PUSH   SI
F32B 57            3874      PUSH   DI
F32C F3            3875      REP     MOVSW       ; SAVE START ADDRESS
F32D A5            3876      POP     DI
F32E 5F            3877      POP     SI         ; RECOVER ADDRESSES
F330 C3            3878      RET
                   3879      N10     ENDP
                   3880
                   3881      ;----- CLEAR_ROW
                   3882
F331              3883      N11     PROC     NEAR
F331 8ACA          3884      MOV     CL,DL       ; GET # COLUMNS TO CLEAR
F333 57            3885      PUSH   D1
F334 F3            3886      REP     STOSW      ; STORE THE FILL CHARACTER
F335 AB            3887
F336 5F            3888      POP     DI
F337 C3            3889      RET
                   3890      N11     ENDP
                   3891      ;----- SCROLL_DOWN
                   3892      ; THIS ROUTINE MOVES THE CHARACTERS WITHIN A
                   3893      ; DEFINED BLOCK DOWN ON THE SCREEN, FILLING THE
                   3894      ; TOP LINES WITH A DEFINED CHARACTER
                   3895      ; INPUT
                   3896      ; (AH) = CURRENT CRT MODE
                   3897      ; (AL) = NUMBER OF LINES TO SCROLL
                   3898      ; (CX) = UPPER LEFT CORNER OF REGION
                   3899      ; (DX) = LOWER RIGHT CORNER OF REGION
                   3900      ; (BH) = FILL CHARACTER
                   3901      ; (DS) = DATA SEGMENT
                   3902      ; (ES) = REGEN SEGMENT
                   3903      ; OUTPUT
                   3904      ; NONE -- SCREEN IS SCROLLED
                   3905      ;-----
F338              3906      SCROLL_DOWN PROC NEAR
F338 FD            3907      STD
F339 8ADB          3908      MOV     BL,AL       ; DIRECTION FOR SCROLL DOWN
F33B 80FC04        3909      CMP     AH,4        ; LINE COUNT TO BL
F33E T208          3910      JC     N12          ; TEST FOR GRAPHICS
F340 80FC07        3911      CMP     AH,7        ; TEST FOR BW CARD
F343 T403          3912      JE     N12
F345 E9A601        3913      JMP     GRAPHICS_DOWN
F348 53            3914      N12:
F349 8BC2          3915      PUSH   BX          ; CONTINUE DOWN
F34B E894FF        3916      MOV     AX,DX       ; SAVE ATTRIBUTE IN BH
F34E T420          3917      CALL   SCROLL_POSITION ; GET REGEN LOCATION
F350 2BF0          3918      JZ     N15
F352 8AE6          3919      SUB     SI,AX       ; SI IS FROM ADDRESS
F354 2AE3          3920      MOV     AH,DH       ; GET TOTAL # ROWS
F356 E8CFFF        3921      SUB     AH,BL       ; COUNT TO MOVE IN SCROLL
F359 2BF5          3922      CALL   N10         ; MOVE ONE ROW
F35B 2BFD          3923      SUB     SI,BP
F35D FECC          3924      SUB     D1,BP
F35F 75F5          3925      DEC     AH
F361 5B            3926      JNZ    N13
F362 B020          3927      N13:
F364 EBCAFF        3928      CALL   N10
F367 2BFD          3929      SUB     D1,BP
F369 FECB          3930      DEC     BL
F36B 75F7          3931      JNZ    N15
F36D E95AFF        3932      JMP     N5
F370 33            3933      N15:
F370 8ADE          3934      MOV     BL,DH
F372 EBED          3935      JMP     N14
                   3936      N14:
F372 EBED          3937      POP     AX
F372 EBED          3938      MOV     AL,' '
F372 EBED          3939      N15:
F372 EBED          3940      CALL   N11
F372 EBED          3941      SUB     D1,BP
F372 EBED          3942      DEC     BL
F372 EBED          3943      JNZ    N15
F372 EBED          3944      JMP     N5
F372 EBED          3945      N16:
F372 EBED          3946      MOV     BL,DH
F372 EBED          3947      JMP     N14
F372 EBED          3948      N14:
F372 EBED          3949      MOV     BL,DH
F372 EBED          3950      JMP     N14
F372 EBED          3951      N14:
F372 EBED          3952      MOV     BL,DH
F372 EBED          3953      JMP     N14
F372 EBED          3954      N14:
F372 EBED          3955      MOV     BL,DH
F372 EBED          3956      JMP     N14
F372 EBED          3957      N14:
F372 EBED          3958      MOV     BL,DH
F372 EBED          3959      JMP     N14
F372 EBED          3960      N14:
F372 EBED          3961      MOV     BL,DH
F372 EBED          3962      JMP     N14
F372 EBED          3963      N14:
F372 EBED          3964      MOV     BL,DH
F372 EBED          3965      JMP     N14
F372 EBED          3966      N14:
F372 EBED          3967      MOV     BL,DH
F372 EBED          3968      JMP     N14
F372 EBED          3969      N14:
F372 EBED          3970      MOV     BL,DH
F372 EBED          3971      JMP     N14
F372 EBED          3972      N14:
F372 EBED          3973      MOV     BL,DH
F372 EBED          3974      JMP     N14
F372 EBED          3975      N14:
F372 EBED          3976      MOV     BL,DH
F372 EBED          3977      JMP     N14
F372 EBED          3978      N14:
F372 EBED          3979      MOV     BL,DH
F372 EBED          3980      JMP     N14
F372 EBED          3981      N14:
F372 EBED          3982      MOV     BL,DH
F372 EBED          3983      JMP     N14
F372 EBED          3984      N14:
F372 EBED          3985      MOV     BL,DH
F372 EBED          3986      JMP     N14
F372 EBED          3987      N14:
F372 EBED          3988      MOV     BL,DH
F372 EBED          3989      JMP     N14
F372 EBED          3990      N14:
F372 EBED          3991      MOV     BL,DH
F372 EBED          3992      JMP     N14
F372 EBED          3993      N14:
F372 EBED          3994      MOV     BL,DH
F372 EBED          3995      JMP     N14
F372 EBED          3996      N14:
F372 EBED          3997      MOV     BL,DH
F372 EBED          3998      JMP     N14
F372 EBED          3999      N14:
F372 EBED          4000      MOV     BL,DH
F372 EBED          4001      JMP     N14

```

```

3941 ;-----
3942 ; READ_AC_CURRENT                               ;
3943 ; TR: ROUTINE READS THE ATTRIBUTE AND CHARACTER ;
3944 ; AT THE CURRENT CURSOR POSITION AND RETURNS THEM ;
3945 ; TO THE CALLER                               ;
3946 ; INPUT                                         ;
3947 ; (AH) = CURRENT CRT MODE                     ;
3948 ; (BH) = DISPLAY PAGE ( ALPHA MODES ONLY )   ;
3949 ; (DS) = DATA SEGMENT                       ;
3950 ; (ES) = REGEN SEGMENT                       ;
3951 ; OUTPUT                                       ;
3952 ; (AL) = CHAR READ                           ;
3953 ; (AH) = ATTRIBUTE READ                       ;
3954 ;-----
3955 ; ASSUME CS:CODE,DS:DATA,ES:DATA
F374 READ_AC_CURRENT PROC NEAR
F374 80FC04 3957 CMP AH,4 ; IS THIS GRAPHICS
F377 7208 3958 JC P1 ;
F379 80FC07 3959 CMP AH,7 ; IS THIS BW CARD
F37C 7403 3960 JE P1 ;
F37E E9A802 3961 JMP GRAPHICS_READ ;
F381 3962 P1: 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
F38A 83C206 3969 ADD DX,6 ; POINT AT STATUS PORT
F38D 06 3970 PUSH DS ;
F38E 1F 3971 POP DS ; GET SEGMENT FOR QUICK ACCESS
F38F 3972 P2: ; WAIT FOR RETRACE LOW
F38F EC 3973 IN AL,DX ; GET STATUS
F390 A801 3974 TEST AL,1 ; IS IT LOW
F392 75FB 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
F39A AD 3981 LODSW ; GET THE CHAR/ATTR
F39B E927FE 3982 JMP VIDEO_RETURN
3983 READ_AC_CURRENT ENDP
3984
F39E FIND_POSITION PROC NEAR
F39E 8ACF 3985 MOV CL,BH ; DISPLAY PAGE TO CX
F3A0 32ED 3987 XOR CH,CH ;
F3A2 8BF1 3988 MOV SI,CX ; MOVE TO SI FOR INDEX
F3A4 D1E6 3989 SAL SI,1 ; * 2 FOR WORD OFFSET
F3A6 8B4450 3990 MOV AX,[SI+OFFSET_CURSOR_POSN] ; GET ROW/COLUMN OF THAT PAGE
F3A9 33DB 3991 XOR BX,BX ; SET START ADDRESS TO ZERO
F3AB E306 3992 JCXZ P5 ; NO PAGE
F3AD 031E4C00 3993 P4: ADD BX,CRT_LEN ; PAGE LOOP
F3B1 E2FA 3994 LOOP P4 ; LENGTH OF BUFFER
F3B3 3995 P5: ; NO PAGE
F3B3 E8CFFE 3996 CALL POSITION ; DETERMINE LOCATION IN REGEN
F3B6 03DB 3998 ADD BX,AX ; ADD TO START OF REGEN
F3B8 C3 3999 RET
4000 FIND_POSITION ENDP
4001 ;-----
4002 ; WRITE_AC_CURRENT                               ;
4003 ; THIS ROUTINE WRITES THE ATTRIBUTE           ;
4004 ; AND CHARACTER AT THE CURRENT CURSOR        ;
4005 ; POSITION                                       ;
4006 ; INPUT                                         ;
4007 ; (AH) = CURRENT CRT MODE                     ;
4008 ; (BH) = DISPLAY PAGE                       ;
4009 ; (CX) = COUNT OF CHARACTERS TO WRITE        ;
4010 ; (AL) = CHAR TO WRITE                       ;
4011 ; (BL) = ATTRIBUTE OF CHAR TO WRITE          ;
4012 ; (DS) = DATA SEGMENT                       ;
4013 ; (ES) = REGEN SEGMENT                       ;
4014 ; OUTPUT                                       ;
4015 ; NONE                                         ;
4016 ;-----
F3B9 WRITE_AC_CURRENT PROC NEAR
F3B9 80FC04 4018 CMP AH,4 ; IS THIS GRAPHICS
F3BC 7208 4019 JC P6 ;
F3BE 80FC07 4020 CMP AH,7 ; IS THIS BW CARD
F3C1 7403 4021 JE P6 ;
F3C3 E9B201 4022 JMP GRAPHICS_WRITE ;
F3C6 8AE3 4024 P6: MOV AH,BL ; WRITE AC_CONTINUE
F3C8 50 4025 PUSH AX ; GET ATTRIBUTE TO AH
F3C9 51 4026 PUSH CX ; SAVE ON STACK
F3CA EBD0FF 4027 CALL FIND_POSITION ; SAVE WRITE COUNT
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 4031 P7: ; WRITE_LOOP
4032
4033 ;----- WAIT FOR HORIZONTAL RETRACE
4034
F3D1 8B166300 4035 MOV DX,ADDR_6845 ; GET BASE ADDRESS
F3D5 83C206 4036 ADD DX,6 ; POINT AT STATUS PORT
F3D8 4037 P8: ;
F3D8 EC 4038 IN AL,DX ; GET STATUS
F3D9 A801 4039 TEST AL,1 ; IS IT LOW
F3DB 75FB 4040 JNZ P8 ; WAIT UNTIL IT IS
F3DD FA 4041 CLI ; NO MORE INTERRUPTS
F3DE 4042 P9: ;
F3DE EC 4043 IN AL,DX ; GET STATUS
F3DF A801 4044 TEST AL,1 ; IS IT HIGH
F3E1 74FB 4045 JZ P9 ; WAIT UNTIL IT IS
F3E3 8BC3 4046 MOV AX,BX ; RECOVER THE CHAR/ATTR
F3E5 AB 4047 STOSW ; PUT THE CHAR/ATTR
F3E6 FB 4048 STI ; INTERRUPTS BACK ON
F3E7 EEB8 4049 JMP VIDEO_RETURN ; AS MANY TIMES AS REQUESTED
F3E9 E9DFD0 4050
4051 WRITE_AC_CURRENT ENDP

```



```

4052 :-----:
4053 : WRITE_C_CURRENT                               :
4054 : THIS ROUTINE WRITES THE CHARACTER AT         :
4055 : THE CURRENT CURSOR POSITION, ATTRIBUTE       :
4056 : UNCHANGED                                    :
4057 : INPUT                                          :
4058 : (AH) = CURRENT CRT MODE                      :
4059 : (BH) = DISPLAY PAGE                         :
4060 : (CX) = COUNT OF CHARACTERS TO WRITE        :
4061 : (AL) = CHAR TO WRITE                        :
4062 : (DI) = DATA SEGMENT                       :
4063 : (ES) = REGEN SEGMENT                       :
4064 : OUTPUT                                        :
4065 : NONE                                         :
4066 :-----:
F3EC      4067 WRITE_C_CURRENT PROC NEAR
F3EC 80FC04 4068     CMP     AH,4           ; IS THIS GRAPHICS
F3EF 7208 4069     JC      P10
F3F1 80FC07 4070     CMP     AH,7           ; IS THIS BW CARD
F3F4 7403 4071     JE      P10
F3F6 E97F01 4072     JMP     GRAPHICS_WRITE
F3F9      P10:
F3F9 50 4074     PUSH    AX           ; SAVE ON STACK
F3FA 51 4075     PUSH    CX           ; SAVE WRITE COUNT
F3FB EBA0FF 4076     CALL   FIND_POSITION
F3FE 8BFB 4077     MOV     DI,BX        ; ADDRESS TO DI
F400 59 4078     POP     CX           ; WRITE COUNT
F401 5B 4079     POP     BX           ; BL HAS CHAR TO WRITE
F402      4080     WRITE_LOOP
4081      4081     P11:
4082 :-----:
F402 8B166300 4082     MOV     DX,ADDR_6845    ; GET BASE ADDRESS
F406 83C206 4083     ADD     DX,6           ; POINT AT STATUS PORT
F409      4084     P12:
F409 EC 4087     IN      AL,DX        ; GET STATUS
F40A A801 4088     TEST   AL,1           ; IS IT LOW
F40C 75FB 4089     JNZ    P12        ; WAIT UNTIL IT IS
F40E FA 4090     CLI      P12        ; NO MORE INTERRUPTS
F40F      4091     P13:
F40F EC 4092     IN      AL,DX        ; GET STATUS
F410 A801 4093     TEST   AL,1           ; IS IT HIGH
F412 74FB 4094     JZ      P13        ; WAIT UNTIL IT IS
F414 8AC3 4095     MOV     AL,BL        ; RECOVER CHAR
F416 AA 4096     STOSB          ; PUT THE CHAR/ATTR
F417 FB 4097     STI      P13        ; INTERRUPTS BACK ON
F418 47 4098     INC     DI           ; BUMP POINTER PAST ATTRIBUTE
F419 E2E7 4099     LOOP   P11           ; AS MANY TIMES AS REQUESTED
F41B E9A7FD 4100     JMP     VIDEO_RETURN
4102      4101     WRITE_C_CURRENT ENDP
4103 :-----:
4103 : READ DOT -- WRITE DOT                         :
4104 : THESE ROUTINES WILL WRITE A DOT, OR READ THE DOT AT :
4105 : THE INDICATED LOCATION                       :
4106 : ENTRY --                                     :
4107 : DX = ROW (0-199) (THE ACTUAL VALUE DEPENDS ON THE MODE) :
4108 : CX = COLUMN ( 0-639) (THE VALUES ARE NOT RANGE CHECKED) :
4109 : AL = DOT VALUE TO WRITE (1,2 OR 4 BITS DEPENDING ON MODE, :
4110 : REQ'D FOR WRITE DOT ONLY, RIGHT JUSTIFIED) :
4111 : BIT 7 OF AL=1 INDICATES XOR THE VALUE INTO THE LOCATION :
4112 : DS = DATA SEGMENT                          :
4113 : ES = REGEN SEGMENT                          :
4114 :                                              :
4115 : EXIT                                         :
4116 : AL = DOT VALUE READ, RIGHT JUSTIFIED, READ ONLY :
4117 :-----:
F41E      4118     ASSUME  CS:CODE,DS:DATA,ES:DATA
F41E E83100 4119     READ_DOT PROC NEAR
F421 268A04 4120     CALL   R3
F422 22C4 4121     MOV     AL,ES:[SI]
F423 D2E0 4122     AND     AL,AH
F424 8ACE 4123     SHL    AL,CL
F425 D2C0 4124     MOV     CL,DH
F426 E96FD 4125     ROL    AL,CL
F427      4126     JMP     VIDEO_RETURN
F42F      4127     READ_DOT ENDP
F42F      4128
F42F 50 4129     WRITE_DOT PROC NEAR
F430 50 4130     PUSH   AX
F431 50 4131     PUSH   AX
F432 E81E00 4132     CALL   R3
F434 D2E8 4133     SHR    AL,CL
F436 22C4 4134     AND     AL,AH
F438 268A0C 4135     MOV     CL,ES:[SI]
F43B 5B 4136     POP     BX
F43C F6C380 4137     TEST   BL,80H
F43F 750D 4138     JNZ    R2
F441 F404 4139     NOT    AH
F443 22CC 4140     AND     CL,AH
F445 0AC1 4141     OR     AL,AH
F447      4142     R1:
F447 268804 4143     MOV     ES:[SI],AL
F44A 5B 4144     POP     AX
F44B E977FD 4145     JMP     VIDEO_RETURN
F44E      4146     R2:
F44E 32C1 4147     XOR    AL,CL
F450 EBF5 4148     JMP     R1
4149      4149     WRITE_DOT ENDP

```

```

4150 ;-----
4151 ; THIS SUBROUTINE DETERMINES THE REGEN BYTE LOCATION ;
4152 ; OF THE INDICATED ROW COLUMN VALUE IN GRAPHICS MODE. ;
4153 ENTRY -- ;
4154 ; DX = ROW VALUE (0-199) ;
4155 ; CX = COLUMN VALUE (0-639) ;
4156 ; EXIT -- ;
4157 ; SI = OFFSET INTO REGEN BUFFER FOR BYTE OF INTEREST ;
4158 ; AH = MASK TO STRIP OFF THE BITS OF INTEREST ;
4159 ; CL = BITS TO SHIFT TO RIGHT JUSTIFY THE MASK IN AH ;
4160 ; DH = # BITS IN RESULT ;
4161 ;-----
F452 R3 PROC NEAR
F452 53 PUSH BX ; SAVE BX DURING OPERATION
F453 50 PUSH AX ; WILL SAVE AL DURING OPERATION
4165 ;-----
4166 ;----- DETERMINE 1ST BYTE IN IDICATED ROW BY MULTIPLYING ROW VALUE BY 40
4167 ;----- ( LOW BIT OF ROW DETERMINES EVEN/ODD, 80 BYTES/ROW
4168 ;-----
F454 B028 MOV AL,40
F456 52 PUSH DX ; SAVE ROW VALUE
F457 80E2FE AND DL,0FEH ; STRIP OFF ODD/EVEN BIT
F45A F6E2 MUL DL ; AX HAS ADDRESS OF 1ST BYTE
4173 ; OF INDICATED ROW
4174 POP DX ; RECOVER IT
F45D F6C201 TEST DL,1 ; TEST FOR EVEN/ODD
F460 7403 JZ R4 ; JUMP IF EVEN ROW
F462 050020 ADD AX,2000H ; STRIP TO LOCATION OF ODD ROWS
F465 ; EVEN ROW
F465 8BF0 R4: MOV SI,AX ; MOVE POINTER TO SI
F467 58 POP AX ; RECOVER AL VALUE
F468 8BD1 MOV DX,CX ; COLUMN VALUE TO DX
4182 ;-----
4183 ;----- DETERMINE GRAPHICS MODE CURRENTLY IN EFFECT
4184 ;-----
4185 ;-----
4186 ; SET UP THE REGISTERS ACCORDING TO THE MODE ;
4187 ; CH = MASK FOR LOW OF COLUMN ADDRESS ( 7/3 FOR HIGH/MED RES) ;
4188 ; CL = # OF ADDRESS BITS IN COLUMN VALUE ( 3/2 FOR H/M) ;
4189 ; BL = MASK TO SELECT BITS FROM POINTED BYTE (80H/COH FOR H/M) ;
4190 ; BH = NUMBER OF VALID BITS IN POINTED BYTE ( 1/2 FOR H/M) ;
4191 ;-----
F46A BBC002 MOV BX,2C0H
F46D B90203 MOV CX,302H ; SET PARMs FOR MED RES
F470 803E490006 JMC CRT_MODE,6
F475 7206 JNC RS ; HANDLE IF MED ARES
F477 BB8001 MOV BX,180H
F47A B90307 MOV CX,703H ; SET PARMs FOR HIGH RES
4199 ;-----
4200 ;----- DETERMINE BIT OFFSET IN BYTE FROM COLUMN MASK
4201 ;-----
F47D R5:
F47D 22EA AND CH,DL ; ADDRESS OF PEL WITHIN BYTE TO CH
4203 ;-----
4204 ;----- DETERMINE BYTE OFFSET FOR THIS LOCATION IN COLUMN
4205 ;-----
F47F D3EA SHR DX,CL ; SHIFT BY CORRECT AMOUNT
F481 03F2 ADD SI,DX ; INCREMENT THE POINTER
F483 8AF7 MOV DH,BH ; GET THE # OF BITS IN RESULT TO DH
4210 ;-----
4211 ;----- MULTIPLY BH (VALID BITS IN BYTE) BY CH (BIT OFFSET)
4212 ;-----
F485 2AC9 SUB CL,CL ; ZERO INTO STORAGE LOCATION
F487 R6:
F487 D0C8 ROR AL,1 ; LEFT JUSTIFY THE VALUE
; IN AL (FOR WRITE)
F489 02CD ADD CL,CH ; ADD IN THE BIT OFFSET VALUE
F48B FECF DEC BH ; LOOP CONTROL
F48D 75F8 JNZ R6 ; ON EXIT, CL HAS SHIFT COUNT
; TO RESTORE BITS
F48F 8AE3 MOV AH,BL ; GET MASK TO AH
F491 D2EC SHR AH,CL ; MOVE THE MASK TO CORRECT LOCATION
F493 5B POP BX ; RECOVER REG
F494 C3 RET ; RETURN WITH EVERYTHING SET UP
4225 R3 ENDP
4226 ;-----
4227 ; SCROLL UP ;
4228 ; THIS ROUTINE SCROLLS UP THE INFORMATION ON THE CRT ;
4229 ; ENTRY ;
4230 ; CH,CL = UPPER LEFT CORNER OF REGION TO SCROLL ;
4231 ; DH,DL = LOWER RIGHT CORNER OF REGION TO SCROLL ;
4232 ; BOTH OF THE ABOVE ARE IN CHARACTER POSITIONS ;
4233 ; BH = FILL VALUE FOR BLANKED LINES ;
4234 ; AL = # LINES TO SCROLL (AL=0 MEANS BLANK THE ENTIRE ;
4235 ; FIELD) ;
4236 ; DS = DATA SEGMENT ;
4237 ; ES = REGEN SEGMENT ;
4238 ; EXIT ;
4239 ; NOTHING, THE SCREEN IS SCROLLED ;
4240 ;-----
F495 GRAPHICS UP PROC NEAR
F495 8ADB MOV BL,AL ; SAVE LINE COUNT IN BL
F497 8BC1 MOV AX,CX ; GET UPPER LEFT POSITION INTO AX REG
4244 ;-----
4245 ;----- USE CHARACTER SUBROUTINE FOR POSITIONING
4246 ;----- ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE
4247 ;-----
F499 E86902 CALL GRAPH_POSN
F49C 8BF8 MOV DI,AX ; SAVE RESULT AS DESTINATION ADDRESS
4250 ;-----
4251 ;----- DETERMINE SIZE OF WINDOW
4252 ;-----
F49E 2BD1 SUB DX,CX
F4A0 81C20101 ADD DX,101H ; ADJUST VALUES
F4A4 D0E6 SAL DH,1 ; MULTIPLY # ROWS BY 4
; SINCE 8 VERT DOTS/CHAR
F4A6 D0E6 SAL DH,1 ; AND EVEN/ODD ROWS
4258 ;-----
4259 ;----- DETERMINE CRT MODE
4260 ;-----
F4A8 803E490006 CMP CRT_MODE,6 ; TEST FOR MEDIUM RES
F4AD 7304 JNC R7 ; FIND_SOURCE
4262 ;-----

```

```

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

4263
4264 |----- MEDIUM RES UP
4265
F44F 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
F4B3 06           4272          | GET SEGMENTS BOTH POINTING TO REGEN
F4B4 1F           4273          POP     DS          |
F4B5 2AED        4274          SUB     CH,CH       | ZERO TO HIGH OF COUNT REG
F4B7 D0E3        4275          SAL    BL,I         | MULTIPLY NUMBER OF LINES BY 4
F4B9 D0E3        4276          SAL    BL,I         |
F4BB 142D        4277          JZ     R11          | IF ZERO, THEN BLANK ENTIRE FIELD
F4BD 8AC3        4278          MOV    AL,BL        | GET NUMBER OF LINES IN AL
F4BF B450        4279          MOV    AH,80        | 80 BYTES/ROW
F4C1 F6E4        4280          MUL    AH           | DETERMINE OFFSET TO SOURCE
F4C3 8BF7        4281          MOV    SI,DI        | SET UP SOURCE
F4C5 03F0        4282          ADD    SI,AX        | ADD IN OFFSET TO IT
F4C7 8AE6        4283          MOV    AH,DH        | NUMBER OF ROWS IN FIELD
F4C9 2AE3        4284          SUB    AH,BL        | DETERMINE NUMBER TO MOVE
4285
4286 |----- LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD FIELDS
4287
F4CB E88000       4288          R8:          | ROW LOOP
F4CE 81E8B01F    4289          CALL   R17          | MOVE ONE ROW
F4D2 81EFB01F    4290          SUB    DI,2000H-80 | MOVE TO NEXT ROW
F4D6 FECC        4291          DEC    AH           | NUMBER OF ROWS TO MOVE
F4D8 75F1        4292          JNZ   R8           | CONTINUE TILL ALL MOVED
4293
4295 |----- FILL IN THE VACATED LINE(S)
4296
F4DA            4297          R9:          | CLEAR ENTRY
F4DA 8AC7        4298          MOV    AL,BH        | ATTRIBUTE TO FILL WITH
F4DC            4299          R10:         |
F4DC E88800       4300          CALL   R18          | CLEAR THAT ROW
F4DE 81EFB01F    4301          SUB    DI,2000H-80 | POINT TO NEXT LINE
F4E3 FECD        4302          DEC    BL           | NUMBER OF LINES TO FILL
F4E5 75F5        4303          JNZ   R10          | CLEAR LOOP
F4E7 E9DBFC      4304          JMP    VIDEO_RETURN | EVERYTHING DONE
F4EA            4305          R11:         | BLANK FIELD
F4EA 8ADE        4306          MOV    BL,DH        | SET BLANK COUNT TO
F4EC EBEC        4307          | EVERYTHING IN FIELD
F4EC EBEC        4308          JMP    R9           | CLEAR THE FIELD
4309          GRAPHICS_UP  ENDP
-----
4311 | SCROLL DOWN
4312 | THIS ROUTINE SCROLLS DOWN THE INFORMATION ON THE CRT
4313 | ENTRY
4314 | CH,CL = UPPER LEFT CORNER OF REGION TO SCROLL
4315 | DH,DL = LOWER RIGHT CORNER OF REGION TO SCROLL
4316 | BOTH OF THE ABOVE ARE IN CHARACTER POSITIONS
4317 | BH = FILL VALUE FOR BLANKED LINES
4318 | AL = # LINES TO SCROLL (AL=0 MEANS BLANK THE ENTIRE
4319 | FIELD)
4320 | DS = DATA SEGMENT
4321 | ES = REGEN SEGMENT
4322 | EXIT
4323 | NOTHING, THE SCREEN IS SCROLLED
4324 |-----
4325          GRAPHICS_DOWN  PROC   NEAR
F4EE            4326          STD
F4EF 8AD8        4327          MOV    BL,AL        | SET DIRECTION
F4F1 8BC2        4328          MOV    AX,DX        | SAVE LINE COUNT IN BL
4329          | GET LOWER RIGHT POSITION INTO AX REG
4330 |----- USE CHARACTER SUBROUTINE FOR POSITIONING
4331 |----- ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE
F4F3 E80F02      4332          CALL   GRAPH_POSN
F4F6 8BF8        4333          MOV    DI,AX        | SAVE RESULT AS DESTINATION ADDRESS
4334
4335 |----- DETERMINE SIZE OF WINDOW
4336
F4F8 2BD1        4337          SUB    DX,CX
F4FA 81C20101    4338          ADD    DX,101H     | ADJUST VALUES
F4FE D0E6        4339          SAL    DH,I         | MULTIPLY # ROWS BY 4
4340          | SINCE 8 VERT DOTS/CHAR
4341          | AND EVEN/ODD ROWS
F500 D0E6        4342          SAL    DH,I         |
4343
4344 |----- DETERMINE CRT MODE
4345
F502 803E490006  4346          CMP    CRT_MODE,6  | TEST FOR MEDIUM RES
F507 1305        4347          JNC   R12          | FIND_SOURCE_DOWN
4348
4349 |----- MEDIUM RES DOWN
4350
F509 D0E2        4351          SAL    DL,I         | # COLUMNS * 2, SINCE
4352          | 2 BYTES/CHAR (OFFSET OK)
F50B D1E7        4353          SAL    DI,I         | OFFSET *2 SINCE 2 BYTES/CHAR
F50D 47          4354          INC    DI           | POINT TO LAST BYTE
4355
4356 |----- DETERMINE THE SOURCE ADDRESS IN THE BUFFER
4357
F50E            4358          R12:         | FIND_SOURCE_DOWN
F50E 06           4359          PUSH   ES           | BOTH SEGMENTS TO REGEN
F50F 1F           4360          POP    DS           |
F510 2AED        4361          SUB    CH,CH       | ZERO TO HIGH OF COUNT REG
F512 81C1F000    4362          ADD    DI,240      | POINT TO LAST ROW OF PIXELS
F516 D0E3        4363          SAL    BL,I         | MULTIPLY NUMBER OF LINES BY 4
F518 D0E3        4364          SAL    BL,I         |
F51A 742E        4365          JZ     R16          | IF ZERO, THEN BLANK ENTIRE FIELD
F51C 8AC3        4366          MOV    AL,BL        | GET NUMBER OF LINES IN AL
F51E B450        4367          MOV    AH,80        | 80 BYTES/ROW
F520 F6E4        4368          MUL    AH           | DETERMINE OFFSET TO SOURCE
F522 8BF7        4369          MOV    SI,DI        | SET UP SOURCE
F524 2BF0        4370          SUB    SI,AX        | SUBTRACT THE OFFSET
F526 8AE6        4371          MOV    AH,DH        | NUMBER OF ROWS IN FIELD
F528 2AE3        4372          SUB    AH,BL        | DETERMINE NUMBER TO MOVE

```

```

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: ; ROW LOOP DOWN
F52A E82100         4377 ; MOVE ONE ROW
F52D 81EE5020      4378 SUB S1,2000H+80 ; MOVE TO NEXT ROW
F531 81EF5020      4379 SUB D1,2000H+80
F535 FECC          4380 DEC AH ; NUMBER OF ROWS TO MOVE
F537 75F1          4381 JNZ R13 ; CONTINUE TILL ALL MOVED
4382
4383 ;----- FILL IN THE VACATED LINE(S)
4384
F539                4385 R14: ; CLEAR ENTRY DOWN
F539 8AC7          4386 MOV AL,BH ; ATTRIBUTE TO FILL WITH
F53B EB2900        4387 R15: ; CLEAR_LOOP_DOWN
F53B EB2900        4388 CALL R18 ; CLEAR_A_ROW
F53C 81EF5020      4389 SUB D1,2000H+80 ; POINT TO NEXT LINE
F542 FECC          4390 DEC BL ; NUMBER OF LINES TO FILL
F544 75F5          4391 JNZ R15 ; CLEAR_LOOP_DOWN
F546 FC            4392 CLD ; RESET THE DIRECTION FLAG
F547 E97BFC        4393 JMP VIDEO_RETURN ; EVERYTHING DONE
F54A              4394 R16: ; BLANK FIELD DOWN
F54A 8ADE          4395 MOV BL,DH ; SET BLANK COUNT TO EVERYTHING
F54C EBEB         4396 ; IN FIELD
4397 ; CLEAR THE FIELD
4398 GRAPHICS_DOWN 4397 JMP R14
4399 ENDP
4400 ;----- ROUTINE TO MOVE ONE ROW OF INFORMATION
4401
F54E                4402 R17 PROC NEAR
F54E 8ACA          4403 MOV CL,DL ; NUMBER OF BYTES IN THE ROW
F550 56           4404 PUSH SI
F551 57           4405 PUSH DI ; SAVE POINTERS
F552 F3           4406 REP MOVSB ; MOVE THE EVEN FIELD
F553 A4           4407 POP DI
F554 5F           4408 POP SI
F555 5E           4409 ADD SI,2000H
F556 81C60020     4410 ADD D1,2000H ; POINT TO THE ODD FIELD
F55A 81C70020     4411 PUSH SI
F555 56           4412 PUSH DI ; SAVE THE POINTERS
F560 8ACA          4413 MOV CL,DL ; COUNT BACK
F562 F3           4414 REP MOVSB ; MOVE THE ODD FIELD
F563 A4           4415 POP DI
F564 5F           4416 POP SI ; POINTERS BACK
F565 5E           4417 RETI ; RETURN TO CALLER
F566 C3           4418 R17 ENDP
4419
4420 ;----- CLEAR A SINGLE ROW
4421
F567                4422 R18 PROC NEAR
F567 8ACA          4423 MOV CL,DL ; NUMBER OF BYTES IN FIELD
F569 57           4424 PUSH DI ; SAVE_POINTER
F56A F3           4425 REP STOSB ; STORE THE NEW VALUE
F56B AA           4426 POP DI ; POINTER BACK
F56C 5F           4427 D1,2000H ; POINT TO ODD FIELD
F56D 81C70020     4428 PUSH DI
F571 57           4429 MOV CL,DL
F572 8ACA          4430 REP STOSB ; FILL THE ODD FIELD
F574 F3           4431 POP DI
F575 AA           4432 RETI ; RETURN TO CALLER
F576 5F           4433 R18 ENDP
F577 C3           4434 ;-----
4435 ; GRAPHICS WRITE
4436 ; THIS ROUTINE WRITES THE ASCII CHARACTER TO THE
4437 ; CURRENT POSITION ON THE SCREEN.
4438 ; ENTRY
4439 ; AL = CHARACTER TO WRITE
4440 ; BL = COLOR ATTRIBUTE TO BE USED FOR FOREGROUND COLOR
4441 ; IF BIT 7 IS SET, THE CHAR IS XOR'D INTO THE REGEN
4442 ; BUFFER (0 IS USED FOR THE BACKGROUND COLOR)
4443 ; CX = NUMBER OF CHARS TO WRITE
4444 ; DS = DATA SEGMENT
4445 ; ES = REGEN SEGMENT
4446 ; EXIT
4447 ; NOTHING IS RETURNED
4448 ;
4449 ; GRAPHICS READ
4450 ; THIS ROUTINE READS THE ASCII CHARACTER AT THE CURRENT
4451 ; CURSOR POSITION ON THE SCREEN BY MATCHING THE DOTS ON
4452 ; THE SCREEN TO THE CHARACTER GENERATOR CODE POINTS
4453 ; ENTRY
4454 ; NONE ( 0 IS ASSUMED AS THE BACKGROUND COLOR
4455 ; EXIT
4456 ; AL = CHARACTER READ AT THAT POSITION (0 RETURNED IF
4457 ; NONE FOUND)
4458 ;
4459 ; FOR BOTH ROUTINES, THE IMAGES USED TO FORM CHARS ARE
4460 ; CONTAINED IN ROW FOR THE 1ST 128 CHARS, TO ACCESS CHARS
4461 ; IN THE SECOND HALF, THE USER MUST INITIALIZE THE VECTOR AT
4462 ; INTERRUPT 1FH (LOCATION 0007CH) TO POINT TO THE USER
4463 ; SUPPLIED TABLE OF GRAPHIC IMAGES (8X8 BOXES).
4464 ; FAILURE TO DO SO WILL CAUSE IN STRANGE RESULTS.
4465 ;-----
4466 ; ASSUME CS:CODE,DS:DATA,ES:DATA
4467 GRAPHICS_WRITE PROC NEAR
4468 MOV AH,0
4469 PUSH AX ; ZERO TO HIGH OF CODE POINT
4470 ; SAVE CODE POINT VALUE
4471 ;----- DETERMINE POSITION IN REGEN BUFFER TO PUT CODE POINTS
4472
F57B EB8401        4473 CALL S26 ; FIND LOCATION IN REGEN BUFFER
F57E BFB8         4474 MOV DI,AX ; REGEN POINTER IN DI
4475
4476 ;----- DETERMINE REGION TO GET CODE POINTS FROM
4477
F580 58           4478 POP AX ; RECOVER CODE POINT
F581 3C80         4479 CMP AL,80H ; IS IT IN SECOND HALF
F583 7306         4480 JAE S1 ; YES

```

```

LOC OBJECT          LINE SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82
4481
4482 ;----- IMAGE 15 IN FIRST HALF, CONTAINED IN ROM
4483
F585 BE6EFA        4484 MOV     SI,0FA6EH          ; CRT CHAR_GEN (OFFSET OF IMAGES)
F588 0E           4485 PUSH   CS                 ; SAVE SEGMENT ON STACK
F589 EB0F         4486 JMP     SHORT S2          ; DETERMINE_MODE
4487
4488 ;----- IMAGE 15 IN SECOND HALF, IN USER RAM
4489
F58B             4490 S1:
F58B 2C80        4491 SUB     AL,80H            ; EXTEND_CHAR
F58D 1E          4492 PUSH   DS                ; ZERO ORIGIN FOR SECOND HALF
F58E 2BF6        4493 MOV     SI,S1             ; SAVE DATA POINTER
F590 8EDE        4494 MOV     DS,S1             ;
F592 C5367C00    4495 ASSUME DS:ABS0           ; ESTABLISH VECTOR ADDRESSING
F596 8CDA        4496 LDS     SI,EXT_PTR        ; GET THE OFFSET OF THE TABLE
F598 1F          4497 MOV     DX,DS             ; GET THE SEGMENT OF THE TABLE
F599 52           4498 ASSUME DS:DATA           ;
F599 52           4499 POP     DS                ;
F599 52           4500 PUSH   DX                ; RECOVER DATA SEGMENT
F599 52           4501 PUSH   DX                ; SAVE TABLE SEGMENT ON STACK
4502
4503 ;----- DETERMINE GRAPHICS MODE IN OPERATION
4504
F59A             4504 S2:
F59A D1E0        4505 SAL     AX,1              ; DETERMINE_MODE
F59C D1E0        4506 TEST   AX,1              ; MULTIPLY CODE POINT
F59E D1E0        4507 SAL     AX,1              ; VALUE BY 8
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
F5A8 722C        4511 JC     S7                ; TEST FOR MEDIUM RESOLUTION MODE
4512
4513 ;----- HIGH RESOLUTION MODE
4514
F5AA             4515 S3:
F5AA 57          4516 PUSH   DI                ; HIGH_CHAR
F5AB 56          4517 PUSH   SI                ; SAVE REGEN POINTER
F5AC B604        4518 MOV     DH,4              ; SAVE CODE POINTER
F5AE             4519 S4:
F5AE AC          4520 LODSB                    ; NUMBER OF TIMES THROUGH LOOP
F5AF F6C380      4521 TEST   BL,80H            ; GET BYTE FROM CODE POINTS
F5B2 7516        4522 JNZ    S6                ; SHOULD WE USE THE FUNCTION
F5B4 AA          4523 STOSB                    ; TO PUT CHAR IN
F5B5 AC          4524 LODSB                    ; STORE IN REGEN BUFFER
F5B6             4525 S5:
F5B6 268885FF1F  4526 MOV     ES:[DI+2000H-1],AL ; STORE IN SECOND HALF
F5B8 B3C74F      4527 ADD     DI,79             ; MOVE TO NEXT ROW IN REGEN
F5B9 FECE        4528 DEC     DH                ; DONE WITH LOOP
F5C0 75EC        4529 JNZ    S1                ;
F5C2 5E          4530 POP     SI                ;
F5C3 5F          4531 POP     DI                ;
F5C4 47          4532 INC     DI                ; RECOVER REGEN POINTER
F5C5 E2E3        4533 LOOP   S3                ; POINT TO NEXT CHAR POSITION
F5C7 E9FBFB      4534 JMP     VIDEO_RETURN      ; MORE CHARS TO WRITE
F5CA             4535 S6:
F5CA 263205      4536 XOR     AL,ES:[DI]        ; EXCLUSIVE OR WITH CURRENT
F5CD AA          4537 STOSB                    ; STORE THE CODE POINT
F5CE AC          4538 STOSB                    ; AGAIN FOR ODD FIELD
F5CF 263285FF1F  4539 XOR     AL,ES:[DI+2000H-1] ;
F5D4 EB0E        4540 JMP     S5                ; BACK TO MAINSTREAM
4541
4542 ;----- MEDIUM RESOLUTION WRITE
4543
F5D6             4544 S7:
F5D6 8AD3        4545 MOV     DL,BL             ; MED RES WRITE
F5D8 D1E7        4546 SAL     DI,1             ; SAVE HIGH COLOR BIT
F5DA E8D100      4547 CALL   S19               ; EXPAND BL TO FULL WORD OF COLOR
F5DD             4548 S8:
F5DD 57          4549 PUSH   DI                ; MED_CHAR
F5DE 56          4550 PUSH   SI                ; SAVE REGEN POINTER
F5DF B604        4551 MOV     DH,4              ; SAVE THE CODE POINTER
F5E1             4552 S9:
F5E1 AC          4553 LODSB                    ; NUMBER OF LOOPS
F5E2 E8DE00      4554 CALL   S21               ; GET CODE POINT
F5E5 23C3        4555 AND     AX,BX             ; DOUBLE UP ALL THE BITS
F5E7 F6C280      4556 TEST   DL,80H            ; CONVERT THEM TO FOREGROUND
F5EA 7407        4557 JZ     S10               ; COLOR ( 0 BACK )
F5EC 263225      4558 XOR     AH,ES:[DI]        ; IS THIS XOR FUNCTION
F5EF 26324501    4559 XOR     AL,ES:[DI+1]      ; NO, STORE IT IN AS IT IS
F5F3             4560 S10:
F5F3 268825      4561 MOV     ES:[DI],AH        ; DO FUNCTION WITH HALF
F5F3 268825      4562 MOV     ES:[DI+1],AL      ; AND WITH OTHER HALF
F5FA AC          4563 LODSB                    ; STORE FIRST BYTE
F5FB E8C500      4564 CALL   S21               ; STORE SECOND BYTE
F5FE 23C3        4565 AND     AX,BX             ; GET CODE POINT
F600 F6C280      4566 TEST   DL,80H            ; CONVERT TO COLOR
F603 740A        4567 JZ     S11               ; AGAIN, IS THIS XOR FUNCTION
F605 2632A50020  4568 XOR     AH,ES:[DI+2000H]  ; NO, JUST STORE THE VALUES
F60A 2632850120  4569 XOR     AL,ES:[DI+2001H]  ; FUNCTION WITH FIRST HALF
F60F             4570 S11:
F60F 2688A50020  4571 MOV     ES:[DI+2000H],AH  ; AND WITH SECOND HALF
F614 2688850120  4572 MOV     ES:[DI+2000H+1],AL ;
F619 83C750      4573 ADD     DI,80             ; STORE IN SECOND PORTION OF BUFFER
F61C FECE        4574 DEC     DH                ; POINT TO NEXT LOCATION
F61E 75C1        4575 JZ     S9                ;
F620 5E          4576 POP     SI                ; KEEP GOING
F621 5F          4577 POP     DI                ; RECOVER CODE POINTER
F622 47          4578 INC     DI                ; RECOVER REGEN POINTER
F623 47          4579 INC     DI                ; POINT TO NEXT CHAR POSITION
F624 E2B7        4580 LOOP   S8                ;
F626 E99CFB      4581 JMP     VIDEO_RETURN      ; MORE TO WRITE
4582
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 S2A ; 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 ; READ CODE POINT
F631 8BEC 4592 MOV BP,SP ; POINTER TO SAVE AREA
4593
4594 :----- DETERMINE GRAPHICS MODES
4595
F633 803E490006 4596 CMP CRT_MODE,6
F638 06 4597 PUSH ES
F639 1F 4598 POP DS ; POINT TO REGEN SEGMENT
F63A 721A 4599 JC S13 ; MEDIUM RESOLUTION
4600
4601 :----- HIGH RESOLUTION READ
4602
4603 :----- GET VALUES FROM REGEN BUFFER AND CONVERT TO CODE POINT
4604
F63C B604 4605 MOV DH,4 ; NUMBER OF PASSES
F63E 4606
S12: 4607 MOV AL,[SI] ; GET FIRST BYTE
4608 MOV [BP],AL ; SAVE IN STORAGE AREA
F643 45 4609 INC BP ; NEXT LOCATION
F644 BA840020 4610 MOV AL,[SI+2000H] ; GET LOWER REGION BYTE
F648 B84000 4611 MOV [BP],AL ; ADJUST AND STORE
F64B 45 4612 INC BP
F64C 83C650 4613 ADD SI,80 ; POINTER INTO REGEN
F64F FECE 4614 DEC SI ; 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: ; MED RES READ
F656 D1E6 4621 SAL SI,1 ; OFFSET*2 SINCE 2 BYTES/CHAR
F658 B604 4622 MOV DH,4 ; NUMBER OF PASSES
F65A E88800 4623 CALL S23 ; GET PAIR BYTES FROM REGEN
4624 ; INTO SINGLE SAVE
4625
F65D 81C60020 4626 ADD SI,2000H ; GO TO LOWER REGION
F661 E88100 4627 CALL S23 ; GET THIS PAIR INTO SAVE
F664 81EEB01F 4628 SI,2000H-80 ; ADJUST POINTER BACK INTO UPPER
F668 FECE 4629 DEC DH
F66A 75EE 4630 JNZ S14 ; KEEP GOING UNTIL ALL 8 DONE
4631
4632 :----- SAVE AREA HAS CHARACTER IN IT, MATCH IT
4633
F66C 4634 S15: ; FIND CHAR
F66C BF6EFA90 4635 MOV DI,OFFSET CRT_CHAR_GEN ; ESTABLISH ADDRESSING
F670 0E 4636 PUSH CS
F671 07 4637 POP ES ; CODE POINTS IN CS
F672 83ED08 4638 SUB BP,8 ; ADJUST POINTER TO BEGINNING
4639 ; OF SAVE AREA
F675 8BF5 4640 MOV SI,BP
F677 FC 4641 CLD ; ENSURE DIRECTION
F678 B000 4642 MOV AL,0 ; CURRENT CODE POINT BEING MATCHED
F67A 16 4643 PUSH SS ; ESTABLISH ADDRESSING TO STACK
F67B 1F 4644 POP DS ; FOR THE STRING COMPARE
F67C BA8000 4645 MOV DX,128 ; NUMBER TO TEST AGAINST
F67F 4646
S17: 4647 PUSH SI ; SAVE SAVE AREA POINTER
F680 57 4648 PUSH DI ; SAVE CODE POINTER
F681 B90800 4649 MOV CX,8 ; NUMBER OF BYTES TO MATCH
F684 F3 4650 REPE CMPSB ; COMPARE THE 8 BYTES
F685 A6 4651 ; RECOVER THE POINTERS
F686 5F 4652 POP DI
F687 5E 4653 POP SI
F688 741E 4654 JZ S18 ; IF ZERO FLAG SET, THEN MATCH OCCURRED
F68A FE00 4655 INC AL ; NO MATCH, MOVE ON TO NEXT
F68C 83C708 4656 ADD DI,8 ; NEXT CODE POINT
F68F 4A 4657 DEC DX ; LOOP CONTROL
F690 75ED 4658 JNZ S17 ; DO ALL OF THEM
4659
4660 :----- CHAR NOT MATCHED, MIGHT BE IN USER SUPPLIED SECOND HALF
4661
F692 3C00 4662 CMP AL,0 ; AL <> 0 IF ONLY 1ST HALF SCANNED
F694 7412 4663 JE S18 ; IF = 0, THEN ALL HAS BEEN SCANNED
F696 2BC0 4664 SUB AX,AX ; ESTABLISH ADDRESSING TO VECTOR
F698 8ED8 4665 MOV DS,AX
4666 ASSUME DS:ABS0
F69A C437C00 4667 LES DI,EXT_PTR ; GET POINTER
F69E 8CC0 4668 MOV AX,ES ; SEE IF THE POINTER REALLY EXISTS
F6A0 0BC7 4669 OR AX,DI ; IF ALL 0, THEN DOESN'T EXIST
F6A2 7404 4670 JZ S18 ; NO SENSE LOOKING
F6A4 B080 4671 MOV AL,128 ; ORIGIN FOR SECOND HALF
F6A6 EBD2 4672 JMP S16 ; GO BACK AND TRY FOR IT
4673 ASSUME DS:DATA
4674
4675 :----- CHARACTER IS FOUND ( AL=0 IF NOT FOUND )
4676
F6A8 4677 S18:
F6A8 83C408 4678 ADD SP,8 ; READJUST THE STACK, THROW AWAY SAVE
F6AB E917FB 4679 JMP VIDEO_RETURN ; ALL DONE
4680 GRAPHICS_READ ENDP

```

```

4681 -----
4682 ; EXPAND_MED_COLOR :
4683 ; THIS ROUTINE EXPANDS THE LOW 2 BITS IN BL TO :
4684 ; FILL THE ENTIRE BX REGISTER :
4685 ; ENTRY :
4686 ; BL = COLOR TO BE USED ( LOW 2 BITS ) :
4687 ; EXIT :
4688 ; BX = COLOR TO BE USED ( 8 REPLICATIONS OF THE :
4689 ; 2 COLOR BITS ) :
4690 -----
F6AE 4691 S19 PROC NEAR :
F6AE 80E303 4692 AND BL,3 ; ISOLATE THE COLOR BITS
F6B1 8AC3 4693 MOV AL,BL ; COPY TO AL
F6B3 51 4694 PUSH AX ; SAVE REGISTER
F6B4 B93030 4695 MOV CX,3 ; NUMBER OF TIMES TO DO THIS
F6B7 4696 S20:
F6B7 D0E0 4697 SAL AL,1
F6B9 D0E0 4698 SAL AL,1 ; LEFT SHIFT BY 2
F6BB 0AD8 4699 OR BL,AL ; ANOTHER COLOR VERSION INTO BL
F6BD E2F8 4700 LOOP S20 ; FILL ALL OF BL
F6BF 8AF8 4701 MOV BH,BL ; FILL UPPER PORTION
F6C1 59 4702 POP CX ; REGISTER BACK
F6C2 C3 4703 RET ; ALL DONE
4704 S19 ENDP
4705 -----
4706 ; EXPAND_BYTE :
4707 ; THIS ROUTINE TAKES THE BYTE IN AL AND DOUBLES :
4708 ; ALL OF THE BITS, TURNING THE 8 BITS INTO :
4709 ; 16 BITS. THE RESULT IS LEFT IN AX :
4710 -----
F6C3 4711 S21 PROC NEAR :
F6C3 52 4712 PUSH DX ; SAVE REGISTERS
F6C4 51 4713 CX ;
F6C5 53 4714 PUSH BX
F6C6 2BD2 4715 SUB DX,DX ; RESULT REGISTER
F6CB B90400 4716 MOV CX,1 ; MASK REGISTER
F6CB 8BD8 4718 MOV BX,AX ; BASE INTO TEMP
F6CD 23D9 4719 AND BX,CX ; USE MASK TO EXTRACT A BIT
F6CF 0BD3 4720 OR DX,BX ; PUT INTO RESULT REGISTER
F6D1 D1E0 4721 SHL AX,1
F6D3 D1E1 4722 SHL CX,1 ; SHIFT BASE AND MASK BY 1
F6D5 8BD8 4723 MOV BX,AX ; BASE TO TEMP
F6D7 23D9 4724 AND BX,CX ; EXTRACT THE SAME BIT
F6D9 0BD3 4725 OR DX,BX ; PUT INTO RESULT
F6DB D1E1 4726 SHL CX,1 ; SHIFT ONLY MASK NOW,
; MOVING TO NEXT BASE
F6DD 73EC 4728 JNC S22 ; USE MASK BIT COMING OUT TO TERMINATE
F6DF 8BC2 4729 MOV AX,DX ; RESULT TO PARM REGISTER
F6E1 5B 4730 POP BX
F6E2 59 4731 POP CX ; RECOVER REGISTERS
F6E3 5A 4732 POP DX
F6E4 C3 4733 RET ; ALL DONE
4734 S21 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 -----
F6E5 4749 S23 PROC NEAR :
F6E5 8A24 4750 MOV AH,[S1] ; GET FIRST BYTE
F6E7 8A4401 4751 MOV AL,[S1+1] ; GET SECOND BYTE
F6EA B90C00 4752 MOV CX,0C000H ; 2 BIT MASK TO TEST THE ENTRIES
F6ED B200 4753 MOV DL,0 ; RESULT REGISTER
F6EF 4754 S24:
F6EF 85C1 4755 TEST AX,CX ; IS THIS SECTION BACKGROUND?
F6F1 F8 4756 JLC ; CLEAR CARRY IN HOPES THAT IT IS
F6F2 7401 4757 JZ S25 ; IF ZERO, IT IS BACKGROUND
F6F4 F9 4758 STC ; WASN'T, SO SET CARRY
F6F5 D0D2 4759 RCL DL,1 ; MOVE THAT BIT INTO THE RESULT
F6F7 D1E9 4760 SHR CX,1
F6F9 D1E9 4761 SHR CX,1 ; MOVE THE MASK TO THE RIGHT BY 2 BITS
F6FB 73F2 4762 JNC S24 ; DO IT AGAIN IF MASK DIDN'T FALL OUT
F6FD 8B5600 4763 MOV [BP],DL ; STORE RESULT IN SAVE AREA
F700 45 4764 INC BP ; ADJUST POINTER
F701 C3 4765 RET ; ALL DONE
4766 S23 ENDP
4767 -----
4768 ; V4_POSITION :
4769 ; THIS ROUTINE TAKES THE CURSOR POSITION :
4770 ; CONTAINED IN THE MEMORY LOCATION, AND :
4771 ; CONVERTS IT INTO AN OFFSET INTO THE :
4772 ; REGEN BUFFER, ASSUMING ONE BYTE/CHAR. :
4773 ; FOR MEDIUM RESOLUTION GRAPHICS, :
4774 ; THE NUMBER MUST BE DOUBLED. :
4775 ; ENTRY :
4776 ; NO REGISTERS, MEMORY LOCATION :
4777 ; CURSOR_POSN IS USED :
4778 ; EXIT :
4779 ; AX CONTAINS OFFSET INTO REGEN BUFFER :
4780 -----
F702 4781 S26 PROC NEAR :
F702 A15000 4782 MOV AX,CURSOR_POSN ; GET CURRENT CURSOR
F705 4783 GRAPH_POSN LABEL NEAR :
F705 53 4784 PUSH BX ; SAVE REGISTER
F707 8B08 4785 MOV BX,AX ; SAVE A COPY OF CURRENT CURSOR
F708 8AC4 4786 MOV AL,AH ; GET ROWS TO AL
F70A F264A00 4787 MUL BYTE PTR CRT_COLS ; MULTIPLY BY BYTES/COLUMN
F70E D1E0 4788 SHL AX,1 ; MULTIPLY * 4 SINCE 4 ROWS/BYTE
F710 D1E0 4789 SHL AX,1
F712 2AFF 4790 SUB BH,BH ; ISOLATE COLUMN VALUE
F714 03C3 4791 ADD AX,BX ; DETERMINE OFFSET
F716 5B 4792 POP BX ; RECOVER POINTER
F717 C3 4793 RET ; ALL DONE
4794 S26 ENDP

```

```

4795 :-----
4796 : WRITE TTY
4797 : THIS INTERFACE PROVIDES A TELETYPE LIKE INTERFACE TO THE VIDEO
4798 : CARD. THE INPUT CHARACTER IS WRITTEN TO THE CURRENT CURSOR
4799 : POSITION, AND THE CURSOR IS MOVED TO THE NEXT POSITION. IF THE
4800 : CURSOR LEAVES THE LAST COLUMN OF THE FIELD, THE COLUMN IS SET
4801 : TO ZERO, AND THE ROW VALUE IS INCREMENTED. IF THE ROW VALUE
4802 : LEAVES THE FIELD, THE CURSOR IS PLACED ON THE LAST ROW, FIRST
4803 : COLUMN, AND THE ENTIRE SCREEN IS SCROLLED UP ONE LINE. WHEN
4804 : THE SCREEN IS SCROLLED UP, THE ATTRIBUTE FOR FILLING THE NEWLY
4805 : BLANKED LINE IS READ FROM THE CURSOR POSITION ON THE PREVIOUS
4806 : LINE BEFORE THE SCROLL, IN CHARACTER MODE. IN GRAPHICS MODE,
4807 : THE 0 COLOR IS USED.
4808 : ENTRY
4809 : (AH) = CURRENT CRT MODE
4810 : (AL) = CHARACTER TO BE WRITTEN
4811 : NOTE THAT BACK SPACE, CAR RET, BELL AND LINE FEED ARE HANDLED
4812 : AS COMMANDS RATHER THAN AS DISPLAYABLE GRAPHICS
4813 : (BL) = FOREGROUND COLOR FOR CHAR WRITE IF CURRENTLY IN A
4814 : GRAPHICS MODE
4815 : EXIT
4816 : ALL REGISTERS SAVED
4817 :-----
4818 : ASSUME CS:CODE,DS:DATA
4819 : WRITE_TTY PROC NEAR
4820 : PUSH AX ; SAVE REGISTERS
4821 : PUSH AX ; SAVE CHAR TO WRITE
4822 : MOV AH,3
4823 : MOV BH,ACTIVE_PAGE ; GET THE CURRENT ACTIVE PAGE
4824 : INT 10H ; READ THE CURRENT CURSOR POSITION
4825 : POP AX ; RECOVER CHAR
4826
4827 :----- DX NOW HAS THE CURRENT CURSOR POSITION
4828
4829 : CMP AL,8 ; IS IT A BACKSPACE
4830 : JE UB ; BACK SPACE
4831 : CMP AL,0DH ; IS IT CARRIAGE RETURN
4832 : JE U9 ; CAR RET
4833 : CMP AL,0AH ; IS IT A LINE FEED
4834 : JE U10 ; LINE FEED
4835 : CMP AL,07H ; IS IT A BELL
4836 : JE U11 ; BELL
4837
4838 :----- WRITE THE CHAR TO THE SCREEN
4839
4840
4841 : MOV AH,10 ; WRITE CHAR ONLY
4842 : MOV CX,1 ; ONLY ONE CHAR
4843 : INT 10H ; WRITE THE CHAR
4844
4845 :----- POSITION THE CURSOR FOR NEXT CHAR
4846
4847 : INC DL
4848 : CMP DL,BYTE PTR CRT_COLS ; TEST FOR COLUMN OVERFLOW
4849 : JNZ U7 ; SET CURSOR
4850 : MOV DL,0 ; COLUMN FOR CURSOR
4851 : DH,24
4852 : JNZ U6 ; SET_CURSOR_INC
4853
4854 :----- SCROLL REQUIRED
4855
4856 : U1:
4857 : MOV AH,2 ; SET THE CURSOR
4858 : INT 10H
4859
4860 :----- DETERMINE VALUE TO FILL WITH DURING SCROLL
4861
4862 : MOV AL,CRT_MODE ; GET THE CURRENT MODE
4863 : CMP AL,4
4864 : JC U2 ; READ-CURSOR
4865 : CMP AL,7
4866 : MOV BH,0 ; FILL WITH BACKGROUND
4867 : JNE U3 ; SCROLL-UP
4868 : MOV AH,8 ; READ-CHAR
4869 : INT 10H ; READ CHAR/ATTR AT CURRENT CURSOR
4870 : MOV BH,AH ; STORE IN BH
4871 : U3:
4872 : MOV AX,601H ; SCROLL-UP
4873 : SUB CX,CX ; SCROLL ONE LINE
4874 : CX,CX ; UPPER LEFT CORNER
4875 : MOV DH,24 ; LOWER RIGHT ROW
4876 : MOV DL,BYTE PTR CRT_COLS ; LOWER RIGHT COLUMN
4877 : DEC DL
4878 : U4:
4879 : INT 10H ; VIDEO-CALL-RETURN
4880 : U5:
4881 : POP AX ; SCROLL UP THE SCREEN
4882 : JMP VIDEO_RETURN ; TTY-RETURN
4883 : U6: ; RESTORE THE CHARACTER
4884 : INC DH ; RETURN TO CALLER
4885 : U7: ; SET-CURSOR-INC
4886 : MOV AH,2 ; NEXT ROW
4887 : JMP U4 ; SET-CURSOR
4888
4889 :----- BACK SPACE FOUND
4890
4891 : U8:
4892 : CMP DL,0 ; ALREADY AT END OF LINE
4893 : JE U7 ; SET CURSOR
4894 : DEC DL ; NO "-- JUST MOVE IT BACK
4895 : JMP U7 ; SET CURSOR
4896
4897 :----- CARRIAGE RETURN FOUND
4898
4899 : U9:
4899 : MOV DL,0 ; MOVE TO FIRST COLUMN
4900 : JMP U7 ; SET_CURSOR
4901
4902 :----- LINE FEED FOUND
4903
4904
4905 : U10:
4906 : CMP DH,24 ; BOTTOM OF SCREEN
4907 : JNE U6 ; YES, SCROLL THE SCREEN
4908 : JMP U1 ; NO, JUST SET THE CURSOR

```


LOC OBJECT

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

```

4909
4910 ;----- BELL FOUND
4911
F78D
F78D B302
F78F E87602
F792 E8DB
4912 U11:
4913     MOV     BL,2           ; SET UP COUNT FOR BEEP
4914     CALL   U5             ; SOUND THE POD BELL
4915     JMP    U5             ; TTY_RETURN
4916 WRITE_TTY     ENDP
4917
4918 ; LIGHT PEN
4919 ; THIS ROUTINE TESTS THE LIGHT PEN SWITCH AND THE LIGHT
4920 ; PEN TRIGGER. IF BOTH ARE SET, THE LOCATION OF THE LIGHT
4921 ; PEN IS DETERMINED. OTHERWISE, A RETURN WITH NO
4922 ; INFORMATION IS MADE.
4923 ; ON EXIT
4924 ; (AH) = 0 IF NO LIGHT PEN INFORMATION IS AVAILABLE
4925 ; BX,CX,DX ARE DESTROYED
4926 ; (AH) = 1 IF LIGHT PEN IS AVAILABLE
4927 ; (DH,DL) = ROW,COLUMN OF CURRENT LIGHT PEN
4928 ; POSITION
4929 ; (CH) = RASTER POSITION
4930 ; (BX) = BEST GUESS AT PIXEL HORIZONTAL POSITION
4931
4932     ASSUME CS:CODE,DS:DATA
4933 ;----- SUBTRACT_TABLE
4934 V1: LABEL     BYTE
4935     DB     3,3,5,5,3,3,3,4 ;

F794
F794 03
F795 03
F796 05
F797 05
F798 03
F799 03
F79A 03
F79B 04
F79C
4936 READ_LPEN     PROC     NEAR
4937
4938 ;----- WAIT FOR LIGHT PEN TO BE DEPRESSED
4939
F79C B400
F79E 8B166300
F7A2 83C206
F7A5 EC
F7A6 A804
F7A8 751E
4940     MOV     AH,0           ; SET NO LIGHT PEN RETURN CODE
4941     MOV     DX,ADDR_6845   ; GET BASE ADDRESS OF 6845
4942     AND     DX,4           ; POINT TO STATUS REGISTER
4943     IN      AL,DX          ; GET STATUS REGISTER
4944     TEST    AL,4           ; TEST LIGHT PEN SWITCH
4945     JNZ    V6             ; NOT SET, RETURN
4946
4947 ;----- NOW TEST FOR LIGHT PEN TRIGGER
4948
F7AA A802
F7AC 7503
F7AE E98100
4949     TEST    AL,2           ; TEST LIGHT PEN TRIGGER
4950     JTA    V7A           ; RETURN WITHOUT RESETTING TRIGGER
4951     JMP    V7
4952
4953 ;----- TRIGGER HAS BEEN SET, READ THE VALUE IN
4954
F7B1
F7B1 B410
4955 V7A:
4956     MOV     AH,16          ; LIGHT PEN REGISTERS ON 6845
4957
4958 ;----- INPUT REGS POINTED TO BY AH, AND CONVERT TO ROW COLUMN IN DX
4959
F7B3 8B166300
F7B7 8AC4
F7B9 EE
F7BA 42
F7BB EC
F7BC 8AE8
F7BE 4A
F7BF FEC4
F7C1 8AC4
F7C3 EE
F7C4 42
F7C5 EC
F7C6 8AE5
4960     MOV     DX,ADDR_6845   ; ADDRESS REGISTER FOR 6845
4961     MOV     AL,AH          ; REGISTER TO READ
4962     OUT     DX,AL          ; SET IT UP
4963     INC     DX             ; DATA REGISTER
4964     IN      AL,DX          ; GET THE VALUE
4965     MOV     CH,AL          ; SAVE IN CH
4966     DEC     DX             ; ADDRESS REGISTER
4967     INC     AH             ;
4968     MOV     AL,AH          ; SECOND DATA REGISTER
4969     OUT     DX,AL          ;
4970     INC     DX             ; POINT TO DATA REGISTER
4971     IN      AL,DX          ; GET SECOND DATA VALUE
4972     MOV     AH,CH          ; AX HAS INPUT VALUE
4973
4974 ;----- AX HAS THE VALUE READ IN FROM THE 6845
4975
F7C8 8A1E4900
F7CC 2AFF
F7CE 2E8A9F94F7
F7D3 2BC3
F7D5 8B1E4E00
F7D9 D1EB
F7DB 2BC3
F7DD 7902
F7DF 2BC0
4976     MOV     BL,CRT_MODE   ; BL,CRT_MODE
4977     SUB     BH,BH          ; MODE VALUE TO BX
4978     MOV     BL,CS:V1[BX]   ; DETERMINE AMOUNT TO SUBTRACT
4979     SUB     AX,BX          ; TAKE IT AWAY
4980     MOV     BX,CRT_START
4981     SHR     BX,1
4982     SUB     AX,BX
4983     JNS    V2             ; IF POSITIVE, DETERMINE MODE
4984     SUB     AX,AX          ; <0 PLAYS AS 0
4985
4986 ;----- DETERMINE MODE OF OPERATION
4987
F7E1
F7E1 B103
F7E3 803E490004
F7E8 722A
F7EA 803E490007
F7EF 7423
4988     V2:
4989     MOV     CL,3           ; DETERMINE MODE
4990     CMP     CRT_MODE,4     ; SET *8 SHIFT COUNT
4991     JNB    V4             ; DETERMINE IF GRAPHICS OR ALPHA
4992     CMP     CRT_MODE,7     ; ALPHA_PEN
4993     JE     V4             ; ALPHA_PEN
4994
4995 ;----- GRAPHICS MODE
4996
F7F1 B228
F7F3 F6F2
4997     MOV     DL,40          ; DIVISOR FOR GRAPHICS
4998     DIV     DL             ; DETERMINE ROW(AL) AND COLUMN(AH)
4999
5000 ;----- DETERMINE GRAPHIC ROW POSITION
5001
F7F5 8AE8
F7F7 02ED
F7F9 8ADC
F7FB 2AFF
F7FD 803E490006
F802 7504
F804 B104
F806 D0E4
F808
F808 D3E3
5002     MOV     CH,AL          ; SAVE ROW VALUE IN CH
5003     ADD     CH,CH          ; *2 FOR EVEN/ODD FIELD
5004     MOV     BL,AH          ; COLUMN VALUE TO BX
5005     SUB     BH,BH          ; MULTIPLY BY 8 FOR MEDIUM RES
5006     CMP     CRT_MODE,6     ; DETERMINE MEDIUM OR HIGH RES
5007     JNE    V3             ; NOT HIGH RES
5008     JNE    V3             ; NOT HIGH RES
5009     MOV     CL,4           ; SHIFT VALUE FOR HIGH RES
5010     SAL     AH,1           ; COLUMN VALUE TIMES 2 FOR HIGH RES
5011     V3:
5012     SHL     BX,CL          ; MULTIPLY *16 FOR HIGH RES

```

SECTION 5

LOC OBJECT

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

```

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

```

```

5051 |-----|
5052 | 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 AMOUNT OF 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 | ASSUME CS:CODE,DS:DATA |
5072 | ORG OF841H |
5073 | MEMORY_SIZE_DET PROC FAR |
5074 | STI | | INTERRUPTS BACK ON |
5075 | PUSH DS | | SAVE SEGMENT |
5076 | CALL DDS | | |
5077 | MOV AX, MEMORY_SIZE | | GET VALUE |
5078 | POP DS | | RECOVER SEGMENT |
5079 | IRET | | RETURN TO CALLER |
5080 | MEMORY_SIZE_DET ENDP |
5081 |-----|
5082 |--- INT 11 ---|
5083 | EQUIPMENT DETERMINATION |
5084 | THIS ROUTINE ATTEMPTS TO DETERMINE WHAT OPTIONAL |
5085 | DEVICES ARE ATTACHED TO THE SYSTEM. |
5086 | INPUT |
5087 | NO REGISTERS |
5088 | THE EQUIP_FLAG_VARIABLE IS SET DURING THE POWER ON |
5089 | DIAGNOSTICS USING THE FOLLOWING HARDWARE ASSUMPTIONS: |
5090 | PORT 60 = LOW ORDER BYTE OF EQUIPMENT |
5091 | PORT 3FA = INTERRUPT ID REGISTER OF 8250 |
5092 | BITS 7-3 ARE ALWAYS 0 |
5093 | PORT 378 = OUTPUT PORT OF PRINTER -- 8255 PORT THAT |
5094 | CAN BE READ AS WELL AS WRITTEN |
5095 | OUTPUT |
5096 | (AX) IS SET, BIT SIGNIFICANT, TO INDICATE ATTACHED I/O |
5097 | BIT 15,14 = NUMBER OF PRINTERS ATTACHED |
5098 | BIT 13 NOT USED |
5099 | BIT 12 = GAME I/O ATTACHED |
5100 | BIT 11,10,9 = NUMBER OF RS232 CARDS ATTACHED |
5101 | BIT 8 UNUSED |
5102 | BIT 7,6 = NUMBER OF DISKETTE DRIVES |
5103 | 00=1, 01=2, 10=3, 11=4 ONLY IF BIT 0 = 1 |
5104 | BIT 5,4 = INITIAL VIDEO MODE |
5105 | 00 - UNUSED |
5106 | 01 - 40X25 BW USING COLOR CARD |
5107 | 10 - 80X25 BW USING COLOR CARD |
5108 | 11 - 80X25 BW USING BW CARD |
5109 | BIT 3,2 = PLANAR RAM SIZE (00=16K,01=32K,10=48K,11=64K) |
5110 | BIT 1 NOT USED |
5111 | BIT 0 = IPL FROM DISKETTE -- THIS BIT INDICATES THAT |
5112 | THERE ARE DISKETTE DRIVES ON THE SYSTEM |
5113 | |
5114 | NO OTHER REGISTERS AFFECTED |
5115 |-----|
5116 | ASSUME CS:CODE,DS:DATA |
5117 | ORG OF84DH |
5118 | EQUIPMENT PROC FAR |
5119 | STI | | INTERRUPTS BACK ON |
5120 | PUSH DS | | SAVE SEGMENT REGISTER |
5121 | CALL DDS | | |
5122 | MOV AX, EQUIP_FLAG | | GET THE CURRENT SETTINGS |
5123 | POP DS | | RECOVER SEGMENT |
5124 | IRET | | RETURN TO CALLER |
5125 | EQUIPMENT ENDP |
5126 |-----|
5127 |--- INT 15 ---|
5128 | DUMMY CASSETTE IO ROUTINE-RETURNS 'INVALID CMD' IF THE ROUTINE IS |
5129 | IS EVER CALLED BY ACCIDENT (AH=86H, CARRY FLAG=1) |
5130 |-----|
5131 | ORG OF859H |
5132 | CASSETTE_IO PROC FAR |
5133 | STC | | CARRY INDICATOR=1 |
5134 | MOV AH, 86H |
5135 | RET 2 |
5136 | CASSETTE_IO ENDP |

```

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

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

F859
F859
F859 F9
F85A B486
F85C CA0200

```

LOC OBJECT          LINE SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82
5138
5140 -----
5141 : NON-MASKABLE INTERRUPT ROUTINE:
5142 : THIS ROUTINE WILL PRINT A PARITY CHECK 1 OR 2 MESSAGE :
5143 : AND ATTEMPT TO FIND THE STORAGE LOCATION CONTAINING THE :
5144 : BAD PARITY. IF FOUND, THE SEGMENT ADDRESS WILL BE :
5145 : PRINTED. IF NO PARITY ERROR CAN BE FOUND (INTERMITTANT :
5146 : READ PROBLEM) ?????<-WILL BE PRINTED WHERE THE ADDRESS :
5147 : WOULD NORMALLY GO.
5148 : IF ADDRESS IN ERROR IS IN THE I/O EXPANSION BOX, THE :
5149 : ADDRESS WILL BE FOLLOWED BY 'E1E1', IF IN SYSTEM UNIT, :
5150 : A '(S)' WILL FOLLOW THE ADDRESS
5151 -----
F85F 5151 NMI_INT PROC NEAR
5152 : ASSUME DS:DATA
F85F 50 5153 PUSH AX ; SAVE ORIG CONTENTS OF AX
F860 E462 5154 IN AL,PORT_C
F862 A8C0 5155 TEST AL,0C0H ; PARITY CHECK?
F864 7503 5156 JNZ NMI_1 ; NO, EXIT FROM ROUTINE
F866 E98700 5157 JMP D14
F869 5158 NMI_1:
F869 BA4000 5159 MOV DX,DATA
F86C BEDA 5160 MOV DS,DX
F86E BE15F990 5161 MOV SI,OFFSET D1 ; ADDR OF ERROR MSG
F872 A840 5162 TEST AL,40H ; I/O PARITY CHECK
F874 7504 5163 JNZ D13 ; DISPLAY ERROR MSG
F876 BE25F990 5164 MOV SI,OFFSET D2 ; MUST BE PLANAR
F87A 5165
F87A B400 5166 MOV AH,0 ; INIT AND SET MODE FOR VIDEO
F87C AD4900 5167 MOV AL,CRT_MODE
F87F CD10 5168 INT 10H ; CALL VIDEO_IO PROCEDURE
F881 EB4601 5169 CALL P_MSG ; PRINT ERROR MSG
5170
5171 :----- SEE IF LOCATION THAT CAUSED PARITY CHECK CAN BE FOUND
5172
F884 B000 5173 MOV AL,00H ; DISABLE TRAP
F886 E6A0 5174 OUT GADH,AL
F888 E461 5175 IN AL,PORT_B
F88A 0C30 5176 OR AL,00110000B ; TOGGLE PARITY CHECK ENABLES
F88C E661 5177 OUT PORT_B,AL
F88E 24CF 5178 AND AL,17011111B
F890 E661 5179 OUT PORT_B,AL
F892 8B1E1300 5180 MOV BX,MEMORY_SIZE ; GET MEMORY SIZE WORD
F896 FC 5181 CLD ; SET DIR FLAG TO INCREMENT
F897 2BD2 5182 SUB DX,DX ; POINT DX AT START OF MEM
F899 5183
F899 BEDA 5184 NMI_LOOP:
F89B BEC2 5185 MOV DS,DX
F89D B90040 5186 MOV CX,4000H ; SET FOR 16KB SCAN
F8A0 2BF6 5187 SUB SI,SI ; SET SI TO BE REALTIME TO
; START OF ES
; READ 16KB OF MEMORY
F8A2 F3 5188 REP LODSB
F8A3 AC 5189
F8A4 E462 5190 IN AL,PORT_C ; SEE IF PARITY CHECK HAPPENED
F8A6 24C0 5191 AND AL,11000000B
F8A8 7512 5192 JNZ PRT_NMI ; GO PRINT ADDRESS IF IT DID
F8AA 81C20004 5193 AND DX,0400H ; POINT TO NEXT 16K BLOCK
F8AE 83EB10 5194 SUB BX,16D
F8B1 75E6 5195 JNZ NMI_LOOP
F8B3 BE35F990 5196 MOV SI,OFFSET D2A1 ; PRINT ROW OF ????? IF PARITY
F8B7 EB1001 5197 CALL P_MSG ; CHECK COULD NOT BE RE-CREATED
F8BA FA 5198 CLI
F8BB F4 5199 HLT ; HALT SYSTEM
F8BC 5200
F8BC 8CDA 5201 PRT_NMI:
F8BE E81907 5202 MOV DX,DS
F8C1 BA1302 5203 CALL PRT_SEG ; PRINT SEGMENT VALUE
F8C4 B000 5204 MOV AL,00
F8C6 EE 5205 OUT DX,AL ; DISPLAY EXPANSION BOX
; (CAN'T WRITE TO MEM)
F8C7 B028 5206 MOV AL,'('
F8C9 E8D000 5207 CALL PRT_HEX
F8CC B85AA5 5208 MOV AX,0A55AH
F8CF 8BC8 5209 MOV CX,AX
F8D1 2BD8 5210 SUB BX,BX
F8D3 8907 5211 MOV [BX],AX ; WRITE A WORD TO SEGMENT THAT
F8D5 90 5212 NOP
F8D6 90 5213 NOP
F8D7 8B07 5214 MOV AX,[BX] ; HAD THE ERROR
F8D9 3BC1 5215 CMP AX,CX ; IS IT THERE?
F8DB 7407 5216 JE SYS_BOX_ERR ; YES- MUST BE SYS UNIT
F8DD B045 5217 MOV AL,'E' ; NO-MUST BE IN EXP. BOX
F8DF E8BA00 5218 CALL PRT_HEX
F8E2 EB05 5219 JMP SHORT HLT_NMI
F8E4 5220
F8E4 B053 5221 MOV AL,'S'
F8E6 E8B300 5222 CALL PRT_HEX
F8E9 5223
F8E9 B029 5224 HLT_NMI:
F8EB E8AE00 5225 MOV AL,')'
F8EE FA 5226 CALL PRT_HEX ; HALT SYSTEM
F8EF F4 5227 CLI
F8F0 5228 HLT
F8F0 58 5229 D14:
F8F1 CF 5230 POP AX ; RESTORE ORIG CONTENTS OF AX
5231
5231 NMI_INT ENDP
5232
5233 :-----
5234 : ROS_CHECKSUM SUBROUTINE
5235 :
5236
F8F2 5236 ROS_CHECKSUM PROC NEAR
F8F2 B90020 5237 MOV CX,8192 ; NEXT ROS MODULE
F8F5 5238 ROS_CHECKSUM_CNT: ; NUMBER OF BYTES TO ADD
F8F5 32C0 5239 XOR AL,AL ; ENTRY FOR OPTIONAL ROS TEST
F8F7 5240
F8F7 5241 C26: ADD AL,DS:[BX]
F8F7 0207 5242 INC BX ; POINT TO NEXT BYTE
F8F9 43 5243 LOOP C26 ; ADD ALL BYTES IN ROS MODULE
F8FA E2FB 5244 OR AL,AL ; SUM = 0?
F8FC 0AC0 5245 RET
F8FE C3 5246 ROS_CHECKSUM ENDP

```

LOC OBJECT

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

```

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

F8FF 313031
F902 0D
F903 0A
F904 20323031
F908 0D
F909 0A
F90A 524F4D
F90D 0D
F90E 0A
F90F 31383031
F913 0D
F914 0A
F915 50415249545920
434845434B2032
F923 0D
F924 0A
F925 50415249545920
434845434B2031
F933 0D
F934 0A
F935 3F3F3F3F3F
F93A 0D
F93B 0A

5251 E1 DB '201',13,10 ; MEMORY ERROR

5252 F3A DB 'ROM',13,10 ; ROM CHECKSUM ERROR

5253 F3C DB '1801',13,10 ; EXPANSION IO BOX ERROR

5254 D1 DB 'PARITY CHECK 2',13,10

5255 D2 DB 'PARITY CHECK 1',13,10

5256 D2A DB '?????',13,10

5257 |-----|
5258 | BLINK LED PROCEDURE FOR MFG RUN-IN TESTS |
5259 | IF LED IS ON, TURN IT OFF. IF OFF, TURN ON. |
5260 |-----|
5261 | ASSUME DS:DATA |
5262 |-----|
F93C BLINK_INT PROC NEAR
F93C FB STI
F93D 50 PUSH AX ; SAVE AX REG CONTENTS
F93E E461 IN AL,PORT_B ; READ CURRENT VAL OF PORT B
F940 8AE0 MOV AH,AL
F942 F6D0 NOT AL ; FLIP ALL BITS
F944 2440 AND AL,01000000B ; ISOLATE CONTROL BIT
F946 80E4BF AND AH,01111111B ; MASK OUT OF ORIGINAL VAL
F949 0AC4 OR AL,AH ; OR NEW CONTROL BIT IN
F94B E661 OUT PORT_B,AL
F94D B020 MOV AL,EDI
F94F E620 OUT INTA00,AL
F951 58 POP AX ; RESTORE AX REG
F952 CF IRET
BLINK_INT ENDP
5278
5279 |-----|
5280 | THIS ROUTINE CHECKS OPTIONAL ROM MODULES AND |
5281 | IF CHECKSUM IS OK, CALLS INIT/TEST CODE IN MODULE |
5282 |-----|
F953 ROM_CHECK PROC NEAR
F953 B84000 MOV AX,DATA
F956 8EC0 MOV ES,AX ; POINT ES TO DATA AREA
F958 2AE4 SUB AH,AH ; ZERO OUT AH
F95A 8A4702 MOV AL,[BX+2] ; GET LENGTH INDICATOR
F95D B109 MOV CL,09H ; MULTIPLY BY 512
F95F D3E0 SHL AX,CL
F961 8BC8 MOV CX,AX ; SET COUNT
F963 51 PUSH CX ; SAVE COUNT
F964 B90400 MOV CX,4 ; ADJUST
F967 D3E8 SHR AX,CL
F969 0300 ADD DX,AX ; SET POINTER TO NEXT MODULE
F96B 59 POP CX ; RETRIEVE COUNT
F96C E86FF CALL ROS_CHECKSUM_CNT ; DO CHECKSUM
F96F 7406 JZ ROM_CHECK_1
F971 E857ED CALL ROM_ERR ; POST CHECKSUM ERROR
F974 EB1490 JMP ROM_CHECK_END
F977 5300 ROM_CHECK_1:
F977 52 PUSH DX ; SAVE POINTER
F978 26C70667000300 MOV ES:10_ROM_INIT,0003H ; LOAD OFFSET
F97F 268C1E4900 MOV ES:10_ROM_SEG,05 ; LOAD SEGMENT
F984 26FF1E6700 5304 CALL DWORD PTR ES:10_ROM_INIT ; CALL INIT./TEST ROUTINE
F989 5A POP DX
F98A 5306 ROM_CHECK_END:
F98A C3 RET ; RETURN TO CALLER
5309 ROM_CHECK ENDP
5310 |-----|
5311 | CONVERT AND PRINT ASCII CODE |
5312 | AL MUST CONTAIN NUMBER TO BE CONVERTED. |
5313 | AX AND BX DESTROYED. |
5314 |-----|
F98B XPC_BYTE 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
F996 5323 XLAT_PR PROC NEAR ; FALL INTO LOW NIBBLE CONVERSION
F996 0490 5324 ADD AL,090H ; CONVERT 00-OF TO ASCII CHARACTER
F998 27 5325 DAA ; ADD FIRST CONVERSION FACTOR
F999 1440 5326 DAA AL,040H ; ADJUST FOR NUMERIC AND ALPHA RANGE
F99B 27 5327 DAA ; ADD CONVERSION AND ADJUST LOW NIBBLE
F99C 5328 PRT_HEX PROC NEAR ; ADJUST HIGH NIBBLE TO ASCII RANGE
F99C B40E 5329 MOV AH,14 ; DISPLAY CHARACTER IN AL
F99E B700 5330 MOV BH,0
F9A0 CD10 5331 INT 10H ; CALL VIDEO_10
F9A2 C3 5332 RET
5333 PRT_HEX ENDP
5334 XLAT_PR ENDP
5335 XPC_BYTE ENDP
5336
5337 F4 LABEL WORD ; PRINTER SOURCE TABLE
F9A3 BC03 5338 DW 38CH
F9A5 7803 5339 DW 378H
F9A7 7802 5340 DW 278H
F9A9 5341 F4E LABEL WORD
5342

```

LOC OBJECT

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

```

5343 |-----|
5344 | THIS SUBROUTINE WILL PRINT A MESSAGE ON THE DISPLAY |
5345 | |
5346 | ENTRY REQUIREMENTS: |
5347 | S1 = OFFSET (ADDRESS) OF MESSAGE BUFFER |
5348 | CX = MESSAGE BYTE COUNT |
5349 | MAXIMUM MESSAGE LENGTH IS 36 CHARACTERS |
5350 |-----|
F9A9 5351 E_MSG PROC NEAR
F9A9 8BEE 5352 MOV BP,S1 ; SET BP NON-ZERO TO FLAG ERR
F9AB E81C00 5353 CALL P_MSG ; PRINT MESSAGE
F9AE 1E 5354 PUSH D5
F9AF E8A700 5355 CALL DDS
F9B2 A01000 5356 MOV AL,BYTE PTR EQUIP_FLAG ; LOOP/HALT ON ERROR
F9B5 2401 5357 AND AL,DIH ; SWITCH ON?
F9B7 750F 5358 JNZ G12 ; NO - RETURN
F9B9 5359 MFG_HALT:
F9B9 FA 5360 CLD ; YES - HALT SYSTEM
F9BA B0B9 5361 MOV AL,B9H
F9BC E663 5362 OUT CMD_PORT,AL
F9BE B0B5 5363 MOV AL,T0000101B ; DISABLE KB
F9C0 E661 5364 OUT PORT_B,AL
F9C2 A01500 5365 MOV AL,MFG_ERR_FLAG ; RECOVER ERROR INDICATOR
F9C5 E660 5366 OUT PORT_A,AL ; SET INTO 8255 REG
F9C7 F4 5367 HLT ; HALT SYS
F9C8 5368 G12:
F9C8 1F 5369 POP D5 ; WRITE_MSG:
F9C9 C3 5370 RET
5371 E_MSG ENDP
5372
F9CA 5373 P_MSG PROC NEAR
F9CA 5374 GT2A:
F9CA 2E8A04 5375 MOV AL,CS:[S1] ; PUT CHAR IN AL
F9CD 46 5376 INC SI ; POINT TO NEXT CHAR
F9CE 50 5377 PUSH AX ; SAVE PRINT CHAR
F9CF E8CAFF 5378 CALL PRT_HEX ; CALL VIDED IO
F9D2 58 5379 POP AX ; RECOVER PRINT CHAR
F9D3 3C0A 5380 CMP AL,10 ; WAS IT LINE FEED?
F9D5 75F3 5381 JNE G12A ; NO,KEEP PRINTING STRING
F9D7 C3 5382 RET
5383 P_MSG ENDP
5384
5385 |-----|
5386 | INITIAL RELIABILITY TEST -- SUBROUTINES |
5387 | |
5388 | ASSUME CS:CODE,DS:DATA |
5389 |-----|
5390 | SUBROUTINES FOR POWER ON DIAGNOSTICS |
5391 |-----|
5392 | THIS PROCEDURE WILL ISSUE ONE LONG TONE (3 SECS) AND ONE OR |
5393 | MORE SHORT TONES (1 SEC) TO INDICATE A FAILURE ON THE PLANAR |
5394 | BOARD, A BAD RAM MODULE, OR A PROBLEM WITH THE CRT. |
5395 | PARAMETERS: |
5396 | DH = NUMBER OF LONG TONES TO BEEP. |
5397 | DL = NUMBER OF SHORT TONES TO BEEP. |
5398 |-----|
F9D8 5399 ERR_BEEP PROC NEAR
F9D8 9C 5400 PUSHF ; SAVE FLAGS
F9D9 FA 5401 CLD ; DISABLE SYSTEM INTERRUPTS
F9DA 1E 5402 PUSH D5 ; SAVE DS REG CONTENTS
F9DB E87B00 5403 CALL DDS
F9DE 0AF6 5404 OR DH,DH ; ANY LONG ONES TO BEEP
F9E0 7414 5405 JZ G3 ; NO, DO THE SHORT ONES
F9E2 5406 G1:
F9E2 B306 5407 MOV BL,6 ; LONG BEEP:
F9E4 E82100 5408 CALL BEEP ; COUNTER FOR BEEPS
F9E7 5409 G2: ; DO THE BEEP
F9E7 E2FE 5410 LOOP G2 ; DELAY BETWEEN BEEPS
F9E9 FECE 5411 DEC DH ; ANY MORE TO DO
F9EB 75F5 5412 JNZ G1 ; DO IT
F9ED 803E120001 5413 CMP MFG_TST,1 ; MFG TEST MODE?
F9F2 7502 5414 JNE G3 ; YES - CONTINUE BEEPING SPEAKER
F9F4 EBC3 5415 JMP MFG_HALT ; STOP BLINKING LED
F9F6 B301 5416 G3: ; SHORT BEEP:
F9F6 B301 5417 MOV BL,1 ; COUNTER FOR A SHORT BEEP
F9F8 E80D00 5418 CALL BEEP ; DO THE SOUND
F9FB 5419 G4:
F9FB E2FE 5420 LOOP G4 ; DELAY BETWEEN BEEPS
F9FD FECA 5421 DEC DL ; DONE WITH SHORTS
F9FF 75F5 5422 JNZ G3 ; DO SOME MORE
FA01 5423 G5:
FA01 E2FE 5424 LOOP G5 ; LONG DELAY BEFORE RETURN
FA03 5425 G6:
FA03 E2FE 5426 LOOP G6
FA05 1F 5427 POP DS ; RESTORE ORIG CONTENTS OF DS
FA06 9D 5428 POPF ; RESTORE FLAGS TO ORIG SETTINGS
FA07 C3 5429 RET ; RETURN TO CALLER
5430 ERR_BEEP ENDP
5431
5432 |-----|
5433 |-----|
5434 |-----|
5435 |-----|
5436 |-----|
5437 |-----|
5438 |-----|
5439 |-----|
5440 |-----|
5441 |-----|
5442 |-----|
5443 |-----|
5444 |-----|
5445 |-----|
5446 |-----|
5447 |-----|
5448 |-----|
5449 |-----|
5450 |-----|
5451 |-----|
5452 |-----|
5453 |-----|
5454 |-----|
5455 |-----|
5456 |-----|
5457 |-----|
5458 |-----|
5459 |-----|
5460 |-----|
5461 |-----|
5462 |-----|
5463 |-----|
5464 |-----|
5465 |-----|
5466 |-----|
5467 |-----|
5468 |-----|
5469 |-----|
5470 |-----|
5471 |-----|
5472 |-----|
5473 |-----|
5474 |-----|
5475 |-----|
5476 |-----|
5477 |-----|
5478 |-----|
5479 |-----|
5480 |-----|
5481 |-----|
5482 |-----|
5483 |-----|
5484 |-----|
5485 |-----|
5486 |-----|
5487 |-----|
5488 |-----|
5489 |-----|
5490 |-----|
5491 |-----|
5492 |-----|
5493 |-----|
5494 |-----|
5495 |-----|
5496 |-----|
5497 |-----|
5498 |-----|
5499 |-----|
5500 |-----|
5501 |-----|
5502 |-----|
5503 |-----|
5504 |-----|
5505 |-----|
5506 |-----|
5507 |-----|
5508 |-----|
5509 |-----|
5510 |-----|
5511 |-----|
5512 |-----|
5513 |-----|
5514 |-----|
5515 |-----|
5516 |-----|
5517 |-----|
5518 |-----|
5519 |-----|
5520 |-----|
5521 |-----|
5522 |-----|
5523 |-----|
5524 |-----|
5525 |-----|
5526 |-----|
5527 |-----|
5528 |-----|
5529 |-----|
5530 |-----|
5531 |-----|
5532 |-----|
5533 |-----|
5534 |-----|
5535 |-----|
5536 |-----|
5537 |-----|
5538 |-----|
5539 |-----|
5540 |-----|
5541 |-----|
5542 |-----|
5543 |-----|
5544 |-----|
5545 |-----|
5546 |-----|
5547 |-----|
5548 |-----|
5549 |-----|
5550 |-----|
5551 |-----|
5552 |-----|
5553 |-----|
5554 |-----|
5555 |-----|
5556 |-----|
5557 |-----|
5558 |-----|
5559 |-----|
5560 |-----|
5561 |-----|
5562 |-----|
5563 |-----|
5564 |-----|
5565 |-----|
5566 |-----|
5567 |-----|
5568 |-----|
5569 |-----|
5570 |-----|
5571 |-----|
5572 |-----|
5573 |-----|
5574 |-----|
5575 |-----|
5576 |-----|
5577 |-----|
5578 |-----|
5579 |-----|
5580 |-----|
5581 |-----|
5582 |-----|
5583 |-----|
5584 |-----|
5585 |-----|
5586 |-----|
5587 |-----|
5588 |-----|
5589 |-----|
5590 |-----|
5591 |-----|
5592 |-----|
5593 |-----|
5594 |-----|
5595 |-----|
5596 |-----|
5597 |-----|
5598 |-----|
5599 |-----|
5600 |-----|
5601 |-----|
5602 |-----|
5603 |-----|
5604 |-----|
5605 |-----|
5606 |-----|
5607 |-----|
5608 |-----|
5609 |-----|
5610 |-----|
5611 |-----|
5612 |-----|
5613 |-----|
5614 |-----|
5615 |-----|
5616 |-----|
5617 |-----|
5618 |-----|
5619 |-----|
5620 |-----|
5621 |-----|
5622 |-----|
5623 |-----|
5624 |-----|
5625 |-----|
5626 |-----|
5627 |-----|
5628 |-----|
5629 |-----|
5630 |-----|
5631 |-----|
5632 |-----|
5633 |-----|
5634 |-----|
5635 |-----|
5636 |-----|
5637 |-----|
5638 |-----|
5639 |-----|
5640 |-----|
5641 |-----|
5642 |-----|
5643 |-----|
5644 |-----|
5645 |-----|
5646 |-----|
5647 |-----|
5648 |-----|
5649 |-----|
5650 |-----|
5651 |-----|
5652 |-----|
5653 |-----|
5654 |-----|
5655 |-----|
5656 |-----|
5657 |-----|
5658 |-----|
5659 |-----|
5660 |-----|
5661 |-----|
5662 |-----|
5663 |-----|
5664 |-----|
5665 |-----|
5666 |-----|
5667 |-----|
5668 |-----|
5669 |-----|
5670 |-----|
5671 |-----|
5672 |-----|
5673 |-----|
5674 |-----|
5675 |-----|
5676 |-----|
5677 |-----|
5678 |-----|
5679 |-----|
5680 |-----|
5681 |-----|
5682 |-----|
5683 |-----|
5684 |-----|
5685 |-----|
5686 |-----|
5687 |-----|
5688 |-----|
5689 |-----|
5690 |-----|
5691 |-----|
5692 |-----|
5693 |-----|
5694 |-----|
5695 |-----|
5696 |-----|
5697 |-----|
5698 |-----|
5699 |-----|
5700 |-----|
5701 |-----|
5702 |-----|
5703 |-----|
5704 |-----|
5705 |-----|
5706 |-----|
5707 |-----|
5708 |-----|
5709 |-----|
5710 |-----|
5711 |-----|
5712 |-----|
5713 |-----|
5714 |-----|
5715 |-----|
5716 |-----|
5717 |-----|
5718 |-----|
5719 |-----|
5720 |-----|
5721 |-----|
5722 |-----|
5723 |-----|
5724 |-----|
5725 |-----|
5726 |-----|
5727 |-----|
5728 |-----|
5729 |-----|
5730 |-----|
5731 |-----|
5732 |-----|
5733 |-----|
5734 |-----|
5735 |-----|
5736 |-----|
5737 |-----|
5738 |-----|
5739 |-----|
5740 |-----|
5741 |-----|
5742 |-----|
5743 |-----|
5744 |-----|
5745 |-----|
5746 |-----|
5747 |-----|
5748 |-----|
5749 |-----|
5750 |-----|
5751 |-----|
5752 |-----|
5753 |-----|
5754 |-----|
5755 |-----|
5756 |-----|
5757 |-----|
5758 |-----|
5759 |-----|
5760 |-----|
5761 |-----|
5762 |-----|
5763 |-----|
5764 |-----|
5765 |-----|
5766 |-----|
5767 |-----|
5768 |-----|
5769 |-----|
5770 |-----|
5771 |-----|
5772 |-----|
5773 |-----|
5774 |-----|
5775 |-----|
5776 |-----|
5777 |-----|
5778 |-----|
5779 |-----|
5780 |-----|
5781 |-----|
5782 |-----|
5783 |-----|
5784 |-----|
5785 |-----|
5786 |-----|
5787 |-----|
5788 |-----|
5789 |-----|
5790 |-----|
5791 |-----|
5792 |-----|
5793 |-----|
5794 |-----|
5795 |-----|
5796 |-----|
5797 |-----|
5798 |-----|
5799 |-----|
5800 |-----|
5801 |-----|
5802 |-----|
5803 |-----|
5804 |-----|
5805 |-----|
5806 |-----|
5807 |-----|
5808 |-----|
5809 |-----|
5810 |-----|
5811 |-----|
5812 |-----|
5813 |-----|
5814 |-----|
5815 |-----|
5816 |-----|
5817 |-----|
5818 |-----|
5819 |-----|
5820 |-----|
5821 |-----|
5822 |-----|
5823 |-----|
5824 |-----|
5825 |-----|
5826 |-----|
5827 |-----|
5828 |-----|
5829 |-----|
5830 |-----|
5831 |-----|
5832 |-----|
5833 |-----|
5834 |-----|
5835 |-----|
5836 |-----|
5837 |-----|
5838 |-----|
5839 |-----|
5840 |-----|
5841 |-----|
5842 |-----|
5843 |-----|
5844 |-----|
5845 |-----|
5846 |-----|
5847 |-----|
5848 |-----|
5849 |-----|
5850 |-----|
5851 |-----|
5852 |-----|
5853 |-----|
5854 |-----|
5855 |-----|
5856 |-----|
5857 |-----|
5858 |-----|
5859 |-----|
5860 |-----|
5861 |-----|
5862 |-----|
5863 |-----|
5864 |-----|
5865 |-----|
5866 |-----|
5867 |-----|
5868 |-----|
5869 |-----|
5870 |-----|
5871 |-----|
5872 |-----|
5873 |-----|
5874 |-----|
5875 |-----|
5876 |-----|
5877 |-----|
5878 |-----|
5879 |-----|
5880 |-----|
5881 |-----|
5882 |-----|
5883 |-----|
5884 |-----|
5885 |-----|
5886 |-----|
5887 |-----|
5888 |-----|
5889 |-----|
5890 |-----|
5891 |-----|
5892 |-----|
5893 |-----|
5894 |-----|
5895 |-----|
5896 |-----|
5897 |-----|
5898 |-----|
5899 |-----|
5900 |-----|
5901 |-----|
5902 |-----|
5903 |-----|
5904 |-----|
5905 |-----|
5906 |-----|
5907 |-----|
5908 |-----|
5909 |-----|
5910 |-----|
5911 |-----|
5912 |-----|
5913 |-----|
5914 |-----|
5915 |-----|
5916 |-----|
5917 |-----|
5918 |-----|
5919 |-----|
5920 |-----|
5921 |-----|
5922 |-----|
5923 |-----|
5924 |-----|
5925 |-----|
5926 |-----|
5927 |-----|
5928 |-----|
5929 |-----|
5930 |-----|
5931 |-----|
5932 |-----|
5933 |-----|
5934 |-----|
5935 |-----|
5936 |-----|
5937 |-----|
5938 |-----|
5939 |-----|
5940 |-----|
5941 |-----|
5942 |-----|
5943 |-----|
5944 |-----|
5945 |-----|
5946 |-----|
5947 |-----|
5948 |-----|
5949 |-----|
5950 |-----|
5951 |-----|
5952 |-----|
5953 |-----|
5954 |-----|
5955 |-----|
5956 |-----|
5957 |-----|
5958 |-----|
5959 |-----|
5960 |-----|
5961 |-----|
5962 |-----|
5963 |-----|
5964 |-----|
5965 |-----|
5966 |-----|
5967 |-----|
5968 |-----|
5969 |-----|
5970 |-----|
5971 |-----|
5972 |-----|
5973 |-----|
5974 |-----|
5975 |-----|
5976 |-----|
5977 |-----|
5978 |-----|
5979 |-----|
5980 |-----|
5981 |-----|
5982 |-----|
5983 |-----|
5984 |-----|
5985 |-----|
5986 |-----|
5987 |-----|
5988 |-----|
5989 |-----|
5990 |-----|
5991 |-----|
5992 |-----|
5993 |-----|
5994 |-----|
5995 |-----|
5996 |-----|
5997 |-----|
5998 |-----|
5999 |-----|
6000 |-----|

```

```

5454 ;-----
5455 ; THIS PROCEDURE WILL SEND A SOFTWARE RESET TO THE KEYBOARD. ;
5456 ; SCAN CODE 14 SHOULD BE RETURNED TO THE CPU. ;
5458 ;-----
FA2A 5459 KBD_RESET PROC NEAR
5460 DS:ABS0
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
G8: 5464 MOV G8,0 ;
5465 LOOP G8 ; LOOP FOR 20 MS
FA33 5466 MOV AL,0C8H ; SET CLK, ENABLE LINES HIGH
FA35 5467 OUT PORT_B,AL ;
FA37 5468 SP_TEST: MOV MOV AL,48H ; ENTRY FOR MANUFACTURING TEST 2
FA39 5470 OUT PORT_B,AL ; SET KBD CLK HIGH, ENABLE LOW
FA3B 5471 MOV AL,0FDH ; ENABLE KEYBOARD INTERRUPTS
FA3D 5472 INTA:JAL INTA:JAL ; WRITE 8255 IMR
FA3F 5473 MOV DATA_AREA[OFFSET INTR_FLAG] ; RESET INTERRUPT INDICATOR
FA44 5474 STI ; ENABLE INTERRUPTS
FA45 5475 SUB CX,CX ; SETUP INTERRUPT TIMEOUT CNT
FA47 5476
G9: 5477 TEST DATA_AREA[OFFSET INTR_FLAG],02H ; DID A KEYBOARD INTR OCCUR?
FA4C 5478 JNZ G10 ; YES - READ SCAN CODE RETURNED
FA4E 5479 LOOP G9 ; NO - LOOP TILL TIMEOUT
FA50 5480
G10: 5481 IN AL,PORT_A ; READ KEYBOARD SCAN CODE
FA52 5482 MOV BL,AL ; SAVE SCAN CODE JUST READ
FA54 5483 MOV AL,0C8H ; CLEAR KEYBOARD
FA56 5484 OUT PORT_B,AL ;
FA58 5485 RET ; RETURN TO CALLER
5486 KBD_RESET ENDP
5487
FA59 5488 DDS PROC NEAR
FA59 5489 PUSH AX ; SAVE AX
FA5A 5490 MOV AX,DATA ;
FA5B 5491 MOV AX,AX ; SET SEGMENT
FA5F 5492 POP AX ; RESTORE AX
FA60 5493 RET
5494 DDS ENDP
5495
-----
5497 ; CHARACTER GENERATOR GRAPHICS FOR 320X200 AND 640X200 GRAPHICS ;
5498 ;-----
FA6E 5499 ORG OFA6EH BYTE
FA6E 5500 CRT_CHAR_GEN LABEL BYTE
FA76 5501 DB 000H,000H,000H,000H,000H,000H,000H,000H ; D_00
FA77 5502 DB 07EH,08H,045H,081H,08DH,099H,081H,07EH ; D_01
FA7E 5503 DB 07EH,0FFH,0DBH,0FFH,0C3H,0E7H,0FFH,07EH ; D_02
FA86 5504 DB 06CH,0FEH,0FEH,0FEH,07CH,078H,010H,000H ; D_03
FA8E 5505 DB 010H,038H,07CH,0FEH,07CH,038H,010H,000H ; D_04
FA96 5506 DB 038H,07CH,038H,0FEH,0FEH,07CH,038H,07CH ; D_05
FA9E 5507 DB 010H,010H,038H,07CH,0FEH,07CH,038H,07CH ; D_06
FAA6 5508 DB 000H,000H,018H,03CH,03CH,018H,000H,000H ; D_07
FAAE 5509 DB 0FFH,0FFH,0E7H,0C3H,0C3H,0E7H,0FFH,0FFH ; D_08
FAB6 5510 DB 000H,0C0H,0F8H,0FEH,0FEH,0FEH,0E0H,0E0H ; D_09
FABE 5511 DB 0FFH,0C3H,099H,08DH,08DH,099H,0C3H,0FFH ; D_0A
FAC6 5512 DB 00FH,007H,00FH,07DH,0CCH,0CCH,00FH,07DH ; D_0B
FACB 5513 DB 03CH,066H,066H,066H,03CH,018H,07EH,018H ; D_0C
FAD3 5514 DB 03FH,03H,03FH,03H,03H,03H,070H,0F0H,0E0H ; D_0D
FADE 5515 DB 07FH,063H,07FH,063H,063H,067H,0E6H,0C0H ; D_0E
FAE6 5516 DB 099H,05AH,03CH,0E7H,0E7H,03CH,05AH,099H ; D_0F
FAEE 5517 DB 0E0H,0E0H,0F8H,0FEH,0FEH,0FEH,0E0H,0E0H ; D_10
FAF6 5518 DB 002H,00EH,03EH,00EH,03EH,00EH,00EH,000H ; D_11
FAFE 5519 DB 018H,03CH,07EH,018H,018H,07EH,03CH,018H ; D_12
FB06 5520 DB 066H,066H,066H,066H,066H,000H,066H,000H ; D_13
FB0E 5521 DB 07FH,0DBH,0DBH,07BH,018H,018H,018H,000H ; D_14
FB16 5522 DB 03EH,063H,038H,06CH,06CH,038H,0C0H,078H ; D_15
FB1E 5523 DB 000H,000H,000H,000H,07EH,07EH,07EH,000H ; D_16
FB26 5524 DB 018H,03CH,07EH,018H,07EH,03CH,018H,0FFH ; D_17
FB2E 5525 DB 018H,03CH,07EH,018H,018H,018H,018H,000H ; D_18
FB36 5526 DB 018H,018H,018H,018H,07EH,03CH,018H,000H ; D_19
FB3E 5527 DB 000H,018H,00CH,0FEH,00CH,018H,000H,000H ; D_1A
FB46 5528 DB 000H,030H,060H,0FEH,060H,030H,000H,000H ; D_1B
FB4E 5529 DB 000H,000H,0C0H,0C0H,0C0H,0FEH,000H,000H ; D_1C
FB56 5530 DB 000H,024H,066H,07FH,066H,024H,000H,000H ; D_1D
FB5E 5531 DB 000H,018H,03CH,07EH,0FFH,0FFH,000H,000H ; D_1E
FB66 5532 DB 000H,0FFH,0FFH,07EH,03CH,018H,000H,000H ; D_1F
FB6E 5533 DB 000H,000H,000H,000H,000H,000H,000H,000H ; SP D_20
FB76 5534 DB 030H,078H,078H,030H,030H,000H,030H,000H ; D_21
FB7E 5535 DB 06CH,06CH,06CH,000H,000H,000H,000H,000H ; D_22
FB86 5536 DB 06CH,06CH,06CH,06CH,06CH,06CH,06CH,06CH ; D_23
FB8E 5537 DB 030H,07CH,0C0H,078H,0C0H,0F8H,030H,000H ; D_24
FB96 5538 DB 000H,0C6H,0CCH,018H,030H,066H,0C6H,000H ; PER CENT D_25
FB9E 5539 DB 038H,06CH,038H,076H,0DCH,0CCH,076H,000H ; A D_26
FBA6 5540 DB 060H,060H,0C0H,000H,000H,000H,000H,000H ; D_27
FBAE 5541 DB 018H,030H,060H,060H,060H,030H,018H,000H ; D_28
FBB6 5542 DB 060H,030H,018H,018H,018H,030H,060H,000H ; D_29
FBBE 5543 DB 000H,066H,03CH,0FFH,03CH,066H,000H,000H ; D_2A
FBC6 5544 DB 000H,030H,030H,0FCH,030H,030H,000H,000H ; D_2B
FBCF 5545 DB 000H,000H,000H,000H,000H,000H,000H,000H ; D_2C
FBD0 5546 DB 000H,000H,000H,0FCH,000H,000H,000H,000H ; D_2D
FBD1 5547 DB 000H,000H,000H,000H,000H,000H,000H,000H ; D_2E
FBD2 5548 DB 066H,0C0H,018H,030H,066H,0C0H,000H,000H ; D_2F
FBD3 5549 DB 07CH,0C6H,0CEH,0DEH,0F6H,0E6H,07CH,000H ; D_30
FBD4 5550 DB 038H,070H,030H,030H,030H,030H,0FCH,000H ; D_31
FBD5 5551 DB 078H,0CCH,0CCH,038H,064H,0CCH,0F8H,000H ; D_32
FBD6 5552 DB 078H,0CCH,0CCH,038H,0CCH,0CCH,078H,000H ; D_33
FBD7 5553 DB 01CH,03CH,06CH,0CCH,0FEH,0CCH,01EH,000H ; A D_34
FBD8 5554 DB 0FCH,0CCH,0F8H,0CCH,0CCH,0CCH,078H,000H ; D_35
FBD9 5555 DB 038H,066H,0CCH,0F8H,0CCH,0CCH,038H,000H ; D_36
FBD0 5556 DB 0FCH,0CCH,0CCH,018H,030H,030H,000H,000H ; D_37
FBD1 5557 DB 078H,0CCH,0CCH,078H,0CCH,0CCH,078H,000H ; D_38
FBD2 5558 DB 078H,0CCH,0CCH,078H,0CCH,018H,070H,000H ; D_39
FBD3 5559 DB 030H,030H,030H,000H,000H,030H,030H,000H ; D_3A
FBD4 5560 DB 000H,030H,030H,000H,000H,030H,030H,060H ; D_3B
FBD5 5561 DB 018H,030H,060H,0C0H,060H,030H,018H,000H ; D_3C
FBD6 5562 DB 000H,000H,030H,0FCH,0CCH,0CCH,000H,000H ; D_3D
FBD7 5563 DB 060H,030H,018H,0CCH,018H,030H,060H,000H ; D_3E
FBD8 5564 DB 078H,0CCH,0CCH,018H,030H,000H,030H,000H ; D_3F

```

```

FC6E 7C6DEDEDEDC07800 5565 DB 07CH,0C6H,0DEH,0DEH,0DEH,0CCH,07BH,000H ; M D_40
FC76 3078CCCFC0CCCC00 5566 DB 030H,078H,0CCH,0CCH,0FCH,0CCH,0CCH,000H ; A D_41
FC7E FC6667C6666FC00 5567 DB 0FCH,066H,066H,07CH,066H,066H,0FCH,000H ; B D_42
FC86 3C66C0CC0C663C00 5568 DB 030H,066H,0CCH,0CCH,0CCH,066H,03CH,000H ; C D_43
FC8E FB6666666666800 5569 DB 0FBH,06CH,066H,066H,066H,0FBH,000H ; D D_44
FC96 FE626786862FE00 5570 DB 0FEH,062H,068H,078H,068H,062H,0FEH,000H ; E D_45
FC9E FE626786860F000 5571 DB 0FEH,062H,068H,078H,068H,060H,0FEH,000H ; F D_46
FCA6 3C66C0CC0C663C00 5572 DB 030H,066H,0CCH,0CCH,0CCH,066H,03CH,000H ; G D_47
FCBA CCCCCFC0CCCC00 5573 DB 0CCH,0CCH,0CCH,0FCH,0CCH,0CCH,0CCH,000H ; H D_48
FCBE 7830303030307800 5574 DB 078H,030H,030H,030H,030H,030H,078H,000H ; I D_49
FCBE IE00C0CCCC7800 5575 DB 01EH,00CCH,00CCH,00CCH,0CCH,0CCH,078H,000H ; J D_4A
FCc6 36666C78666C00 5576 DB 0CCH,066H,066H,078H,066H,066H,066H,000H ; K D_4B
FCC6 F0606060626FE00 5577 DB 0F0H,060H,060H,060H,062H,066H,0FEH,000H ; L D_4C
FCD6 C6E6FE6DE6C6C600 5578 DB 0C6H,0EEH,0FEH,0FEH,0D6H,0C6H,0C6H,000H ; M D_4D
FCDE C6E6F6DECE6C6600 5579 DB 0C6H,0EEH,0FEH,0D6H,0C6H,0C6H,0C6H,000H ; N D_4E
FCe6 3B66C0CC0C663C00 5580 DB 03BH,06CH,066H,066H,066H,03BH,000H ; O D_4F
FCE6 FC66667C606F000 5581 DB 0FCH,066H,066H,07CH,060H,060H,0F0H,000H ; P D_50
FCF6 78CCCCC781C000 5582 DB 078H,0CCH,0CCH,0CCH,0CCH,078H,01CH,000H ; Q D_51
FCFE FC66667C666E600 5583 DB 0FCH,066H,066H,07CH,066H,066H,066H,000H ; R D_52
FD06 18CCE0710CC7800 5584 DB 078H,0CCH,066H,07CH,01CH,0CCH,078H,000H ; S D_53
FD0E FCB4303030307800 5585 DB 0FCH,0B4H,030H,030H,030H,030H,078H,000H ; T D_54
FD16 CCCCCCCCCCFC00 5586 DB 0CCH,0CCH,0CCH,0CCH,0CCH,0CCH,0CCH,000H ; U D_55
FD1E CCCCCCCCC783000 5587 DB 0CCH,0CCH,0CCH,0CCH,0CCH,078H,030H,000H ; V D_56
FD26 C6C6C6FEE6E6C00 5588 DB 0C6H,0C6H,0C6H,0D6H,0FEH,0EEH,0C6H,000H ; W D_57
FD2E C6C6C63838C6C00 5589 DB 0C6H,0C6H,06CH,03BH,03BH,06CH,0C6H,000H ; X D_58
FD36 CCCCC7830307800 5590 DB 0CCH,0CCH,0CCH,078H,030H,030H,078H,000H ; Y D_59
FD3E F6C68183266FE00 5591 DB 0FEH,066H,06CH,01BH,032H,066H,0FEH,000H ; Z D_5A
FD46 7860606060607800 5592 DB 078H,060H,060H,060H,060H,060H,078H,000H ; [ D_5B
FD4E C06030180C060200 5593 DB 0C0H,060H,030H,01BH,00CH,060H,002H,000H ; BACKSLASH D_5C
FD56 7818181818187800 5594 DB 078H,018H,018H,018H,018H,018H,078H,000H ; ] D_5D
FD5E 1038CC6600000000 5595 DB 010H,038H,0C6H,0C6H,0C6H,000H,000H,000H ; CIRCUMFLEX D_5E
FD66 00000000000000FF 5596 DB 000H,000H,000H,000H,000H,000H,000H,000H ; _ D_5F
FD6E 3030180000000000 5597 DB 030H,030H,018H,000H,000H,000H,000H,000H ; ^ D_60
FD76 0000780C7C7C7600 5598 DB 000H,000H,078H,0CCH,07CH,0CCH,078H,000H ; LOWER CASE A D_61
FD7E E60607C6660C00 5599 DB 0E6H,066H,06CH,07CH,066H,066H,0CCH,000H ; L C C B D_62
FD86 000078CC0CC7800 5600 DB 000H,000H,078H,0CCH,0CCH,0CCH,078H,000H ; L C C C D_63
FD8E IC0C07C0CC7C600 5601 DB 01CH,0CCH,0CCH,07CH,0CCH,0CCH,076H,000H ; L C C D D_64
FD96 000078CFCFC07800 5602 DB 000H,000H,078H,0CCH,0FCH,0CCH,078H,000H ; L C C E D_65
FD9E 386C60F960F000 5603 DB 038H,0C6H,066H,066H,066H,066H,066H,000H ; L C C F D_66
FDA6 000078CC7C0CF8 5604 DB 000H,000H,076H,0CCH,0CCH,07CH,0CCH,0FBH ; L C C G D_67
FDAE E0606766666E60 5605 DB 0E0H,060H,06CH,076H,066H,066H,0E6H,000H ; L C C H D_68
FDB6 30007030307800 5606 DB 030H,000H,070H,030H,030H,030H,078H,000H ; L C C I D_69
FDBE 0C000C0C0CC0C78 5607 DB 0CCH,000H,0CCH,0CCH,0CCH,0CCH,0CCH,078H ; L C C J D_6A
FDC6 E060666786CE60 5608 DB 0E0H,060H,066H,06CH,078H,06CH,0E6H,000H ; L C C K D_6B
FDD6 70303030307800 5609 DB 070H,030H,030H,030H,030H,030H,078H,000H ; L C C L D_6C
FDE6 0000CCFEEF0C600 5610 DB 000H,000H,0CCH,0FEH,0FEH,0D6H,0C6H,000H ; L C C M D_6D
FDEE 000F80CC0CC000 5611 DB 000H,000H,0FBH,0CCH,0CCH,0CCH,000H ; L C C N D_6E
FDEE 000078CC0CC780 5612 DB 000H,000H,078H,0CCH,0CCH,0CCH,078H,000H ; L C C O D_6F
FDEE 0000C66667C60F 5613 DB 000H,000H,0CCH,066H,066H,07CH,060H,0F0H ; L C C P D_70
FDF6 000076CC0CC0CIE 5614 DB 000H,000H,076H,0CCH,0CCH,07CH,066H,01EH ; L C C Q D_71
FDFE 0000C76666F000 5615 DB 000H,000H,0CCH,0CCH,0CCH,0CCH,0CCH,01EH ; L C C R D_72
FE06 00007C70780CF80 5616 DB 000H,000H,07CH,0C0H,078H,0C0H,0FBH,000H ; L C C S D_73
FE0E 10307C3030341800 5617 DB 010H,030H,07CH,030H,030H,034H,018H,000H ; L C C T D_74
FE16 0000CC0CC0CC78 5618 DB 000H,000H,0CCH,0CCH,0CCH,0CCH,016H,000H ; L C C U D_75
FE1E 0000CCCCC783000 5619 DB 000H,000H,0CCH,0CCH,0CCH,078H,030H,000H ; L C C V D_76
FE26 0000C6D6FEFE6C00 5620 DB 000H,000H,0C6H,0D6H,0FEH,0FEH,066H,000H ; L C C W D_77
FE2E 0000C6C386CC600 5621 DB 000H,000H,0C6H,06CH,03BH,06CH,0C6H,000H ; L C C X D_78
FE36 0000CC0CC7C0CF8 5622 DB 000H,000H,0CCH,0CCH,0CCH,07CH,0CCH,0FBH ; L C C Y D_79
FE3E 0000F993064FC00 5623 DB 000H,000H,0FCH,09BH,030H,064H,0FCH,000H ; L C C Z D_7A
FE46 IC3030E030301C00 5624 DB 01CH,030H,030H,0E0H,030H,030H,01CH,000H ; [ D_7B
FE4E 1818180018181800 5625 DB 018H,018H,018H,000H,018H,018H,018H,000H ; ] D_7C
FE56 E030301C3030E000 5626 DB 0E0H,030H,030H,01CH,030H,030H,0E0H,000H ; ^ D_7D
FE5E 76D0000000000000 5627 DB 076H,0DCH,000H,000H,000H,000H,000H,000H ; TILDE D_7E
FE66 0010386C6C6FE00 5628 DB 000H,010H,038H,06CH,0C6H,0C6H,0FEH,000H ; DELTA D_7F

```

```

5629
5630 ---- INT IA
5631 ; TIME_OF_DAY
5632 ; THIS ROUTINE ALLOWS THE CLOCK TO BE SET/READ
5633 ;
5634 ; INPUT
5635 ; (AH) = 0 READ THE CURRENT CLOCK SETTING
5636 ; RETURNS CX = HIGH PORTION OF COUNT
5637 ; DX = LOW PORTION OF COUNT
5638 ; AL = 0 IF TIMER HAS NOT PASSED
5639 ; 24 HOURS SINCE LAST READ
5640 ; <=0 IF ON ANOTHER DAY
5641 ; (AH) = 1 SET THE CURRENT CLOCK
5642 ; CX = HIGH PORTION OF COUNT
5643 ; DX = LOW PORTION OF COUNT
5644 ; NOTE: COUNTS OCCUR AT THE RATE OF
5645 ; 1/1921845533 COUNTS/SEC
5646 ; (OR ABOUT 18.2 PER SECOND -- SEE EQUATES BELOW)
5647
5648 ASSUME CS:CODE,DS:DATA
FE6E 5649 ORG OFE6EH
FE6E 5650 PROC FAR
FE6E FB 5651 STI ; INTERRUPTS BACK ON
FE6F IE 5652 PUSH DS ; SAVE SEGMENT
FE70 E8E6FB 5653 CALL DDS
FE73 0AE4 5654 OR AH,AH ; AH=0
FE75 7407 5655 JZ T2 ; READ_TIME
FE77 FECC 5656 AH DEC AH ; READ_TIME
FE79 7416 5657 JZ T3 ; SET_TIME
FE7B FB 5658 T1: STI ; TOD_RETURN
FE7C IF 5659 POP DS ; INTERRUPTS BACK ON
FE7D CF 5661 IRET ; RECOVER SEGMENT
FE7E 5662 T2: ; RETURN TO CALLER
FE7E FA 5663 CLI ; READ_TIME
FE7E A07000 5664 MOV AL,TIMER_OFL ; NO TIMER INTERRUPTS WHILE READING
FE82 C06070000 5665 MOV TIMER_OFL,0 ; GET OVERFLOW, AND RESET THE FLAG
FE87 8B0E6E00 5666 MOV CX,TIMER_HIGH
FE8B 8B166C00 5667 MOV DX,TIMER_LOW
FE8F EBEA 5668 JMP T1 ; TOD_RETURN
FE91 5669 T3: ; SET_TIME
FE91 FA 5670 CLI ; NO INTERRUPTS WHILE WRITING
FE92 89166C00 5671 MOV TIMER_LOW,DX
FE96 890E6E00 5672 MOV TIMER_HIGH,CX
FE9A C06070000 5673 MOV TIMER_OFL,0 ; SET THE TIME
FE9F EBD4 5674 JMP T1 ; RESET OVERFLOW
5675 5675 TIME_OF_DAY ENDP ; TOD_RETURN

```



```

5676 ;
5677 ; -----
5678 ; THIS ROUTINE HANDLES THE TIMER INTERRUPT FROM ;
5679 ; CHANNEL 0 OF THE 8253 TIMER. INPUT FREQUENCY ;
5680 ; IS 1.19318 MHZ AND THE DIVISOR IS 65536, RESULTING ;
5681 ; IN APPROX. 18.2 INTERRUPTS EVERY SECOND. ;
5682 ;
5683 ; THE INTERRUPT HANDLER MAINTAINS A COUNT OF INTERRUPTS ;
5684 ; SINCE POWER ON TIME, WHICH MAY BE USED TO ESTABLISH ;
5685 ; TIME OF DAY. ;
5686 ; THE INTERRUPT HANDLER ALSO DECREMENTS THE MOTOR ;
5687 ; CONTROL COUNT OF THE DISKETTE, AND WHEN IT EXPIRES, ;
5688 ; WILL TURN OFF THE DISKETTE MOTOR, AND RESET THE ;
5689 ; MOTOR RUNNING FLAGS. ;
5690 ; THE INTERRUPT HANDLER WILL ALSO INVOKE A USER ROUTINE ;
5691 ; THROUGH INTERRUPT ICH AT EVERY TIME TICK. THE USER ;
5692 ; MUST CODE A ROUTINE AND PLACE THE CORRECT ADDRESS IN ;
5693 ; THE VECTOR TABLE. ;
5694 ; -----
FEA5 5695 ORG OFEA5H
FEA5 5696 TIMER_INT PROC FAR
FEA5 FB 5697 STI ; INTERRUPTS BACK ON
FEA6 1E 5698 PUSH DS
FEA7 50 5699 PUSH AX
FEA8 52 5700 PUSH DX ; SAVE MACHINE STATE
FEA9 EBADF8 5701 CALL DDS
FEAC FF06C00 5702 INC TIMER_LOW ; INCREMENT TIME
FEB0 7504 5703 JNZ T4 ; TEST DAY
FEB2 FF06E00 5704 INC TIMER_HIGH ; INCREMENT HIGH WORD OF TIME
FEB6 833E6E0018 5705 ;
FEBB 7515 5706 JNZ T4 ; TEST FOR COUNT EQUALING 24 HOURS
FEBD 813E6C00B000 5707 CMP TIMER_HIGH,018H ; DISKETTE_CTL
FEC3 750D 5708 JNZ T5 ; DISKETTE_CTL
5709
5710 ; ----- TIMER HAS GONE 24 HOURS
5711
5712
5713 SUB AX,AX
5714 MOV TIMER_HIGH,AX
5715 MOV TIMER_LOW,AX
5716 MOV TIMER_OFL,1
5717
5718 ; ----- TEST FOR DISKETTE TIME OUT
5719
5720 T5:
5721 DEC MOTOR_COUNT ; DISKETTE_CTL
5722 JNZ T6 ; RETURN IF COUNT NOT OUT
5723 AND MOTOR_STATUS,0F0H ; TURN OFF MOTOR RUNNING BITS
5724 MOV AL,0CH
5725 MOV DX,03F2H ; FDC CTL PORT
5726 OUT DX,AL ; TURN OFF THE MOTOR
5727 T6:
5728 INT 1CH ; TIMER RET:
5729 MOV AL,EO1 ; TRANSFER CONTROL TO A USER ROUTINE
5730 OUT 020H,AL ; END OF INTERRUPT TO 8259
5731 POP DX
5732 POP AX
5733 POP DS ; RESET MACHINE STATE
5734 IRET ; RETURN FROM INTERRUPT
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 OFF5BH
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 D11 ; 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 VIDEO_IO ; INTERRUPT 10H
5756 DW OFFSET EQUIPMENT ; INTERRUPT 11H
5757 DW OFFSET MEMORY_SIZE_DET ; INTERRUPT 12H
5758 DW OFFSET DISKETTE_IO ; INTERRUPT 13H
5759 DW OFFSET RS232_IO ; INTERRUPT 14H
5760 DW CASSETTE_IO ; INTERRUPT 15H(FORMER CASSETTE IO)
5761 DW OFFSET KEYBOARD_IO ; INTERRUPT 16H
5762 DW OFFSET PRINTER_IO ; INTERRUPT 17H
5763
5764 DW 00000H ; INTERRUPT 18H
5765 ; DW 0F600H ; MUST BE INSERTED INTO TABLE LATER
5766
5767 DW OFFSET BOOT_STRAP ; INTERRUPT 19H
5768 DW TIME_OF_DAY ; INTERRUPT 1AH -- TIME OF DAY
5769 DW DUMMY_RETURN ; INTERRUPT 1BH -- KEYBOARD BREAK ADDR
5770 DW DUMMY_RETURN ; INTERRUPT 1C -- TIMER BREAK ADDR
5771 DW VIDEO_PARMS ; INTERRUPT 1D -- VIDEO PARAMETERS
5772 DW OFFSET DISK_BASE ; INTERRUPT 1E -- DISK PARMS
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. LOCATION "INTR_FLAG" WILL ;
5780 ; CONTAIN EITHER: 1. LEVEL OF HARDWARE INT. THAT ;
5781 ; CAUSED CODE TO BE EXEC. ;
5782 ; 2. "FF" FOR NON-HARDWARE INTERRUPTS THAT WAS ;
5783 ; EXECUTED ACCIDENTLY. ;
5784 -----
FF23 D11 PROC NEAR
5785 ASSUME DS:DATA
5786 PUSH DS
5787 PUSH DX
5788 ; SAVE REG AX CONTENTS
5789 PUSH AX
5790 CALL DD5 ; READ IN-SERVICE REG
5791 MOV AL,0BH ; (FIND OUT WHAT LEVEL BEING
5792 OUT INTA00,AL ; SERVICED)
5793 NOP ; GET LEVEL
5794 IN AL,INTA00 ; SAVE IT
5795 MOV AH,AL ; 00? (NO HARDWARE ISR ACTIVE)
5796 OR AL,AH
5797 JNZ HW_INT ; SET FLAG TO FF IF NON-HDWARE
5798 MOV AH,OFFH ;
5799 JMP SHORT SET_INTR_FLAG ;
5800 HW_INT:
5801 IN AL,INTA01 ; GET MASK VALUE
5802 OR AL,AH ; MASK OFF LVL BEING SERVICED
5803 OUT INTA01,AL
5804 MOV AL,E01
5805 OUT INTA00,AL
5806 SET_INTR_FLAG:
5807 MOV INTR_FLAG,AH ; SET FLAG
5808 POP AX ; RESTORE REG AX CONTENTS
5809 POP DX
5810 POP DS
5811 DUMMY_RETURN:
5812 IRET ; NEED IRET FOR VECTOR TABLE
5813 D11 ENDP
5814
5815 -----
5816 ; DUMMY RETURN FOR ADDRESS COMPATIBILITY ;
5817 -----
5818 ORG OFF53H
5819 IRET
5820

```

```

5821 :--- INT 5
5822 : THIS LOGIC WILL BE INVOKED BY INTERRUPT 05H TO PRINT THE
5823 : SCREEN. THE CURSOR POSITION AT THE TIME THIS ROUTINE IS INVOKED
5824 : WILL BE SAVED AND RESTORED UPON COMPLETION. THE ROUTINE IS
5825 : INTENDED TO RUN WITH INTERRUPTS ENABLED. IF A SUBSEQUENT
5826 : 'PRINT SCREEN' KEY IS DEPRESSED DURING THE TIME THIS ROUTINE
5827 : IS PRINTING IT WILL BE IGNORED.
5828 : ADDRESS 50:0 CONTAINS THE STATUS OF THE PRINT SCREEN:
5829 :
5830 : 50:0 =0 EITHER PRINT SCREEN HAS NOT BEEN CALLED
5831 : OR UPON RETURN FROM A CALL THIS INDICATES
5832 : A SUCCESSFUL OPERATION.
5833 : =1 PRINT SCREEN IS IN PROGRESS
5834 : =255 ERROR ENCOUNTERED DURING PRINTING
5835 :---
5836 : ASSUME CS:CODE,DS:XXDATA
5837 : ORG OFF54H
5838 PRINT_SCREEN PROC FAR
5839 STI
5840 PUSH DS ; MUST RUN WITH INTERRUPTS ENABLED
5841 PUSH AX ; MUST USE 50:0 FOR DATA AREA STORAGE
5842 PUSH BX
5843 PUSH CX ; WILL USE THIS LATER FOR CURSOR LIMITS
5844 PUSH DX ; WILL HOLD CURRENT CURSOR POSITION
5845 MOV AX,XXDATA ; HEX 50
5846 MOV DS,AX
5847 CMP STATUS_BYTE,1 ; SEE IF PRINT ALREADY IN PROGRESS
5848 JZ EXIT ; JUMP IF PRINT ALREADY IN PROGRESS
5849 MOV STATUS_BYTE,1 ; INDICATE PRINT NOW IN PROGRESS
5850 MOV AH,15 ; WILL REQUEST THE CURRENT SCREEN MODE
5851 INT 10H ; [AL]=MODE
5852 ; [AH]=NUMBER COLUMNS/LINE
5853 ; [BH]=VISUAL PAGE
5854 :---
5855 : AT THIS POINT WE KNOW THE COLUMNS/LINE ARE IN
5856 : [AX] AND THE PAGE IF APPLICABLE IS IN[BH]. THE STACK
5857 : HAS DS,AX,BX,CX,DX PUSHED. [A] HAS VIDEO MODE
5858 :---
5859 MOV CL,AH ; WILL MAKE USE OF [CX] REGISTER TO
5860 MOV CH,25 ; CONTROL ROW & COLUMNS
5861 CALL CRLF ; CARRIAGE RETURN LINE FEED ROUTINE
5862 PUSH CX ; SAVE SCREEN BOUNDS
5863 MOV AH,3 ; WILL NOW READ THE CURSOR.
5864 INT 10H ; AND PRESERVE THE POSITION
5865 POP CX ; RECALL SCREEN BOUNDS
5866 PUSH DX ; RECALL [BH]=VISUAL PAGE
5867 XOR DX,DX ; WILL SET CURSOR POSITION TO [0,0]
5868 :---
5869 : THE LOOP FROM PRI10 TO THE INSTRUCTION PRIOR TO PRI20
5870 : IS THE LOOP TO READ EACH CURSOR POSITION FROM THE
5871 : SCREEN AND PRINT.
5872 :---
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 :---
5882 PRI15:
5883 PUSH DX ; SAVE CURSOR POSITION
5884 XOR DX,DX ; INDICATE PRINTER I
5885 MOV AH,AH ; TO INDICATE PRINT CHAR IN [AL]
5886 INT 17H ; PRINT THE CHARACTER
5887 POP DX ; RECALL CURSOR POSITION
5888 TEST AH,25H ; TEST FOR PRINTER ERROR
5889 JNC ERR10 ; JUMP IF ERROR DETECTED
5890 INC DL ; ADVANCE TO NEXT COLUMN
5891 CMP CL,DL ; SEE IF AT END OF LINE
5892 JNZ PRI10 ; IF NOT PROCEED
5893 XOR DL,DL ; BACK TO COLUMN 0
5894 MOV AH,DL ; [AH]=0
5895 PUSH DX ; SAVE NEW CURSOR POSITION
5896 CALL CRLF ; LINE FEED CARRIAGE RETURN
5897 POP DX ; RECALL CURSOR POSITION
5898 DH ; ADVANCE TO NEXT LINE
5899 CMP CH,DH ; FINISHED?
5900 JNZ PRI10 ; IF NOT CONTINUE
5901 :---
5902 PRI20:
5903 POP DX ; RECALL CURSOR POSITION
5904 MOV AH,2 ; TO INDICATE CURSOR SET REQUEST
5905 INT 10H ; TO INDICATE CURSOR SET REQUEST
5906 MOV STATUS_BYTE,0 ; CURSOR POSITION RESTORED
5907 JMP SHORT EXIT ; INDICATE FINISHED
5908 :---
5909 ERR10:
5910 POP DX ; GET CURSOR POSITION
5911 MOV AH,2 ; TO REQUEST CURSOR SET
5912 INT 10H ; CURSOR POSITION RESTORED
5913 :---
5914 ERR20:
5915 MOV STATUS_BYTE,0FFH ; INDICATE ERROR
5916 :---
5917 EXIT:
5918 POP DX ; RESTORE ALL THE REGISTERS USED
5919 POP CX
5920 POP BX
5921 POP AX
5922 POP DS
5923 IRET
5924 PRINT_SCREEN ENDP
5925 :----- CARRIAGE RETURN, LINE FEED SUBROUTINE
5926 :---
5927 CRLF PROC NEAR
5928 XOR DX,DX ; PRINTER 0
5929 MOV AH,AH ; WILL NOW SEND INITIAL LF,CR
5930 XOR AH,AH ; TO PRINTER
5931 LF
5932 INT 17H ; SEND THE LINE FEED
5933 MOV AH,AH ; NOW FOR THE CR
5934 XOR AH,15Q ; CR
5935 INT 17H ; SEND THE CARRIAGE RETURN
5936 RET
5937 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 PRT_SEG PROC NEAR
FFDA 8AC6 MOV AL,DIH ;GET MSB
FFDC E8ACF9 CALL XPC_BYTE
FFDF 8AC2 MOV AL,DL ;LSB
FFE1 E8A7F9 CALL XPC_BYTE
FFE4 B030 MOV AL,'0' ; PRINT A '0 '
FFE6 E8B3F9 CALL PRT_HEX
FFE9 B020 MOV AL,' '
FFEB E8AEF9 CALL PRT_HEX
FFEE C3 RET
----- PRT_SEG ENDP
5950
5951 CODE ENDS
5952
5953 |-----|
5954 | POWER ON RESET VECTOR |
5955 |-----|
5956 VECTOR SEGMENT AT 0FFFFH
5957
5958 |----- POWER ON RESET
5959
0000 EA5BE00F0 JMP RESET
5960
0005 31312F30382F38 DB '11/08/82' ; RELEASE MARKER
32
-----
5963 VECTOR ENDS
5964 END

```

SECTION 6. INSTRUCTION SET

8088 Register Model	6-3
Operand Summary	6-4
Second Instruction Byte Summary	6-4
Memory Segmentation Model	6-5
Segment Override Prefix	6-6
Use of Segment Override	6-6
8088 Instruction Set	6-7
Data Transfer	6-7
Arithmetic	6-10
Logic	6-13
String Manipulation	6-15
Control Transfer	6-16
8088 Instruction Set Matrix	6-20
8088 Conditional Transfer Operations	6-22
Processor Control	6-23
8087 Coprocessor Instruction Set	6-24
Data Transfer	6-24
Comparison	6-25
Arithmetic	6-26
Transcendental	6-28
Constants	6-28
Processor Control	6-29

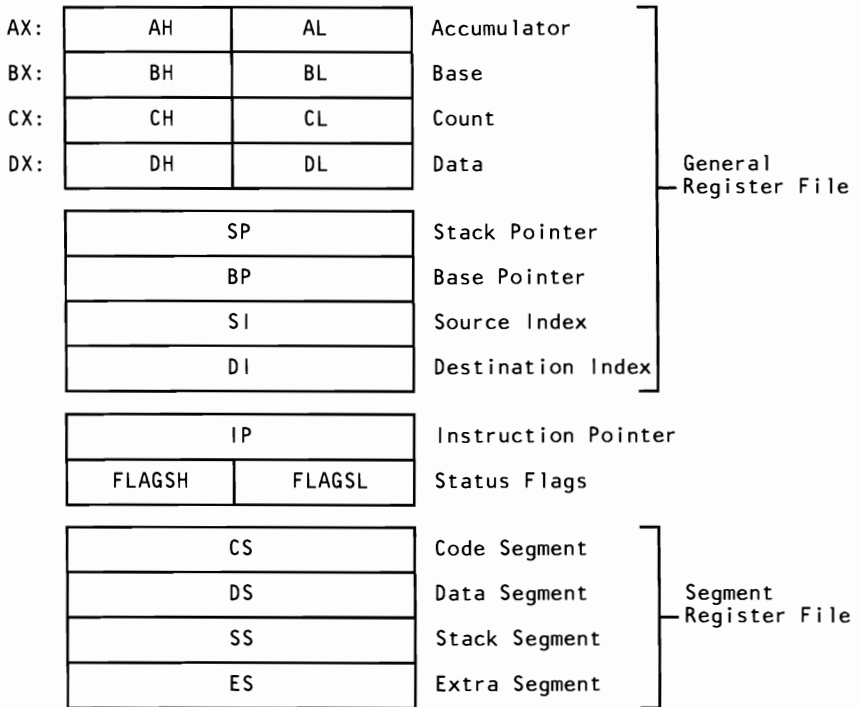
Notes:



8088 Register Model

Notes:

if d = 1 then "to"; if d = 0 then "from"
 if w = 1 then word instruction; if w = 0 then byte instruction
 if s:w = 01 then 16 bits of immediate data from the operand
 if s:w = 11 then an immediate data byte is signed extended to form the 16-bit operand
 if v = 0 the "count" = 1; if v = 1 the "count" is in (CL) or (CX)
 x = don't care
 z is used for string primitives for comparison with ZF FLAG
 AL = 8-bit accumulator
 AX = 16-bit accumulator
 CX = Count register
 DS = Data segment
 ES = Extra segment
 Above/below refers to unsigned value
 Greater = more positive;
 Less = less positive (more negative) signed values



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



X = Don't Care

AF: Auxiliary Carry - BCD	}	8080 Flags
CF: Carry Flag		
PF: Parity Flag		
SF: Sign Flag		
ZF: Zero Flag		
DF: Direction Flag	}	8088 Flags
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

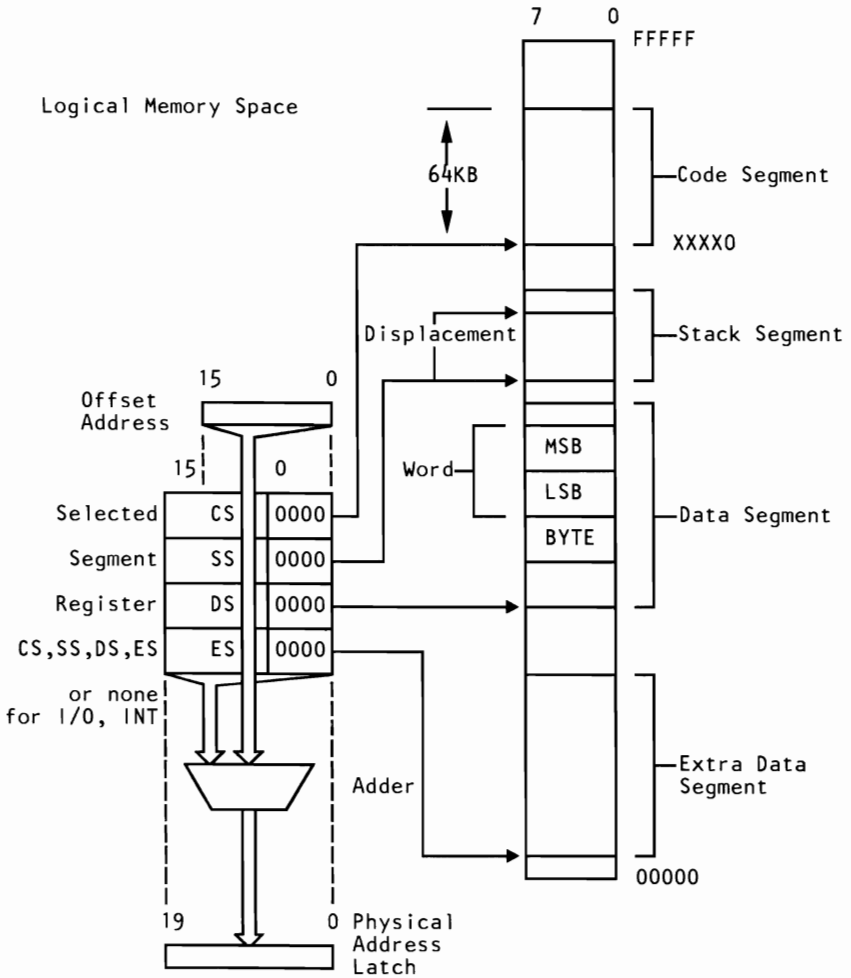
mod	xxx	r/m
-----	-----	-----

mod | Displacement

00	DISP = 0*, disp-low and disp-high are absent
01	DISP = disp-low sign-extended to 16-bits, disp-high is absent
10	DISP = disp-high: disp-low
11	r/m is treated as a "reg" field

DISP follows 2nd byte of instruction (before data if required)
 *except if mod=00 and r/m=110 then EA=disp-high: disp-low.

Memory Segmentation Model



Segment Override Prefix

001reg110

Use of Segment Override

Operand Register	Default	With Override Prefix
IP (Code Address)	CS	Never
SP (Stack Address)	SS	Never
BP (Stack Address or Stack Marker)	SS	BP + DS or ES, or CS
SI or DI (not including strings)	DS	ES, SS, or CS
SI (Implicit Source Address for strings)	DS	ES, SS, or CS
DI (Implicit Destination Address for strings)	ES	Never

8088 Instruction Set

Data Transfer

MOV = Move

Register/Memory to/from Register

100010dw	mod reg r/m
----------	-------------

Immediate to Register/Memory

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

Immediate to Register

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

Memory to Accumulator

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

Accumulator to Memory

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

Register/Memory to Segment Register

10001110	mod 0 reg r/m
----------	---------------

Segment Register to Register/Memory

10001100	mod 0 reg r/m
----------	---------------

PUSH = Push

Register/Memory

11111111	mod 110 r/m
----------	-------------

Register

01010 reg

Segment Register

000 reg 110

POP = Pop

Register/Memory

10001111	mod 000 r/m
----------	-------------

Register

01011reg

Segment Register

000 reg 111

XCHG = Exchange

Register/Memory with Register

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

Register with Accumulator

10010reg

IN = Input to AL/AX from

Fixed Port

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

Variable Port

1110110w

OUT = Output from AL/AX to

Fixed Port

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

Variable Port (DX)

1110110w

XLAT = Translate Byte to AL

11010111

LEA = Load EA to Register

10001101 mod reg r/m

LDS = Load Pointer to DS

11000101 mod reg r/m

LES = Load Pointer to ES

11000100 mod reg r/m

LAHF = Load AH with Flags

10011111

SAHF = Store AH with Flags

10011110

PUSHF = Push Flags

10011100

POPF = Pop Flags

10011101

Arithmetic

ADD = Add

Register/Memory with Register to Either

00000dw	mod reg r/m
---------	-------------

Immediate to Register Memory

10000sw	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

10000sw	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

10000sw	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

10000sw	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
HI 0	ADD b,b,r/m	ADD w,f,r/m	ADD b,t,r/m	ADD w,t,r/m	ADD b,ia	ADD w,ia	PUSH ES	POP ES
1	ADC b,f,r/m	ADC w,f,r/m	ADC b,t,r/m	ADC w,t,r/m	ADC b,i	ADC w,i	PUSH SS	POP SS
2	AND b,f,r/m	AND w,f,r/m	AND b,t,r/m	AND w,t,r/m	AND b,i	AND w,i	DEG =ES	DAA
3	XOR b,f,r/m	XOR w,f,r/m	XOR b,t,r/m	XOR w,t,r/m	XOR b,i	XOR w,i	SEG =S+	AAA
4	INC AX	INC CX	INC DX	INC BX	INC SP	INC BP	INC SI	INC DI
5	PUSH AX	PUSH CX	PUSH DX	PUSH BX	PUSH SP	PUSH BP	PUSH SI	PUSH DI
6								
7	JO	JNO	JB/ JNAE	JNB/ JAE	JE/ JZ	JNE/ JNZ	JBE/ JNA	JNBE/ JA
8	Immed b,r/m	Immed w,r/m	Immed b,r/m	Immed is,r/m	TEST b,r/m	TEST w,r/m	XCHG b,r/m	XCHG w,r/m
9	NOP	XCHG CX	XCHG DX	XCHG BX	XCHG SP	XCHG BP	XCHG SI	XCHG DI
A	MOV m AL	MOV m AL	MOV AL m	MOV AL m	MOVS b	MOVS w	CMPS b	CMPS w
B	MOV i AL	MOV i CL	MOV i DL	MOV i BL	MOV i AH	MOV i CH	MOV i DH	MOV i BH
C			RET (I+SP)	RET	LES	LDS	MOV b,i,r/m	MOV w,i,r/m
D	Shift b	Shift w	Shift b,v	Shift w,v	AAM	AAD		XLAT
E	LOOPNZ/ LOOPNE	LOOPZ/ LOOPPE	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

	8	9	A	B	C	D	E	F
HI 0	OR b,f,r/m	w,f,r/m	OR b,t,r/m	OR w,t,r/m	OR b,i	OR w,i	PUSH CS	
1	SBB b,f,r/m	SBB w,f,r/m	SBB b,t,r/m	SBB w,t,r/m	SBB b,i	SBB w,i	PUSH DS	POP DS
2	SUB b,f,r/m	SUB w,f,r/m	SUB b,t,r/m	SUB w,t,r/m	SUB b,i	SUB w,i	SEG= CS	DAS
3	CMP b,f,r/m	CMP w,f,r/m	CMP b,t,r/m	CMP w,t,r/m	CMP b,i	CMP w,i	SEG= CS	AAS
4	DEC AX	DEC CX	DEC DX	DEC BX	DEC SP	DEC BP	DEC SI	DEC DI
5	POP AX	POP CX	POP DX	POP BX	POP SP	POP BP	POP SI	POP DI
6								
7	JS	JNS	JP/ JPE	JNP/ JPO	JL/ JNGE	JNL/ JGE	JLE/ JNG	JNLE/ JG
8	MOV b,f,r/m	MOV w,f,r/m	MOV b,t,r/m	MOV w,t,r/m	MOV sr,t,r/m	LEA	MOV sr,f,r/m	POP r/m
9	CBW	CWD CX	CALL l,d	WAIT BX	PUSHF SP	POPF BP	SAHF SI	LAHF DI
A	TEST b,i	TEST w,i	STOS b	STOS w	LODS b	LODS w	SCAS b	SCAS w
B	MOV i AX	MOV i CX	MOV i DX	MOV i BX	MOV i SP	MOV i BP	MOV i SI	MOV i DI
C			RET l,(l+SP)	RET l	INT Type 3	INT (Any)	INTO	IRET
D	ESC 0	ESC 1	ESC 2	ESC 3	ESC 4	ESC 5	ESC 6	ESC 7
E	CALL d	JMP d	JMP l,d	JMP si,d	IN v,b	IN v,w	OUT v,b	OUT v,w
F	CLC	STC	CLI	STI	CLD	STD	Grp 2 b,r/m	Grp 3 w,r/m

where:

mod r/m	000	001	010	011	100	101	110	111
Immed	ADD	OR	ADC	SBB	AND	SUB	XOR	CMP
Shift	ROL	ROR	RCL	RCR	SHL/SAL	SHR	--	SAR
Grp 1	TEST	--	NOT	NEG	MUL	IMUL	DIV	DIV
Grp 2	INC	DEC	CALL id	CALL l,id	JMP id	JMP l,id	PUSH	--

8088 Conditional Transfer Operations

Instruction	Condition	Interpretation
JE or JZ	ZF = 1	"equal" or "zero"
JL or JNGE	(SF xor OF) = 1	"less" or "not greater or equal"
JLE or JNG	((SF xor OF) or ZF) = 1	"less or equal" or "not greater"
JB or JNAE or JC	CF = 1	"below" or "not above or equal"
JBE or JNA	(CF or ZF) = 1	"below or equal" or "not above"
JP or JPE	PF = 1	"parity" or "parity even"
JO	OF = 1	"overflow"
JS	SF = 1	"sign"
JNE or JNZ	ZF = 0	"not equal" or "not zero"
JNL or JGE	(SF xor OF) = 0	"not less" or "greater or equal"
JNLE or JG	((SF xor OF) or ZF) = 0	"not less or equal" or "greater"
JNB or JAE or JNC	CF = 0	"not below" or "above or equal"
JNBE or JA	(CF or ZF) = 0	"not below or equal" or "above"
JNP or JPO	PF = 0	"not parity" or "parity odd"
JNO	OF = 0	"not overflow"
JNS	SF = 0	"not sign"

"Above" and "below" refer to the relation between two unsigned values, while "greater" and "less" refer to the relation between two signed values.

INT = Interrupt

Type Specified

11001101	Type
----------	------

Type 3

11001100

INTO = Interrupt on Overflow

11001110

IRET = Interrupt Return

11001111

Processor Control

CLC = Clear Carry

11111000

STC = Set Carry

11111001

CMC = Complement Carry

11110101

NOP = No Operation

10010000

CLD = Clear Direction

11111100

STD = Set Direction

11111101

CLI = Clear Interrupt

11111010

STI = Set Interrupt

11111011

HLT = Halt

11110100

WAIT = Wait

10011011

LOCK = Bus lock prefix

11110000

ESC = Escape (to 8087)

11011xxx	mod xxx r/m
----------	-------------

8087 Coprocessor Instruction Set

The following is an instruction set summary for the 8087 coprocessor. In the following, the bit pattern for escape is 11011.

MF = Memory format	r/m	Operand Address
00 - 32-bit Real	000	(BX) + (SI) + DISP
01 - 32-bit Integer	001	(BX) + (DI) + DISP
10 - 64-bit Real	010	(BP) + (SI) + DISP
11 - 64-bit Integer	011	(BP) + (DI) + DISP
	100	(SI) + DISP
	101	(DI) + DISP
	110	(BP) + DISP*
	100	(BX) + DISP

DISP follows 2nd byte of instruction (before data if required)
 *except if mod=00 and r/m=110 then EA=disp-high: disp-low.

Data Transfer

FLD = Load

Integer/Real Memory to ST(0)

escape MF 1	mod 000 r/m	disp-low	disp-high
-------------	-------------	----------	-----------

Long Integer Memory to ST(0)

escape 111	mod 101 r/m	disp-low	disp-high
------------	-------------	----------	-----------

Temporary Real Memory to ST(0)

escape 011	mod 101 r/m	disp-low	disp-high
------------	-------------	----------	-----------

BCD Memory to ST(0)

escape 111	mod 100 r/m	disp-low	disp-high
------------	-------------	----------	-----------

ST(i) to ST(0)

escape 001	11000ST(i)		
------------	------------	--	--

FST = Store

ST(0) to Integer/Real Memory

escape MF 1	mod 010 r/m	disp-low	disp-high
-------------	-------------	----------	-----------

ST(0) to ST(i)

escape 101	11010 ST(i)
------------	-------------

FSTP = Store and Pop

ST(0) to Integer/Real Memory

escape MF 1	mod 011 r/m	disp-low	disp-high
-------------	-------------	----------	-----------

ST(0) to Long Integer Memory

escape 111	mod 111 r/m	disp-low	disp-high
------------	-------------	----------	-----------

ST(0) to Temporary Real Memory

escape 011	mod 111 r/m	disp-low	disp-high
------------	-------------	----------	-----------

ST(0) to BCD Memory

escape 111	mod 110 r/m	disp-low	disp-high
------------	-------------	----------	-----------

ST(0) to ST(i)

escape 101	11011 ST(i)
------------	-------------

FXCH = Exchange ST(i) and ST(0)

escape 001	11001 ST(i)
------------	-------------

Comparison

FCOM = Compare

Integer/Real Memory to ST(0)

escape MF 0	mod 010 r/m	disp-low	disp-high
-------------	-------------	----------	-----------

ST(i) to ST(0)

escape 000	11010 ST(i)
------------	-------------

FCOMP = Compare and Pop

Integer/Real Memory to ST(0)

escape MF 0	mod 011 r/m	disp-low	disp-high
-------------	-------------	----------	-----------

ST(i) to ST(0)

escape 000	11010 ST(i)
------------	-------------

FCOMP = 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
------------	----------

FEXTRACT = 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 $\text{Log}_2 10$ into ST(0)

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

FLDLG2 = Load $\text{Log}_{10} 2$ into ST(0)

escape 001	11101100
------------	----------

FLDLN2 = Load $\text{Log}_e 2$ into ST(0)

escape 001	11101101
------------	----------

Processor Control

FINIT = Initialize NDP

escape 011	11100011
------------	----------

FENI = Enable Interrupts

escape 011	11100000
------------	----------

FDISI = Disable Interrupts

escape 011	11100001
------------	----------

FLDCW = Load Control Word

escape 001	mod101 r/m	disp-low	disp-high
------------	------------	----------	-----------

FSTCW = Store Control Word

escape 001	mod111 r/m	disp-low	disp-high
------------	------------	----------	-----------

FSTSW = Store Status Word

escape 101	mod111 r/m	disp-low	disp-high
------------	------------	----------	-----------

FCLEX = Clear Exceptions

escape 011	11100010
------------	----------

FSTENV = Store Environment

escape 001	mod110 r/m	disp-low	disp-high
------------	------------	----------	-----------

FLDENV = Load Environment

escape 001	mod100 r/m	disp-low	disp-high
------------	------------	----------	-----------

FSAVE = Save State

escape 101	mod110 r/m	disp-low	disp-high
------------	------------	----------	-----------

FRSTOR = Restore State

escape 101	mod100 r/m	disp-low	disp-high
------------	------------	----------	-----------

FINCSTP = Increment Stack Pointer

escape 001	11110111
------------	----------

FDECSTP = Decrement Stack Pointer

escape 001	11110110
------------	----------

FFREE = Free ST(i)

escape 001	11000ST(i)
------------	------------

FNOP = No Operation

escape 001	11010000
------------	----------

FWAIT = CPU Wait for NDP

10011011

Notes:

ST(0) = Current Stack top

ST(i) = i^{th} register below Stack top

d = Destination

0—Destination is ST(0)

1—Destination is ST(i)

P = POP

0—No Pop

1—Pop ST(0)

R = Reverse

0—Destination (op) Source

1—Source (op) Destination

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

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

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

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

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

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

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

Notes:



SECTION 7. CHARACTERS, KEYSTROKES, AND COLORS

Character Codes	7-3
Quick Reference	7-14

Notes:



Character Codes

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

Value		As Characters			As Text Attributes		
					Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground	
18	24	↑	Ctrl X		Blue	Dark Grey	High Intensity
19	25	↓	Ctrl Y		Blue	Light Blue	High Intensity Underline
1A	26	→	Ctrl Z		Blue	Light Green	High Intensity
1B	27	←	Ctrl [, Esc, Shift Esc, Ctrl Esc		Blue	Light Cyan	High Intensity
1C	28	└─	Ctrl \		Blue	Light Red	High Intensity
1D	29	↔	Ctrl]		Blue	Light Magenta	High Intensity
1E	30	▲	Ctrl 6		Blue	Yellow	High Intensity
1F	31	▼	Ctrl —		Blue	White	High Intensity
20	32	Blank Space	Space Bar, Shift, Space, Ctrl Space, Alt Space		Green	Black	Normal
21	33	!	!	Shift	Green	Blue	Underline
22	34	”	”	Shift	Green	Green	Normal
23	35	#	#	Shift	Green	Cyan	Normal
24	36	\$	\$	Shift	Green	Red	Normal
25	37	%	%	Shift	Green	Magenta	Normal
26	38	&	&	Shift	Green	Brown	Normal
27	39	'	'		Green	Light Grey	Normal
28	40	((Shift	Green	Dark Grey	High Intensity
29	41))	Shift	Green	Light Blue	High Intensity Underline
2A	42	*	*	Note 1	Green	Light Green	High Intensity
2B	43	+	+	Shift	Green	Light Cyan	High Intensity
2C	44	,	,		Green	Light Red	High Intensity
2D	45	-	-		Green	Light Magenta	High Intensity
2E	46	.	.	Note 2	Green	Yellow	High Intensity

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

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

Value		As Characters			As Text Attributes		
					Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground	
67	103	g	g	Note 5	Brown	Light Grey	Normal
68	104	h	h	Note 5	Brown	Dark Grey	High Intensity
69	105	i	i	Note 5	Brown	Light Blue	High Intensity Underline
6A	106	j	j	Note 5	Brown	Light Green	High Intensity
6B	107	k	k	Note 5	Brown	Light Cyan	High Intensity
6C	108	l	l	Note 5	Brown	Light Red	High Intensity
6D	109	m	m	Note 5	Brown	Light Magenta	High Intensity
6E	110	n	n	Note 5	Brown	Yellow	High Intensity
6F	111	o	o	Note 5	Brown	White	High Intensity
70	112	p	p	Note 5	Light Grey	Black	Reverse Video
71	113	q	q	Note 5	Light Grey	Blue	Underliné
72	114	r	r	Note 5	Light Grey	Green	Normal
73	115	s	s	Note 5	Light Grey	Cyan	Normal
74	116	t	t	Note 5	Light Grey	Red	Normal
75	117	u	u	Note 5	Light Grey	Magenta	Normal
76	118	v	v	Note 5	Light Grey	Brown	Normal
77	119	w	w	Note 5	Light Grey	Light Grey	Normal
78	120	x	x	Note 5	Light Grey	Dark Grey	Reverse Video
79	121	y	y	Note 5	Light Grey	Light Blue	High Intensity Underline
7A	122	z	z	Note 5	Light Grey	Light Green	High Intensity
7B	123	{	{	Shift	Light Grey	Light Cyan	High Intensity
7C	124			Shift	Light Grey	Light Red	High Intensity
7D	125	}	}	Shift	Light Grey	Light Magenta	High Intensity
7E	126	~	~	Shift	Light Grey	Yellow	High Intensity
7F	127	△	Ctrl -		Light Grey	White	High Intensity

Value		As Characters			As Text Attributes		
					Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground	
*** 80 to FF Hex are Flashing in both Color & IBM Monochrome ***							
80	128	Ç	Alt 128	Note 6	Black	Black	Non-Display
81	129	ü	Alt 129	Note 6	Black	Blue	Underline
82	130	é	Alt 130	Note 6	Black	Green	Normal
83	131	â	Alt 131	Note 6	Black	Cyan	Normal
84	132	ã	Alt 132	Note 6	Black	Red	Normal
85	133	à	Alt 133	Note 6	Black	Magenta	Normal
86	134	â	Alt 134	Note 6	Black	Brown	Normal
87	135	ç	Alt 135	Note 6	Black	Light Grey	Normal
88	136	ê	Alt 136	Note 6	Black	Dark Grey	Non-Display
89	137	ë	Alt 137	Note 6	Black	Light Blue	High Intensity Underline
8A	138	è	Alt 138	Note 6	Black	Light Green	High Intensity
8B	139	ï	Alt 139	Note 6	Black	Light Cyan	High Intensity
8C	140	î	Alt 140	Note 6	Black	Light Red	High Intensity
8D	141	ì	Alt 141	Note 6	Black	Light Magenta	High Intensity
8E	142	Ä	Alt 142	Note 6	Black	Yellow	High Intensity
8F	143	Å	Alt 143	Note 6	Black	White	High Intensity
90	144	É	Alt 144	Note 6	Blue	Black	Normal
91	145	æ	Alt 145	Note 6	Blue	Blue	Underline
92	146	Æ	Alt 146	Note 6	Blue	Green	Normal
93	147	ô	Alt 147	Note 6	Blue	Cyan	Normal
94	148	ö	Alt 148	Note 6	Blue	Red	Normal
95	149	ò	Alt 149	Note 6	Blue	Magenta	Normal
96	150	û	Alt 150	Note 6	Blue	Brown	Normal
97	151	ù	Alt 151	Note 6	Blue	Light Grey	Normal
98	152	ÿ	Alt 152	Note 6	Blue	Dark Grey	High Intensity
99	153	Ö	Alt 153	Note 6	Blue	Light Blue	High Intensity Underline
9A	154	Ü	Alt 154	Note 6	Blue	Light Green	High Intensity

Value		As Characters			As Text Attributes		
					Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground	
9B	155	¢	Alt 155	Note 6	Blue	Light Cyan	High Intensity
9C	156	£	Alt 156	Note 6	Blue	Light Red	High Intensity
9D	157	¥	Alt 157	Note 6	Blue	Light Magenta	High Intensity
9E	158	Pt	Alt 158	Note 6	Blue	Yellow	High Intensity
9F	159	<i>f</i>	Alt 159	Note 6	Blue	White	High Intensity
A0	160	á	Alt 160	Note 6	Green	Black	Normal
A1	161	í	Alt 161	Note 6	Green	Blue	Underline
A2	162	ó	Alt 162	Note 6	Green	Green	Normal
A3	163	ú	Alt 163	Note 6	Green	Cyan	Normal
A4	164	ñ	Alt 164	Note 6	Green	Red	Normal
A5	165	Ñ	Alt 165	Note 6	Green	Magenta	Normal
A6	166	<u>a</u>	Alt 166	Note 6	Green	Brown	Normal
A7	167	<u>o</u>	Alt 167	Note 6	Green	Light Grey	Normal
A8	168	¿	Alt 168	Note 6	Green	Dark Grey	High Intensity
A9	169	┌	Alt 169	Note 6	Green	Light Blue	High Intensity Underline
AA	170	└	Alt 170	Note 6	Green	Light Green	High Intensity
AB	171	½	Alt 171	Note 6	Green	Light Cyan	High Intensity
AC	172	¼	Alt 172	Note 6	Green	Light Red	High Intensity
AD	173	i	Alt 173	Note 6	Green	Light Magenta	High Intensity
AE	174	<<	Alt 174	Note 6	Green	Yellow	High Intensity
AF	175	>>	Alt 175	Note 6	Green	White	High Intensity
B0	176	⋮	Alt 176	Note 6	Cyan	Black	Normal
B1	177	⋈	Alt 177	Note 6	Cyan	Blue	Underline
B2	178	⋉	Alt 178	Note 6	Cyan	Green	Normal
B3	179	⋊	Alt 179	Note 6	Cyan	Cyan	Normal
B4	180	⋋	Alt 180	Note 6	Cyan	Red	Normal
B5	181	⋌	Alt 181	Note 6	Cyan	Magenta	Normal
B6	182	⋍	Alt 182	Note 6	Cyan	Brown	Normal

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

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

Notes:

1. Asterisk (*) can be typed using two methods: press the (PrtSc/*) key or, in the shift mode, press the 8 key.
2. Period (.) can be typed using two methods: press the . key or, in the shift or Num Lock mode, press the Del key.
3. Numeric characters 0-9 can be typed using two methods: press the numeric keys on the top row of the keyboard or, in the shift or Num Lock mode, press the numeric keys in the keypad portion of the keyboard.
4. Uppercase alphabetic characters (A-Z) can be typed in two modes: the shift mode or the Caps Lock mode.
5. Lowercase alphabetic characters (a-z) can be typed in two modes: in the normal mode or in Caps Lock and shift mode combined.
6. The three digits after the Alt key must be typed from the numeric keypad. Character codes 001-255 may be entered in this fashion (with Caps Lock activated, character codes 97-122 will display uppercase).

Quick Reference

DECIMAL VALUE	➡	0	16	32	48	64	80	96	112
↙	HEXA-DECIMAL VALUE	0	1	2	3	4	5	6	7
0	0	BLANK (NULL)	▶	BLANK (SPACE)	0	@	P	'	p
1	1	😊	◀	!	1	A	Q	a	q
2	2	😁	↕		2	B	R	b	r
3	3	♥	!!	#	3	C	S	c	s
4	4	♦	¶	\$	4	D	T	d	t
5	5	♣	§	%	5	E	U	e	u
6	6	♠	▬	&	6	F	V	f	v
7	7	•	↕	'	7	G	W	g	w
8	8	●	↑	(8	H	X	h	x
9	9	○	↓)	9	I	Y	i	y
10	A	◉	→	*	:	J	Z	j	z
11	B	♂	←	+	;	K	[k	{
12	C	♀	└	,	<	L	\	l	
13	D	🎵	↔	—	=	M		m	}
14	E	🎶	▲	.	>	N	^	n	~
15	F	☀	▼	/	?	O	_	o	△

DECIMAL VALUE	➡	128	144	160	176	192	208	224	240
↙	HEXA-DECIMAL VALUE	8	9	A	B	C	D	E	F
0	0	Ç	É	á	⋮			∞	≡
1	1	ü	æ	í	⋮			β	±
2	2	é	Æ	ó	⋮			Γ	≥
3	3	â	ô	ú				π	≤
4	4	ä	ö	ñ				Σ	∫
5	5	à	ò	Ñ				σ	∫
6	6	â	û	à				μ	÷
7	7	ç	ù	ó				γ	≈
8	8	ê	ÿ	¿				Φ	°
9	9	ë	Ö	┐				Θ	•
10	A	è	Ü	┐				Ω	•
11	B	ï	¢	½				δ	√
12	C	î	£	¼				∞	n
13	D	ì	¥	¡				φ	²
14	E	Ä	℞	«				€	■
15	F	Å	f	»				∩	BLANK 'FF'

Notes:



Glossary

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

μ . Prefix micro; 0.000 001.

μ s. Microsecond; 0.000 001 second.

A. Ampere.

ac. Alternating current.

accumulator. A register in which the result of an operation is formed.

active high. Designates a signal that has to go high to produce an effect. Synonymous with positive true.

active low. Designates a signal that has to go low to produce an effect. Synonymous with negative true.

adapter. An auxiliary device or unit used to extend the operation of another system.

address bus. One or more conductors used to carry the binary-coded address from the processor throughout the rest of the system.

algorithm. A finite set of well-defined rules for the solution of a problem in a finite number of steps.

all points addressable (APA). A mode in which all points of a displayable image can be controlled by the user.

alphameric. Synonym for alphanumeric.

alphanumeric (A/N). Pertaining to a character set that contains letters, digits, and usually other characters, such as punctuation marks. Synonymous with alphameric.

alternating current (ac). A current that periodically reverses its direction of flow.

American National Standard Code for Information Interchange (ASCII). The standard code, using a coded character set consisting of 7-bit coded characters (8 bits including parity check), used for information exchange between data processing systems, data communication systems, and associated equipment. The ASCII set consists of control characters and graphic characters.

ampere (A). The basic unit of electric current.

A/N. Alphanumeric

analog. (1) Pertaining to data in the form of continuously variable physical quantities. (2) Contrast with digital.

AND. A logic operator having the property that if P is a statement, Q is a statement, R is a statement,..., then the AND of P, Q, R,...is true if all statements are true, false if any statement is false.

AND gate. A logic gate in which the output is 1 only if all inputs are 1.

AND operation. The boolean operation whose result has the boolean value 1, if and only if, each operand has the boolean value 1. Synonymous with conjunction.

APA. All points addressable.

ASCII. American National Standard Code for Information Interchange.

assemble. To translate a program expressed in an assembler language into a computer language.

assembler. A computer program used to assemble.

assembler language. A computer-oriented language whose instructions are usually in one-to-one correspondence with computer instructions.

asynchronous transmission. (1) Transmission in which the time of occurrence of the start of each character, or block of characters, is arbitrary; once started, the time of occurrence of each signal representing a bit within a character, or block, has the same relationship to significant instants of a fixed time frame. (2) Transmission in which each information character is individually transmitted (usually timed by the use of start elements and stop elements).

audio frequencies. Frequencies that can be heard by the human ear (approximately 15 hertz to 20,000 hertz).

auxiliary storage. (1) A storage device that is not main storage. (2) Data storage other than main storage; for example, storage on magnetic disk. (3) Contrast with main storage.

BASIC. Beginner's all-purpose symbolic instruction code.

basic input/output system (BIOS). The feature of the IBM Personal Computer that provides the level control of the major I/O devices, and relieves the programmer from concern about hardware device characteristics.

baud. (1) A unit of signaling speed equal to the number of discrete conditions or signal events per second. For example, one baud equals one bit per second in a train of binary signals, one-half dot cycle per second in Morse code, and one 3-bit value per second in a train of signals each of which can assume one of eight different states. (2) In asynchronous transmission, the unit of modulation rate corresponding to one unit of interval per second; that is, if the duration of the unit interval is 20 milliseconds, the modulation rate is 50 baud.

BCC. Block-check character.

beginner's all-purpose symbolic instruction code (BASIC). A programming language with a small repertoire of commands and a simple syntax, primarily designed for numeric applications.

binary. (1) Pertaining to a selection, choice, or condition that has two possible values or states. (2) Pertaining to a fixed radix numeration system having a radix of 2.

binary digit. (1) In binary notation, either of the characters 0 or 1. (2) Synonymous with bit.

binary notation. Any notation that uses two different characters, usually the binary digits 0 and 1.

binary synchronous communications (BSC). A uniform procedure, using a standardized set of control characters and control character sequences for synchronous transmission of binary-coded data between stations.

BIOS. Basic input/output system.

bit. Synonym for binary digit

bits per second (bps). A unit of measurement representing the number of discrete binary digits transmitted by a device in one second.

block. (1) A string of records, a string of words, or a character string formed for technical or logic reasons to be treated as an entity. (2) A set of things, such as words, characters, or digits, treated as a unit.

block-check character (BCC). In cyclic redundancy checking, a character that is transmitted by the sender after each message block and is compared with a block-check character computed by the receiver to determine if the transmission was successful.

boolean operation. (1) Any operation in which each of the operands and the result take one of two values. (2) An operation that follows the rules of boolean algebra.

bootstrap. A technique or device designed to bring itself into a desired state by means of its own action; for example, a machine routine whose first few instructions are sufficient to bring the rest of itself into the computer from an input device.

bps. Bits per second.

BSC. Binary synchronous communications.

buffer. (1) An area of storage that is temporarily reserved for use in performing an input/output operation, into which data is read or from which data is written. Synonymous with I/O area. (2) A portion of storage for temporarily holding input or output data.

bus. One or more conductors used for transmitting signals or power.

byte. (1) A sequence of eight adjacent binary digits that are operated upon as a unit. (2) A binary character operated upon as a unit. (3) The representation of a character.

C. Celsius.

capacitor. An electronic circuit component that stores an electric charge.

Cartesian coordinates. A system of coordinates for locating a point on a plane by its distance from each of two intersecting lines, or in space by its distance from each of three mutually perpendicular planes.

CAS. Column address strobe.

cathode ray tube (CRT). A vacuum tube in which a stream of electrons is projected onto a fluorescent screen producing a luminous spot. The location of the spot can be controlled.

cathode ray tube display (CRT display). (1) A CRT used for displaying data. For example, the electron beam can be controlled to form alphanumeric data by use of a dot matrix. (2) Synonymous with monitor.

CCITT. International Telegraph and Telephone Consultative Committee.

Celsius (C). A temperature scale. Contrast with Fahrenheit (F).

central processing unit (CPU). Term for processing unit.

channel. A path along which signals can be sent; for example, data channel, output channel.

character generator. (1) In computer graphics, a functional unit that converts the coded representation of a graphic character into the shape of the character for display. (2) In word processing, the means within equipment for generating visual characters or symbols from coded data.

character set. (1) A finite set of different characters upon which agreement has been reached and that is considered complete for some purpose. (2) A set of unique representations called characters. (3) A defined collection of characters.

characters per second (cps). A standard unit of measurement for the speed at which a printer prints.

check key. A group of characters, derived from and appended to a data item, that can be used to detect errors in the data item during processing.

clipping. In computer graphics, removing parts of a display image that lie outside a window.

closed circuit. A continuous unbroken circuit; that is, one in which current can flow. Contrast with open circuit.

CMOS. Complementary metal oxide semiconductor.

code. (1) A set of unambiguous rules specifying the manner in which data may be represented in a discrete form. Synonymous with coding scheme. (2) A set of items, such as abbreviations, representing the members of another set. (3) To represent data or a computer program in a symbolic form that can be accepted by a data processor. (4) Loosely, one or more computer programs, or part of a computer program.

coding scheme. Synonym for code.

collector. An element in a transistor toward which current flows.

color cone. An arrangement of the visible colors on the surface of a double-ended cone where lightness varies along the axis of

the cone, and hue varies around the circumference. Lightness includes both the intensity and saturation of color.

column address strobe (CAS). A signal that latches the column addresses in a memory chip.

compile. (1) To translate a computer program expressed in a problem-oriented language into a computer-oriented language. (2) To prepare a machine-language program from a computer program written in another programming language by making use of the overall logic structure of the program, or generating more than one computer instruction for each symbolic statement, or both, as well as performing the function of an assembler.

complement. A number that can be derived from a specified number by subtracting it from a second specified number.

complementary metal oxide semiconductor (CMOS). A logic circuit family that uses very little power. It works with a wide range of power supply voltages.

computer. A functional unit that can perform substantial computation, including numerous arithmetic operations or logic operations, without human intervention during a run.

computer instruction code. A code used to represent the instructions in an instruction set. Synonymous with machine code.

computer program. A sequence of instructions suitable for processing by a computer.

computer word. A word stored in one computer location and capable of being treated as a unit.

configuration. (1) The arrangement of a computer system or network as defined by the nature, number, and the chief characteristics of its functional units. More specifically, the term configuration may refer to a hardware configuration or a software configuration. (2) The devices and programs that make up a system, subsystem, or network.

conjunction. Synonym for AND operation.

contiguous. Touching or joining at the edge or boundary; adjacent.

control character. A character whose occurrence in a particular context initiates, modifies, or stops a control operation.

control operation. An action that affects the recording, processing, transmission, or interpretation of data; for example, starting or stopping a process, carriage return, font change, rewind, and end of transmission.

control storage. A portion of storage that contains microcode.

coordinate space. In computer graphics, a system of Cartesian coordinates in which an object is defined.

cps. Characters per second.

CPU. Central processing unit.

CRC. Cyclic redundancy check.

CRT. Cathode ray tube.

CRT display. Cathode ray tube display.

CTS. Clear to send. Associated with modem control.

cursor. (1) In computer graphics, a movable marker that is used to indicate position on a display. (2) A displayed symbol that acts as a marker to help the user locate a point in text, in a system command, or in storage. (3) A movable spot of light on the screen of a display device, usually indicating where the next character is to be entered, replaced, or deleted.

cyclic redundancy check (CRC). (1) A redundancy check in which the check key is generated by a cyclic algorithm. (2) A system of error checking performed at both the sending and receiving station after a block-check character has been accumulated.

cylinder. (1) The set of all tracks with the same nominal distance from the axis about which the disk rotates. (2) The

tracks of a disk storage device that can be accessed without repositioning the access mechanism.

daisy-chained cable. A type of cable that has two or more connectors attached in series.

data. (1) A representation of facts, concepts, or instructions in a formalized manner suitable for communication, interpretation, or processing by human or automatic means. (2) Any representations, such as characters or analog quantities, to which meaning is, or might be assigned.

data base. A collection of data that can be immediately accessed and operated upon by a data processing system for a specific purpose.

data processing system. A system that performs input, processing, storage, output, and control functions to accomplish a sequence of operations on data.

data transmission. Synonym for transmission.

dB. Decibel.

dBa. Adjusted decibels.

dc. Direct current.

debounce. (1) An electronic means of overcoming the make/break bounce of switches to obtain one smooth change of signal level. (2) The elimination of undesired signal variations caused by mechanically generated signals from contacts.

decibel. (1) A unit that expresses the ratio of two power levels on a logarithmic scale. (2) A unit for measuring relative power.

decoupling capacitor. A capacitor that provides a low impedance path to ground to prevent common coupling between circuits.

Deutsche Industrie Norm (DIN). (1) German Industrial Norm. (2) The committee that sets German dimension standards.

digit. (1) A graphic character that represents an integer; for example, one of the characters 0 to 9. (2) A symbol that

represents one of the non-negative integers smaller than the radix. For example, in decimal notation, a digit is one of the characters 0 to 9.

digital. (1) Pertaining to data in the form of digits.
(2) Contrast with analog.

DIN. Deutsche Industrie Norm.

DIN connector. One of the connectors specified by the DIN committee.

DIP. Dual in-line package.

DIP switch. One of a set of small switches mounted in a dual in-line package.

direct current (dc). A current that always flows in one direction.

direct memory access (DMA). A method of transferring data between main storage and I/O devices that does not require processor intervention.

disable. To stop the operation of a circuit or device.

disabled. Pertaining to a state of a processing unit that prevents the occurrence of certain types of interruptions. Synonymous with masked.

disk. Loosely, a magnetic disk.

diskette. A thin, flexible magnetic disk and a semirigid protective jacket, in which the disk is permanently enclosed. Synonymous with flexible disk.

diskette drive. A device for storing data on and retrieving data from a diskette.

display. (1) A visual presentation of data. (2) A device for visual presentation of information on any temporary character imaging device. (3) To present data visually. (4) See cathode ray tube display.

display attribute. In computer graphics, a particular property that is assigned to all or part of a display; for example, low intensity, green color, blinking status.

display element. In computer graphics, a basic graphic element that can be used to construct a display image; for example, a dot, a line segment, a character.

display group. In computer graphics, a collection of display elements that can be manipulated as a unit and that can be further combined to form larger groups.

display image. In computer graphics, a collection of display elements or display groups that are represented together at any one time in a display space.

display space. In computer graphics, that portion of a display surface available for a display image. The display space may be all or part of a display surface.

display surface. In computer graphics, that medium on which display images may appear; for example, the entire screen of a cathode ray tube.

DMA. Direct memory access.

dot matrix. (1) In computer graphics, a two-dimensional pattern of dots used for constructing a display image. This type of matrix can be used to represent characters by dots. (2) In word processing, a pattern of dots used to form characters. This term normally refers to a small section of a set of addressable points; for example, a representation of characters by dots.

dot printer. Synonym for matrix printer.

dot-matrix character generator. In computer graphics, a character generator that generates character images composed of dots.

drawing primitive. A group of commands that draw defined geometric shapes.

DSR. Data set ready. Associated with modem control.

DTR. In the IBM Personal Computer, data terminal ready. Associated with modem control.

dual in-line package (DIP). A widely used container for an integrated circuit. DIPs have pins in two parallel rows. The pins are spaced 1/10 inch apart. See also DIP switch.

duplex. (1) In data communication, pertaining to a simultaneous two-way independent transmission in both directions.
(2) Contrast with half-duplex.

duty cycle. In the operation of a device, the ratio of on time to idle time. Duty cycle is expressed as a decimal or percentage.

dynamic memory. RAM using transistors and capacitors as the memory elements. This memory requires a refresh (recharge) cycle every few milliseconds. Contrast with static memory.

EBCDIC. Extended binary-coded decimal interchange code.

ECC. Error checking and correction.

edge connector. A terminal block with a number of contacts attached to the edge of a printed-circuit board to facilitate plugging into a foundation circuit.

EIA. Electronic Industries Association.

electromagnet. Any device that exhibits magnetism only while an electric current flows through it.

enable. To initiate the operation of a circuit or device.

end of block (EOB). A code that marks the end of a block of data.

end of file (EOF). An internal label, immediately following the last record of a file, signaling the end of that file. It may include control totals for comparison with counts accumulated during processing.

end-of-text (ETX). A transmission control character used to terminate text.

end-of-transmission (EOT). A transmission control character used to indicate the conclusion of a transmission, which may have included one or more texts and any associated message headings.

end-of-transmission-block (ETB). A transmission control character used to indicate the end of a transmission block of data when data is divided into such blocks for transmission purposes.

EOB. End of block.

EOF. End of file.

EOT. End-of-transmission.

EPROM. Erasable programmable read-only memory.

erasable programmable read-only memory (EPROM). A PROM in which the user can erase old information and enter new information.

error checking and correction (ECC). The detection and correction of all single-bit errors, plus the detection of double-bit and some multiple-bit errors.

ESC. The escape character.

escape character (ESC). A code extension character used, in some cases, with one or more succeeding characters to indicate by some convention or agreement that the coded representations following the character or the group of characters are to be interpreted according to a different code or according to a different coded character set.

ETB. End-of-transmission-block.

ETX. End-of-text.

extended binary-coded decimal interchange code (EBCDIC). A set of 256 characters, each represented by eight bits.

F. Fahrenheit.

Fahrenheit (F). A temperature scale. Contrast with Celsius (C).

falling edge. Synonym for negative-going edge.

FCC. Federal Communications Commission.

fetch. To locate and load a quantity of data from storage.

FF. The form feed character.

field. (1) In a record, a specified area used for a particular category of data. (2) In a data base, the smallest unit of data that can be referred to.

field-programmable logic sequencer (FPLS). An integrated circuit containing a programmable, read-only memory that responds to external inputs and feedback of its own outputs.

FIFO (first-in-first out). A queuing technique in which the next item to be retrieved is the item that has been in the queue for the longest time.

fixed disk drive. In the IBM Personal Computer, a unit consisting of nonremovable magnetic disks, and a device for storing data on and retrieving data from the disks.

flag. (1) Any of various types of indicators used for identification. (2) A character that signals the occurrence of some condition, such as the end of a word. (3) Deprecated term for mark.

flexible disk. Synonym for diskette.

flip-flop. A circuit or device containing active elements, capable of assuming either one of two stable states at a given time.

font. A family or assortment of characters of a given size and style; for example, 10 point Press Roman medium.

foreground. (1) In multiprogramming, the environment in which high-priority programs are executed. (2) On a color display screen, the characters as opposed to the background.

form feed. (1) Paper movement used to bring an assigned part of a form to the printing position. (2) In word processing, a

function that advances the typing position to the same character position on a predetermined line of the next form or page.

form feed character. A control character that causes the print or display position to move to the next predetermined first line on the next form, the next page, or the equivalent.

format. The arrangement or layout of data on a data medium.

FPLS. Field-programmable logic sequencer.

frame. (1) In SDLC, the vehicle for every command, every response, and all information that is transmitted using SDLC procedures. Each frame begins and ends with a flag. (2) In data transmission, the sequence of contiguous bits bracketed by and including beginning and ending flag sequences.

g. Gram.

G. (1) Prefix giga; 1,000,000,000. (2) When referring to computer storage capacity, 1,073,741,824. ($1,073,741,824 = 2$ to the 30th power.)

gate. (1) A combinational logic circuit having one output channel and one or more input channels, such that the output channel state is completely determined by the input channel states. (2) A signal that enables the passage of other signals through a circuit.

Gb. 1,073,741,824 bytes.

general-purpose register. A register, usually explicitly addressable within a set of registers, that can be used for different purposes; for example, as an accumulator, as an index register, or as a special handler of data.

giga (G). Prefix 1,000,000,000.

gram (g). A unit of weight (equivalent to 0.035 ounces).

graphic. A symbol produced by a process such as handwriting, drawing, or printing.

graphic character. A character, other than a control character, that is normally represented by a graphic.

half-duplex. (1) In data communication, pertaining to an alternate, one way at a time, independent transmission. (2) Contrast with duplex.

hardware. (1) Physical equipment used in data processing, as opposed to programs, procedures, rules, and associated documentation. (2) Contrast with software.

head. A device that reads, writes, or erases data on a storage medium; for example, a small electromagnet used to read, write, or erase data on a magnetic disk.

hertz (Hz). A unit of frequency equal to one cycle per second.

hex. Common abbreviation for hexadecimal. Also, hexadecimal can be noted as X' '.

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

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

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

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

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

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

Hz. Hertz

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

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

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

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

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

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

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

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

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

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

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

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

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

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

I/O. Input/output.

I/O area. Synonym for buffer.

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

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

k. Prefix kilo; 1000.

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

KB. 1024 bytes.

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

kg. Kilogram; 1000 grams.

kHz. Kilohertz; 1000 hertz.

kilo (k). Prefix 1000

kilogram (kg). 1000 grams.

kilohertz (kHz). 1000 hertz

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

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

LED. Light-emitting diode.

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

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

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

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

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

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

LUT. Look-up table.

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

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

mA. Milliampere; 0.001 ampere.

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

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

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

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

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

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

masked. Synonym for disabled.

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

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

MB. 1,048,576 bytes.

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

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

memory. Term for main storage.

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

MFM. Modified frequency modulation.

MHz. Megahertz; 1,000,000 hertz.

micro (μ). Prefix 0.000,001.

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

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

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

microsecond (μ s). 0.000,001 second.

milli (m). Prefix 0.001.

milliampere (mA). 0.001 ampere.

millisecond (ms). 0.001 second.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

ms. Millisecond; 0.001 second.

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

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

n. Prefix nano; 0.000,000,001.

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

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

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

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

negative true. Synonym for active low.

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

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

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

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

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

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

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

ns. Nanosecond; 0.000,000,001 second.

NUL. The null character.

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

odd-even check. Synonym for parity check.

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

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

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

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

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

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

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

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

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

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

overcurrent. A current of higher than specified strength.

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

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

overvoltage. A voltage of higher than specified value.

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

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

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

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

PEL. Picture element.

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

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

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

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

port. An access point for data entry or exit.

positive true. Synonym for active high.

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

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

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

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

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

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

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

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

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

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

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

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

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

PROM. Programmable read-only memory.

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

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

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

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

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

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

RAM. Random access memory. Read/write memory.

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

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

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

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

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

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

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

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

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

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

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

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

RF. Radio frequency.

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

RGBI. Red-green-blue-intensity.

rising edge. Synonym for positive-going edge.

ROM. Read-only memory.

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

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

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

RTS. Request to send. Associated with modem control.

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

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

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

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

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

SDL. Shielded Data Link

SDLC. Synchronous Data Link Control.

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

SERDES. Serializer/deserializer.

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

serializer/deserializer (SERDES). A device that serializes output from, and deserializes input to, a business machine.

setup. (1) In a computer that consists of an assembly of individual computing units, the arrangement of interconnections between the units, and the adjustments needed for the computer to operate. (2) The preparation of a computing system to perform a job or job step. Setup is usually performed by an operator and often involves performing routine functions, such as mounting tape reels. (3) The preparation of the system for normal operation.

short circuit. A low-resistance path through which current flows, rather than through a component or circuit.

signal. A variation of a physical quantity, used to convey data.

sink. A device or circuit into which current drains.

software. (1) Computer programs, procedures, and rules concerned with the operation of a data processing system. (2) Contrast with hardware.

source. The origin of a signal or electrical energy.

square wave. An alternating or pulsating current or voltage whose waveshape is square.

square wave generator. A signal generator delivering an output signal having a square waveform.

SS. Start-stop.

start bit. (1) A signal to a receiving mechanism to get ready to receive data or perform a function. (2) In a start-stop system, a signal preceding a character or block that prepares the receiving device for the reception of the code elements.

start-of-text (STX). A transmission control character that precedes a text and may be used to terminate the message heading.

start-stop system. A data transmission system in which each character is preceded by a start bit and is followed by a stop bit.

start-stop (SS) transmission. (1) Asynchronous transmission such that a group of signals representing a character is preceded by a start bit and followed by a stop bit. (2) Asynchronous transmission in which a group of bits is preceded by a start bit that prepares the receiving mechanism for the reception and registration of a character and is followed by at least one stop bit that enables the receiving mechanism to come to an idle condition pending the reception of the next character.

static memory. RAM using flip-flops as the memory elements. Data is retained as long as power is applied to the flip-flops. Contrast with dynamic memory.

stop bit. (1) A signal to a receiving mechanism to wait for the next signal. (2) In a start-stop system, a signal following a character or block that prepares the receiving device for the reception of a subsequent character or block.

storage. (1) A storage device. (2) A device, or part of a device, that can retain data. (3) The retention of data in a storage device. (4) The placement of data into a storage device.

strobe. An instrument that emits adjustable-rate flashes of light. Used to measure the speed of rotating or vibrating objects.

STX. Start-of-text.

symbol. (1) A conventional representation of a concept. (2) A representation of something by reason of relationship, association, or convention.

synchronization. The process of adjusting the corresponding significant instants of two signals to obtain the desired phase relationship between these instants.

Synchronous Data Link Control (SDLC). A protocol for management of data transfer over a data link.

synchronous transmission. (1) Data transmission in which the time of occurrence of each signal representing a bit is related to a fixed time frame. (2) Data transmission in which the sending and receiving devices are operating continuously at substantially the same frequency and are maintained, by means of correction, in a desired phase relationship.

syntax. (1) The relationship among characters or groups of characters, independent of their meanings or the manner of their interpretation and use. (2) The structure of expressions in a language. (3) The rules governing the structure of a language. (4) The relationships among symbols.

text. In ASCII and data communication, a sequence of characters treated as an entity if preceded and terminated by one STX and one ETX transmission control character, respectively.

time-out. (1) A parameter related to an enforced event designed to occur at the conclusion of a predetermined elapsed time. A time-out condition can be cancelled by the receipt of an appropriate time-out cancellation signal. (2) A time interval allotted for certain operations to occur; for example, response to polling or addressing before system operation is interrupted and must be restarted.

track. (1) The path or one of the set of paths, parallel to the reference edge on a data medium, associated with a single reading or writing component as the data medium moves past the component. (2) The portion of a moving data medium such as a drum, or disk, that is accessible to a given reading head position.

transistor-transistor logic (TTL). A popular logic circuit family that uses multiple-emitter transistors.

translate. To transform data from one language to another.

transmission. (1) The sending of data from one place for reception elsewhere. (2) In ASCII and data communication, a series of characters including headings and text. (3) The dispatching of a signal, message, or other form of intelligence by wire, radio, telephone, or other means. (4) One or more blocks or messages. For BSC and start-stop devices, a transmission is terminated by an EOT character. (5) Synonymous with data transmission.

TTL. Transistor-transistor logic.

typematic key. A keyboard key that repeats its function when held pressed.

V. Volt.

vector. In computer graphics, a directed line segment.

video. Computer data or graphics displayed on a cathode ray tube, monitor, or display.

view point. In computer graphics, the origin from which angles and scales are used to map virtual space into display space.

viewing reference point. In computer graphics, a point in the modeling coordinate space that is a defined distance from the view point.

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

viewplane. The visible plane of a CRT display screen that completely contains a defined window.

viewport. In computer graphics, a predefined part of the CRT display space.

volt. The basic practical unit of electric pressure. The potential that causes electrons to flow through a circuit.

W. Watt.

watt. The practical unit of electric power.

window. (1) A predefined part of the virtual space. (2) The visible area of a viewplane.

word. (1) A character string or a bit string considered as an entity. (2) See computer word.

write. To make a permanent or transient recording of data in a storage device or on a data medium.

write precompensation. The varying of the timing of the head current from the outer tracks to the inner tracks of the diskette to keep a constant 'write' signal.

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

Notes:



Bibliography

Intel Corporation. *The 8086 Family User's Manual*. This manual introduces the 8086 family of microcomputing components and serves as a reference in system design and implementation.

Intel Corporation. *8086/8087/8088 Macro Assembly Reference Manual for 8088/8085 Based Development System*. This manual describes the 8086/8087/8088 Macro Assembly Language and is intended for persons who are familiar with assembly language.

Intel Corporation. *Component Data Catalog*. This book describes Intel components and their technical specifications.

Motorola, Inc. *The Complete Microcomputer Data Library*. This book describes Motorola components and their technical specifications.

National Semiconductor Corporation. *250 Asynchronous Communications Element*. This book documents physical and operating characteristics of the INS 8250.

Notes:



Index

A

- AAA 6-10, 6-11
- AAD 6-13
- AAM 6-12
- AAS 6-12
- adapter card with ROM 5-10
- ADC 6-10
- ADD 6-10
- address
 - bits 0 to 19
 - (A0–A19) 1-20
 - enable (AEN), I/O channel 1-20
 - latch enable (ALE), I/O channel 1-20
 - map, I/O channel 1-25
 - map, I/O planar 1-24
- AEN (address enable) 1-20
- ALE (address latch enable), I/O channel 1-20
- alternate key 4-41
- AND 6-14
- arithmetic instructions 6-10, 6-26
- ASCII characters 7-3
- ASCII, extended 4-34

B

- bandwidth formula 1-14
 - specifications I/O channel 1-15
- BASIC
 - DEF SEG 5-8
 - reserved interrupt 5-7, 5-8
- basic assurance test 4-25
- BASIC reserved interrupts 5-7
- BAT (basic assurance test) 4-25
- BAT Completion Code command 4-26
- BAT Failure Code command 4-26
- binary integers
 - (coprocessor) 2-3, 2-4
- BIOS
 - parameter passing 5-4
 - quick reference 5-11, 5-111
 - software interrupt 5-5
 - system ROM 5-11, 5-111
 - use of 5-3
- bit map, I/O 8255A 1-27
- block diagram
 - system timer 1-10
- block diagram (coprocessor) 2-6
- break 4-11
- break code 4-24
- break key 4-42
- buffer, keyboard 4-24

C

cabling 4-23
CALL 6-16
caps lock key 4-10, 4-41
card specifications 1-31
CBW 6-13
CH CK, negative (-channel check), I/O channel 1-22
channel check, negative (-CH CK), I/O channel 1-22
channel, I/O
 pin assignments 1-17
character codes 4-6, 4-34
character codes
 (keyboard) 4-8
characters 7-3
CLC 6-23
CLD 6-23
CLI 6-23
CLK, I/O channel 1-21
clock (CLK), I/O
 channel 1-21
clock and data signals 4-32
 data output 4-33
 data stream 4-33
CMC 6-23
CMP 6-12
CMPS 6-15
codes
 character 4-34
 extended 4-38
commands from the system
 Reset 4-26
commands to the system
 BAT (basic assurance test)
 Completion Code 4-26
 BAT Failure 4-26
 Key Detection Error 4-27
 Overrun 4-27
comparison instructions 6-25

component diagram, system
 board 1-19
connector specifications 4-19
connectors
 J-1 through J-8 1-16
 keyboard 1-33
 power supply 1-32
 speaker 1-33
 system board 1-32
connectors (power supply) 3-6, 3-9
constants instructions 6-28
control key 4-40
control transfer
 instructions 6-16
Ctrl state 4-38
CWD 6-13

D

DAS 6-12
data
 bits 0 to 7 (D0-D7) 1-21
 flow, system board
 diagram 1-6
 data output 4-33
 data stream 4-33
 data transfer
 instructions 6-7, 6-24
DEC 6-11
decimal integers
 (coprocessor) 2-3, 2-4
delay, typematic 4-24
description 4-22
 buffer 4-24
 cabling 4-23
 key-code scanning 4-23
 keys 4-24
 sequencing key-code
 scanning 4-23
description I/O channel 1-20

diagram, system board 1-19
diagrams
 logic, 101/102-key
 keyboard 4-52
 logic, 83-key
 keyboard 4-21
 logics, 256/640K 1-46
 logics, 64/256K 1-34
DIV 6-12
DMA request 1 to 3
 (DRQ1-DRQ3) 1-21
DOS
 keyboard function 5-7

E

encoding, keyboard 4-33
ESC 6-23
extended ASCII 4-6, 4-34
extended codes 4-9, 4-38

F

FABS 6-28
FADD 6-26
FCHS 6-28
FCLEX 6-30
FCOM 6-25
FCOMP 6-26
FCOMPP 6-26
FDECSTP 6-30
FDISI 6-29
FDIV 6-27
FENI 6-29
FFREE 6-30
FIFO 4-24
FINCSTP 6-30
FINIT 6-29
FLD 6-24

FLDCW 6-29
FLDENV 6-30
FLDLG2 6-29
FLDLN2 6-29
FLDL2T 6-29
FLDP1 6-29
FLDZ 6-28
FLD1 6-28
FMUL 6-27
FNOP 6-30
FPATAN 6-28
FPREM 6-27
FPTAN 6-28
French keyboard 4-13, 4-45
FRNDINT 6-27
FRSTOR 6-30
FSAVE 6-30
FSCALE 6-27
FSQRT 6-27
FST 6-25
FSTCW 6-29
FSTENV 6-30
FSTP 6-25
FSTSW 6-29
FSUB 6-26
FTST 6-26
FWAIT 6-30
FXAM 6-26
FXCH 6-25
FXTRACT 6-27
FYL2X 6-28
FYL2XP1 6-28
F2XM1 6-28

G

generator, refresh
 request 1-10
German keyboard 4-14, 4-46

H

HLT 6-23

I

I/O channel

- address map,
 - channel 1-25
- address map, planar 1-24
- ALE (address latch enable) 1-20
- bit map 8255A 1-27
- CH CK (-I/O Channel Check) 1-22
- CH RDY (I/O Channel Ready), I/O channel 1-22
- check (-CH CK) 1-22
- CLK 1-21
- description 1-20
- I/O channel 1-15
- oscillator (OSC) 1-23
- pin assignments 1-17
- read command (-IOR) 1-22
- Reset Drive (RESET DRV) 1-23
- Terminal Count (T/C) 1-23
- Write Command (-IOW) 1-22

I/O channel connectors 1-17

IDIV 6-12

IMUL 6-12

IN 6-8

INC 6-10

instructions

arithmetic 6-10, 6-26

comparison 6-25

constants 6-28

control transfer 6-16

data transfer 6-7, 6-24

logic 6-13

rotate 6-13

shift 6-13

string manipulation 6-15

INT 6-22

Intel 8048 4-3

Intel 8088 microprocessor,

- arithmetic 6-8, 6-19

- comparison 6-19

- conditional transfer operations 6-15

- constants 6-21

- control transfer 6-12

- data transfer 6-6, 6-17

- instruction set index 6-27

- instruction set

- matrix 6-25

- instruction set

- extensions 6-17

- logic 6-10

- memory segmentation

- model 6-5

- operand summary 6-4

- processor control 6-16, 6-22

- register model 6-3

- second instruction byte summary 6-4

- string manipulation 6-11

- transcendental 6-21

- use of segment

- override 6-5

interrupt request 2 to 7 (IRQ2-IRQ7) 1-22

INTO 6-22

IRET 6-22

Italian keyboard 4-15, 4-47

J

JB/JNAE 6-17
 JBE/JNA 6-17
 JCXZ 6-19
 JE/JZ 6-17
 JL/JNGE 6-17
 JLE/JNG 6-17
 JMP 6-16
 JNB/JAE 6-18
 JNBE/JA 6-18
 JNE/JNZ 6-18
 JNL/JGE 6-18
 JNLE/JG 6-18
 JNO 6-18
 JNP/JPO 6-18
 JNS 6-19
 JO 6-18
 JP/JPE 6-18
 JS 6-18

K

key-code scanning 4-23
 Key Detection Error
 command 4-27
 keyboard 4-3
 connector 1-33, 4-19
 encoding 4-33
 interface 4-5
 layout 4-12, 4-35
 power-on self test 4-4
 routine 4-43
 keyboard buffer 4-24
 keyboard data output 4-33
 keyboard extended codes
 alt 4-10
 alternate 4-41
 break 4-11, 4-42
 caps lock 4-10, 4-41

combinations 4-41
 ctrl 4-9, 4-40
 number lock 4-41
 pause 4-11, 4-42
 print screen 4-11, 4-42
 scroll lock 4-10, 4-41
 shift 4-9, 4-40
 system request 4-42
 system reset 4-11

keyboard layouts

French 4-13, 4-45
 German 4-14, 4-46
 Italian 4-15, 4-47
 Spanish 4-16, 4-48
 UK English 4-17, 4-49
 US English 4-18, 4-50

keyboard scan 4-3

keyboard scan codes 4-6,
 4-28

keyboard, French 4-13, 4-45

keyboard, German 4-14,
 4-46

keyboard, Italian 4-15, 4-47

keyboard, Spanish 4-16, 4-48
 keyboard, UK English 4-17,
 4-49

keyboard, US English 4-18,
 4-50

keys 4-24

keys, typematic 4-4, 4-24

L

LAHF 6-9

layout, keyboard 4-35

layouts

French 4-13, 4-45
 German 4-14, 4-46
 Italian 4-15, 4-47
 Spanish 4-16, 4-48
 UK English 4-17, 4-49

US English 4-18, 4-50
LDS 6-9
LEA 6-9
LES 6-9
line protocol 4-25
LOCK 6-23
LODS 6-15
logic diagrams 4-52
logic diagrams, system board,
256/640K 1-46
logic diagrams, system board,
64/256K 1-34
logic instructions 6-13
LOOP 6-19
LOOPNZ/LOOPNE 6-19
LOOPZ/LOOPE 6-19

M

make code 4-4, 4-24
make/break 4-24
math coprocessor
binary integers 2-3, 2-4
block diagram 2-6
control word 2-5
decimal integers 2-3, 2-4
hardware interface 2-4
NMI 2-5
QS0 2-4
QS1 2-4
real numbers 2-3, 2-4
memory locations
reserved 5-8
memory map
BIOS 5-8
memory map, system 1-8
memory read command
(-MEMR) 1-23
memory write command
(-MEMW) 1-23
-MEMR (memory read
command) 1-23

-MEMW (memory write
command) 1-23
modules, RAM 1-12
modules,
ROM/EPROM 1-13
MOV 6-7
MOVS 6-15
MUL 6-12

N

NEG 6-11
NMI (coprocessor) 2-5
NOP 6-23
NOT 6-13
Num Lock key 4-9, 4-11
Num Lock state 4-38
number lock key 4-41

O

OR 6-14
OSC (oscillator), I/O
channel 1-23
oscillator (OSC), I/O
channel 1-23
OUT 6-8
output, keyboard 4-33
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

T

terminal count (T/C), I/O
channel 1-23
TEST 6-14
timer/counters 1-10
timers, system 1-10
tone generation,
speaker 1-10
typematic delay 4-24
typematic keys 4-4, 4-24
typematic rate 4-24

U

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

V

vectors with special
meanings 5-5

W

WAIT 6-23
write command (-IOW), I/O
channel 1-22
write memory command
(-MEMW) 1-23

X

XCHG 6-8
XLAT 6-9
XOR 6-15

Numerics

8088, (see also Intel 8088
microprocessor) 1-4
8254-2 1-10
8255A bit map 1-27





Reader's Comment Form

Technical Reference

6139821

Your comments assist us in improving the usefulness of our publication; they are an important part of the input used for revisions.

IBM may use and distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Please do not use this form for technical questions regarding the IBM Personal Computer or programs for the IBM Personal Computer, or for requests for additional publications; this only delays the response. Instead, direct your inquiries or request to your authorized IBM Personal Computer dealer.

Comments:



**NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES**

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 40 ARMONK, NEW YORK



POSTAGE WILL BE PAID BY ADDRESSEE

IBM PERSONAL COMPUTER
READER COMMENT DEPARTMENT
P.O. BOX 1328-C
BOCA RATON, FLORIDA 33429-9960



Fold here

Tape

Please do not staple

Tape