



*Personal Computer
Hardware Reference
Library*

Technical Reference



*Personal Computer
Hardware Reference
Library*

Technical Reference

Revised Edition (April 1984)

The following paragraph does not apply to the United Kingdom or any country where such provisions are inconsistent with local law: International Business Machines Corporation provides this manual "as is," without warranty of any kind, either expressed or implied, including, but not limited to the particular purpose. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this manual at any time.

This product could include technical inaccuracies or typographical errors. Changes are made periodically to the information herein; these changes will be incorporated in new editions of the publication.

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

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

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

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

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



Preface

This publication describes the various units of the IBM 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.

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 the IBM Personal Computer power supply.

Section 4, "Keyboard," discusses the hardware make up, function, and layout of the IBM Personal Computer keyboard.

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. In addition, keyboard encoding and usage is discussed.

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

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

Section 8, "Communications," describes communications hardware and discusses communications interface standards and the sequence of events to establish communications.

A glossary, bibliography, and index are also provided.

The 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 page length hard tabs with the above category descriptions, separate the groups of modules.

The term "*Technical Reference* manual" in the option and adapter manual, refers to the IBM Personal Computer *Technical Reference* system manual.

The term “*Guide to Operations* manual” in the option and adapter manual, refers to the IBM Personal Computer *Guide to Operations* manual.

Prerequisite Publications

- IBM Personal Computer *Guide to Operations*

Suggested Reading

- *BASIC for the IBM Personal Computer*
- *Disk Operating System (DOS), Version 1.1*
- *Disk Operating System (DOS), Version 2.1*
- IBM Personal Computer *Hardware Maintenance and Service*
- *MACRO Assembler for the IBM Personal Computer*

Contents

SECTION 1. SYSTEM BOARD	1-1
Description	1-3
Microprocessor	1-3
Data Flow Diagrams	1-5
System Memory Map	1-8
System Timers	1-11
System Interrupts	1-11
ROM	1-13
RAM	1-13
DMA	1-13
I/O Channel	1-14
System Board Diagram	1-16
I/O Channel Diagram	1-18
I/O Channel Description	1-20
I/O Address Map	1-24
Other Circuits	1-25
Speaker Circuit	1-25
Cassette Interface	1-27
Cassette Circuit Block Diagrams	1-28
8255A I/O Bit Map	1-31
System-Board Switch Settings	1-33
Specifications	1-34
Card Specifications	1-34
Logic Diagrams	1-36
SECTION 2. COPROCESSOR	2-1
Description	2-3
Programming Interface	2-3
Hardware Interface	2-4
SECTION 3. POWER SUPPLY	3-1
Description	3-3
Input Requirements	3-4
Outputs	3-4
Vdc Output	3-4
Vac Output	3-5
Overvoltage/Overcurrent Protection	3-5

Primary (Input)	3-5
Secondary (Output)	3-5
Power Good Signal	3-6
Power Supply Connectors and Pin Assignments	3-6
SECTION 4. KEYBOARD	4-1
Description	4-3
Block Diagram	4-4
Keyboard Diagrams	4-5
Connector Specifications	4-12
Keyboard Logic Diagram	4-13
SECTION 5. SYSTEM BIOS	5-1
System BIOS Usage	5-3
Keyboard Encoding and Usage	5-14
BIOS Cassette Logic	5-25
System BIOS Listing	5-29
Quick Reference	5-29
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
Use of Segment Override	6-5
Data Transfer	6-6
Arithmetic	6-8
Logic	6-10
String Manipulation	6-11
Control Transfer	6-12
8088 Conditional Transfer Operations	6-15
Processor Control	6-16
8087 Extensions to the 8088 Instruction Set	6-17
Data Transfer	6-17
Comparison	6-19
Arithmetic	6-19
Transcendental	6-21
Constants	6-21
Processor Control	6-22
8088 Instruction Set Matrix	6-25
Instruction Set Index	6-27

SECTION 7. CHARACTERS, KEYSTROKES, AND COLORS 7-1

SECTION 8. COMMUNICATIONS 8-1

 Communications 8-3

 Establishing a Communications Link 8-5

 Establishing Link on Nonswitched Point-to-Point Line 8-6

 Establishing Link on Nonswitched Multipoint Line ... 8-8

 Establishing Link on Switched Point-to-Point Line .. 8-10

Glossary **Glossary-1**

Bibliography **Bibliography-1**

Index **Index-1**



INDEX TAB LISTING

Section 1. System Board

Section 2. Coprocessor

Section 3. Power Supply

Section 4. Keyboard

Section 5. System BIOS

Section 6. Instruction Set

Section 1

Section 2

Section 3

Section 4

Section 5

Section 6



Section 7. Characters, Keystrokes, and Colors

Section 7

Section 8. Communications

Section 8

Glossary

Glossary

Bibliography

Bibliography

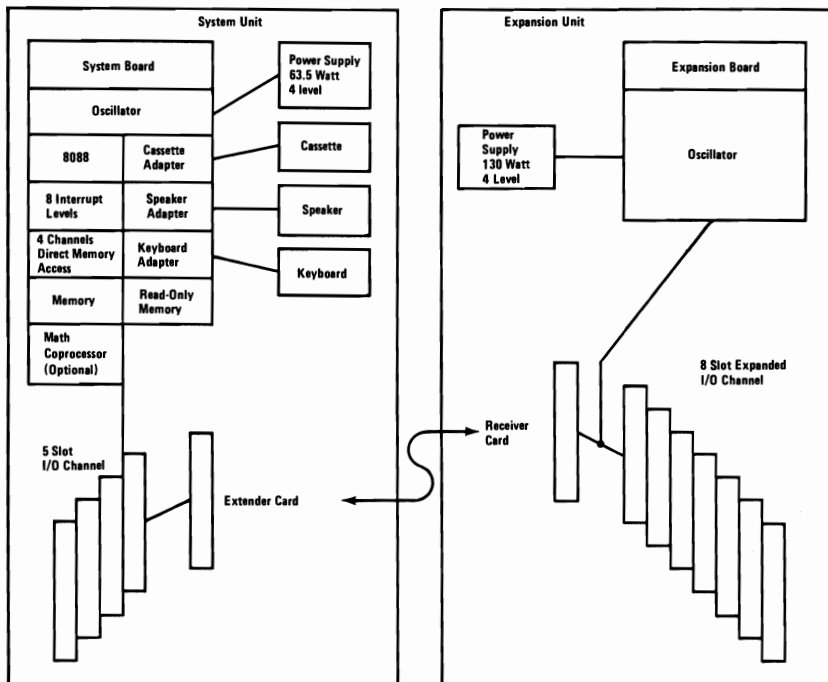
Index

Index



System Block Diagram

The following is a system block diagram of the IBM 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.



SECTION 1. SYSTEM BOARD

Contents

Description	1-3
Microprocessor	1-3
Data Flow Diagrams	1-5
System Memory Map	1-8
System Timers	1-11
System Interrupts	1-11
ROM	1-13
RAM	1-13
DMA	1-13
I/O Channel	1-14
System Board Diagram	1-16
I/O Channel Diagram	1-18
I/O Channel Description	1-20
I/O Address Map	1-24
Other Circuits	1-25
Speaker Circuit	1-25
Cassette Interface	1-27
Cassette Circuit Block Diagrams	1-28
8255A I/O Bit Map	1-31
System-Board Switch Settings	1-33

Specifications	1-34
Card Specifications	1-34
Logic Diagrams	1-36



Description

The system board fits horizontally in the base of the system unit and is approximately 215.5 mm (8-1/2 in.) x 304.8 mm (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 six-pin connectors. Other connectors on the board are for attaching the keyboard, audio cassette and speaker. Five 62-pin card edge-sockets are also mounted on the board. The I/O channel is bussed across these five I/O slots.

Two dual-in-line package (DIP) switches (two eight-switch packs) are mounted on the board and can be read under program control. The DIP switches provide the system software with information about the installed options, how much storage the system board has, what type of display adapter is installed, what operation 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 consists of five functional areas: the microprocessor subsystem and its support elements, the read-only memory (ROM) subsystem, the read/write (R/W) memory subsystem, integrated I/O adapters, and the I/O channel. The read/write memory is also referred to as random access memory (RAM). 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 supports 20 bits of addressing (1 megabyte of storage). It also operates in maximum mode, so a comicroprocessor can be added as a feature. The microprocessor operates at 4.77-MHz. This frequency, which is derived from a

14.31818-MHz crystal, is divided by 3 for the microprocessor clock, and by 4 to obtain the 3.58-MHz color burst signal required for color televisions.

At the 4.77-MHz clock rate, the 8088 bus cycles are four clocks of 210-ns, or 840-ns. I/O cycles take five 210-ns clocks or 1.05- μ s.

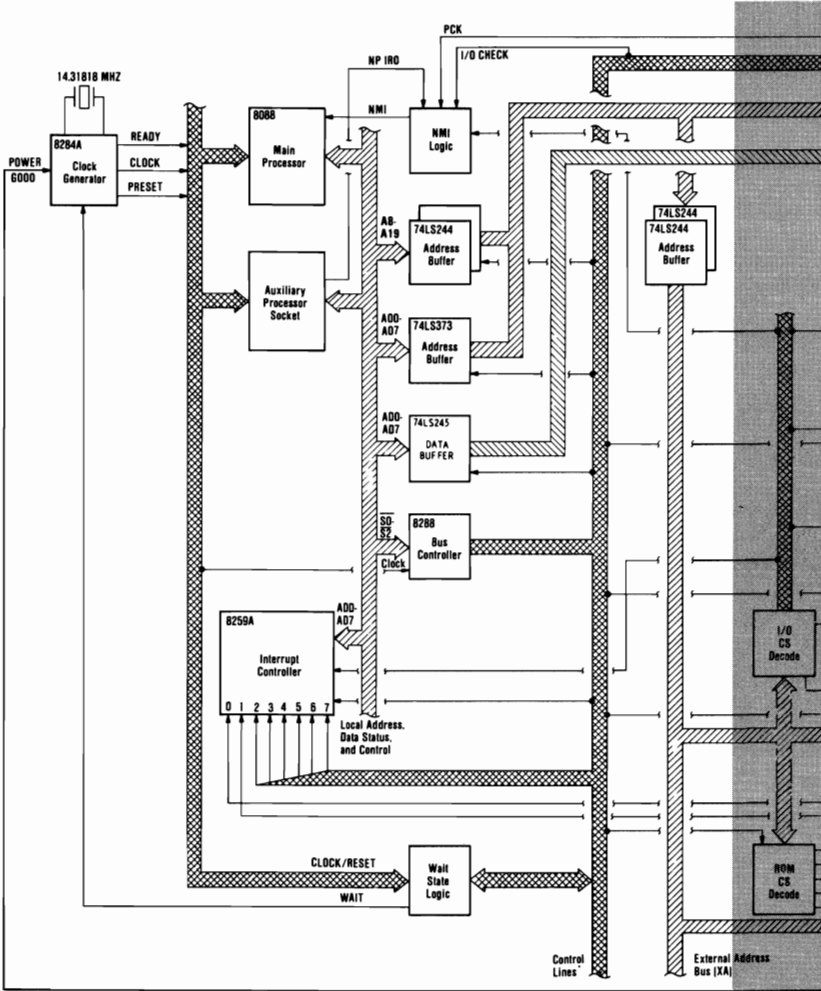
The system board contains circuits for attaching an audio cassette, the keyboard, and the speaker. The cassette adapter allows the attachment of any good quality audio cassette through the earphone output and either the microphone or auxiliary inputs. The system board has a jumper for either input. This interface also provides a cassette motor control for transport starting and stopping under program control. This interface reads and writes the audio cassette at a data rate of between 1,000 and 2,000 baud. The baud rate is variable and depend on data content, because a different bit-cell time is used for 0's and 1's. For diagnostic purposes, the tape interface can loop read to write for testing the system board's circuits. The ROM cassette software blocks cassette data and generates a cyclic redundancy check (CRC) to check this data.

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.

Both the keyboard and cassette interfaces on the system board are 5-pin DIN connectors that extends through the rear panel of the system unit.

Data Flow Diagrams

The following pages contain the system-board Data Flow Diagrams.



System Board Data Flow (1 of 2)

System Memory Map

The following pages contain the System Memory Map.



Start Address		Function
Decimal	Hex	
0	00000	64 to 256K Read/Write Memory on System Board
16K	04000	
32K	08000	
48K	0C000	
64K	10000	
80K	14000	
96K	18000	
112K	1C000	
128K	20000	
144K	24000	
160K	28000	
176K	2C000	
192K	30000	
208K	34000	
224K	38000	
240K	3C000	
256K	40000	Up to 384K Read/Write Memory in I/O Channel
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 for 64/256K System Board (Part 1 of 2)

Start Address		Function	
Decimal	Hex		
640K	A0000	128K Reserved	
656K	A4000		
672K	A8000		
688K	AC000		
704K	B0000	Monochrome	
720K	B4000		
736K	B8000	Color/Graphics	
752K	BC000		
768K	C0000	192K Read Only Memory Expansion and Control	
784K	C4000		
800K	C8000		Fixed Disk Control
816K	CC000		
832K	D0000		
848K	D4000		
864K	D8000		
880K	DC000		
896K	E0000		
912K	E4000		
928K	E8000		
944K	EC000		
960K	F0000	Reserved	
976K	F4000	48K Base System ROM	
992K	F8000		
1008K	FC000		

System Memory Map for 64/256K System Board (Part 2 of 2)

System Timers

Three programmable timer/counters are used by the system as follows: Channel 0 is a general-purpose timer providing a constant time base for implementing a time-of-day clock, Channel 1 times and requests refresh cycles from the Direct Memory Access (DMA) channel, and Channel 2 supports the tone generation for the speaker. Each channel has a minimum timing resolution of 1.05- μ s.

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 highest 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.

Number	Usage
NMI	Parity 8087
0	Timer
1	Keyboard
2	Reserved
3	Asynchronous Communications (Alternate)
	SDLC Communications
	BSC Communications
	Cluster (Primary)
4	Asynchronous Communications (Primary)
	SDLC Communications
	BSC Communications
5	Fixed Disk
6	Diskette
7	Printer
	Cluster (Alternate)

8088 Hardware Interrupt Listing

ROM

The system board supports both Read Only Memory (ROM) and Random Access Memory (RAM). It has space for up to 512K of ROM or Erasable Programmable ROM (EPROM). Six module sockets are provided, each of which can accept an 8K or 8 byte device. Five sockets are populated with 40K of ROM. This ROM contains the cassette BASIC interpreter, cassette operating system, power-on selftest, Input/Output (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 of 250-ns and a cycle time of 375-ns.

RAM

The RAM on the system board is as shown in the following chart.

System Board	Minimum Storage	Maximum Storage	Memory Modules	Soldered (Bank 0)	Pluggable (Bank 1-3)
64/256K	64K	256K	64K by 1 Bit	1 Bank of 9	3 Banks of 9

Memory greater than the system board's maximum is obtained by adding memory cards in the expansion slots. All memory is parity-checked and consists of dynamic 64K by 1 bit chips with an access time of 250-ns and a cycle time of 410-ns.

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 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 storage both on the system board and in the system expansion slots. All DMA data transfers, except the refresh channel, take five microprocessor clocks of 210-ns, or 1.05- μ s if the microprocessor ready line is not deactivated. Refresh DMA cycles take four clocks or 840-ns.

The three programmable timer/counter devices 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 1 is used to time and request refresh cycles from the DMA channel; and Channel 2 is used to support the tone generation for the speaker. Each channel has a minimum timing resolution of 1.05-us.

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 highest priority, is attached to Channel 0 of the timer/counter device 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.

I/O Channel

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

The I/O channel contains an 8-bit bidirectional data bus, 20 address lines, 6 levels of interrupt, control lines for memory and I/O read or write, clock and timing lines, 3 channels of DMA

control lines, memory refresh timing control lines, a channel-check line, and power and ground for the adapters. Four voltage levels are provided for I/O cards: +5 Vdc, -5 Vdc, +12 Vdc, and -12 Vdc. These functions are provided in a 62-pin connector with 100-mil card tab spacing.

A 'ready' line is available on the I/O channel to allow operation with slow I/O or memory devices. If the channel's ready line is not activated by an addressed device, all microprocessor-generated memory read and write cycles take four 210-ns clocks or 840-ns/byte. All microprocessor-generated I/O read and write cycles require five clocks for a cycle time of 1.05- μ s/byte. All DMA transfers require five clocks for a cycle time of 1.05- μ s/byte. Refresh cycles occur once every 72 clocks (approximately 15- μ s) and require four clocks or approximately 7% of the bus bandwidth.

I/O devices are addressed using I/O mapped address space. The channel is designed so that 512 I/O device addresses are available to the I/O channel cards.

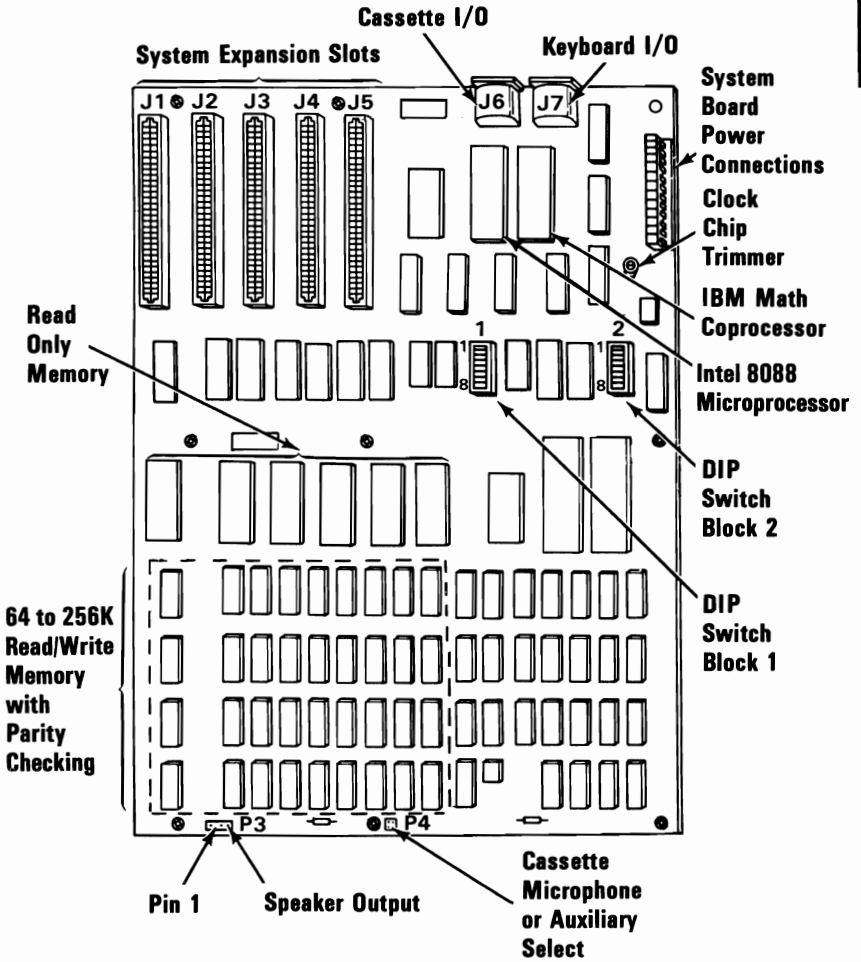
A 'channel check' line exists for reporting error conditions to the microprocessor. Activating this line results in a non-maskable interrupt (NMI) to the 8088 microprocessor. Memory expansion options use this line to report parity errors.

The I/O channel is repowered to provide sufficient drive to power all five system unit expansion slots, assuming two low-power Schottky loads per slot. The IBM I/O adapters typically use only one load.

System Board Diagram

The following shows the system board's component layout.



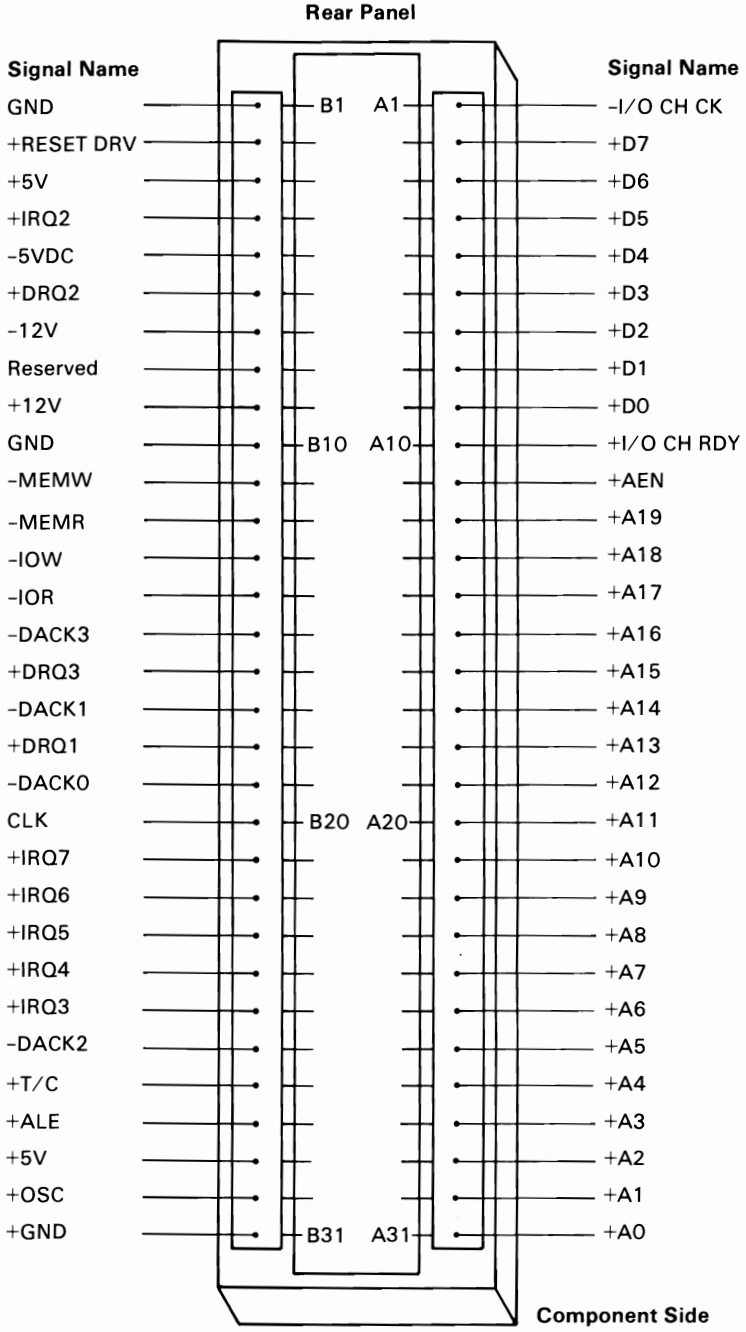


System Board Component Diagram

I/O Channel Diagram

The following page contains the I/O Channel Diagram. All lines are TTL-compatible.





I/O Channel Diagram

I/O Channel Description

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

Signal	I/O	Description
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.
CLK	O	System clock: It is a divide-by-three of the oscillator and has a period of 210-ns (4.77-MHz) 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 least significant bit (LSB) and D7 is the most significant bit (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 (210-ns).

- | | | |
|------------------|----------|---|
| -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 Command: 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 Command: 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 70-ns period (14.31818-MHz). 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.

Hex Range*	Usage
000-00F	DMA Chip 8237A-5
020-021	Interrupt 8259A
040-043	Timer 8253-5
060-063	PPI 8255A-5
080-083	DMA Page Registers
0AX**	NMI Mask Register
200-20F	Game Control
210-217	Expansion Unit
2F8-2FF	Asynchronous Communications (Secondary)
300-31F	Prototype Card
320-32F	Fixed Disk
378-37F	Printer
380-38C***	SDLC Communications
380-389***	Binary Synchronous Communications (Secondary)
390-393	Cluster
3A0-3A9	Binary Synchronous Communications (Primary)
3B0-3BF	IBM Monochrome Display/Printer
3D0-3DF	Color/Graphics
3F0-3F7	Diskette
3F8-3FF	Asynchronous Communications (Primary)
790-793	Cluster (Adapter 1)
B90-B93	Cluster (Adapter 2)
1390-1393	Cluster (Adapter 3)
2390-2393	Cluster (Adapter 4)

* 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)

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

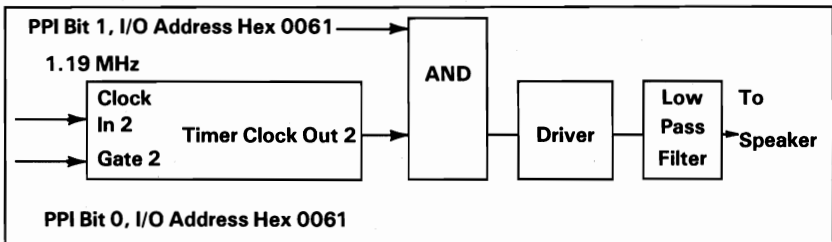
I/O Address Map

Other Circuits

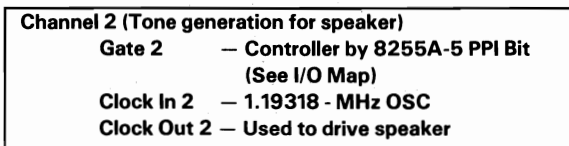
Speaker Circuit

The sound system has a small, permanent magnet, 57.15 mm (2-1/4 in.) speaker. The speaker can be driven from one or both of two sources:

- An 8255A-5 programmable peripheral interface (PPI) output bit. The address and bit are defined in the “I/O Address Map”.
- A timer clock channel, the output of which is programmable within the functions of the 8253-5 timer when using a 1.19-MHz clock input. The timer gate also is controlled by an 8255A-5 PPI output-port bit. Address and bit assignment are in the “I/O Address Map”.



Speaker Drive System Block Diagram



Speaker Tone Generation

The speaker connection is a 4-pin Berg connector. See "System Board Component Diagram," earlier in this section, for speaker connection or placement.

Pin	Function
1	Data
2	Key
3	Ground
4	+ 5 Volts

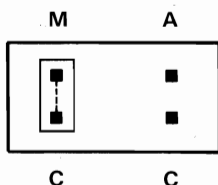
Speaker Connector

The speaker drive circuit is capable of about 1/2 watt of power. The control circuits allow the speaker to be driven three 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 device may be programmed to generate a waveform to the speaker; (3) the clock input to the timer/counter device can be modulated with a program-controlled I/O register bit. All three methods may be performed simultaneously.

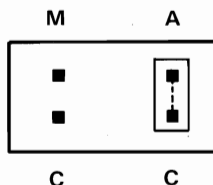
Cassette Interface

The cassette interface is controlled through software. An output from the 8253 timer controls the data to the cassette recorder through pin 5 of the cassette DIN connector at the rear of the system board. The cassette input data is read by an input port bit of the 8255A-5 PPI. This data is received through pin 4 of the cassette connector. Software algorithms are used to generate and read cassette data. The cassette drive motor is controlled through pins 1 and 3 of the cassette connector. The drive motor on/off switching is controlled by an 8255A-5 PPI output-port bit (hex 61, bit 3). The 8255A-5 address and bit assignments are defined in "I/O Address Map" earlier in this section.

A 2 by 2 Berg pin and a jumper are used on the cassette 'data out' line. The jumper allows use of the 'data out' line as a 0.075-Vdc microphone input when placed across the M and C of the Berg Pins. A 0.68-Vdc auxiliary input to the cassette recorder is available when the jumper is placed across the A and C of the Berg Pins. The "System Board Component Diagram" shows the location of the cassette Berg pins.



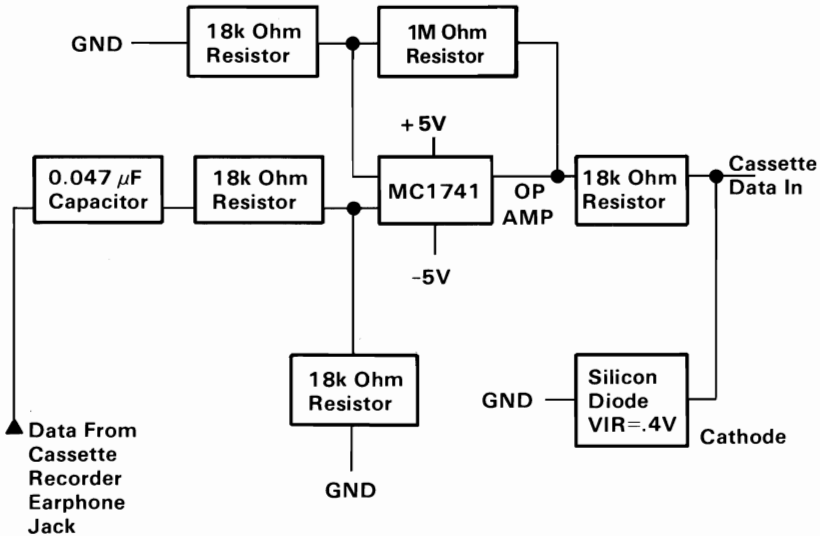
Microphone Input
(0.075 Vdc)



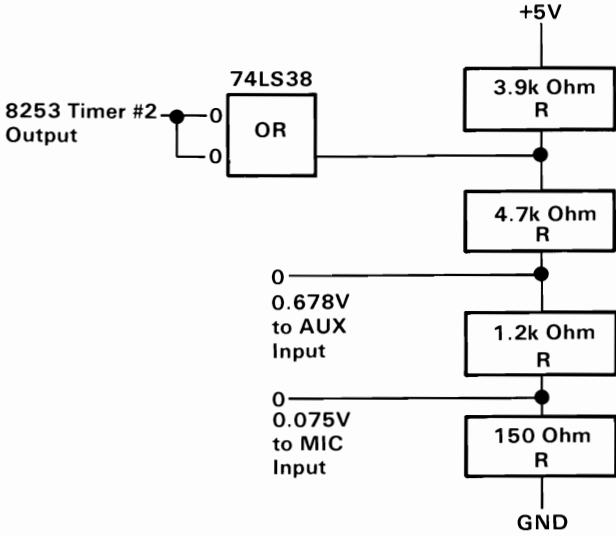
Auxiliary Input
(0.68 Vdc)

Cassette Circuit Block Diagrams

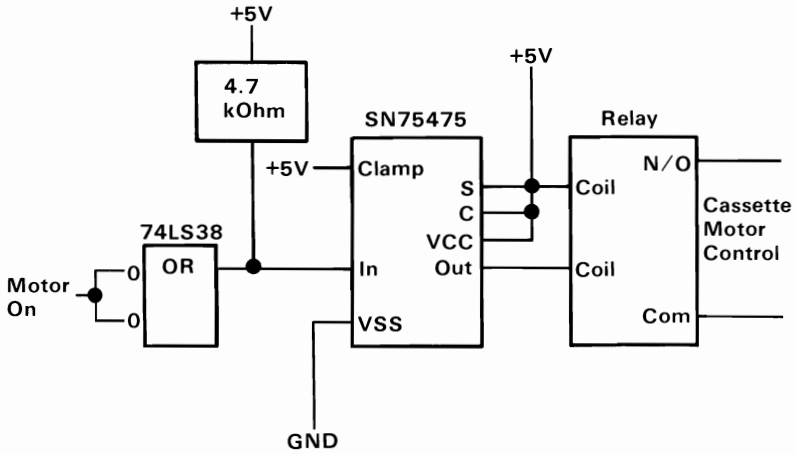
Circuit block diagrams for the cassette-interface read hardware, write hardware, and motor control are illustrated below.



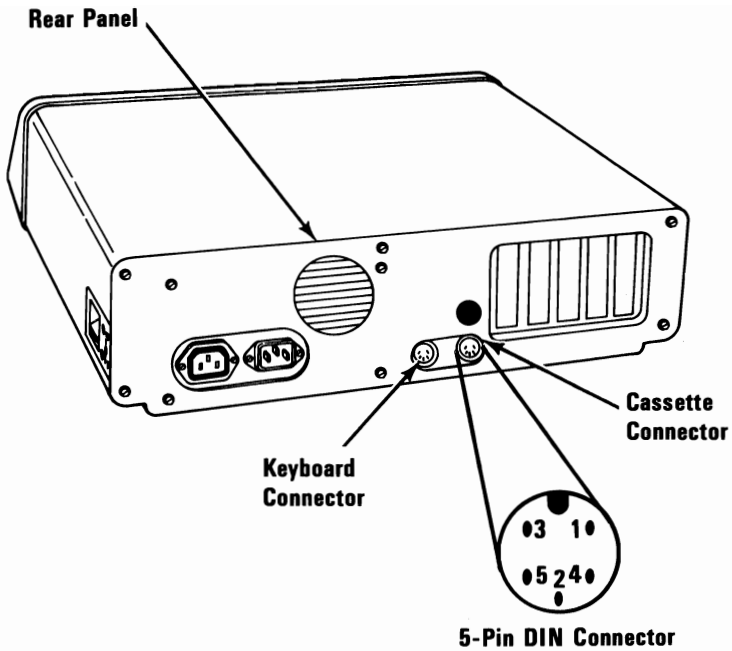
Cassette-Interface Read Hardware Block Diagram



Cassette Interface Write Hardware Block Diagram



Cassette Motor Control Block Diagram



Pin	Signal	Electrical Characteristics
1	Motor Control	Common from Relay
2	Ground	
3	Motor Control	Relay N.O. (6 Vdc at 1A)
4	Data In	500nA at $\pm 13V$ - at 1,000 - 2,000 Baud
5	Data Out (Microphone or Auxiliary)	250 μA at 0.68 Vdc or ** 0.075 Vdc

* All voltages and currents are maximum ratings and should not be exceeded.

** Data out can be chosen using a jumper located on the system board.
(Auxiliary \rightarrow 0.68 Vdc or Microphone \rightarrow 0.075 Vdc).

Interchange of these voltages on the cassette recorder could lead to damage of recorder inputs.

Cassette Interface Connector Specifications

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	Or	IPL 5-1/4 Diskette Drive (SW1-1)																	
		1		1		Reserved (SW1-2)																	
		2		2		System Board Read/Write Memory Size *(SW1-3)																	
		3		3		System Board Read/Write Memory Size *(SW1-4)																	
		4		4		+ Display Type 1 **(SW1-5)																	
		5		5		+ Display Type 2 **(SW1-6)																	
		6		6		No. of 5-1/4 Drives *** (SW1-7)																	
7		7	No. of 5-1/4 Drives *** (SW1-8)																				
0061	O U T P U T	PB0	+ Timer 2 Gate Speaker																				
		1	+ Speaker Data																				
		2	+ (Read Read/Write Memory Size) or (Read Spare Key)																				
		3	+ Cassette Motor Off																				
		4	- Enable Read/Write Memory																				
		5	- Enable I/O Channel Check																				
		6	- Hold Keyboard Clock Low																				
7	- (Enable Keyboard or + (Clear Keyboard and Enable Sense Switches)																						
0062	I N P U T	PC0	I/O Read/Write Memory (Sw2-1)	Binary Value X 32K	Or	I/O Read/Write Memory (Sw2-2)																	
			I/O Read/Write Memory (Sw2-2)				I/O Read/Write Memory (Sw2-3)																
			I/O Read/Write Memory (Sw2-3)					I/O Read/Write Memory (Sw2-4)															
			I/O Read/Write Memory (Sw2-4)																				
		4	+ Cassette Data In																				
		5	+ Timer Channel 2 Out																				
		6	+ I/O Channel Check																				
7	+ Read/Write Memory Parity Check																						
0063	Command/Mode Register		Hex 99																				
Mode Register Value		<table border="1"> <tr> <td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td> </tr> <tr> <td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td> </tr> </table>						7	6	5	4	3	2	1	0	1	0	0	1	1	0	0	1
7	6	5	4	3	2	1	0																
1	0	0	1	1	0	0	1																

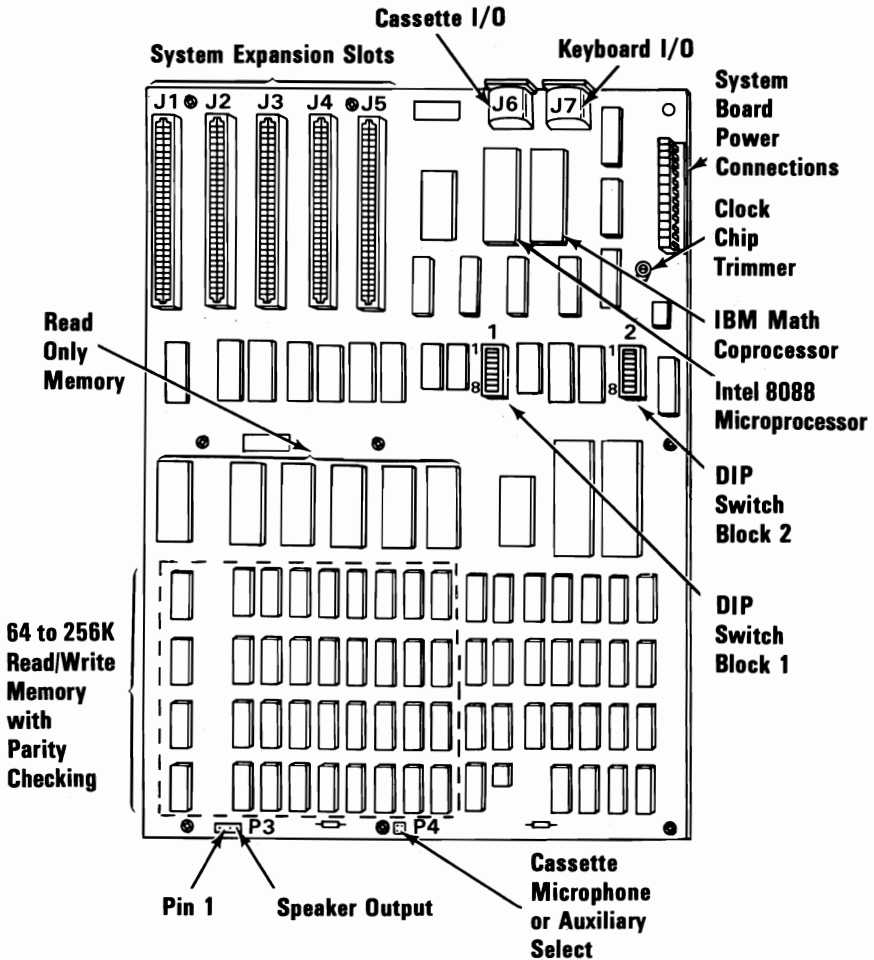
*	PA3 Sw1-4	PA2 Sw1-3	Amount of Memory Located on System Board 64 to 256K
**	PA5 Sw1-6	PA4 Sw1-5	Display at Power-Up Mode Reserved Color 40 X 25 (BW Mode) Color 80 X 25 (BW Mode) IBM Monochrome (80 X 25)
***	PA7 Sw1-8	PA6 Sw1-7	Number of 5-1/4" Drives in System 1 2 3 4
Note:	A plus (+) indicates a bit value of 1 performs the specified function. A minus (-) indicates a bit value of 0 performs the specified function. PA Bit = 0 implies switch "ON." PA bit = 1 implies switch "OFF."		

8255A I/O Bit Map

1-32 System Board

System-Board Switch Settings

All system board switch settings for total system memory, number of diskette drives, and type of display adapter are described under "Switch Settings" in the IBM Personal Computer *Guide to Operations*. The diagram showing the system board switch locations follows.



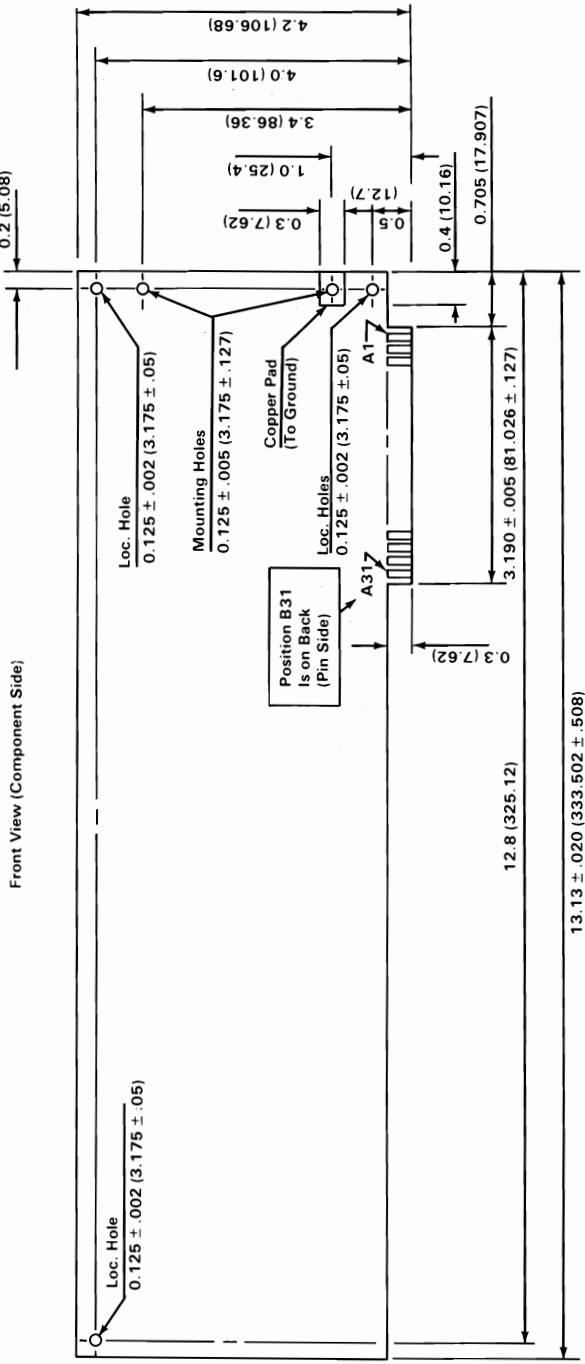
Specifications

The following voltages are available on the system-board I/O channel:

- + 5 Vdc \pm 5% on 2 connector pins
- 5 Vdc \pm 10% on 1 connector pin
- +12 Vdc \pm 5% on 1 connector pin
- 12 Vdc \pm 10% on 1 connector pin
- GND (Ground) on 3 connector pins

Card Specifications

The specifications for option cards follow.



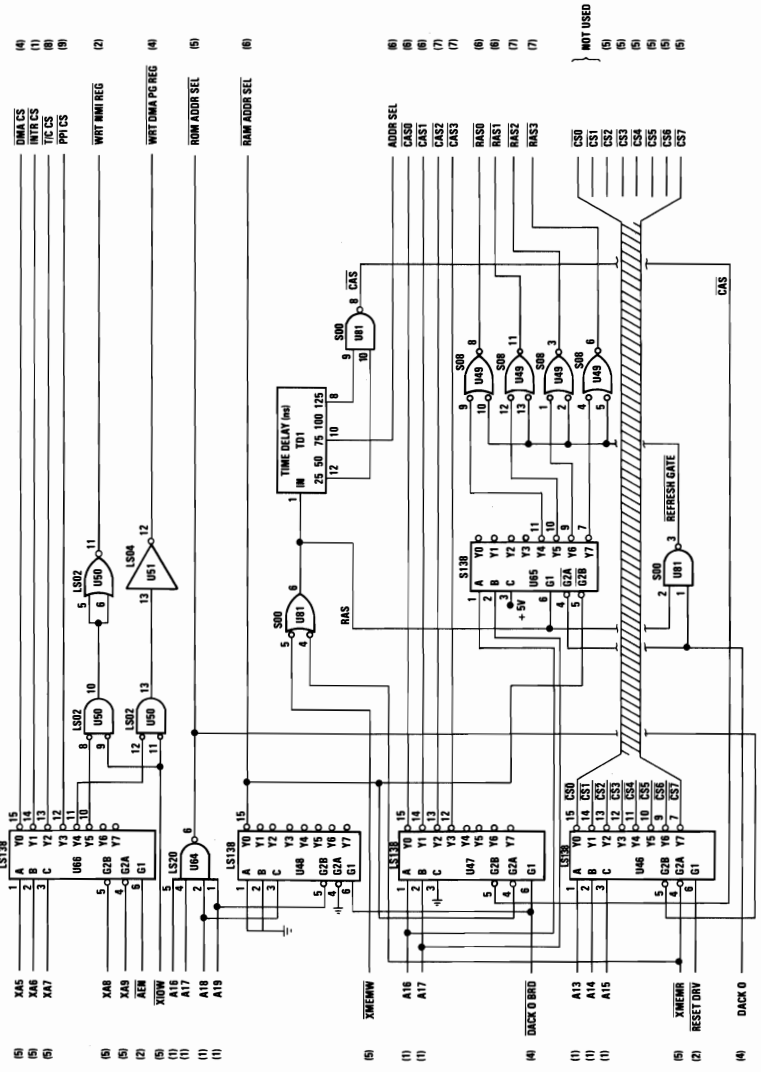
Front View (Component Side)

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

Logic Diagrams

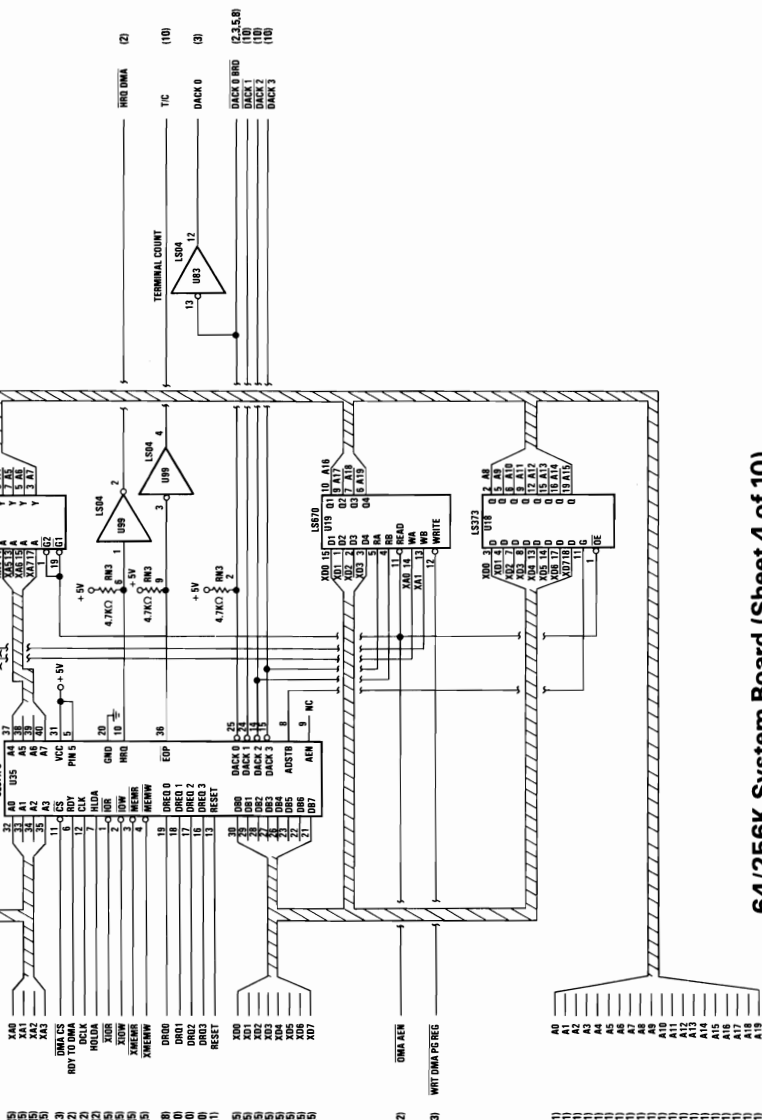
The following pages contain the logic diagrams for the system board.





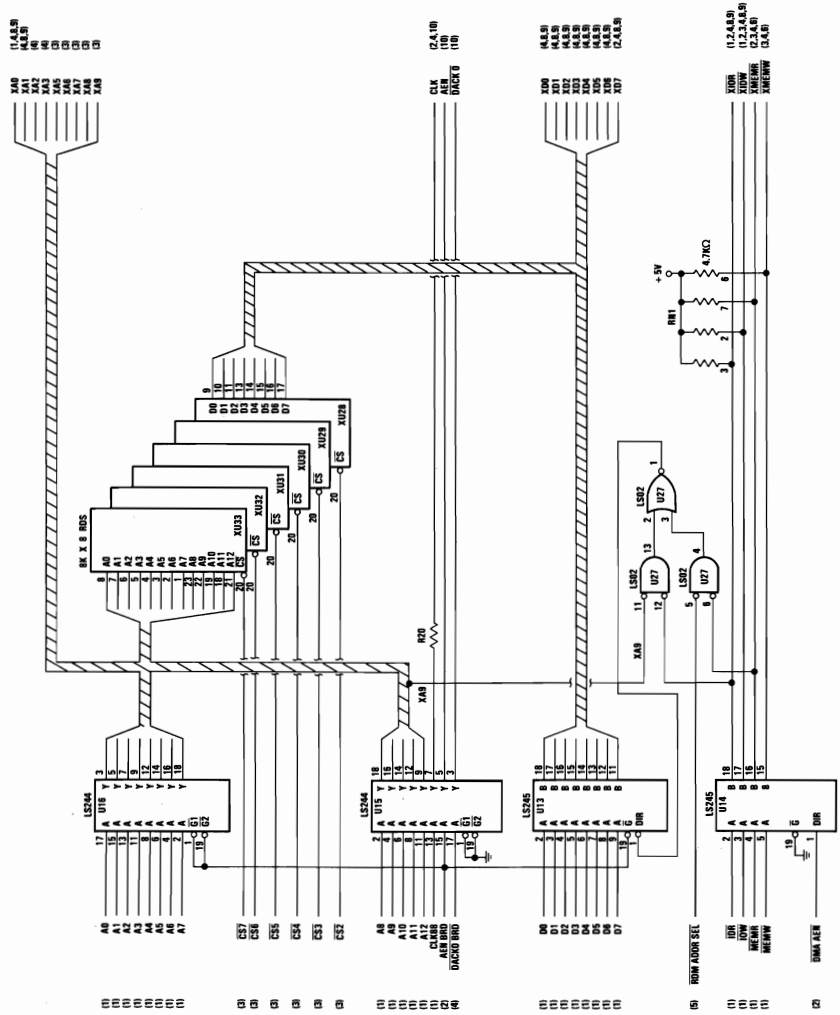
64/256K System Board (Sheet 3 of 10)

64/256K System Board (Sheet 4 of 10)

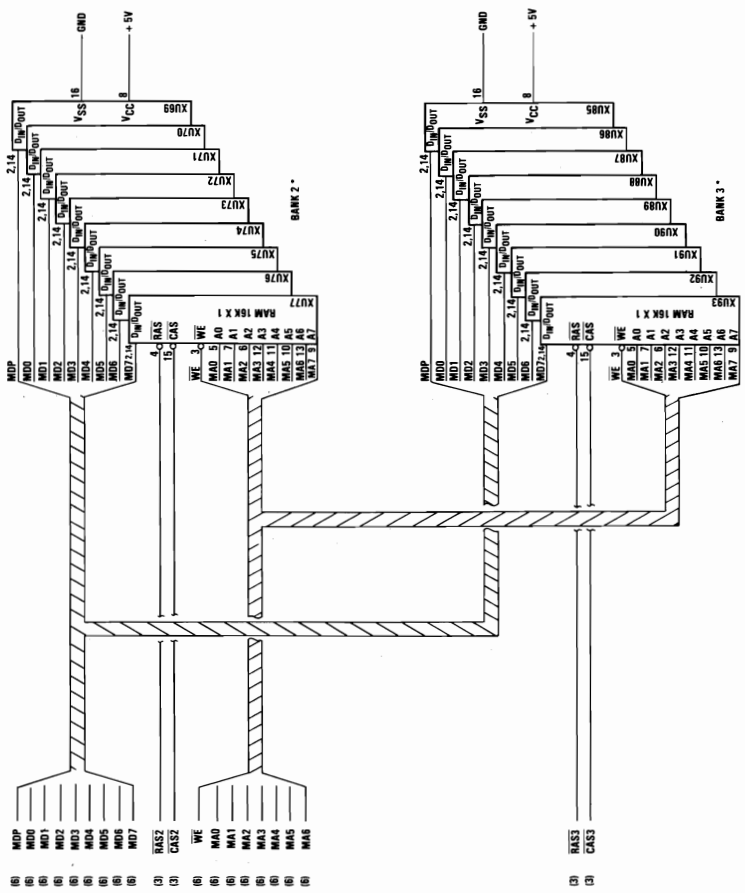


- (5) X40
- (5) X41
- (5) X42
- (5) X43
- (3) DMVLS
- (2) DCK
- (2) HOLD
- (5) XD0R
- (5) XD0R
- (5) XD0R
- (5) XD0R
- (5) XD0R
- (8) DR00
- (10) DR01
- (10) DR02
- (10) DR03
- (10) DR04
- (10) DR05
- (10) DR06
- (10) DR07
- (2) OMA LEN
- (3) WRIT OMA PD RES

- (1) A1
- (1) A2
- (1) A3
- (1) A4
- (1) A5
- (1) A6
- (1) A7
- (1) A8
- (1) A9
- (1) A10
- (1) A11
- (1) A12
- (1) A13
- (1) A14
- (1) A15
- (1) A16
- (1) A17
- (1) A18
- (1) A19

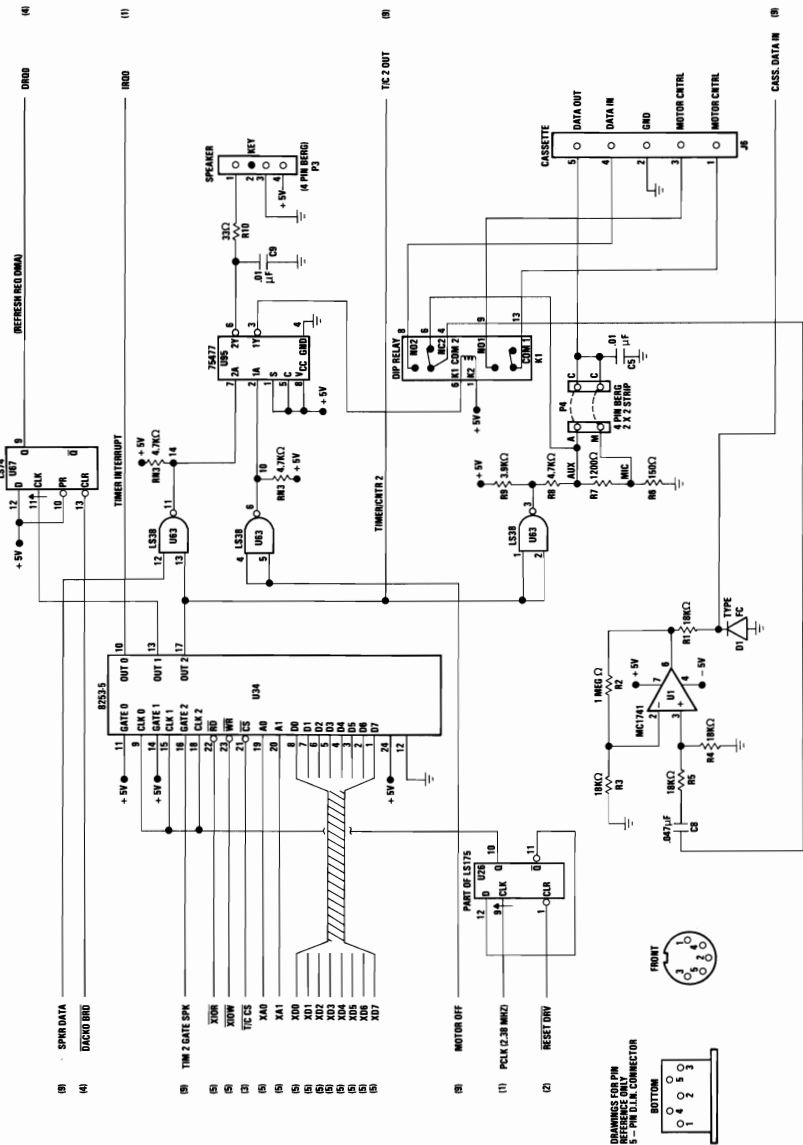


64/256K System Board (Sheet 5 of 10)

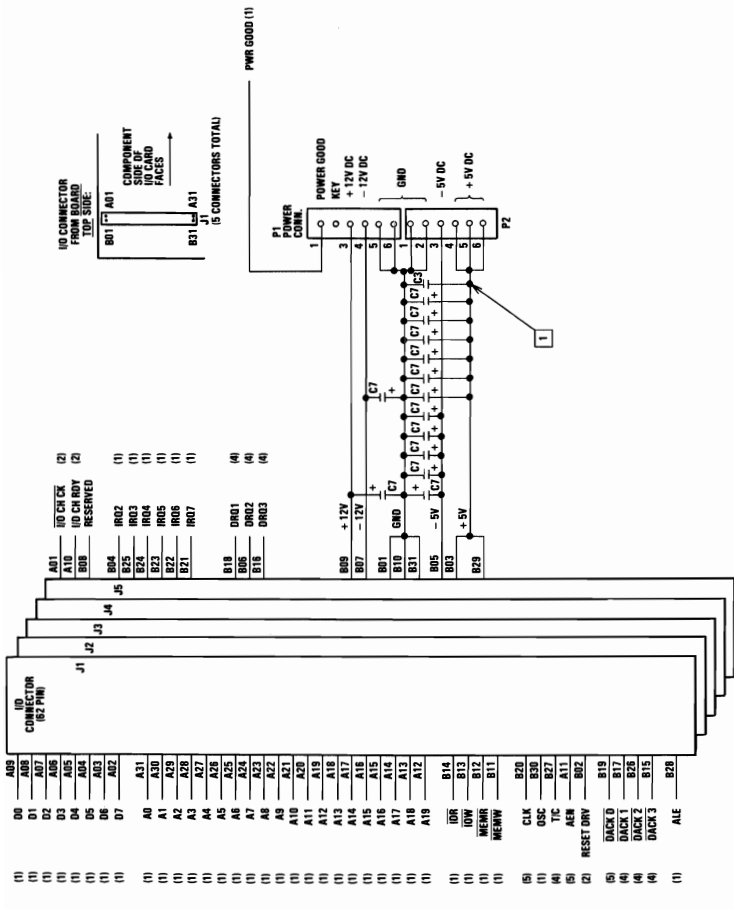


*BANKS 2 & 3 ARE FEATURES.

64/256K System Board (Sheet 7 of 10)



64/256K System Board (Sheet 8 of 10)



64/256K System Board (10 of 10)

NOTE: 1 C3 QTY = 23 CAPS

SECTION 2. COPROCESSOR

Contents

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



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

*The short 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 coprocessor is wired directly into 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

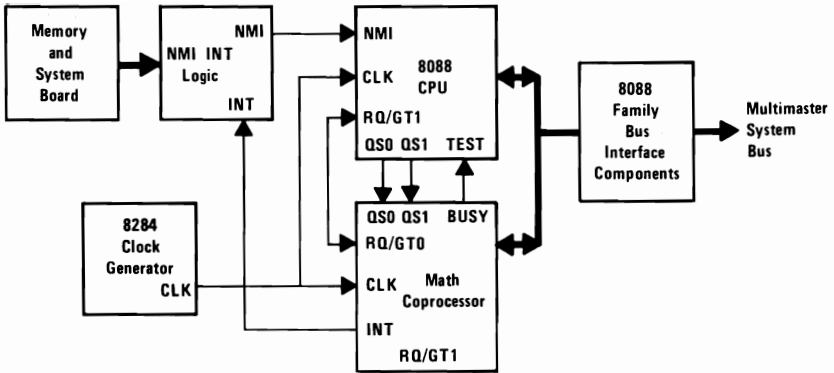
signal the microprocessor with an interrupt. There are three conditions that will disable the coprocessor interrupt to the microprocessor:

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 (NMI) register (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 SUPPLY

Contents

- Description** 3-3
- Input Requirements** 3-4
- Outputs** 3-4
 - Vdc Output 3-4
 - Vac Output 3-5
- Overvoltage/Overcurrent Protection** 3-5
 - Primary (Input) 3-5
 - Secondary (Output) 3-5
- Power Good Signal** 3-6
- Power Supply Connectors and Pin Assignments** 3-6

Section 3



Description

The system power supply is located at the right rear of the system unit. It is an integral part of the system-unit chassis. Its housing provides support for the rear panel, and its fan furnishes cooling for the whole system.

It supplies the power and reset signal necessary for the operation of the system board, installed options, and the keyboard. It also provides a switched ac socket for the IBM Monochrome Display and two separate connectors for power to the 5-1/4 inch diskette drives.

The two different power supplies available are designed for continuous operation at 63.5 Watts. They have a fused 120 Vac or 220/240 Vac input and provide four regulated dc output voltages: 7 A at +5 Vdc, 2 A at +12 Vdc, 0.3 A at -5 Vdc, and 0.25 A at -12 Vdc. These outputs are overvoltage, overcurrent, open-circuit, and short-circuit protected. If a dc overload or overvoltage condition occurs, all dc outputs are shut down as long as the condition exists.

The +12 Vdc and -12 Vdc power the EIA drivers and receivers on the asynchronous communications adapter. The +12 Vdc also powers the system's dynamic memory and the two internal 5-1/4 inch diskette drive motors. It is assumed that only one drive is active at a time. The +5 Vdc powers the logic on the system board and diskette drives and allows about 4 A of +5 Vdc for the adapters in the system-unit expansion slots. The -5 Vdc is for dynamic memory bias voltage; it tracks the +5 Vdc and +12 Vdc very quickly at power-on and has a longer decay on power-off than the +5 Vdc and +12 Vdc outputs. All four power supply dc voltages are bussed across each of the five system-unit expansion slots.

Input Requirements

The following are the input requirements for the system unit power supply.

Voltage (Vac)			Frequency (Hz)	Current (Amps)
Nominal	Minimum	Maximum	+ / - 3Hz	Maximum
120	104	127	60	2.5 at 104 Vac
220/240	180	259	50	1.0 at 180 Vac

Outputs

Vdc Output

The following are the dc outputs for the system unit power supply.

Voltage (Vdc)	Current (Amps)		Regulation (Tolerance)	
	Minimum	Maximum	+ %	- %
+ 5.0	2.3	7.0	5	4
- 5.0	0.0	0.3	10	8
+ 12.0	0.4	2.0	5	4
- 12.0	0.0	0.25	10	9

Vac Output

The power supply provides a filtered, fused, ac output that is switched on and off with the main power switch. The maximum current available at this output is 0.75 A. The receptacle provided at the rear of the power supply for this ac output is a nonstandard connector designed to be used only for the IBM Monochrome Display.

Overvoltage/Overcurrent Protection

The system power supply employs the protection features which are described below.

Primary (Input)

The following table describes the primary (input voltage) protection for the system-unit power supply.

Voltage (Nominal Vac)	Type Protection	Rating (Amps)
120	Fuse	2
220/240	Fuse	1

Secondary (Output)

On overvoltage, the power supply is designed to shut down all outputs when either the +5 Vdc or the +12 Vdc output exceeds 200% of its maximum rated voltage. On overcurrent, the supply will turn off if any output exceeds 130% of its nominal value.

Power Good Signal

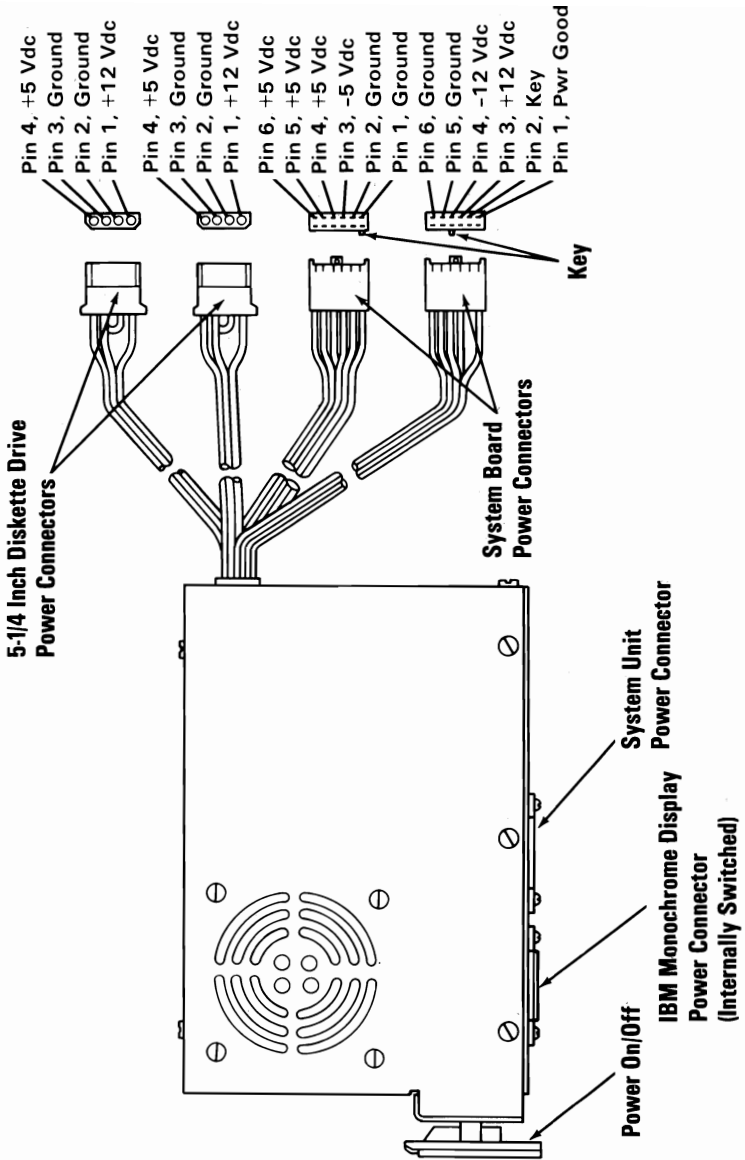
When the power supply is turned on after it has been off for a minimum of 5 seconds, it generates a 'power good' signal that indicates there is adequate power for processing. When the four output voltages are above the minimum sense levels, as described below, the signal sequences to a TTL-compatible up level (2.4 Vdc to 5.5 Vdc), is capable of sourcing 60 μ A. When any of the four output voltages is below its minimum sense level or above its maximum sense level, the 'power good' signal will be TTL-compatible down level (0.0 Vdc to 0.4 Vdc) capable of supplying 500 μ A. The 'power good' signal has a turn-on delay of 100-ms after the output voltages have reached their respective minimum sense levels.

Output Voltage	Under-Voltage Nominal Sense Level	Over-Voltage Nominal Sense Level
+ 5 Vdc	+ 4.0 Vdc	+ 5.9 Vdc
- 5 Vdc	- 4.0 Vdc	- 5.9 Vdc
+ 12 Vdc	+ 9.6 Vdc	+ 14.2 Vdc
- 12 Vdc	- 9.6 Vdc	- 14.2 Vdc

Power Supply Connectors 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 configuration and locations follow.

Power Supply and Connectors





SECTION 4. KEYBOARD

Contents

Description 4-3

Block Diagram 4-4

Keyboard Diagrams 4-5

Connector Specifications 4-12

Keyboard Logic Diagram 4-13



Description

The IBM Personal Computer 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, two bidirectional signal lines, and one wire used as a 'reset' line. The cable is approximately 182.88 cm (6 ft) long and is coiled, like that of a telephone handset.

The keyboard uses 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 orientation).

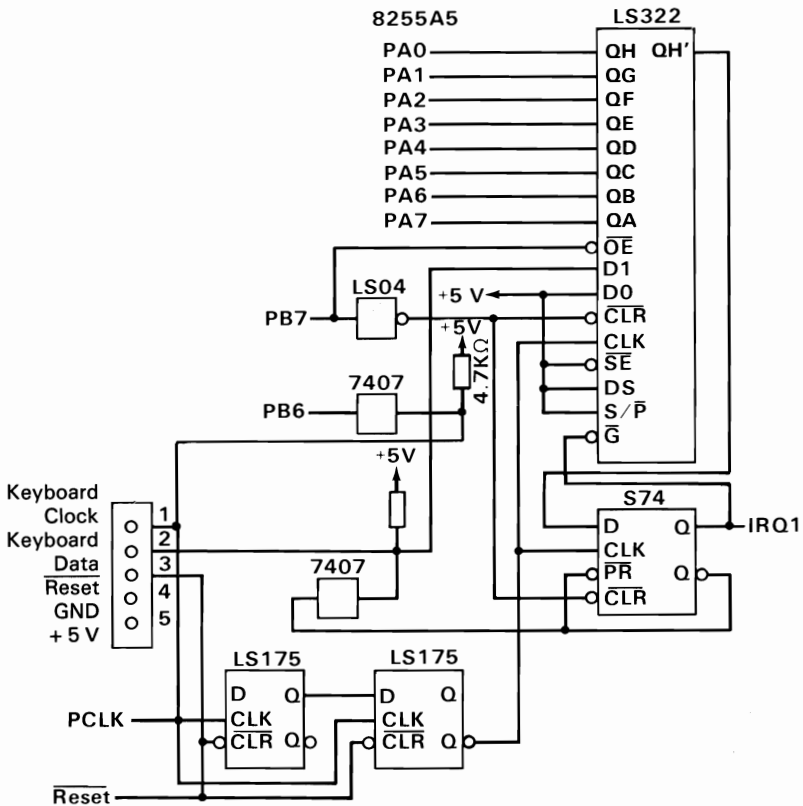
The keyboard has 83 keys arranged in three major groupings. The central portion of the keyboard is a standard typewriter keyboard layout. On the left side are 10 function keys. These keys are defined by the software. On the right is a 15-key keypad. These keys are also defined by the software, but have 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 microprocessor (Intel 8048) in the keyboard performs several functions, including a power-on self test when requested by the system unit. This test checks the microprocessor (Intel 8048) 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 keyboard arrangements are available. These are illustrated on the following pages. For information about the keyboard routines required to implement non-U.S. keyboards, refer to the *Guide to Operations* and *DOS* manuals.

Block Diagram



Keyboard Interface Block Diagram

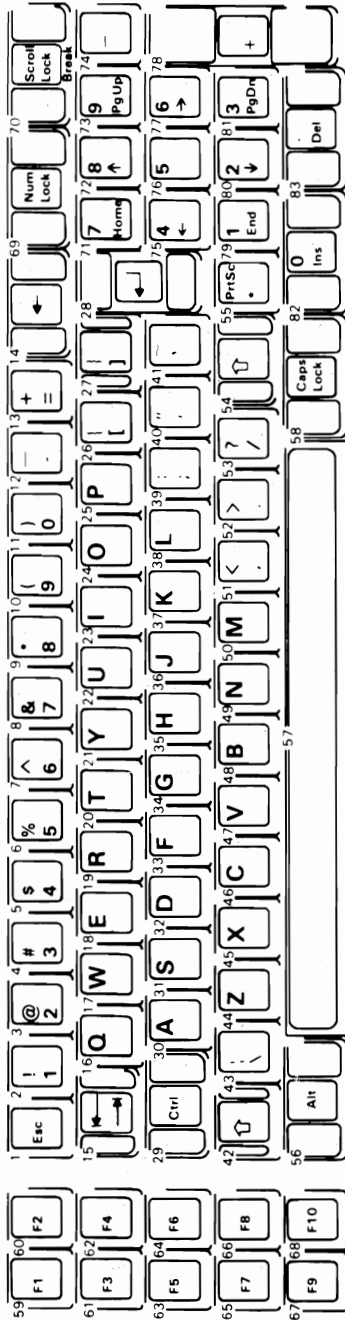
Keyboard Diagrams

The IBM Personal Computer keyboard is available in six layouts:

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

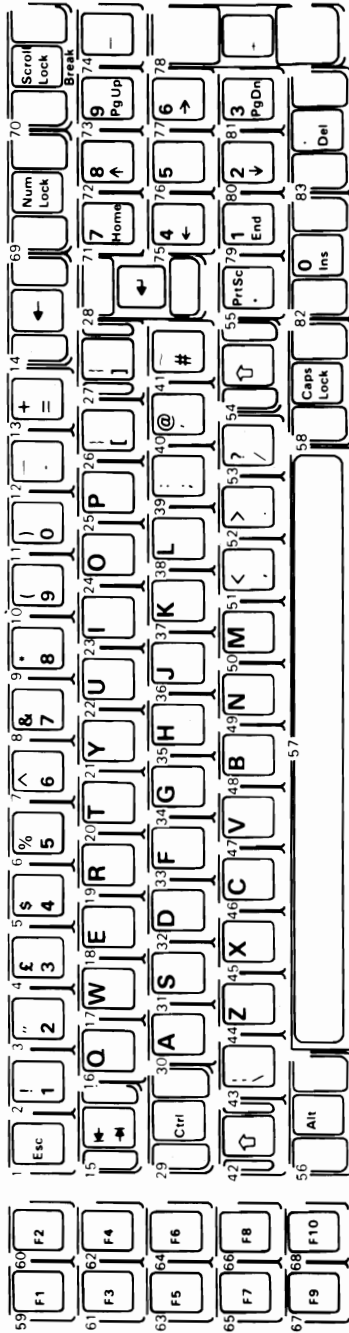
The following pages show all six keyboard layouts.

U.S. English Keyboard Diagram



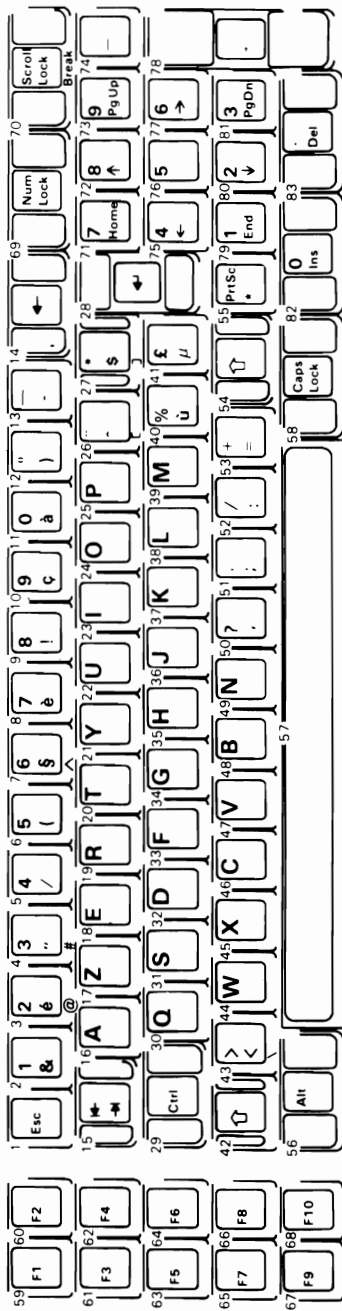
Note: Nomenclature is on both the top and front face of keybuttons as shown. The number to the upper left designates the button position.

U.K. English Keyboard Diagram



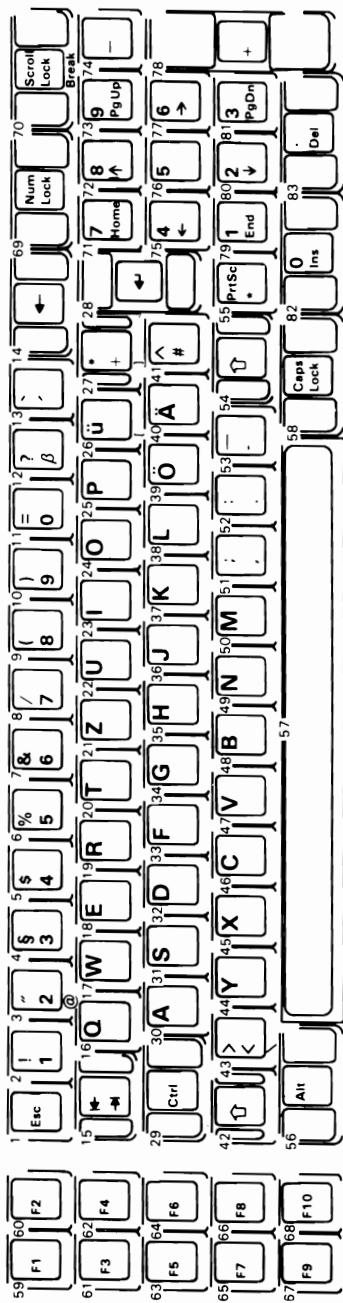
Note: Nomenclature is on both the top and front face of keybuttons as shown. The number to the upper left designates the button position.

French Keyboard Diagram



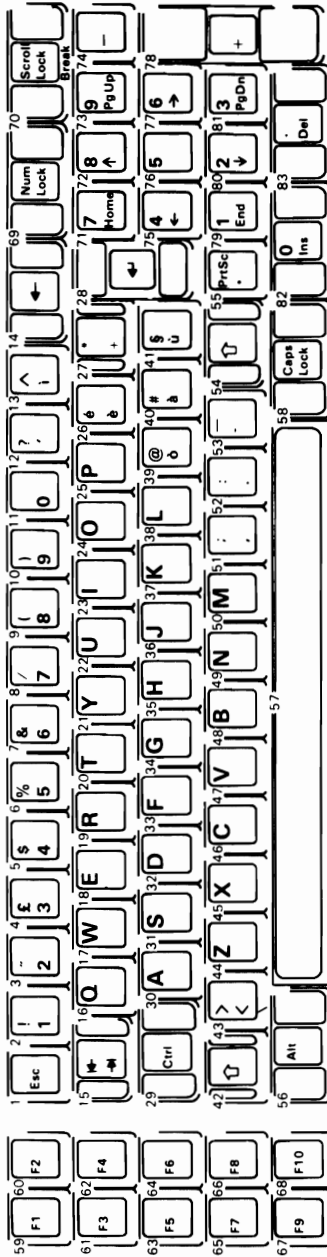
Note: Nomenclature is on both the top and front face of keybuttons as shown. The number to the upper left designates the button position.

German Keyboard Diagram



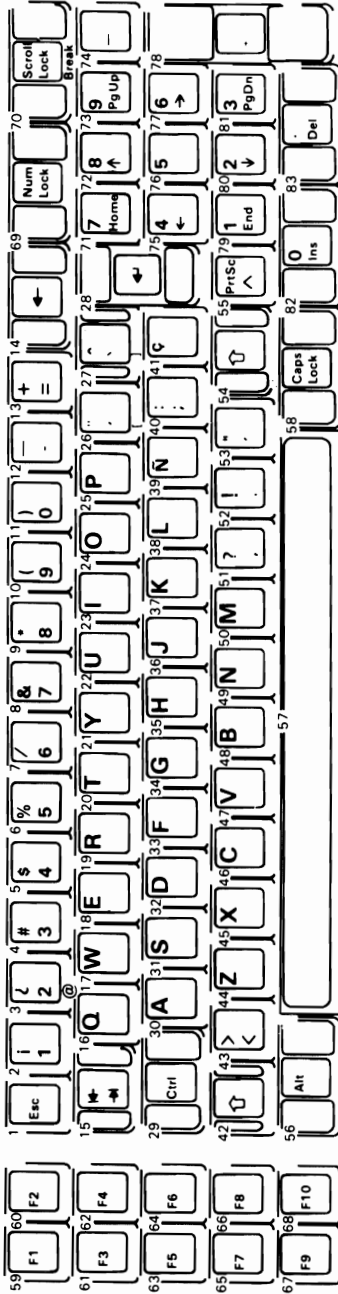
Note: Nomenclature is on both the top and front face of keybuttons as shown. The number to the upper left designates the button position.

Italian Keyboard Diagram



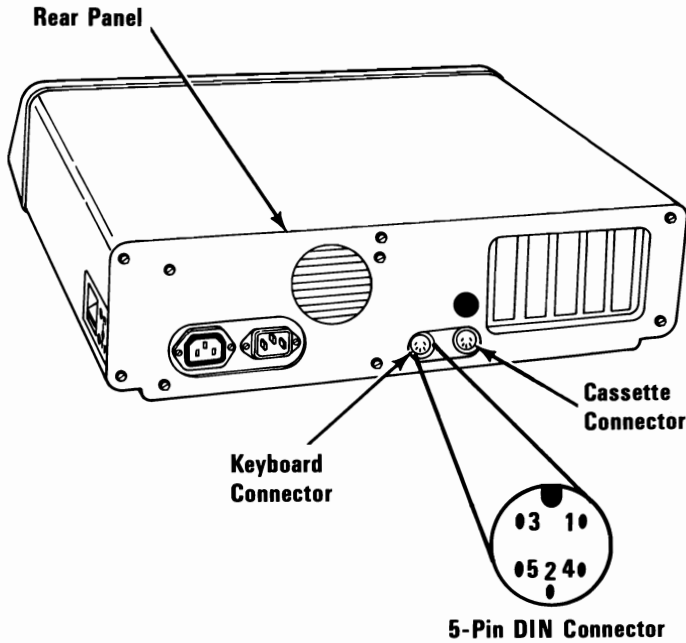
Note: Nomenclature is on both the top and front face of keybuttons as shown. The number to the upper left designates the button position.

Spanish Keyboard Diagram



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

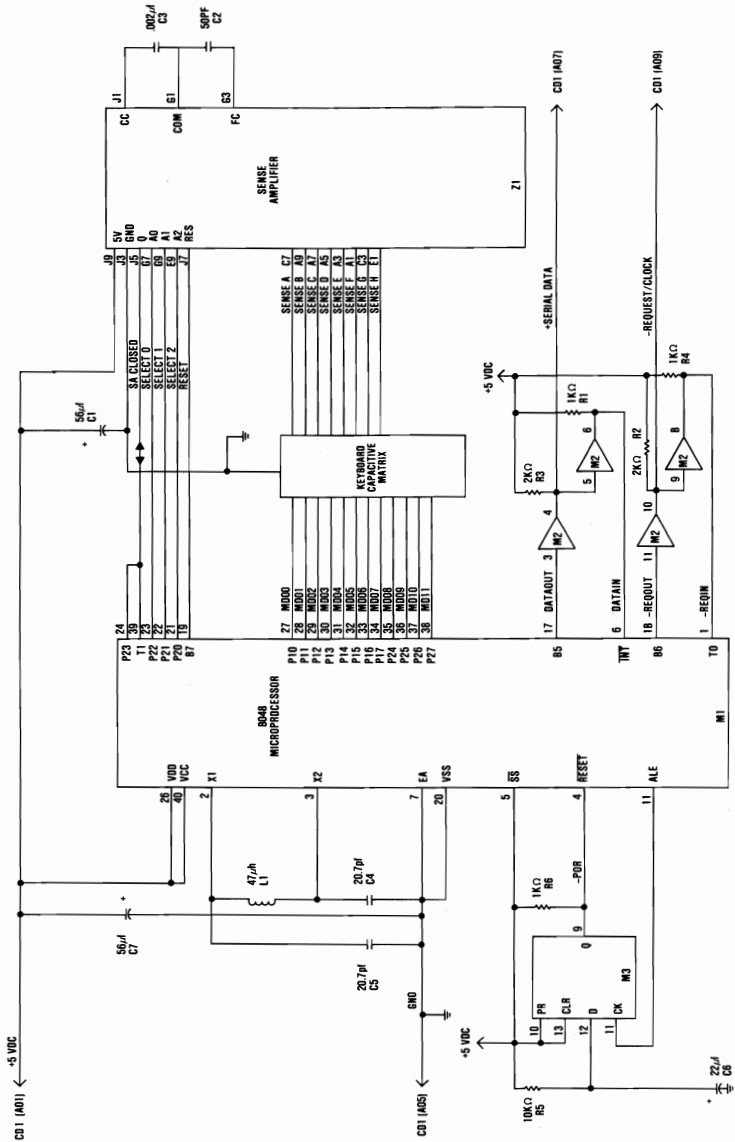
Connector Specifications



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

Keyboard Interface Connector Specifications

Keyboard Logic Diagram



IBM Keyboard (Sheet 1 of 1)

Section 4



SECTION 5. SYSTEM BIOS

Contents

System BIOS Usage	5-3
Vectors with Special Meanings	5-6
Other Read/Write Memory Usage	5-8
BIOS Programming Hints	5-13
Adapter Cards with System-Accessible ROM Modules	5-13
Keyboard Encoding and Usage	5-14
Encoding	5-14
Extended Codes	5-18
Shift States	5-19
Special Handling	5-20
Extended Functions	5-21
Keyboard Usage	5-22
BIOS Cassette Logic	5-25
Software Algorithms - Interrupt Hex 15	5-25
Cassette Write	5-25
Cassette Read	5-26
Data Record Architecture	5-27
Error Recovery	5-28
System BIOS Listing	5-29
Quick Reference	5-29

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 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.

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 at input to indicate the desired operation. For example, to set the time of day, the following code is required:

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

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

```
MOV DX,LOW_COUNT
```

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

To read the time of day:

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

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

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.

Address (Hex)	Interrupt Number	Name	BIOS Entry
0-3	0	Divide by Zero	D_EOI
4-7	1	Single Step	D_EOI
8-B	2	Nonmaskable	NMI_INT
C-F	3	Breakpoint	D_EOI
10-13	4	Overflow	D_EOI
14-17	5	Print Screen	PRINT_SCREEN
18-1B	6	Reserved	D_EOI
1D-1F	7	Reserved	D_EOI
20-23	8	Time of Day	TIMER_INT
24-27	9	Keyboard	KB_INT
28-2B	A	Reserved	D_EOI
2C-2F	B	Communications	D_EOI
30-33	C	Communications	D_EOI
34-37	D	Disk	D_EOI
38-3B	E	Diskette	DISK_INT
3C-3F	F	Printer	D_EOI
40-43	10	Video	VIDEO_IO
44-47	11	Equipment Check	EQUIPMENT
48-4B	12	Memory	MEMORY_SIZE _DETERMINE
4C-4F	13	Diskette/Disk	DISKETTE_IO
50-53	14	Communications	RS232_IO
54-57	15	Cassette	CASSETTE_IO
58-5B	16	Keyboard	KEYBOARD_IO
5C-5F	17	Printer	PRINTER_IO
60-63	18	Resident BASIC	F600:0000
64-67	19	Bootstrap	BOOT_STRAP
68-6B	1A	Time of Day	TIME_OF_DAY
6C-6F	1B	Keyboard Break	DUMMY_RETURN
70-73	1C	Timer Tick	DUMMY_RETURN
74-77	1D	Video Initialization	VIDEO_PARAMS
78-7B	1E	Diskette Parameters	DISK_BASE
7C-7F	1F	Video Graphics Characters	0
100-103	40	Diskette pointer save area for Fixed Disk	
104-107	41	Fixed Disk Parameters	FD_TBL
168-16B	5A	Cluster	D000:XXXX
16C-16F	5B	Used by Cluster Program	N/A
180-19F	60-67	Reserved for User Programs	N/A

8088 Software Interrupt Listing

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 407 contain the base addresses of any RS-232C cards attached to the system. Locations hex 408 to 40F contain the base addresses of the Printer Adapter.

Memory locations hex 300 to 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.

Address (Hex)	Interrupt (Hex)	Function
80-83	20	DOS Program Terminate
84-87	21	DOS Function Call
88-8B	22	DOS Terminate Address
8C-8F	23	DOS Ctrl Break Exit Address
90-93	24	DOS Fatal Error Vector
94-97	25	DOS Absolute Disk Read
98-9B	26	DOS Absolute Disk Write
9C-9F	27	DOS Terminate, Fix In Storage
A0-FF	28-3F	Reserved for DOS
100-17F	40-5F	Reserved
180-19F	60-67	Reserved for User Software Interrupts
1A0-1FF	68-7F	Not Used
200-217	80-85	Reserved by BASIC
218-3C3	86-F0	Used by BASIC Interpreter while BASIC is running
3C4-3FF	F1-FF	Not Used

BASIC and DOS Reserved Interrupts

Address (Hex)	Mode	Function
400-48F 490-4EF 4F0-4FF	ROM BIOS	See BIOS Listing Reserved Reserved as Intra-Application Communication Area for any application
500-5FF 500	DOS	Reserved for DOS and BASIC Print Screen Status Flag Store 0-Print Screen Operation 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 (Hex Value)	Length		
Line number of current line being executed	2E	2		
Line number of last error	347	2		
Offset into segment of start of program text	30	2		
Offset into segment of start of variables (end of program text 1-1)	358	2		
Keyboard buffer contents if 0-no characters in buffer if 1-characters in buffer	6A	1		
Character color in graphics mode Set to 1, 2, or 3 to get text in colors 1 to 3. Do not set to 0. (Default = 3)	4E	1		
<p>Example</p> <pre>100 Print PEEK (&H2E) + 256*PEEK (&H2F)</pre> <p style="margin-left: 40px;"> { L H </p> <p style="margin-left: 40px;">100</p> <table border="1" style="margin-left: 100px;"> <tr> <td style="width: 50px; text-align: center;">Hex 64</td> <td style="width: 50px; text-align: center;">Hex 00</td> </tr> </table>			Hex 64	Hex 00
Hex 64	Hex 00			

BASIC Workspace Variables

Starting Address in Hex

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 start-up.

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.

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

Character Codes

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

Key Number	Base Case	Upper Case	Ctrl	Alt
1	Esc	Esc	Esc	- 1
2	1	!	- 1	Note 1
3	2	@	Nul (000) Note 1	Note 1
4	3	#	- 1	Note 1
5	4	\$	- 1	Note 1
6	5	%	- 1	Note 1
7	6	^	RS(030)	Note 1
8	7	&	- 1	Note 1
9	8	*	- 1	Note 1
10	9	(- 1	Note 1
11	0)	- 1	Note 1
12	-	_	US(031)	Note 1
13	=	+	- 1	Note 1
14	Backspace (008)	Backspace (008)	Del (127)	- 1
15	→ (009)	← (Note 1)	- 1	- 1
16	q	Q	DC1 (017)	Note 1
17	w	W	ETB (023)	Note 1

Character Codes (Part 1 of 3)

Key Number	Base Case	Upper Case	Ctrl	Alt
18	e	E	ENQ (005)	Note 1
19	r	R	DC2 (018)	Note 1
20	t	T	DC4 (020)	Note 1
21	y	Y	EM (025)	Note 1
22	u	U	NAK (021)	Note 1
23	i	I	HT (009)	Note 1
24	o	O	SI (015)	Note 1
25	p	P	DLE (016)	Note 1
26	[{	Esc (027)	- 1
27]	}	GS (029)	- 1
28	CR	CR	LF (010)	- 1
29 Ctrl	- 1	- 1	- 1	- 1
30	a	A	SOH (001)	Note 1
31	s	S	DC3 (019)	Note 1
32	d	D	EOT (004)	Note 1
33	f	F	ACK (006)	Note 1
34	g	G	BEL (007)	Note 1
35	h	H	BS (008)	Note 1
36	j	J	LF (010)	Note 1
37	k	K	VT (011)	Note 1
38	l	L	FF (012)	Note 1
39	;	:	- 1	- 1
40	'	"	- 1	- 1
41	,	~	- 1	- 1
42 Shift	- 1	- 1	- 1	- 1
43	\		FS (028)	- 1
44	z	Z	SUB (026)	Note 1
45	x	X	CAN (024)	Note 1
46	c	C	ETX (003)	Note 1
47	v	V	SYN (022)	Note 1
48	b	B	STX (002)	Note 1
49	n	N	SO (014)	Note 1
50	m	M	CR (013)	Note 1
51	,	<	- 1	- 1
52	.	>	- 1	- 1
53	/	?	- 1	- 1
54 Shift	- 1	- 1	- 1	- 1
55	*	(Note 2)	(Note 1)	- 1
56 Alt	- 1	- 1	- 1	- 1
57	SP	SP	SP	SP
58 Caps Lock	- 1	- 1	- 1	- 1
59	Nul (Note 1)	Nul (Note 1)	Nul (Note 1)	Nul (Note 1)
60	Nul (Note 1)	Nul (Note 1)	Nul (Note 1)	Nul (Note 1)
61	Nul (Note 1)	Nul (Note 1)	Nul (Note 1)	Nul (Note 1)
62	Nul (Note 1)	Nul (Note 1)	Nul (Note 1)	Nul (Note 1)
63	Nul (Note 1)	Nul (Note 1)	Nul (Note 1)	Nul (Note 1)
64	Nul (Note 1)	Nul (Note 1)	Nul (Note 1)	Nul (Note 1)

Character Codes (Part 2 of 3)

Key Number	Base Case	Upper Case	Ctrl	Alt
65	Nul (Note 1)	Nul (Note 1)	Nul (Note 1)	Nul (Note 1)
66	Nul (Note 1)	Nul (Note 1)	Nul (Note 1)	Nul (Note 1)
67	Nul (Note 1)	Nul (Note 1)	Nul (Note 1)	Nul (Note 1)
68	Nul (Note 1)	Nul (Note 1)	Nul (Note 1)	Nul (Note 1)
69 Num Lock	- 1	- 1	Pause (Note 2)	- 1
70	- 1	- 1	Break (Note 2)	- 1
Scroll Lock				

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

Character Codes (Part 3 of 3)

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.

Key Number	Num Lock	Base Case	Alt	Ctrl
71	7	Home (Note 1)	- 1	Clear Screen
72	8	↑ (Note 1)	- 1	- 1
73	9	Page Up (Note 1)	- 1	Top of Text and Home
74	-	-----	- 1	- 1
75	4	←(Note 1)	- 1	Reverse Word (Note 1)
76	5	- 1	- 1	- 1
77	6	→(Note 1)	- 1	Advance Word (Note 1)
78	+	+	- 1	- 1
79	1	End (Note 1)	- 1	Erase to EOL (Note 1)
80	2	↓ (Note 1)	- 1	- 1
81	3	Page Down (Note 1)	- 1	Erase to EOS (Note 1)
82	0	Ins	- 1	- 1
83		Del (Notes 1,2)	Note 2	Note 2

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

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 (Nul) 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.

Second Code	Function
3	Nul Character
15	←
16-25	Alt Q, W, E, R, T, Y, U, I, O, P
30-38	Alt A, S, D, F, G, H, J, K, L
44-50	Alt Z, X, C, V, B, N, M
59-68	F1 to F10 Function Keys Base Case
71	Home
72	↑
73	Page Up and Home Cursor
75	←
77	→
79	End
80	↓
81	Page Down and Home Cursor
82	Ins (Insert)
83	Del (Delete)
84-93	F11 to F20 (Uppercase F1 to F10)
94-103	F21 to F30 (Ctrl F1 to F10)
104-113	F31 to F40 (Alt F1 to F10)
114	Ctrl PrtSc (Start/Stop Echo to Printer)
115	Ctrl←(Reverse Word)
116	Ctrl→(Advance Word)
117	Ctrl End [Erase to End of Line (EOL)]
118	Ctrl PgDn [Erase to End of Screen (EOS)]
119	Ctrl Home (Clear Screen and Home)
120-131	Alt 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, -, = (Keys 2-13)
132	Ctrl PgUp (Top 25 Lines of Text and Home Cursor)

Keyboard Extended Functions

Shift States

Most shift states are handled within the keyboard routine, transparent 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 diagram in Section 4. 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 is 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 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 0 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 1A. 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 (key 55) keys will result in an interrupt invoking the print screen routine. This routine works in the alphanumeric 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 Usage

This section is intended to outline a set of guidelines of key usage when performing commonly used functions.

Function	Key(s)	Comment
Home Cursor	Home	Editors; word processors
Return to outermost menu	Home	Menu driven applications
Move cursor up	↑	Full screen editor, word processor
Page up, scroll backward 25 lines and home	PgUp	Editors; word processors
Move cursor left	←Key 75	Text, command entry
Move cursor right	→	Text, command entry
Scroll to end of text Place cursor at end of line	End	Editors; word processors
Move cursor down	↓	Full screen editor, word processor
Page down, scroll forward 25 lines and home	Pg Dn	Editors; word processors
Start/Stop insert text at cursor, shift text right in buffer	Ins	Text, command entry
Delete character at cursor	Del	Text, command entry
Destructive backspace	←Key 14	Text, command entry
Tab forward	→	Text entry
Tab reverse	←	Text entry
Clear screen and home	Ctrl Home	Command entry
Scroll up	↑	In scroll lock mode
Scroll down	↓	In scroll lock mode
Scroll left	←	In scroll lock mode
Scroll right	→	In scroll lock mode
Delete from cursor to EOL	Ctrl End	Text, command entry
Exit/Escape	Esc	Editor, 1 level of menu, and so on
Start/Stop Echo screen to printer	Ctrl Prt Sc (Key 55)	Any time
Delete from cursor to EOS	Ctrl PgDn	Text, command entry
Advance word	Ctrl →	Text entry
Reverse word	Ctrl ←	Text entry
Window Right	Ctrl →	When text is too wide to fit screen
Window Left	Ctrl ←	When text is too wide to fit screen
Enter insert mode	Ins	Line editor

Keyboard - Commonly Used Functions (Part 1 of 2)

5-22 System BIOS

Function	Key(s)	Comment
Exit insert mode	Ins	Line editor
Cancel current line	Esc	Command entry, text entry
Suspend system (pause)	Ctrl Num Lock	Stop list, stop program, and so on Resumes on any key
Break interrupt	Ctrl Break	Interrupt current process
System reset	Alt Ctrl Del	Reboot
Top of document and home cursor	Ctrl PgUp	Editors, word processors
Standard function keys	F1-F10	Primary function keys
Secondary function keys	Shift F1-F10 Ctrl F1-F10 Alt F1-F10	Extra function keys if 10 are not sufficient
Extra function keys	Alt Keys 2-13 (1-9,0,-,=)	Used when templates are put along top of keyboard
Extra function keys	Alt A-Z	Used when function starts with same letter as one of the alpha keys

Keyboard - Commonly Used Functions (Part 2 of 2)

Function	Key
Carriage return	↵
Line feed	Ctrl ↵
Bell	Ctrl G
Home	Home
Cursor up	↑
Cursor down	↓
Cursor left	←
Cursor right	→
Advance one word	Ctrl →
Reverse one word	Ctrl ←
Insert	Ins
Delete	Del
Clear screen	Ctrl Home
Freeze output	Ctrl Num Lock
Tab advance	→
Stop execution (break)	Ctrl Break
Delete current line	Esc
Delete to end of line	Ctrl End
Position cursor to end of line	End

BASIC Screen Editor Special Functions

Function	Key
Suspend	Ctrl Num Lock
Echo to printer	Ctrl PrtSc
Stop echo to printer	(Key 55 any case) Ctrl PrtSc
Exit current function (break)	Ctrl Break
Backspace	← Key 14
Line feed	Ctrl ↵
Cancel line	Esc
Copy character	F1 or →
Copy until match	F2
Copy remaining	F3
Skip character	Del
Skip until match	F4
Enter insert mode	Ins
Exit insert mode	Ins
Make new line the template	F5
String separator in REPLACE	F6
End of file in keyboard input	F6

DOS Special Functions

BIOS Cassette Logic

Software Algorithms - Interrupt Hex 15

The cassette routine is called by the request type in AH. The address of the bytes to be read from or written to the tape is specified ES:BX and the number of bytes to be read or written is specified by CX. The number of bytes read is returned in DX. The read block and write block automatically turn the cassette motor on at the start and off at the end. The request types in AH and the cassette status descriptions follow:

Request Type	Function
AH = 0	Turn Cassette Motor On
AH = 1	Turn Cassette Motor Off
AH = 2	Read Tape Block Read CX bytes into memory starting at Address ES:BX Return actual number of bytes read in DX Return Cassette Status in AH
AH = 3	Write Tape Block Write CX bytes onto cassette starting at Address DS:BX Return Cassette Status in AH

Cassette Status	Description
AH = 00	No Errors
AH = 01	Cyclic Redundancy Check (CRC) Error in Read Block
AH = 02	No Data Transitions
AH = 04	No Leader
AH = 80	Invalid Command

Notes: The carry flag will be set on any error.

Cassette Write

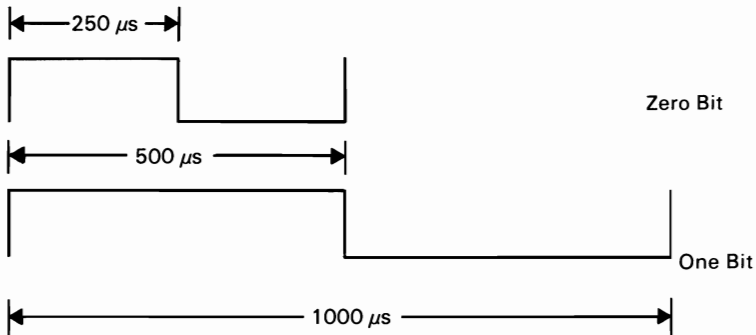
The write-block routine writes a tape block onto the cassette tape. The tape block is described in "Data Record Architecture" later in this section.

The write-block routine turns on the cassette drive motor and a synchronization bit(0) and then writes the leader(256 bytes of all 1's) to the tape. Next, the routine writes the number of data

blocks specified by CX. After each data block of 256 bytes, a 2-byte cyclic redundancy check (CRC) is written. The data bytes are taken from the memory location pointed at by ES.

The write-byte routine disassembles and writes the byte a bit at a time to the cassette. The method used is to set Timer 2 to the period of the desired data bit. The timer is set to a period of 1.0-ms for a 1 bit and 0.5-ms for a 0 bit.

The timer is set to mode 3, which means the timer outputs a square wave with a period given by its count register. The timer's period is changed on the fly for each data bit written to the cassette. If the number of data bytes to be written is not an integral multiple of 256, then, after the last desired data byte from memory has been written, the data block is extended to 256 bytes of writing multiples of the last data byte. The last block is closed with two CRC bytes as usual. After the last data block, a trailer consisting of four bytes of all 1 bits is written. Finally, the cassette motor is turned off, if there are no errors reported by the routine.



Cassette Read

The read-block routine turns on the cassette drive motor and then delays for about 0.5 second to allow the motor to come up to speed.

The read-block routine then searches for the leader and must detect all 1 bits for approximately 1/4 of the leader length before it can look for the sync (0) bit. After the sync bit is detected, the

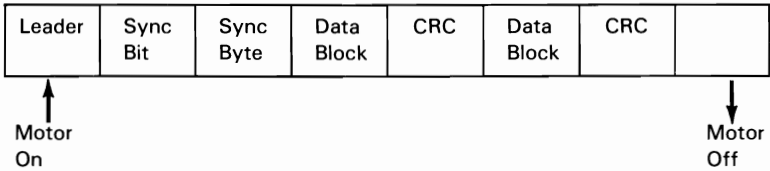
sync byte (ASCII character hex 16) is read. If the sync byte is read correctly, the data portion can be read. If a correct sync byte is not found, the routine goes back and searches for the leader again. The data is read a bit at a time and assembled into bytes. After each byte is assembled, it is written into memory at location ES:BX and BX is incremented by 1.

After each multiple of 256 data bytes is read, the CRC is read and compared to the CRC generated. If a CRC error is detected, the routine exits with the carry flag set to indicate an error and the status of AH set to hex 01. DX contains the number of bytes written to memory.

The time of day interrupt(IRQ0) is disabled during the cassette-read operation.

Data Record Architecture

The write-block routine uses the following format to record a tape block onto a cassette tape.



Component	Description
Leader	256 Bytes (of All 1's)
Sync Bit	One 0 Bit
Sync Byte	ASCII Character Hex 16
Data Blocks	256 Bytes in Length
CRC	2 Bytes for each Data Block

Data Record Components

Error Recovery

Error recovery is handled through software. A CRC is used to detect errors. The polynomial used is $G(X) = X^{16} + X^{12} + X^5 + 1$, which is the polynomial used by the synchronous data link control interface. Essentially, as bits are written to or read from the cassette tape, they are passed through the CRC register in software. After a block of data is written, the complemented value of the calculated CRC register is written on the tape. On reading the cassette data, the CRC bytes are read and compared to the generated CRC value. If the read CRC does not equal the generated CRC, the processor's carry flag is set and the status of AH is set to hex 01, which indicates a CRC error has occurred. The routine is exited on a CRC error.

System BIOS Listing

Quick Reference

	Page	Line Number
System ROM BIOS		
Equates	5-30	12
8088 Interrupt Locations	5-30	34
Stack	5-30	66
Data Areas	5-30	74
Power-On Self-Test	5-33	229
Boot Strap Loader	5-49	1493
I/O Support		
Asynchronous Communications (RS-232C)	5-50	1551
Keyboard	5-54	1818
Diskette	5-64	2426
Printer	5-74	3201
Display	5-75	3327
System Configuration Analysis		
Memory Size Determination	5-101	5177
Equipment Determination	5-101	5208
Cassette I/O Support	5-102	5253
Graphics Character Generator	5-108	5769
Time of Day	5-110	5903
Print Screen	5-112	6077

```

1 $TITLE(BIOS FOR IBM PERSONAL COMPUTER)
2
3 -----
4 | THE BIOS ROUTINES ARE MEANT TO BE ACCESSED THROUGH |
5 | SOFTWARE INTERRUPTS ONLY. ANY ADDRESSES PRESENT IN |
6 | THE LISTINGS ARE INCLUDED ONLY FOR COMPLETENESS, |
7 | NOT FOR REFERENCE. APPLICATIONS WHICH REFERENCE |
8 | ABSOLUTE ADDRESSES WITHIN THE CODE SEGMENT |
9 | VIOLATE THE STRUCTURE AND DESIGN OF BIOS. |
10 -----
11
12 |-----|
13 | EQUATES |
14 |-----|
0060 15 PORT_A EQU 60H ; 8255 PORT A ADDR
0061 16 PORT_B EQU 61H ; 8255 PORT B ADDR
0062 17 PORT_C EQU 62H ; 8255 PORT C ADDR
0063 18 CMD_PORT EQU 63H
0020 19 INTA00 EQU 20H ; 8259 PORT
0021 20 INTA01 EQU 21H ; 8259 PORT
0020 21 EO1 EQU 20H
0040 22 TIMER EQU 40H
0043 23 TIM_CTL EQU 43H ; 8253 TIMER CONTROL PORT ADDR
0040 24 TIMERO EQU 40H ; 8253 TIMER/CENTER 0 PORT ADDR
0001 25 THINT EQU 01 ; TIMER 0 INTR RECVD MASK
0008 26 DMA08 EQU 08 ; DMA STATUS REG PORT ADDR
0000 27 DMA EQU 00 ; DMA CHANNEL 0 ADDR REG PORT ADDR
0540 28 MAX_PERIOD EQU 540H
0410 29 MIN_PERIOD EQU 410H
0060 30 KBD_IN EQU 60H ; KEYBOARD DATA IN ADDR PORT
0002 31 KBDINT EQU 02 ; KEYBOARD INTR MASK
0060 32 KB_DATA EQU 60H ; KEYBOARD SCAN CODE PORT
0061 33 KB_CTL EQU 61H ; CONTROL BITS FOR KB SENSE DATA
34
35 | 8088 INTERRUPT LOCATIONS |
36 |-----|
37 ABS0 SEGMENT AT 0
0000 38 STG_LOCO LABEL BYTE
0008 39 ORG 2*6
0008 40 NMI_PTR LABEL WORD
0014 41 ORG 5*6
0014 42 INT5_PTR LABEL WORD
0020 43 ORG 8*6
0020 44 INT_ADDR LABEL WORD
0020 45 INT_PTR LABEL DWORD
0040 46 ORG 10H*6
0040 47 VIDEO_INT LABEL WORD
0074 48 ORG 10H*6
0074 49 PARM_PTR LABEL DWORD ; POINTER TO VIDEO PARMS
0060 50 ORG 18H*6
0060 51 BASIC_PTR LABEL WORD ; ENTRY POINT FOR CASSETTE BASIC
0078 52 ORG 01EH*6 ; INTERRUPT IEH
0078 53 DISK_POINTER LABEL DWORD
007C 54 ORG 01FH*6 ; LOCATION OF POINTER
007C 55 EXT_PTR LABEL DWORD ; POINTER TO EXTENSION
0100 56 ORG 040H*4 ; ROUTINE
0100 57 IO_ROM_INIT DW ? ;
0102 58 IO_ROM_SEG DW ? ; OPTIONAL ROM SEGMENT
0400 59 ORG 400H
0400 60 DATA_AREA LABEL BYTE ; ABSOLUTE LOCATION OF DATA SEGMENT
0400 61 DATA_WORD LABEL WORD
7C00 62 ORG 7C00H
7C00 63 BOOT_LOCH LABEL FAR
---- 64 ABS0 ENDS
65
66 |-----|
67 | STACK -- USED DURING INITIALIZATION ONLY |
68 |-----|
---- 69 STACK SEGMENT AT 30H
0000 (128 70 DW 128 DUP(?)
????
)
0100 71 TOS LABEL WORD
---- 72 STACK ENDS
73
74 |-----|
75 | ROM BIOS DATA AREAS |
76 |-----|
---- 77 DATA SEGMENT AT 40H

```

```

LOC OBJ      LINE   SOURCE
0000 {4      78   RS232_BASE   DW    4 DUP(?)      ; ADDRESSES OF RS232 ADAPTERS
    ????)
    )
0008 {4      79   PRINTER_BASE DW    4 DUP(?)      ; ADDRESSES OF PRINTERS
    ????)
    )
0010 ?????   80   EQUIP_FLAG   DW    ?             ; INSTALLED HARDWARE
0012 ??      81   MFG_TST     DB    ?             ; INITIALIZATION FLAG
0013 ?????   82   MEMORY_SIZE DW    ?             ; MEMORY SIZE IN K BYTES
0015 ?????   83   IO_RAM_SIZE DW    ?             ; MEMORY IN I/O CHANNEL
0016 ?????   84   ;-----
0017 ??      85   ;           KEYBOARD DATA AREAS           ;
0018 ??      86   ;-----
0019 ??      87   KB_FLAG     DB    ?             ;
0020 ??      88   ;
0021 ??      89   ;----- SHIFT FLAG EQUATES WITHIN KB_FLAG
0022 ??      90   ;
0023 0080      91   INS_STATE   EQU    80H          ; INSERT STATE IS ACTIVE
0024 0040      92   CAPS_STATE  EQU    40H          ; CAPS LOCK STATE HAS BEEN TOGGLED
0025 0020      93   NUM_STATE   EQU    20H          ; NUM LOCK STATE HAS BEEN TOGGLED
0026 0010      94   SCROLL_STATE EQU    10H          ; SCROLL LOCK STATE HAS BEEN TOGGLED
0027 0008      95   ALT_SHIFT   EQU    08H          ; ALTERNATE SHIFT KEY DEPRESSED
0028 0004      96   CTL_SHIFT   EQU    04H          ; CONTROL SHIFT KEY DEPRESSED
0029 0002      97   LEFT_SHIFT  EQU    02H          ; LEFT SHIFT KEY DEPRESSED
0030 0001      98   RIGHT_SHIFT EQU    01H          ; RIGHT SHIFT KEY DEPRESSED
0031 0018 ??   100  KB_FLAG_1   DB    ?             ; SECOND BYTE OF KEYBOARD STATUS
0032 0010      101  ;
0033 0080      102  INS_SHIFT   EQU    80H          ; INSERT KEY IS DEPRESSED
0034 0040      103  CAPS_SHIFT  EQU    40H          ; CAPS LOCK KEY IS DEPRESSED
0035 0020      104  NUM_SHIFT   EQU    20H          ; NUM LOCK KEY IS DEPRESSED
0036 0010      105  SCROLL_SHIFT EQU    10H          ; SCROLL LOCK KEY IS DEPRESSED
0037 0008      106  HOLD_STATE  EQU    08H          ; SUSPEND KEY HAS BEEN TOGGLED
0038 0019 ??   107  ;
0039 001A ????? 108  ALT_INPUT   DB    ?             ; STORAGE FOR ALTERNATE KEYPAD ENTRY
0040 001C ????? 109  BUFFER_HEAD DW    ?             ; POINTER TO HEAD OF KEYBOARD BUFFER
0041 001E {16   110  BUFFER_TAIL DW    ?             ; POINTER TO TAIL OF KEYBOARD BUFFER
    ????)
    )
0042 003E      111  KB_BUFFER   DW    16 DUP(?)      ; ROOM FOR 15 ENTRIES
0043 003E      112  KB_BUFFER_END LABEL WORD
0044 003E      113  ;
0045 0045      114  ;----- HEAD = TAIL INDICATES THAT THE BUFFER IS EMPTY
0046 0046      115  ;
0047 0045      116  NUM_KEY     EQU    69          ; SCAN CODE FOR NUMBER LOCK
0048 0046      117  SCROLL_KEY  EQU    70          ; SCROLL LOCK KEY
0049 0036      118  ALT_KEY     EQU    56          ; ALTERNATE SHIFT KEY SCAN CODE
0050 001D      119  CTL_KEY     EQU    29          ; SCAN CODE FOR CONTROL KEY
0051 003A      120  CAPS_KEY    EQU    58          ; SCAN CODE FOR SHIFT LOCK
0052 002A      121  LEFT_KEY    EQU    42          ; SCAN CODE FOR LEFT SHIFT
0053 0036      122  RIGHT_KEY   EQU    54          ; SCAN CODE FOR RIGHT SHIFT
0054 0052      123  INS_KEY     EQU    82          ; SCAN CODE FOR INSERT KEY
0055 0053      124  DEL_KEY     EQU    83          ; SCAN CODE FOR DELETE KEY
0056 0055      125  ;
0057 0056      126  ;-----
0058 0056      127  ;           DISKETTE DATA AREAS           ;
0059 0056      128  ;-----
0060 003E ??   129  SEEK_STATUS DB    ?             ; DRIVE RECALIBRATION STATUS
0061 003E ??   130  ;           BIT 3-0 = DRIVE 3-0 NEEDS RECAL BEFORE
0062 003E ??   131  ;           NEXT SEEK IF BIT IS = 0
0063 0080      132  INT_FLAG    EQU    080H        ; INTERRUPT OCCURRENCE FLAG
0064 003F ??   133  MOTOR_STATUS DB    ?             ; MOTOR STATUS
0065 003F ??   134  ;           BIT 3-0 = DRIVE 3-0 IS CURRENTLY RUNNING
0066 003F ??   135  ;           BIT 7 = CURRENT OP IS A WRITE, REQUIRES DELAY
0067 0040 ??   136  MOTOR_COUNT DB    ?             ; TIME OUT COUNTER FOR DRIVE TURN OFF
0068 0025      137  MOTOR_WAIT  EQU    37          ; TWO SEC OF COUNT FOR MOTOR TURN OFF
0069 0041 ??   138  ;
0070 0080      139  DISKETTE_STATUS DB    ?             ; BYTE OF RETURN CODE INFO FOR STATUS
0071 0040      140  TIME_OUT    EQU    80H          ; ATTACHMENT FAILED TO RESPOND
0072 0040      141  BAD_SEEK    EQU    40H          ; SEEK OPERATION FAILED
0073 0020      142  BAD_NEC     EQU    20H          ; NEC CONTROLLER HAS FAILED
0074 0010      143  BAD_CRC     EQU    10H          ; BAD CRC ON DISKETTE READ
0075 0009      144  DMA_BOUNDARY EQU    09H          ; ATTEMPT TO DMA ACROSS 64K BOUNDARY
0076 0008      145  BAD_DMA     EQU    08H          ; DMA OVERRUN ON OPERATION
0077 0004      146  RECORD_NOT_FND EQU    04H          ; REQUESTED SECTOR NOT FOUND
0078 0003      147  WRITE_PROTECT EQU    03H          ; WRITE ATTEMPTED ON WRITE PROT DISK
0079 0002      148  BAD_ADDR_MARK EQU    02H          ; ADDRESS MARK NOT FOUND

```



```

LOC OBJ          LINE  SOURCE
0001             149  BAD_CMD      EQU    01H      ; BAD COMMAND PASSED TO DISKETTE I/O
                  150
0042 ( ?         151  NEC_STATUS   DB      7 DUP(?) ; STATUS BYTES FROM NEC
  ??
  )
                  152
                  153 ;-----
                  154 ; VIDEO DISPLAY DATA AREA :
                  155 ;-----
0049 ??          156  CRT_MODE    DB      ?      ; CURRENT CRT MODE
004A ????        157  CRT_COLS    DW      ?      ; NUMBER OF COLUMNS ON SCREEN
004C ????        158  CRT_LEN     DW      ?      ; LENGTH OF REGEN IN BYTES
004E ????        159  CRT_START   DW      ?      ; STARTING ADDRESS IN REGEN BUFFER
0050 ( 8         160  CURSOR_POSN DW      8 DUP(?) ; CURSOR FOR EACH OF UP TO 8 PAGES
  ????
  )
0060 ????        161  CURSOR_MODE DW      ?      ; CURRENT CURSOR MODE SETTING
0062 ??          162  ACTIVE_PAGE DB      ?      ; CURRENT PAGE BEING DISPLAYED
0063 ????        163  ADDR_6845   DW      ?      ; BASE ADDRESS FOR ACTIVE DISPLAY CARD
0065 ??          164  CRT_MODE_SET DB      ?      ; CURRENT SETTING OF THE 3x8 REGISTER
0066 ??          165  CRT_PALETTE DB      ?      ; CURRENT PALETTE SETTING COLOR CARD
                  166
                  167 ;-----
                  168 ; CASSETTE DATA AREA :
                  169 ;-----
0067 ????        170  EDGE_CNT    DW      ?      ; TIME COUNT AT DATA EDGE
0069 ????        171  CRC_REG     DW      ?      ; CRC REGISTER
006B ??          172  LAST_VAL    DB      ?      ; LAST INPUT VALUE
                  173
                  174 ;-----
                  175 ; TIMER DATA AREA :
                  176 ;-----
006C ????        177  TIMER_LOW   DW      ?      ; LOW WORD OF TIMER COUNT
006E ????        178  TIMER_HIGH  DW      ?      ; HIGH WORD OF TIMER COUNT
0070 ??          179  TIMER_OFL   DB      ?      ; TIMER HAS ROLLED OVER SINCE LAST READ
                  180 ;COUNTS_SEC EQU    18
                  181 ;COUNTS_MIN EQU    1092
                  182 ;COUNTS_HOUR EQU   65543
                  183 ;COUNTS_DAY EQU    1573040 = 1800B0H
                  184
                  185 ;-----
                  186 ; SYSTEM DATA AREA :
                  187 ;-----
0071 ??          188  BIOS_BREAK  DB      ?      ; BIT 7 = 1 IF BREAK KEY WAS DEPRESSED
0072 ????        189  RESET_FLAG  DW      ?      ; WORD = 1234H IF KB RESET UNDERWAY
                  190
                  191 ;-----
                  192 ; FIXED DISK DATA AREA :
                  193 ;-----
0074 ????        193             DW      ?      ;
0076 ????        194             DW      ?      ;
                  195 ;-----
                  196 ; PRINTER AND RS232 TIMEOUT CTPS :
                  197 ;-----
0078 ( 4         198  PRINT_TIM_OUT DB      4 DUP(?) ; PRINTER TIME OUT COUNTER
  ??
  )
007C ( 4         199  RS232_TIM_OUT DB      4 DUP(?) ; RS232 TIME OUT COUNTER
  ??
  )
                  200 ;-----
                  201 ; EXTRA KEYBOARD DATA AREA :
                  202 ;-----
0080 ????        203  BUFFER_START DW      ?      ;
0082 ????        204  BUFFER_END   DW      ?      ;
-----         205  DATA ENDS
                  206 ;-----
                  207 ; EXTRA DATA AREA :
                  208 ;-----
----           209  XXDATA SEGMENT AT 50H
0000 ??         210  STATUS_BYTE DB      ?      ;
-----         211  XXDATA ENDS
                  212
                  213 ;-----
                  214 ; VIDEO DISPLAY BUFFER :
                  215 ;-----
----           216  VIDEO_RAM SEGMENT AT 0B800H

```

```

LOC OBJ          LINE  SOURCE
0000             217  REGEN          LABEL  BYTE
0000             218  REGENM        LABEL  WORD
0000 (16384      219  DB            DB      16384 DUP(?)
    )
    ??
    )
----            220  VIDEO_RAM     ENDS
221             ;-----
222             ;          ROM RESIDENT CODE          :
223             ;-----
----            224  CODE          SEGMENT AT 0F000H
0000 (57344      225  DB            DB      57344 DUP(?)          ; FILL LOWEST 56K
    )
    ??
    )
0000 31353031343736 226
20434F5052E20      227  DB            DB      '1501476 COPR. IBM 1951'          ; COPYRIGHT NOTICE
49424D20313936
32
228
229             ;-----
230             ;          INITIAL RELIABILITY TESTS -- PHASE 1          :
231             ;-----
232             ;          ASSUME      CS:CODE,SS:CODE,ES:ABS0,DS:DATA
233             ;-----
234             ;          DATA DEFINITIONS          :
235             ;-----
E016 D1E0        236  C1      DW      C11          ; RETURN ADDRESS
237
238             ;-----
239             ;          THIS SUBROUTINE PERFORMS A READ/WRITE STORAGE TEST ON          :
240             ;          A 16K BLOCK OF STORAGE.          :
241             ; ENTRY REQUIREMENTS:          :
242             ;     ES = ADDRESS OF STORAGE SEGMENT BEING TESTED          :
243             ;     DS = ADDRESS OF STORAGE SEGMENT BEING TESTED          :
244             ;     WHEN ENTERING AT STGTST_CNT, CX MUST BE LOADED WITH          :
245             ;     THE BYTE COUNT.          :
246             ; EXIT PARAMETERS:          :
247             ;     ZERO FLAG = 0 IF STORAGE ERROR (DATA COMPARE OR PARITY CHECK.          :
248             ;     AL = 0 DENOTES A PARITY CHECK. ELSE AL=XOR'ED BIT          :
249             ;     PATTERN OF THE EXPECTED DATA PATTERN VS THE          :
250             ;     ACTUAL DATA READ.          :
251             ;     AX,BX,CX,DX,DI, AND SI ARE ALL DESTROYED.          :
252             ;-----
253
E018             254  STGTST  PROC   NEAR
E018 B90040        255  MOV     CX,4000H          ; SETUP CNT TO TEST A 16K BLK
E018             256  STGTST_CNT:
E018 FC           257  CLD                     ; SET DIR FLAG TO INCREMENT
E01C 8BD9        258  MOV     BX,CX            ; SAVE BYTE CNT (4K FOR VIDEO OR 16K)
E01E B8AAAA      259  MOV     AX,0AAAAH        ; GET DATA PATTERN TO WRITE
E021 BA55FF      260  MOV     DX,0FF55H        ; SETUP OTHER DATA PATTERNS TO USE
E024 2BFF       261  SUB     DI,DI            ; DI = OFFSET 0 RELATIVE TO ES REG
E026 F3         262  REP     STOSB           ; WRITE STORAGE LOCATIONS
E027 AA
E028             263  C3:                     ; STG01
E028 4F         264  DEC     DI              ; POINT TO LAST BYTE JUST WRITTEN
E029 FD         265  STD                     ; SET DIR FLAG TO GO BACKWARDS
E02A             266  C4:
E02A 8BF7       267  MOV     SI,DI
E02C 8BCB       268  MOV     CX,BX
E02E             269  C5:                     ; SETUP BYTE CNT
E02E AC         270  LODSB                    ; INNER TEST LOOP
E02F 32C4       271  XOR     AL,AH            ; READ OLD TST BYTE FROM STORAGE [DI]+
E031 7525       272  JNE     C7              ; DATA READ AS EXPECTED ?
E033 8AC2       273  MOV     AL,DL           ; NO - GO TO ERROR ROUTINE
E035 AA         274  STOSB                    ; GET NEXT DATA PATTERN TO WRITE
E036 E2F6       275  LOOP   C5              ; WRITE INTO LOCATION JUST READ [DI]+
276             ; DECREMENT BYTE COUNT AND LOOP CX
E038 22E4       277  AND     AH,AH           ; ENDING ZERO PATTERN WRITTEN TO STG ?
E03A 7416       278  JZ      C6X            ; YES - RETURN TO CALLER WITH AL=0
E03C 8AE0       279  MOV     AH,AL           ; SETUP NEW VALUE FOR COMPARE
E03E 86F2       280  XCHG   DH,DL           ; MOVE NEXT DATA PATTERN TO DL
E040 22E4       281  AND     AH,AH           ; READING ZERO PATTERN THIS PASS ?
E042 7504       282  JNZ     C6              ; CONTINUE TEST SEQUENCE TILL ZERO DATA
E044 8AD4       283  MOV     DL,AH           ; ELSE SET ZERO FOR END READ PATTERN
E046 EBE0       284  JMP     C3              ; AND MAKE FINAL BACKWARDS PASS
E048             285  C6:

```

```

LOC OBJ          LINE  SOURCE
E048 FC          286      CLD                ; SET DIR FLAG TO GO FORWARD
E049 47          287      INC  DI                ; SET POINTER TO BEG LOCATION
E04A 74DE        288      JZ   C4                ; READ/WRITE FORWARD IN STG
E04C 4F          289      DEC  DI                ; ADJUST POINTER
E04D BA0100      290      MOV  DX,00001H        ; SETUP 01 FOR PARITY BIT
291              291              ; AND 00 FOR END
E050 EBD6        292      JMP  C3                ; READ/WRITE BACKWARD IN STG
E052              293      C6X:
E052 E462        294      IN   AL,PORT_C        ; DID A PARITY ERROR OCCUR ?
E054 24C0        295      AND  AL,0C0H          ; ZERO FLAG WILL BE OFF PARITY ERROR
E056 B000        296      MOV  AL,000H          ; AL=0 DATA COMPARE OK
E058              297      C7:
E058 FC          298      CLD                ; SET DEFAULT DIRCTN FLAG BACK TO INC
E059 C3          299      RET
300      STG$T$  ENDP
301      ;-----
302      ;      8088 PROCESSOR TEST
303      ; DESCRIPTION
304      ; VERIFY 8088 FLAGS, REGISTERS AND CONDITIONAL JUMPS
305      ;-----
306      ASSUME CS:CODE,DS:NOTHING,ES:NOTHING,SS:NOTHING
E05B              307      ORG  0E05BH
E05B              308      RESET LABEL FAR
E05B              309      START:
E05B FA          310      CLI                ; DISABLE INTERRUPTS
E05C B405        311      MOV  AH,0D5H         ; SET SF, CF, ZF, AND AF FLAGS ON
E05E 9E          312      SAHF
E05F 734C        313      JNC  ERR01          ; GO TO ERR ROUTINE IF CF NOT SET
E061 754A        314      JNZ  ERR01          ; GO TO ERR ROUTINE IF ZF NOT SET
E063 7B48        315      JNP  ERR01          ; GO TO ERR ROUTINE IF PF NOT SET
E065 7946        316      JNS  ERR01          ; GO TO ERR ROUTINE IF SF NOT SET
E067 9F          317      LAHF                ; LOAD FLAG IMAGE TO AH
E068 B105        318      MOV  CL,5            ; LOAD CNT REG WITH SHIFT CNT
E06A D2EC        319      SHR  AH,CL          ; SHIFT AF INTO CARRY BIT POS
E06C 733F        320      JNC  ERR01          ; GO TO ERR ROUTINE IF AF NOT SET
E06E B040        321      MOV  AL,40H         ; SET THE OF FLAG ON
E070 D0E0        322      SHL  AL,1           ; SETUP FOR TESTING
E072 7139        323      JNO  ERR01          ; GO TO ERR ROUTINE IF OF NOT SET
E074 32E4        324      XOR  AH,AH          ; SET AH = 0
E076 9E          325      SAHF                ; CLEAR SF, CF, ZF, AND PF
E077 7634        326      JBE  ERR01          ; GO TO ERR ROUTINE IF CF ON
327              327              ; OR TO TO ERR ROUTINE IF ZF ON
E079 7832        328      JS   ERR01          ; GO TO ERR ROUTINE IF SF ON
E07B 7A30        329      JP  ERR01          ; GO TO ERR ROUTINE IF PF ON
E07D 9F          330      LAHF                ; LOAD FLAG IMAGE TO AH
E07E B105        331      MOV  CL,5            ; LOAD CNT REG WITH SHIFT CNT
E080 D2EC        332      SHR  AH,CL          ; SHIFT 'AF' INTO CARRY BIT POS
E082 7229        333      JC   ERR01          ; GO TO ERR ROUTINE IF ON
E084 D0E4        334      SHL  AH,1           ; CHECK THAT 'OF' IS CLEAR
E086 7025        335      JO   ERR01          ; GO TO ERR ROUTINE IF ON
336
337      ;----- READ/WRITE THE 8088 GENERAL AND SEGMENTATION REGISTERS
338      ;      WITH ALL ONE'S AND ZEROES'S.
339
E088 B0FFFF      340      MOV  AX,0FFFFH       ; SETUP ONE'S PATTERN IN AX
E08B F9          341      STC
E08C              342      C8:
E08C 8ED8        343      MOV  DS,AX           ; WRITE PATTERN TO ALL REGS
E08E 8CDB        344      MOV  BX,DS
E090 8EC3        345      MOV  ES,BX
E092 8CC1        346      MOV  CX,ES
E094 8ED1        347      MOV  SS,CX
E096 8CD2        348      MOV  DX,SS
E098 8BE2        349      MOV  SP,DX
E09A 8BEC        350      MOV  BP,SP
E09C 8BF5        351      MOV  SI,BP
E09E 8BF0        352      MOV  DI,SI
E0A0 7307        353      JNC  C9              ; TSTIA
E0A2 33C7        354      XOR  AX,DI           ; PATTERN MAKE IT THRU ALL REGS
E0A4 7507        355      JNZ  ERR01          ; NO - GO TO ERR ROUTINE
E0A6 F8          356      CLC
E0A7 EBE3        357      JMP  C8
E0A9              358      C9:
E0A9 0BC7        359      OR   AX,DI           ; ZERO PATTERN MAKE IT THRU?
E0AB 7401        360      JZ   C10            ; YES - GO TO NEXT TEST
E0AD F4          361      ERR01: HLT          ; HALT SYSTEM
362      ;-----

```

```

363 ; ROS CHECKSUM TEST I ;
364 ; DESCRIPTION ;
365 ; A CHECKSUM IS DONE FOR THE 8K ROS MODULE ;
366 ; CONTAINING POD AND BIOS. ;
367 ;-----;
E0AE C10:
368 ;
369 ; ZERO IN AL ALREADY ;
E0AE E6A0 370 OUT 0A0H,AL ; DISABLE NMI INTERRUPTS
E0B0 E683 371 OUT 83H,AL ; INITIALIZE DMA PAGE REG
E0B2 BAD803 372 MOV DX,308H
E0B5 EE 373 OUT DX,AL ; DISABLE COLOR VIDEO
E0B6 FEC0 374 INC AL
E0B8 B2B8 375 MOV DL,0B8H
E0BA EE 376 OUT DX,AL ; DISABLE B/W VIDEO,EN HIGH RES
E0BB B099 377 MOV AL,99H ; SET 8255 A,C-INPUT,B-OUTPUT
E0BD E663 378 OUT CMD_PORT,AL ; WRITE 8255 CMD/MODE REG
E0BF B0FC 379 MOV AL,0FCH ; DISABLE PARITY CHECKERS AND
E0C1 E661 380 OUT PORT_B,AL ; GATE SHS SHS,CASS MOTOR OFF
E0C3 8CC8 381 MOV AX,CS ; SETUP SS SEG REG
E0C5 8ED0 382 MOV SS,AX
E0C7 8ED8 383 MOV DS,AX ; SET UP DATA SEG TO POINT TO
384 ; ROM ADDRESS
385
386 ASSUME SS:CODE
E0C9 B7E0 386 MOV BH,0E0H ; SETUP STARTING ROS ADDR (E0000)
E0CB BC16E0 387 MOV SP,OFFSET C1 ; SETUP RETURN ADDRESS
E0CE E97B0B 388 JMP ROS_CHECKSUM
E0D1 389 C11:
E0D1 75DA 390 JNE ERR01 ; HALT SYSTEM IF ERROR
391 ;-----;
392 ; 8237 DMA INITIALIZATION CHANNEL REGISTER TEST ;
393 ; DESCRIPTION ;
394 ; DISABLE THE 8237 DMA CONTROLLER. VERIFY THAT TIMER 1 ;
395 ; FUNCTIONS OK. WRITE/READ THE CURRENT ADDRESS AND WORD ;
396 ; COUNT REGISTERS FOR ALL CHANNELS. INITIALIZE AND ;
397 ; START DMA FOR MEMORY REFRESH. ;
398 ;-----;
E0D3 B004 399 MOV AL,04 ; DISABLE DMA CONTROLLER
E0D5 E608 400 OUT DMA08,AL
401
402 ;---- VERIFY THAT TIMER 1 FUNCTIONS OK
403
E0D7 B054 404 MOV AL,54H ; SEL TIMER 1,LSB,MODE 2
E0D9 E643 405 OUT TIMER+3,AL
E0DB 8AC1 406 MOV AL,CL ; SET INITIAL TIMER CNT TO 0
E0DD E641 407 OUT TIMER+1,AL
E0DF C12: ; TIMER1_BITS_ON
E0DF B040 408 MOV AL,40H ; LATCH TIMER 1 COUNT
E0E1 E643 410 OUT TIMER+3,AL
E0E3 80FBFF 411 CMP BL,OFFH ; YES - SEE IF ALL BITS GO OFF
E0E6 7407 412 JE C13 ; TIMER1_BITS_OFF
E0E8 E441 413 IN AL,TIMER+1 ; READ TIMER 1 COUNT
E0EA 0AD8 414 OR BL,AL ; ALL BITS ON IN TIMER
E0EC E2F1 415 LOOP C12 ; TIMER1_BITS_ON
E0EE F4 416 HLT ; TIMER 1 FAILURE, HALT SYS
E0EF C13: ; TIMER1_BITS_OFF
E0EF 8AC3 418 MOV AL,BL ; SET TIMER 1 CNT
E0F1 2BC9 419 SUB CX,CX
E0F3 E641 420 OUT TIMER+1,AL
E0F5 C14: ; TIMER_LOOP
E0F5 B040 422 MOV AL,40H ; LATCH TIMER 1 COUNT
E0F7 E643 423 OUT TIMER+3,AL
E0F9 90 424 NOP ; DELAY FOR TIMER
E0FA 90 425 NOP
E0FB E441 426 IN AL,TIMER+1 ; READ TIMER 1 COUNT
E0FD 22D8 427 AND BL,AL
E0FF 7403 428 JZ C15 ; GO TO WRAP_DMA_REG
E101 E2F2 429 LOOP C14 ; TIMER_LOOP
E103 F4 430 HLT ; TIMER ERROR - HALT SYSTEM
431
432 ;---- INITIALIZE TIMER 1 TO REFRESH MEMORY
433
E104 434 C15: ; WRAP_DMA_REG
E104 B012 435 MOV AL,18 ; SETUP DIVISOR FOR REFRESH
E106 E641 436 OUT TIMER+1,AL ; WRITE TIMER 1 CNT REG
E108 E60D 437 OUT DMA+0DH,AL ; SEND MASTER CLEAR TO DMA
438

```

```

LOC OBJ          LINE  SOURCE
439             ;----- WRAP DMA CHANNELS ADDRESS AND COUNT REGISTERS
440
E10A B0FF       441             MOV     AL,0FFH             ; WRITE PATTERN FF TO ALL REGS
E10C            442 C16:
E10C 8AD6       443             MOV     BL,AL             ; SAVE PATTERN FOR COMPARE
E10E 8AF8       444             MOV     BH,AL
E110 B90800     445             MOV     CX,8             ; SETUP LOOP CNT
E113 2BD2       446             SUB     DX,DX             ; SETUP I/O PORT ADDR OF REG (0000)
E115            447 C17:
E115 EE        448             OUT     DX,AL             ; WRITE PATTERN TO REG, LSB
E116 50        449             PUSH   AX
E117 EE        450             OUT     DX,AL             ; HSB OF 16 BIT REG
E118 B80101     451             MOV     AX,0101H         ; AX TO ANOTHER PAT BEFORE RD
E11B EC        452             IN      AL,DX            ; READ 16-BIT DMA CH REG, LSB
E11C 8AE0       453             MOV     AH,AL            ; SAVE LSB OF 16-BIT REG
E11E EC        454             IN      AL,DX            ; READ HSB OF DMA CH REG
E11F 3BD8       455             CHP    BX,AX             ; PATTERN READ AS WRITTEN?
E121 7401       456             JE     C18                ; YES - CHECK NEXT REG
E123 F4        457             HLT
E124            458 C18:
E124 42        459             INC     DX                ; SET I/O PORT TO NEXT CH REG
E125 E2EE       460             LOOP   C17                ; WRITE PATTERN TO NEXT REG
E127 FEC0       461             INC     AL                ; SET PATTERN TO 0
E129 74E1       462             JZ     C16                ; WRITE TO CHANNEL REGS
463
464             ;----- INITIALIZE AND START DMA FOR MEMORY REFRESH.
465
E12B 8EDB       466             MOV     DS,BX             ; SET UP ABS0 INTO DS AND ES
E12D 8EC3       467             MOV     ES,BX
468             ASSUME DS:ABS0,ES:ABS0
469
E12F B0FF       470             MOV     AL,0FFH         ; SET CNT OF 64K FOR RAM REFRESH
E131 E601       471             OUT     DMA+1,AL
E133 50        472             PUSH   AX
E134 E601       473             OUT     DMA+1,AL
E136 B20B       474             MOV     DL,0BH           ; DX=000B
E138 B058       475             MOV     AL,058H         ; SET DMA MODE,CH 0,READ,AUTOINT
E13A EE        476             OUT     DX,AL            ; WRITE DMA MODE REG
E13B B000       477             MOV     AL,0             ; ENABLE DMA CONTROLLER
E13D E608       478             OUT     DMA+8,AL        ; SETUP DMA COMMAND REG
E13F 50        479             PUSH   AX
E140 E60A       480             OUT     DMA+10,AL       ; ENABLE CHANNEL 0 FOR REFRESH
E142 B103       481             MOV     CL,3
E144 B041       482             MOV     AL,41H           ; SET MODE FOR CHANNEL 1
E146            483 C18A:
E146 EE        484             OUT     DX,AL
E147 FEC0       485             INC     AL                ; POINT TO NEXT CHANNEL
E149 E2FB       486             LOOP   C18A
487
488             ;-----
489             ; BASE 16K READ/WRITE STORAGE TEST :
490             ; DESCRIPTION :
491             ; WRITE/READ/VERIFY DATA PATTERNS FF,55,AA,01, AND 00 :
492             ; TO 1ST 16K OF STORAGE. VERIFY STORAGE ADDRESSABILITY. :
493             ; INITIALIZE THE 8259 INTERRUPT CONTROLLER CHIP FOR :
494             ; CHECKING MANUFACTURING TEST 2 MODE. :
495             ;-----
496
497             ;----- DETERMINE MEMORY SIZE AND FILL MEMORY WITH DATA
498
E14B BA1302     498             MOV     DX,0213H         ; ENABLE EXPANSION BOX
E14E B001       499             MOV     AL,01H
E150 EE        500             OUT     DX,AL
E151 8B2E7204   501             MOV     BP,DATA_WORD[OFFSET RESET_FLAG] ; SAVE 'RESET_FLAG' IN BP
E155 81FD3412   502             CMP     BP,1234H         ; WARM START?
E159 740A       503             JE     C18B                ; BYPASS STG TST.
E15B BC41F090   504             MOV     SP,OFFSET C2
E15F E9B6FE     505             JMP     STGTST
E162            506 C24:
E162 7401       507             JE     C18B                ; PROCEED IF STGTST OK
E164 F4        508             HLT                        ; HALT IF NOT
E165            509 C18B:
E165 2BFF       510             SUB     DI,DI
E167 E460       511             IN      AL,PORT_A         ; DETERMINE BASE RAM SIZE
E169 240C       512             AND     AL,0CH            ; ISOLATE RAM SIZE SMS
E16B 0404       513             ADD     AL,4              ; CALCULATE MEMORY SIZE
E16D B10C       514             MOV     CL,12

```

LOC OBJ	LINE	SOURCE
E16F D3E0	515	SHL AX, CL
E171 8BC8	516	MOV CX, AX
E173 FC	517	CLD ; SET DIR FLAG TO INCR
E174	518	C19:
E174 AA	519	STOSB ; FILL BASE RAM WITH DATA
E175 E2FD	520	LOOP C19 ; LOOP TIL ALL ZERO
E177 892E7204	521	MOV DATA_WORD(OFFSET RESET_FLAG),BP
	522	
	523	;----- DETERMINE IO CHANNEL RAM SIZE
	524	
E17B B0F8	525	MOV AL,0F8H ; ENABLE SWITCH 5
E17D E661	526	OUT PORT_B,AL
E17F E462	527	IN AL,PORT_C ; READ SWITCHES
E181 2401	528	AND AL,0000001B ; ISOLATE SWITCH 5
E183 B10C	529	MOV CL,12D
E185 D3C0	530	ROL AX,CL
E187 B0FC	531	MOV AL,0FCH ; DISABLE SW. 5
E189 E661	532	OUT PORT_B,AL
E18B E462	533	IN AL,PORT_C
E18D 240F	534	AND AL,0FH
E18F 0AC4	535	OR AL,AH ; COMBINE SWITCH VALUES
E191 8AD8	536	MOV BL,AL ; SAVE
E193 B420	537	MOV AH,32
E195 F6E4	538	MUL AH ; CALC. LENGTH
E197 A31504	539	MOV DATA_WORD(OFFSET IO_RAM_SIZE),AX ;SAVE IT
E19A 7418	540	JZ C21
E19C BA0010	541	MOV DX,1000H ; SEGMENT FOR I/O RAM
E19F 8AE0	542	MOV AH,AL
E1A1 B000	543	MOV AL,0
E1A3	544	C20: ; FILL_IO:
E1A3 8EC2	545	MOV ES,DX
E1A5 B90080	546	MOV CX,8000H ; FILL 32K BYTES
E1A8 2BFF	547	SUB DI,DI
E1AA F3	548	REP STOSB
E1AB AA		
E1AC 81C20008	549	ADD DX,800H ; NEXT SEGMENT VALUE
E1B0 FECB	550	DEC BL
E1B2 75EF	551	JNZ C20 ; FILL_IO
	552	;
	553	; INITIALIZE THE 8259 INTERRUPT CONTROLLER CHIP :
	554	;
	555	C21:
E1B4	556	
E1B4 B013	556	MOV AL,13H ; ICW1 - EDGE, SNGL, ICW4
E1B6 E620	557	OUT INTA00,AL
E1B8 B008	558	MOV AL,8 ; SETUP ICW2 - INT TYPE 8 (8-F)
E1BA E621	559	OUT INTA01,AL
E1BC B009	560	MOV AL,9 ; SETUP ICW4 - BUFFRD,0086 MODE
E1BE E621	561	OUT INTA01,AL
E1C0 2BC0	562	SUB AX,AX ; POINT ES TO BEGIN
E1C2 8EC0	563	MOV ES,AX ; OF R/W STORAGE
	564	;
	565	; CHECK FOR MANUFACTURING TEST 2 TO LOAD TEST PROGRAMS FROM KEYBOARD.:
	566	;
	567	
	568	;----- SETUP STACK SEG AND SP
	569	
E1C4 B83000	570	MOV AX,STACK ; GET STACK VALUE
E1C7 8ED0	571	MOV SS,AX ; SET THE STACK UP
E1C9 BC0001	572	MOV SP,OFFSET TOS ; STACK IS READY TO GO
E1CC 81FD3412	573	CHP BP,1234H ; RESET_FLAG SET?
E1D0 7425	574	JE C25 ; YES - SKIP MFG TEST
E1D2 2BFF	575	SUB DI,DI
E1D4 8EDF	576	MOV DS, DI
E1D6 BB2400	577	MOV BX, 24H
E1D9 C70747FF	578	MOV WORD PTR [BX],OFFSET D11 ; SET UP KB INTERRUPT
E1DD 43	579	INC BX
E1DE 43	580	INC BX
E1DF 8C0F	581	MOV [BX],CS
E1E1 E85F04	582	CALL KBD_RESET ; READ IN KB RESET CODE TO BL
E1E4 80FB65	583	CHP BL,065H ; IS THIS MANUFACTURING TEST 2?
E1E7 750E	584	JNZ C25 ; JUMP IF NOT MAN. TEST
E1E9 B2FF	585	MOV DL,255 ; READ IN TEST PROGRAM
E1EB	586	C22:
E1EB E86204	587	CALL SP_TEST
E1EE 8AC3	588	MOV AL,BL
E1F0 AA	589	STOSB

LOC OBJ	LINE	SOURCE
E1F1 FECA	590	DEC DL
E1F3 75F6	591	JNZ C22 ; JUMP IF NOT DONE YET
E1F5 CD3E	592	INT 3EH ; SET INTERRUPT TYPE 62 ADDRESS F6H
E1F7	593	C25:
	594	
	595	;----- SET UP THE BIOS INTERRUPT VECTORS TO TEMP INTERRUPT
	596	
E1F7 B92000	597	MOV CX,32 ; FILL ALL 32 INTERRUPTS
E1FA 28FF	598	SUB DI,DI ; FIRST INTERRUPT LOCATOIN
E1FC	599	D3:
E1FC B847FF	600	MOV AX,OFFSET D11 ; MOVE ADDR OF INTR PROC TO TBL
E1FF AB	601	STOSW
E200 8CC8	602	MOV AX,CS ; GET ADDR OF INTR PROC SEG
E202 AB	603	STOSW
E203 E2F7	604	LOOP D3 ; VECTBLO
	605	
	606	;----- SET UP OTHER INTERRUPTS AS NECESSARY
	607	
E205 C7060800C3E2	608	MOV NMI_PTR,OFFSET NMI_INT ; NMI INTERRUPT
E20B C706140054FF	609	MOV INT5_PTR,OFFSET PRINT_SCREEN ; PRINT SCREEN
E211 C706620000F6	610	MOV BASIC_PTR+2,0F600H ; SEGMENT FOR CASSETTE BASIC
	611	
	612	;-----
	613	; 8259 INTERRUPT CONTROLLER TEST ;
	614	; DESCRIPTION ;
	615	; READ/WRITE THE INTERRUPT MASK REGISTER (IMR) WITH ALL ;
	616	; ONES AND ZEROES. ENABLE SYSTEM INTERRUPTS. MASK DEVICE ;
	617	; INTERRUPTS OFF. CHECK FOR HOT INTERRUPTS (UNEXPECTED). ;
	618	;-----
	619	
	620	;----- TEST THE IMR REGISTER
	621	
E217 BA2100	622	MOV DX,0021H ; POINT INTR. CHIP ADDR 21
E21A B000	623	MOV AL,0 ; SET IMR TO ZERO
E21C EE	624	OUT DX,AL
E21D EC	625	IN AL,DX ; READ IMR
E21E 0AC0	626	OR AL,AL ; IMR = 0?
E220 7515	627	JNZ D6 ; GO TO ERR ROUTINE IF NOT 0
E222 B0FF	628	MOV AL,0FFH ; DISABLE DEVICE INTERRUPTS
E224 EE	629	OUT DX,AL ; WRITE TO IMR
E225 EC	630	IN AL,DX ; READ IMR
E226 0401	631	ADD AL,1 ; ALL IMR BIT ON?
E228 7500	632	JNZ D6 ; NO - GO TO ERR ROUTINE
	633	
	634	;----- CHECK FOR HOT INTERRUPTS
	635	
	636	;----- INTERRUPTS ARE MASKED OFF. CHECK THAT NO INTERRUPTS OCCUR.
	637	
E22A 32E4	638	XOR AH,AH ; CLEAR AH REG
E22C FB	639	STI ; ENABLE EXTERNAL INTERRUPTS
E22D 2BC9	640	SUB CX,CX ; WAIT 1 SEC FOR ANY INTRs THAT
E22F	641	D4:
E22F E2FE	642	LOOP D4 ; MIGHT OCCUR
E231	643	D5:
E231 E2FE	644	LOOP D5
E233 0AE4	645	OR AH,AH ; DID ANY INTERRUPTS OCCUR?
E235 7408	646	JZ D7 ; NO - GO TO NEXT TEST
E237	647	D6:
E237 BA0101	648	MOV DX,101H ; BEEP SPEAKER IF ERROR
E23A E89203	649	CALL ERR_BEEP ; GO TO BEEP SUBROUTINE
E23D FA	650	CLI
E23E F4	651	HLT ; HALT THE SYSTEM
	652	;-----
	653	; 8253 TIMER CHECKOUT ;
	654	; DESCRIPTION ;
	655	; VERIFY THAT THE SYSTEM TIMER (0) ;
	656	; DOESN'T COUNT TOO FAST OR TOO SLOW. ;
	657	;-----
	658	D7:
E23F	659	MOV AL,0FEH ; MASK ALL INTRs EXCEPT LVL 0
E23F B0FE	660	OUT DX,AL ; WRITE THE 8259 IMR
E241 EE	661	MOV AL,00010000B ; SEL TIM 0, LSB, MODE 0, BINARY
E242 B010	662	OUT TIM_CTL,AL ; WRITE TIMER CONTROL MODE REG
E244 E643	663	MOV CX,16H ; SET PGH LOOP CNT
E246 B91600	664	MOV AL,CL ; SET TIMER 0 CNT REG
E249 BAC1	665	OUT TIMER0,AL ; WRITE TIMER 0 CNT REG
E24B E640	665	

LOC OBJ	LINE	SOURCE
E24D	666	D0:
E24D F6C4FF	667	TEST AH,OFFH ; DID TIMER 0 INTERRUPT OCCUR?
E250 7504	668	JNZ D9 ; YES - CHECK TIMER OP FOR SLOW TIME
E252 E2F9	669	LOOP D0 ; WAIT FOR INTR FOR SPECIFIED TIME
E254 EBE1	670	JMP D6 ; TIMER 0 INTR DIDN'T OCCUR - ERR
E256	671	
E256 B112	672	D9: MOV CL,16 ; SET PGH LOOP CNT
E258 B0FF	673	MOV AL,OFFH ; WRITE TIMER 0 CNT REG
E25A E640	674	OUT TIMER0,AL
E25C B0FE00	675	MOV AX,0FEH
E25F EE	676	OUT DX,AL
E260	677	
E260 F6C4FF	678	D10: TEST AH,OFFH ; DID TIMER 0 INTERRUPT OCCUR?
E263 75D2	679	JNZ D6 ; YES - TIMER CNTING TOO FAST, ERR
E265 E2F9	680	LOOP D10 ; WAIT FOR INTR FOR SPECIFIED TIME
	681	
	682	;
	683	;----- ESTABLISH BIOS SUBROUTINE CALL INTERRUPT VECTORS
E267 1E	684	
E268 BF4000	685	PUSH DS ; SAVE POINTER TO DATA AREA
E26B 0E	686	MOV DI,OFFSET VIDEO_INT ; SETUP ADDR TO INTR AREA
E26C 1F	687	PUSH CS
E26D B03FF90	688	POP DS ; SETUP ADDR OF VECTOR TABLE
E271 B91000	689	MOV SI,OFFSET VECTOR_TABLE+16 ; START WITH VIDEO ENTRY
	690	
	691	;
	692	;----- SETUP TIMER 0 TO MODE 3
E274 B0FF	693	MOV AL,OFFH ; DISABLE ALL DEVICE INTERRUPTS
E276 EE	694	OUT DX,AL
E277 B036	695	MOV AL,36H ; SEL TIM 0,LSB,MSB,MODE 3
E279 E643	696	OUT TIMER+3,AL ; WRITE TIMER MODE REG
E27B B000	697	MOV AL,0
E27D E640	698	OUT TIMER,AL ; WRITE LSB TO TIMER 0 REG
E27F	699	
E27F A5	700	E1A: MOVSW ; MOVE VECTOR TABLE TO RAM
E280 47	701	INC DI ; MOVE PAST SEGMENT POINTER
E281 47	702	INC DI
E282 E2FB	703	LOOP E1A
E284 E640	704	OUT TIMER,AL ; WRITE MSB TO TIMER 0 REG
E286 1F	705	POP DS ; RECOVER DATA SEG POINTER
	706	
	707	;
	708	;----- SETUP TIMER 0 TO BLINK LED IF MANUFACTURING TEST MODE
E287 E8B903	709	CALL KBD_RESET ; SEND SOFTWARE RESET TO KEYBRD
E28A 80FBAA	710	CMF BL,0AAH ; SCAN CODE 'AA' RETURNED?
E28D 741E	711	JE E6 ; YES - CONTINUE (NON MFG MODE)
E28F B03C	712	MOV AL,3CH ; EN KBD, SET KBD CLK LINE LOW
E291 E661	713	OUT PORT_B,AL ; WRITE 8255 PORT B
E293 90	714	NOP
E294 90	715	NOP
E295 E460	716	IN AL,PORT_A ; WAS A BIT CLOCKED IN?
E297 24FF	717	AND AL,OFFH
E299 750E	718	JNZ E2 ; YES - CONTINUE (NON MFG MODE)
E29B FE061204	719	INC DATA_AREA[OFFSET MFG_TST] ; ELSE SET SW FOR MFG TEST MODE
E29F C70620006DE6	720	MOV INT_ADDR,OFFSET BLINK_INT ; SETUP TIMER INTR TO BLINK LED
E2A5 B0FE	721	MOV AL,0FEH ; ENABLE TIMER INTERRUPT
E2A7 E621	722	OUT INTA01,AL
E2A9	723	E2: ; JUMPER_NOT_IN:
E2A9 B0CC	724	MOV AL,0CCH ; RESET THE KEYBOARD
E2AB E661	725	OUT PORT_B,AL
	726	
	727	;
	728	;-----
	729	; INITIALIZE AND START CRT CONTROLLER (6845) ;
	730	; TEST VIDEO READ/WRITE STORAGE. ;
	731	; DESCRIPTION ;
	732	; RESET THE VIDEO ENABLE SIGNAL. ;
	733	; SELECT ALPHANUMERIC MODE, 40 * 25, B & N. ;
	734	; READ/WRITE DATA PATTERNS TO STG. CHECK STG ;
	735	; ADDRESSABILITY. ;
	736	;-----
E2AD	736	E6: ;
E2AD E460	737	IN AL,PORT_A ; READ SENSE SWITCHES
E2AF B400	738	MOV AH,0
E2B1 A31004	739	MOV DATA_WORD[OFFSET EQUIP_FLAG],AX ; STORE SENSE SW INFO
E2B4	740	
E2B4 2430	741	E6A: AND AL,30H ; ISOLATE VIDEO SMS
E2B6 7529	742	JNZ E7 ; VIDEO SMS SET TO 0?


```

LOC OBJ          LINE  SOURCE
E2B6 C706400053FF 743      MOV    VIDEO_INT,OFFSET DUMMY_RETURN
E2BE E9A200        744      JMP    E10_1          ; SKIP VIDEO TESTS FOR BURN-IN
745
E2C3             746      ORG    0E2C3H
E2C3             747      NMI_INT PROC NEAR
E2C3 50           748      PUSH  AX              ; SAVE ORIG CONTENTS OF AX
E2C4 E462        749      IN    AL,PORT_C
E2C6 A8C0        750      TEST  AL,0C0H        ; PARITY CHECK?
E2C8 7415        751      JZ    D14             ; NO, EXIT FROM ROUTINE
E2CA BEDAFF90    752      MOV  SI,OFFSET D1    ; ADDR OF ERROR MSG
E2CE A840        753      TEST  AL,40H        ; I/O PARITY CHECK
E2D0 7504        754      JNZ  D13             ; DISPLAY ERROR MSG
E2D2 BE23FF90    755      MOV  SI,OFFSET D2    ; MUST BE PLANAR
E2D6            756      D13:
E2D6 2BC0        757      SUB  AX,AX           ; INIT AND SET MODE FOR VIDEO
E2D8 CD10        758      INT  10H            ; CALL VIDEO_IO PROCEDURE
E2DA E0DD03     759      CALL P_MSG          ; PRINT ERROR MSG
E2DD FA         760      CLI
E2DE F4         761      HLT                 ; HALT SYSTEM
E2DF            762      D14:
E2DF 58         763      POP  AX              ; RESTORE ORIG CONTENTS OF AX
E2E0 CF         764      IRET
765      NMI_INT ENDP
E2E1            766      E7:
E2E1 3C30        767      CMP  AL,30H         ; TEST_VIDEO:
E2E3 7408        768      JE   E8              ; B/W CARD ATTACHED?
E2E5 FEC4        769      INC  AH              ; YES - SET MODE FOR B/W CARD
E2E7 3C20        770      CMP  AL,20H         ; SET COLOR MODE FOR COLOR CD
E2E9 7502        771      JNE  E8              ; 80X25 MODE SELECTED?
E2EB B403        772      MOV  AH,3           ; NO - SET MODE FOR 40X25
E2ED            773      E8:
E2ED 86E0        774      XCHG AH,AL          ; SET_MODE
E2EF 50         775      PUSH AX             ; SAVE VIDEO MODE ON STACK
E2F0 2AE4        776      SUB  AH,AH          ; INITIALIZE TO ALPHANUMERIC MD
E2F2 CD10        777      INT  10H            ; CALL VIDEO_IO
E2F4 58         778      POP  AX             ; RESTORE VIDEO SENSE SMS IN AH
E2F5 50         779      PUSH AX             ; RESAVE VALUE
E2F6 B000B0     780      MOV  BX,0B000H     ; BEG VIDEO RAM ADDR B/W CD
E2F9 B8B003     781      MOV  DX,3B8H       ; MODE REG FOR B/W
E2FC B90010     782      MOV  CX,4096       ; RAM BYTE CNT FOR B/W CD
E2FF B001       783      MOV  AL,1           ; SET MODE FOR BW CARD
E301 80FC30     784      CMP  AH,30H        ; B/W VIDEO CARD ATTACHED?
E304 7408        785      JE   E9              ; YES - GO TEST VIDEO STG
E306 B7B8       786      MOV  BH,0B8H       ; BEG VIDEO RAM ADDR COLOR CD
E308 B2D8       787      MOV  DL,0D8H       ; MODE REG FOR COLOR CD
E30A B540       788      MOV  CH,40H        ; RAM BYTE CNT FOR COLOR CD
E30C FEC6       789      DEC  AL             ; SET MODE TO 0 FOR COLOR CD
E30E            790      E9:
E30E EE         791      OUT  DX,AL         ; TEST_VIDEO_STG:
E30F 01FD3412   792      CMP  BP,1234H      ; DISABLE VIDEO FOR COLOR CD
E313 8EC3       793      MOV  ES,BX          ; POD INITIATED BY KBD RESET?
E315 7407       794      JE   E10            ; POINT ES TO VIDEO RAM STG
E317 8EDB       795      MOV  DS,BX          ; YES - SKIP VIDEO RAM TEST
E319 E0FFFC     796      ASSUME DS:NOTHING,ES:NOTHING ; POINT DS TO VIDEO RAM STG
E31C 7532       797      CALL STGTST_CNT    ; GO TEST VIDEO R/W STG
E31E            798      JNE  E17            ; R/W STG FAILURE - BEEP SPK
799
800      ; -----
801      ; SETUP VIDEO DATA ON SCREEN FOR VIDEO LINE TEST.
802      ; DESCRIPTION
803      ; ENABLE VIDEO SIGNAL AND SET MODE.
804      ; DISPLAY A HORIZONTAL BAR ON SCREEN.
805      ; -----
E31E            805      E10:
E31E 58         806      POP  AX              ; GET VIDEO SENSE SMS (AH)
E31F 50         807      PUSH AX             ; SAVE IT
E320 B400       808      MOV  AH,0           ; ENABLE VIDEO AND SET MODE
E322 CD10       809      INT  10H            ; VIDEO
E324 B82070     810      MOV  AX,7020H      ; WRT BLANKS IN REVERSE VIDEO
E327 2BFF       811      SUB  DI,DI          ; SETUP STARTING LOC
E329 B92800     812      MOV  CX,40          ; NO. OF BLANKS TO DISPLAY
E32C F3         813      REP  STOSH          ; WRITE VIDEO STORAGE
814      ; -----
815      ; CRT INTERFACE LINES TEST
816      ; DESCRIPTION
817      ; SENSE ON/OFF TRANSITION OF THE VIDEO ENABLE

```

```

818 ; AND HORIZONTAL SYNC LINES. ;
819 ;-----
E32E 58 020 POP AX ; GET VIDEO SENSE SW INFO
E32F 50 021 PUSH AX ; SAVE IT
E330 80FC30 022 CMP AH,30H ; B/W CARD ATTACHED?
E333 BABA03 023 MOV DX,03BAH ; SETUP ADDR OF BW STATUS PORT
E336 7402 024 JE E11 ; YES - GO TEST LINES
E338 B2DA 025 MOV DL,0DAH ; COLOR CARD IS ATTACHED
E33A 026 E11: ; LINE_TST:
E33A B408 027 MOV AH,8
E33C 028 E12: ; OFLOOP_CNT:
E33C 2BC9 029 SUB CX,CX
E33E 030 E13:
E33E EC 031 IN AL,DX ; READ CRT STATUS PORT
E33F 22C4 032 AND AL,AH ; CHECK VIDEO/HORZ LINE
E341 7504 033 JNZ E14 ; ITS ON - CHECK IF IT GOES OFF
E343 E2F9 034 LOOP E13 ; LOOP TILL ON OR TIMEOUT
E345 EB09 035 JMP SHORT E17 ; GO PRINT ERROR MSG
E347 036 E14:
E347 2BC9 037 SUB CX,CX
E349 038 E15:
E349 EC 039 IN AL,DX ; READ CRT STATUS PORT
E34A 22C4 040 AND AL,AH ; CHECK VIDEO/HORZ LINE
E34C 740A 041 JZ E16 ; ITS ON - CHECK NEXT LINE
E34E E2F9 042 LOOP E15 ; LOOP IF OFF TILL IT GOES ON
E350 043 E17: ; CRT_ERR
E350 BA0201 044 MOV DX,102H
E353 E87902 045 CALL ERR_BEEP ; GO BEEP SPEAKER
E356 EB06 046 JMP SHORT E18
E358 047 E16: ; NXT_LINE
E358 B103 048 MOV CL,3 ; GET NEXT BIT TO CHECK
E35A D2EC 049 SHR AH,CL
E35C 750E 050 JNZ E12 ; GO CHECK HORIZONTAL LINE
E35E 051 E18: ; DISPLAY_CURSOR:
E35E 58 052 POP AX ; GET VIDEO SENSE SMS (AH)
E35F B400 053 MOV AH,0 ; SET MODE AND DISPLAY CURSOR
E361 CD10 054 INT 10H ; CALL VIDEO I/O PROCEDURE
055
E363 056 E18_1:
E363 BA00C0 057 MOV DX,0C000H
E366 058 E18A:
E366 8EDA 059 MOV DS,DX
E368 2B0B 060 SUB BX,BX
E36A 8B07 061 MOV AX,[BX] ; GET FIRST 2 LOCATIONS
E36C 53 062 PUSH BX
E36D 5B 063 POP BX ; LET BUS SETTLE
E36E 3055AA 064 CMP AX,0AA55H ; PRESENT?
E371 7505 065 JNZ E18B ; NO? GO LOOK FOR OTHER MODULES
E373 E80E03 066 CALL ROM_CHECK ; GO SCAN MODULE
E376 EB04 067 JMP SHORT E18C
E378 068 E18B:
E378 81C28000 069 ADD DX,0080H ; POINT TO NEXT 2K BLOCK
E37C 070 E18C:
E37C 81FA00C8 071 CMP DX,0C800H ; TOP OF VIDEO ROM AREA YET?
E380 7CE4 072 JL E18A ; GO SCAN FOR ANOTHER MODULE
073 ;-----
074 ; EXPANSION I/O BOX TEST ;
075 ; CHECK TO SEE IF EXPANSION BOX PRESENT - IF INSTALLED, ;
076 ; TEST DATA AND ADDRESS BUSES TO I/O BOX. ;
077 ; ERROR='1801' ;
078 ;-----
079
080 ;----- DETERMINE IF BOX IS PRESENT
081
E382 082 EXP_IO: ; (CARD WAS ENABLED EARLIER)
E382 BA1002 083 MOV DX,0210H ; CONTROL PORT ADDRESS
E385 B85555 084 MOV AX,5555H ; SET DATA PATTERN
E388 EE 085 OUT DX,AL
E389 B001 086 MOV AL,01H
E38B EC 087 IN AL,DX ; RECOVER DATA
E38C 3AC4 088 CMP AL,AH ; REPLY?
E38E 7534 089 JNE E19 ; NO RESPONSE, GO TO NEXT TEST
E390 F7D0 090 NOT AX ; MAKE DATA=AAAA
E392 EE 091 OUT DX,AL
E393 B001 092 MOV AL,01H
E395 EC 093 IN AL,DX ; RECOVER DATA
E396 3AC4 094 CMP AL,AH

```

LOC OBJ	LINE	SOURCE	
E398 752A	895	JNE E19	; NO ANSWER=NEXT TEST
	896		
	897	;----- CHECK ADDRESS AND DATA BUS	
	898		
E39A	899	EXPI:	
E39A 8BD8	900	MOV BX,AX	
E39C BA1402	901	MOV DX,0214H	; LOAD DATA REG ADDRESS
E39F 2E8807	902	MOV CS:[BX],AL	; WRITE ADDRESS F000+BX
E3A2 EE	903	OUT DX,AL	; WRITE DATA
E3A3 90	904	NOP	
E3A4 EC	905	IN AL,DX	; READ DATA
E3A5 3AC7	906	CMP AL,BH	
E3A7 7514	907	JNE EXP_ERR	
E3A9 42	908	INC DX	; DX=215H (ADDR. HI REG)
E3AA EC	909	IN AL,DX	
E3AB 3AC4	910	CMP AL,AH	; COMPARE TO HI ADDRESS
E3AD 750E	911	JNE EXP_ERR	
E3AF 42	912	INC DX	; DX=216H (ADDR. LOW REG)
E3B0 EC	913	IN AL,DX	
E3B1 3AC4	914	CMP AL,AH	; ADDR. LOW OK?
E3B3 7508	915	JNE EXP_ERR	
E3B5 F7D0	916	NOT AX	; INVERT AX
E3B7 3CAA	917	CMP AL,0AAH	; BACK TO STARTING VALUE (AAAA) YET
E3B9 7409	918	JE E19	; GO ON TO NEXT TEST IF SO
E3BB EB0D	919	JMP EXP1	; LOOP BACK THROUGH WITH DATA OF 5555
E3BD	920	EXP_ERR:	
E3BD BEEDFE90	921	MOV SI,OFFSET F3B	
E3C1 E8F602	922	CALL P_MSG	
	923	;-----	
	924	; ADDITIONAL READ/WRITE STORAGE TEST	:
	925	; DESCRIPTION	:
	926	; WRITE/READ DATA PATTERNS TO ANY READ/WRITE STORAGE	:
	927	; AFTER THE BASIC 16K. STORAGE ADDRESSABILITY IS CHECKED.	:
	928	;-----	
	929	ASSUME DS:DATA	
E3C4	930	E19:	
	931		
	932	;----- DETERMINE RAM SIZE ON PLANAR BOARD	
	933		
E3C4 E8771B	934	CALL DDS	
E3C7 A01000	935	MOV AL,BYTE PTR EQUIP_FLAG	; GET SENSE SMS INFO
E3CA 240C	936	AND AL,0CH	; ISOLATE RAM SIZE SMS
E3CC B404	937	MOV AH,4	
E3CE F6E4	938	MUL AH	
E3D0 0410	939	ADD AL,16	; ADD BASIC 16K
E3D2 8B0D	940	MOV DX,AX	; SAVE PLANAR RAM SIZE IN DX
E3D4 8B08	941	MOV BX,AX	; AND IN BX
	942		
	943	;----- DETERMINE IO CHANNEL RAM SIZE	
	944		
E3D6 A11500	945	MOV AX,IO_RAM_SIZE	; GET IO CHANNEL RAM SIZE
E3D9 83FB40	946	CMP BX,40H	; PLANAR RAM SIZE = 64K?
E3DC 7402	947	JE E20	; YES - ADD IO CHN RAM SIZE
E3DE 2BC0	948	SUB AX,AX	; NO - DON'T ADD ANY IO RAM
E3E0	949		; ADD_IO_SIZE:
E3E0 03C3	950	ADD AX,BX	; SUM TOTAL RAM SIZE
E3E2 A31300	951	MOV MEMORY_SIZE,AX	; SETUP MEMORY SIZE PARM
E3E5 81FD3412	952	CMP BP,1234H	; POD INITIATED BY KBD RESET?
E3E9 1E	953	PUSH DS	; SAVE DATA SEGMENT
E3EA 744F	954	JE TST12	; YES - SKIP MEMORY TEST
	955		
	956	;----- TEST ANY OTHER READ/WRITE STORAGE AVAILABLE	
	957		
E3EC BB0004	958	MOV BX,400H	
E3EF B91000	959	MOV CX,16	
E3F2	960	E21:	
E3F2 3B01	961	CMP DX,CX	; ANY MORE STG TO BE TESTED?
E3F4 762D	962	JBE E23	; NO - GO TO NEXT TEST
E3F6 8E0B	963	MOV DS,BX	; SETUP STG ADDR IN DS AND ES
E3F8 8EC3	964	MOV ES,BX	
E3FA 83C110	965	ADD CX,16	; INCREMENT STG BYTE COUNTER
E3FD 81C30004	966	ADD BX,400H	; SET POINTER TO NEXT 16K BLK
E401 51	967	PUSH CX	; SAVE REGS
E402 53	968	PUSH BX	
E403 52	969	PUSH DX	
E404 E811FC	970	CALL STGTST	; GO TEST A 16K BLK OF STG
E407 5A	971	POP DX	

LOC OBJ	LINE	SOURCE
E408 5B	972	POP BX ; RESTORE REGS
E409 59	973	POP CX
E40A 74E6	974	JE E21 ; CHECK IF MORE STG TO TEST
	975	
	976	;----- PRINT FAILING ADDRESS AND XOR'ED PATTERN IF DATA COMPARE ERROR
	977	
E40C 8CDA	978	MOV DX,DS ; CONVERT FAILING HIGH-ORDER
E40E 8AE8	979	MOV CH,AL ; SAVE FAILING BIT PATTERN
E410 8AC6	980	MOV AL,DH ; GET FAILING ADDR
E412 E81002	981	CALL XPC_BYTE ; CONVERT AND PRINT CODE
E415 8AC5	982	MOV AL,CH ; GET FAILING BIT PATTERN
E417 E80B02	983	CALL XPC_BYTE ; CONVERT AND PRINT CODE
E41A BE67FA90	984	MOV SI,OFFSET E1 ; SETUP ADDRESS OF ERROR MSG
E41E E89902	985	CALL P_MSG ; PRINT ERROR MSG
E421	986	E22:
E421 EB18	987	JMP SHORT TST12 ; GO TO NEXT TEST
E423	988	E23:
E423 1F	989	POP DS ; STG_TEST_DONE
E424 1E	990	PUSH DS ; POINT DS TO DATA SEGMENT
E425 8B161500	991	MOV DX,IO_RAM_SIZE ; GET IO CHANNEL RAM SIZE
E429 0BD2	992	OR DX,DX ; SET FLAG RESULT
E42B 740E	993	JZ TST12 ; NO IO RAM, GO TO NEXT TEST
E42D B90000	994	MOV CX,0
E430 81FB0010	995	CHP BX,1000H ; HAS IO RAM BEEN TESTED
E434 7705	996	JA TST12 ; YES - GO TO NEXT TEST
E436 BB0010	997	MOV BX,1000H ; SETUP BEG LOC FOR IO RAM
E439 EBB7	998	JMP E21 ; GO TEST IO CHANNEL RAM
	999	;
	1000	;-----
	1000	; KEYBOARD TEST ;
	1001	; DESCRIPTION ;
	1002	; RESET THE KEYBOARD AND CHECK THAT SCAN CODE ;
	1003	; 'AA' IS RETURNED TO THE CPU. CHECK FOR STUCK ;
	1004	; KEYS. ;
	1005	;
	1006	ASSUME DS:DATA
E43B	1007	TST12:
E43B 1F	1008	POP DS
E43C 803E120001	1009	CHP MFG_TST,1 ; MANUFACTURING TEST MODE?
E441 742A	1010	JE F7 ; YES - SKIP KEYBOARD TEST
E443 E8FD01	1011	CALL KBD_RESET ; ISSUE SOFTWARE RESET TO KEYBOD
E446 E31E	1012	JCXZ F6 ; PRINT ERR MSG IF NO INTERRUPT
E448 B04D	1013	MOV AL,4DH ; ENABLE KEYBOARD
E44A E661	1014	OUT PORT_B,AL
E44C 80FBAA	1015	CHP BL,0AAH ; SCAN CODE AS EXPECTED?
E44F 7515	1016	JNE F6 ; NO - DISPLAY ERROR MSG
	1017	
	1018	;----- CHECK FOR STUCK KEYS
	1019	
E451 B0CC	1020	MOV AL,0CCH ; CLR KBD, SET CLK LINE HIGH
E453 E661	1021	OUT PORT_B,AL
E455 B04C	1022	MOV AL,4CH ; ENABLE KBD,CLK IN NEXT BYTE
E457 E661	1023	OUT PORT_B,AL
E459 2BC9	1024	SUB CX,CX
E45B	1025	F5: ; KBD_WAIT
E45B E2FE	1026	LOOP F5 ; DELAY FOR A WHILE
E45D E460	1027	IN AL,KBD_IN ; CHECK FOR STUCK KEYS
E45F 3C00	1028	CHP AL,0 ; SCAN CODE = 0?
E461 740A	1029	JE F7 ; YES - CONTINUE TESTING
E463 E8BF01	1030	CALL XPC_BYTE ; CONVERT AND PRINT
E466 BE33FF90	1031	F6: MOV SI,OFFSET F1 ; GET HSG ADDR
E46A E84D02	1032	CALL P_MSG ; PRINT HSG ON SCREEN
	1033	
	1034	;----- SETUP INTERRUPT VECTOR TABLE
	1035	
E46D	1036	F7: ; SETUP_INT_TABLE:
E46D 2BC0	1037	SUB AX,AX
E46F 8EC0	1038	MOV ES,AX
E471 B90800	1039	MOV CX,8 ; GET VECTOR CNT
E474 1E	1040	PUSH DS ; SAVE DATA SEGMENT
E475 0E	1041	PUSH CS ; SETUP DS SEG REG
E476 1F	1042	POP DS
E477 BEF3FE90	1043	MOV SI,OFFSET VECTOR_TABLE
E47B BF2000	1044	MOV DI,OFFSET INT_PTR
E47E	1045	F7A:
E47E A5	1046	MOVSW
E47F 47	1047	INC DI ; SKIP OVER SEGMENT
E480 47	1048	INC DI

```

LOC OBJ          LINE  SOURCE

E481 E2FB        1049          LOOP    F7A
1050          ;-----
1051          ; CASSETTE DATA WRAP TEST          :
1052          ; DESCRIPTION                    :
1053          ; TURN CASSETTE MOTOR OFF. WRITE A BIT OUT TO THE :
1054          ; CASSETTE DATA BUS. VERIFY THAT CASSETTE DATA :
1055          ; READ IS WITHIN A VALID RANGE.    :
1056          ;-----
1057
1058          ;----- TURN THE CASSETTE MOTOR OFF
1059
E483            1060      TST13:
E483 1F          1061          POP     DS
E484 1E          1062          PUSH  DS
E485 B0A0D      1063          MOV    AL,040H          ; SET TIMER 2 SPK OUT, AND CASST
E487 E661      1064          OUT   PORT_B,AL      ; OUT BITS ON, CASSETTE MOT OFF
1065
1066          ;----- WRITE A BIT
1067
E489 B0FF      1068          MOV    AL,0FFH          ; DISABLE TIMER INTERRUPTS
E48B E621      1069          OUT   INTA01,AL
E48D B0B6      1070          MOV    AL,0B6H          ; SEL TIM 2, LSB, MSB, MD 3
E48F E643      1071          OUT   TIMER+3,AL    ; WRITE 8253 CMD/MODE REG
E491 B80304    1072          MOV    AX,1235         ; SET TIMER 2 CNT FOR 1000 USEC
E494 E642      1073          OUT   TIMER+2,AL    ; WRITE TIMER 2 COUNTER REG
E496 8AC4      1074          MOV    AL,AH           ; WRITE MSB
E498 E642      1075          OUT   TIMER+2,AL
1076
1077          ;----- READ CASSETTE INPUT
1078
E49A E462      1079          IN    AL,PORT_C      ; READ VALUE OF CASS IN BIT
E49C 2410      1080          AND   AL,10H          ; ISOLATE FROM OTHER BITS
E49E A26B00    1081          MOV    LAST_VAL,AL
E4A1 E80514    1082          CALL  READ_HALF_BIT
E4A4 E80214    1083          CALL  READ_HALF_BIT
E4A7 E30C      1084          JCXZ  F8             ; CAS_ERR
E4A9 81FB4005  1085          CMP   BX,MAX_PERIOD
E4AD 7306      1086          JNC   F8             ; CAS_ERR
E4AF 81FB1004  1087          CMP   BX,MIN_PERIOD
E4B3 7307      1088          JNC   ROM_SCAN       ; GO TO NEXT TEST IF OK
E4B5           1089      F8:
E4B5 BE39FF90  1090          MOV   SI,OFFSET F2    ; CAS_ERR
E4B9 E8FE01    1091          CALL P_MSG           ; CASSETTE WRAP FAILED
1092          ; GO PRINT ERROR MSG
1093          ;-----
1094          ; CHECK FOR OPTIONAL ROM FROM C8000->F4000 IN 2K INCREMENTS :
1095          ; (A VALID MODULE HAS '55AA' IN THE FIRST 2 LOCATIONS, LENGTH :
1096          ; INDICATOR (LENGTH/512) IN THE 3RD LOCATION AND TEST/INIT. :
1097          ; CODE STARTING IN THE 4TH LOCATION.) :
1098          ;-----
E4BC           1098      ROM_SCAN:
E4BC BA00C8    1099          MOV   DX,0C000H      ; SET BEGINNING ADDRESS
E4BF           1100      ROM_SCAN_1:
E4BF BEDA      1101          MOV   DS,DX
E4C1 2B0B      1102          SUB   BX,BX          ; SET BX=0000
E4C3 8B07      1103          MOV   AX,[BX]
E4C5 3D55AA    1104          CMP   AX,0AA55H     ; = TO ID WORD?
E4C8 7505      1105          JNZ  NEXT_ROM       ; PROCEED TO NEXT ROM IF NOT
E4CA E8B701    1106          CALL ROM_CHECK      ; GO DO CHECKSUM AND CALL
E4CD EB04      1107          JMP  SHORT ARE_ME_DONE ; CHECK FOR END OF ROM SPACE
E4CF           1108      NEXT_ROM:
E4CF 81C28000  1109          ADD   DX,0800H     ; POINT TO NEXT 2K ADDRESS
E4D3           1110      ARE_ME_DONE:
E4D3 81FA00F6  1111          CMP   DX,0F600H     ; AT F6000 YET?
E4D7 7CE6      1112          JL   ROM_SCAN_1    ; GO CHECK ANOTHER ADD. IF NOT
E4D9 EB0190    1113          JMP  BASE_ROM_CHK   ; GO CHECK BASIC ROM
1114          ;-----
1115          ; ROS CHECKSUM II          :
1116          ; DESCRIPTION                    :
1117          ; A CHECKSUM IS DONE FOR THE 4 ROS :
1118          ; MODULES CONTAINING BASIC CODE :
1119          ;-----
E4DC           1120      BASE_ROM_CHK:
E4DC           1121      E4:
E4DC 2B0B      1122          SUB   BX,BX          ; SETUP STARTING ROS ADDR
E4DE BEDA      1123          MOV   DS,DX
E4E0 E86907    1124          CALL ROS_CHECKSUM   ; CHECK ROS

```

LOC OBJ	LINE	SOURCE			
E4E3 7403	1125	JE	E5		; CONTINUE IF OK
E4E5 E62103	1126	CALL	ROM_ERR		; POST ERROR
E4E8	1127	E5:			
E4E8 80C602	1128	ADD	DH,02H		; POINT TO NEXT 8K MODULE
E4EB 80FEFE	1129	CMF	DH,0FEH		
E4EE 75EC	1130	JNZ	E4		; YES - CONTINUE
E4F0 1F	1131	POP	DS		; RECOVER DATA SEG PTR
	1132				-----
	1133				; DISKETTE ATTACHMENT TEST
	1134				; DESCRIPTION
	1135				; CHECK IF IPL DISKETTE DRIVE IS ATTACHED TO SYSTEM. IF ATTACHED,
	1136				; VERIFY STATUS OF NEC FDC AFTER A RESET. ISSUE A RECAL AND SEEK :
	1137				; CMD TO FDC AND CHECK STATUS. COMPLETE SYSTEM INITIALIZATION :
	1138				; THEN PASS CONTROL TO THE BOOT LOADER PROGRAM. :
	1139				-----
E4F1	1140	F9:			
E4F1 A01000	1141	MOV	AL,BYTE PTR EQUIP_FLAG		; GET SENSE SMS INFO
E4F4 A801	1142	TEST	AL,01H		; IPL DISKETTE DRIVE ATTC?
E4F6 750A	1143	JNZ	F10		; NO -SKIP THIS TEST
E4F8 803E120001	1144	CMF	HFG_TST,1		; MANUFACTURING TEST MODE?
E4FD 753D	1145	JNE	F15A		; NO - GO TO BOOT LOADER
E4FF E959FB	1146	JMP	START		; YES - LOOP POWER-ON-DIAGS
E502	1147	F10:			
E502 E421	1148	IN	AL,INTA01		; DISK_TEST
E504 24BF	1149	AND	AL,0BFH		; ENABLE DISKETTE INTERRUPTS
E506 E621	1150	OUT	INTA01,AL		
E508 B400	1151	MOV	AH,0		; RESET NEC FDC
E50A 8AD4	1152	MOV	DL,AH		; (POINT TO DISKETTE)
E50C CD13	1153	INT	13H		; VERIFY STATUS AFTER RESET
E50E 7221	1154	JC	F13		
	1155				
	1156				-----
	1157				; TURN DRIVE 0 MOTOR ON
E510 BAF203	1158	MOV	DX,03F2H		; GET ADDR OF FDC CARD
E513 52	1159	PUSH	DX		; SAVE IT
E514 B01C	1160	MOV	AL,1CH		; TURN MOTOR ON, EN DMA/INT
E516 EE	1161	OUT	DX,AL		; WRITE FDC CONTROL REG
E517 2BC9	1162	SUB	CX,CX		
E519	1163	F11:			; MOTOR_WAIT:
E519 E2FE	1164	LOOP	F11		; WAIT FOR 1 SECOND
E51B	1165	F12:			; MOTOR_WAIT1:
E51B E2FE	1166	LOOP	F12		
E51D 3302	1167	XOR	DX,DX		; SELECT DRIVE 0
E51F B501	1168	MOV	CH,1		; SELECT TRACK 1
E521 88163E00	1169	MOV	SEEK_STATUS,DL		
E525 E85509	1170	CALL	SEEK		; RECALIBRATE DISKETTE
E528 7207	1171	JC	F13		; GO TO ERR SUBROUTINE IF ERR
E52A B522	1172	MOV	CH,34		; SELECT TRACK 34
E52C E84E09	1173	CALL	SEEK		; SEEK TO TRACK 34
E52F 7307	1174	JNC	F14		; OK, TURN MOTOR OFF
E531	1175	F13:			; DSK_ERR:
E531 BEEAFF90	1176	MOV	SI,OFFSET F3		; GET ADDR OF MSG
E535 E88201	1177	CALL	P_MSG		; GO PRINT ERROR MSG
	1178				
	1179				-----
	1180				; TURN DRIVE 0 MOTOR OFF
E538	1181	F14:			; DR0_OFF:
E538 B00C	1182	MOV	AL,0CH		; TURN DRIVE 0 MOTOR OFF
E53A 5A	1183	POP	DX		; RECOVER FDC CTL ADDRESS
E53B EE	1184	OUT	DX,AL		
	1185				
	1186				-----
	1187				; SETUP PRINTER AND RS232 BASE ADDRESSES IF DEVICE ATTACHED
E53C	1188	F15A:			
E53C BE1E00	1189	MOV	SI,OFFSET KB_BUFFER		
E53F 89361A00	1190	MOV	BUFFER_HEAD,SI		; SETUP KEYBOARD PARAMETERS
E543 89361C00	1191	MOV	BUFFER_TAIL,SI		
E547 89368000	1192	MOV	BUFFER_START,SI		; DEFAULT TO STANDARD BUFFER
E54B 83C620	1193	ADD	SI,32		; (32 BYTES LONG)
E54E 89368200	1194	MOV	BUFFER_END,SI		
E552 E421	1195	IN	AL,INTA01		
E554 24FC	1196	AND	AL,0FCH		; ENABLE TIMER AND KBD INTS
E556 E621	1197	OUT	INTA01,AL		
E558 BD3DE690	1198	MOV	BP,OFFSET F4		; PRT_SRC_TBL
E55C 2BF6	1199	SUB	SI,SI		
E55E	1200	F16:			; PRT_BASE:
E55E 2E885600	1201	MOV	DX,CS:[BP]		; GET PRINTER BASE ADDR

LOC OBJ	LINE	SOURCE	
E562 B0AA	1202	MOV AL,0AAH	; WRITE DATA TO PORT A
E564 EE	1203	OUT DX,AL	
E565 52	1204	PUSH DX	
E566 EC	1205	IN AL,DX	; READ PORT A
E567 5A	1206	POP DX	
E568 3CAA	1207	CHP AL,0AAH	; DATA PATTERN SAME
E56A 7505	1208	JNE F17	; NO - CHECK NEXT PRT CD
E56C 895408	1209	MOV PRINTER_BASE[SI],DX	; YES - STORE PRT BASE ADDR
E56F 46	1210	INC SI	; INCREMENT TO NEXT WORD
E570 46	1211	INC SI	
E571	1212	F17:	; NO_STORE:
E571 45	1213	INC BP	; POINT TO NEXT BASE ADDR
E572 45	1214	INC BP	
E573 81FD43E6	1215	CHP BP,OFFSET F4E	; ALL POSSIBLE ADDRS CHECKED?
E577 75E5	1216	JNE F16	; PRT_BASE
E579 2BDB	1217	SUB BX,DX	; POINTER TO RS232 TABLE
E57B BAF03	1218	MOV DX,3FAH	; CHECK IF RS232 CD 1 ATTCH?
E57E EC	1219	IN AL,DX	; READ INTR ID REG
E57F A8F8	1220	TEST AL,0F8H	
E581 7506	1221	JNZ F18	
E583 C707F803	1222	MOV RS232_BASE[BX],3F8H	; SETUP RS232 CD #1 ADDR
E587 43	1223	INC BX	
E588 43	1224	INC BX	
E589	1225	F18:	
E589 B602	1226	MOV DH,02H	; CHECK IF RS232 CD 2 ATTCH (AT 2FA)
E58B EC	1227	IN AL,DX	; READ INTERRUPT ID REG
E58C A8F8	1228	TEST AL,0F8H	
E58E 7506	1229	JNZ F19	; BASE_END
E590 C707F802	1230	MOV RS232_BASE[BX],2F8H	; SETUP RS232 CD #2
E594 43	1231	INC BX	
E595 43	1232	INC BX	
	1233		
	1234	;	----- SET UP EQUIP FLAG TO INDICATE NUMBER OF PRINTERS AND RS232 CARDS
	1235		
E596	1236	F19:	; BASE_END:
E596 8BC6	1237	MOV AX,SI	; SI HAS 2* NUMBER OF RS232
E598 B103	1238	MOV CL,3	; SHIFT COUNT
E59A D2C8	1239	ROR AL,CL	; ROTATE RIGHT 3 POSITIONS
E59C 0AC3	1240	OR AL,BL	; OR IN THE PRINTER COUNT
E59E A21100	1241	MOV BYTE PTR EQUIP_FLAG+1,AL	; STORE AS SECOND BYTE
E5A1 B201	1242	MOV DL,01H	; DX=201
E5A3 EC	1243	IN AL,DX	
E5A4 A80F	1244	TEST AL,0FH	
E5A6 7505	1245	JNZ F20	; NO_GAME_CARD
E5A8 800E110010	1246	OR BYTE PTR EQUIP_FLAG+1,16	
E5AD	1247	F20:	
	1248		
	1249	;	----- SET DEFAULT TIMEOUT VALUES FOR PRINTER AND RS232
	1250		
E5AD 1E	1251	PUSH DS	
E5AE 07	1252	POP ES	
E5AF BF7800	1253	MOV DI,OFFSET PRINT_TIM_OUT	
E5B2 B81414	1254	MOV AX,1414H	; PRINTER DEFAULTS (COUNT=20)
E5B5 AB	1255	STOSW	
E5B6 AB	1256	STOSW	
E5B7 B80101	1257	MOV AX,0101H	; RS232 DEFAULTS=01
E5BA AB	1258	STOSW	
E5BB AB	1259	STOSW	
	1260		
	1261	;	----- ENABLE NMI INTERRUPTS
	1262		
E5BC B080	1263	MOV AL,80H	; ENABLE NMI INTERRUPTS
E5BE E6A0	1264	OUT 0A0H,AL	
E5C0 803E120001	1265	CHP MFG_TST,1	; MFG MODE?
E5C5 7406	1266	JE F21	; LOAD_BOOT_STRAP
E5C7 BA0100	1267	MOV DX,1	
E5CA E80200	1268	CALL ERR_BEEP	; BEEP 1 SHORT TONE
	1269		
E5CD	1270	F21:	; LOAD_BOOT_STRAP:
E5CD CD19	1271	INT 19H	; BOOTSTRAP
	1272		
	1273	;	-----
	1274	;	INITIAL RELIABILITY TEST -- SUBROUTINES ;
	1275	;	-----
	1276	ASSUME CS:CODE,DS:DATA	
	1277	;	-----
	1278	;	SUBROUTINES FOR POWER ON DIAGNOSTICS ;

```

1279 ; THIS PROCEDURE WILL ISSUE ONE LONG TONE (3 SECS) AND ONE OR :
1280 ; MORE SHORT TONES (1 SEC) TO INDICATE A FAILURE ON THE PLANAR :
1281 ; BOARD, A BAD RAM MODULE, OR A PROBLEM WITH THE CRT. :
1282 ; ENTRY PARAMETERS: :
1283 ; DH = NUMBER OF LONG TONES TO BEEP :
1284 ; DL = NUMBER OF SHORT TONES TO BEEP :
1285 ;-----
ESC F 1286 ERR_BEEP PROC NEAR
ESC F 9C 1287 PUSH ; SAVE FLAGS
E50 FA 1288 CLI ; DISABLE SYSTEM INTERRUPTS
E5D1 1E 1289 PUSH DS ; SAVE DS REG CONTENTS
E5D2 E86919 1290 CALL DDS
E5D5 0AF6 1291 OR DH,DH ; ANY LONG ONES TO BEEP
E5D7 7418 1292 JZ G3 ; NO, DO THE SHORT ONES
E5D9 1293 G1: ; LONG_BEEP:
E5D9 B306 1294 MOV BL,6 ; COUNTER FOR BEEPS
E5DB E82500 1295 CALL BEEP ; DO THE BEEP
E5DE E2FE 1296 G2: LOOP G2 ; DELAY BETWEEN BEEPS
E5E0 FECE 1297 DEC DH ; ANY MORE TO DO
E5E2 75F5 1298 JNZ G1 ; DO IT
E5E4 803E120001 1299 CMP HFG_TST,1 ; HFG TEST MODE?
E5E9 7506 1300 JNE G3 ; YES - CONTINUE BEEPING SPEAKER
E5EB B0CD 1301 MOV AL,0CDH ; STOP BLINKING LED
E5ED E661 1302 OUT PORT_B,AL
E5EF EBE8 1303 JMP SHORT G1
E5F1 1304 G3: ; SHORT_BEEP:
E5F1 B301 1305 MOV BL,1 ; COUNTER FOR A SHORT BEEP
E5F3 E80D00 1306 CALL BEEP ; DO THE SOUND
E5F6 1307 G4:
E5F6 E2FE 1308 LOOP G4 ; DELAY BETWEEN BEEPS
E5F8 FECA 1309 DEC DL ; DONE WITH SHORTS
E5FA 75F5 1310 JNZ G3 ; DO SOME MORE
E5FC 1311 G5:
E5FC E2FE 1312 LOOP G5 ; LONG DELAY BEFORE RETURN
E5FE 1313 G6:
E5FE E2FE 1314 LOOP G6
E600 1F 1315 POP DS ; RESTORE ORIG CONTENTS OF DS
E601 9D 1316 POPF ; RESTORE FLAGS TO ORIG SETTINGS
E602 C3 1317 RET ; RETURN TO CALLER
ERR_BEEP 1318 ENDP
1319
1320 ;---- ROUTINE TO SOUND BEEPER
1321
E603 1322 BEEP PROC NEAR
E603 B0B6 1323 MOV AL,10110110B ; SEL TIM 2,LSB,MSB,BINARY
E605 E643 1324 OUT TIMER+3,AL ; WRITE THE TIMER MODE REG
E607 B83305 1325 MOV AX,533H ; DIVISOR FOR 1000 HZ
E60A E642 1326 OUT TIMER+2,AL ; WRITE TIMER 2 CNT - LSB
E60C 8AC4 1327 MOV AL,AH
E60E E642 1328 OUT TIMER+2,AL ; WRITE TIMER 2 CNT - MSB
E610 E461 1329 IN AL,PORT_B ; GET CURRENT SETTING OF PORT
E612 8AE0 1330 MOV AH,AL ; SAVE THAT SETTING
E614 0C03 1331 OR AL,03 ; TURN SPEAKER ON
E616 E661 1332 OUT PORT_B,AL
E618 2BC9 1333 SUB CX,CX ; SET CNT TO WAIT 500 MS
E61A 1334 G7:
E61A E2FE 1335 LOOP G7 ; DELAY BEFORE TURNING OFF
E61C FECE 1336 DEC BL ; DELAY CNT EXPIRED?
E61E 75FA 1337 JNZ G7 ; NO - CONTINUE BEEPING SPK
E620 8AC4 1338 MOV AL,AH ; RECOVER VALUE OF PORT
E622 E661 1339 OUT PORT_B,AL
E624 C3 1340 RET ; RETURN TO CALLER
BEEP 1341 ENDP
1342
1343 ;-----
1344 ; CONVERT AND PRINT ASCII CODE :
1345 ; AL MUST CONTAIN NUMBER TO BE CONVERTED. :
1346 ; AX AND BX DESTROYED. :
1347 ;-----
E625 1348 XPC_BYTE PROC NEAR
E625 50 1349 PUSH AX ; RESAVE FOR LOW NIBBLE DISPLAY
E626 B104 1350 MOV CL,4 ; SHIFT COUNT
E628 D2E8 1351 SHR AL,CL ; NIBBLE SWAP
E62A E80300 1352 CALL XLAT_PR ; DO THE HIGH NIBBLE DISPLAY
E62D 58 1353 POP AX ; RECOVER THE NIBBLE
E62E 240F 1354 AND AL,0FH ; ISOLATE TO LOW NIBBLE
1355 ; FALL INTO LOW NIBBLE CONVERSION

```



```

LOC OBJ          LINE    SOURCE
E630             1356  XLAT_PR PROC   NEAR           ; CONVERT 00-0F TO ASCII CHARACTER
E630 0490        1357          ADD     AL,090H           ; ADD FIRST CONVERSION FACTOR
E632 27          1358          DAA                    ; ADJUST FOR NUMERIC AND ALPHA RANGE
E633 1440        1359          ADC     AL,040H           ; ADD CONVERSION AND ADJUST LOW NIBBLE
E635 27          1360          DAA                    ; ADJUST HI NIBBLE TO ASCII RANGE
E636             1361  PRT_HEX PROC   NEAR
E636 B40E        1362          MOV     AH,14             ; DISPLAY CHAR. IN AL
E636 B700        1363          MOV     BH,0
E63A CD10        1364          INT     10H              ; CALL VIDEO_IO
E63C C3          1365          RET
E636             1366  PRT_HEX ENDP
E636             1367  XLAT_PR ENDP
E636             1368  XPC_BYTE      ENDP
E636             1369
E63D             1370  F4 LABEL WORD           ; PRINTER SOURCE TABLE
E63D BC03        1371          DW     3BCH
E63F 7803        1372          DW     378H
E641 7802        1373          DW     278H
E643             1374  F4E LABEL WORD
E643             1375
E643             1376  ;-----
E643             1377  ;   THIS PROCEDURE WILL SEND A SOFTWARE RESET TO THE KEYBOARD.   ;
E643             1378  ;   SCAN CODE 'AA' SHOULD BE RETURNED TO THE CPU.                 ;
E643             1379  ;-----
E643             1380  KBD_RESET    PROC   NEAR
E643 B00C        1381          MOV     AL,0CH           ; SET KBD CLK LINE LOW
E645 E661        1382          OUT    PORT_B,AL        ; WRITE 8255 PORT B
E647 B95629      1383          MOV     CX,10582        ; HOLD KBD CLK LOW FOR 20 MS
E64A             1384  G8:
E64A E2FE        1385          LOOP   G8                ; LOOP FOR 20 MS
E64C B0CC        1386          MOV     AL,0CCH         ; SET CLK, ENABLE LINES HIGH
E64E E661        1387          OUT    PORT_B,AL
E650             1388  SP_TEST:      ; ENTRY FOR MANUFACTURING TEST 2
E650 B04C        1389          MOV     AL,4CH           ; SET KBD CLK HIGH, ENABLE LOW
E652 E661        1390          OUT    PORT_B,AL
E654 B0FD        1391          MOV     AL,0FDH         ; ENABLE KEYBOARD INTERRUPTS
E656 E621        1392          OUT    INTA01,AL        ; WRITE 8259 IMR
E658 FB          1393          STI                    ; ENABLE SYSTEM INTERRUPTS
E659 B400        1394          MOV     AH,0             ; RESET INTERRUPT INDICATOR
E65B 2BC9        1395          SUB     CX,CX            ; SETUP INTERRUPT TIMEOUT CNT
E65D             1396  G9:
E65D F6C4FF      1397          TEST   AH,OFFH          ; DID A KEYBOARD INTR OCCUR?
E660 7502        1398          JNZ    G10              ; YES - READ SCAN CODE RETURNED
E662 E2F9        1399          LOOP   G9               ; NO - LOOP TILL TIMEOUT
E664             1400  G10:
E664 E460        1401          IN     AL,PORT_A        ; READ KEYBOARD SCAN CODE
E666 8A08        1402          MOV     BL,AL           ; SAVE SCAN CODE JUST READ
E668 B0CC        1403          MOV     AL,0CCH         ; CLEAR KEYBOARD
E66A E661        1404          OUT    PORT_B,AL
E66C C3          1405          RET                       ; RETURN TO CALLER
E66C             1406  KBD_RESET    ENDP
E66C             1407
E66C             1408  ;-----
E66C             1409  ;   BLINK LED PROCEDURE FOR MFG BURN-IN AND RUN-IN TESTS       ;
E66C             1410  ;   IF LED IS ON, TURN IT OFF. IF OFF, TURN ON.                 ;
E66C             1411  ;-----
E66D             1412  BLINK_INT    PROC   NEAR
E66D FB          1413          STI
E66E 50          1414          PUSH  AX                ; SAVE AX REG CONTENTS
E66F E461        1415          IN     AL,PORT_B        ; READ CURRENT VAL OF PORT B
E671 8AE0        1416          MOV     AH,AL
E673 F6D0        1417          NOT    AL                ; FLIP ALL BITS
E675 2440        1418          AND    AL,01000000B     ; ISOLATE CONTROL BIT
E677 80E4BF      1419          AND    AH,10111111B    ; MASK OUT OF ORIGINAL VAL
E67A 0AC4        1420          OR     AL,AH            ; OR NEW CONTROL BIT IN
E67C E661        1421          OUT    PORT_B,AL
E67E B020        1422          MOV     AL,E0I
E680 E620        1423          OUT    INTA00,AL
E682 58          1424          POP    AX                ; RESTORE AX REG
E683 CF          1425          IRET
E683             1426  BLINK_INT    ENDP
E683             1427
E683             1428  ;----- CHECKSUM AND CALL INIT CODE IN OPTIONAL ROMS
E683             1429
E684             1430  ROM_CHECK    PROC   NEAR
E684 B84000      1431          MOV     AX,DATA         ; SET ES=DATA
E687 8EC0        1432          MOV     ES,AX

```

```

LOC OBJ          LINE    SOURCE
E6B9 2AE4        1433    SUB    AH,AH          ; ZERO OUT AH
E6B8 8A4702      1434    MOV    AL,[BX+2]      ; GET LENGTH INDICATOR
E6B6 B109        1435    MOV    CL,09H         ; MULTIPLY BY 512
E690 D3E0        1436    SHL   AX,CL
E692 8BC8        1437    MOV    CX,AX          ; SET COUNT
E694 51          1438    PUSH  CX
E695 B104        1439    MOV    CL,4
E697 D3E8        1440    SHR   AX,CL
E699 03D0        1441    ADD   DX,AX           ; SET POINTER TO NEXT MODULE
E69B 59          1442    POP   CX
1443
E69C E8B005       1444    CALL  ROS_CHECKSUM_CNT ; DO CHECKSUM
E69F 7405        1445    JZ    ROM_CHECK_1
E6A1 E86501       1446    CALL  ROM_ERR         ; PRINT ERROR INFO
E6A4 EB13        1447    JMP   SHORT ROM_CHECK_END
E6A6
ROM_CHECK_1:
E6A6 52          1449    PUSH  DX              ; SAVE POINTER
E6A7 26C70600010300 1450    MOV   ES:IO_ROM_INIT,0003H ; LOAD OFFSET
E6A8 268C1E0201   1451    MOV   ES:IO_ROM_SEG,DS ; LOAD SEGMENT
E6B3 26FF1E0001   1452    CALL  DWORD PTR ES:IO_ROM_INIT ; CALL INIT RTN.
E6B8 5A          1453    POP   DX
E6B9
ROM_CHECK_END:
E6B9 C3          1454    RET
1455
ROM_CHECK      ENDP
1457
1458
;-----
1459 ; THIS SUBROUTINE WILL PRINT A MESSAGE ON THE DISPLAY :
1460 ; :
1461 ; ENTRY REQUIREMENTS: :
1462 ; SI = OFFSET(ADDRESS) OF MESSAGE BUFFER :
1463 ; CX = MESSAGE BYTE COUNT :
1464 ; MAXIMUM MESSAGE LENGTH IS 36 CHARACTERS :
;-----
1465
E6BA
P_MSG PROC NEAR
E6BA E80118        1467    CALL  DDS
E6BD 803E120001   1468    CMP   MFG_TST,1      ; MFG TEST MODE?
E6C2 7505        1469    JNE   G12             ; NO - DISPLAY ERROR MSG
E6C4 B601        1470    MOV   DH,1           ; YES - SETUP TO BEEP SPEAKER
E6C6 E906FF      1471    JMP   ERR_BEEP       ; YES - BEEP SPEAKER
E6C9
G12:
E6C9 2E8A04        1472    MOV   AL,CS:[SI]     ; WRITE_MSG:
; PUT CHAR IN AL
E6CC 46          1473    INC   SI              ; POINT TO NEXT CHAR
E6CD 50          1474    PUSH  AX              ; SAVE PRINT CHAR
E6CE E865FF      1475    CALL  PRT_HEX        ; CALL VIDEO_IO
E6D1 58          1476    POP   AX              ; RECOVER PRINT CHAR
E6D2 3C0A        1477    CMP   AL,10          ; HAS IT LINE FEED
E6D4 75F3        1478    JNE   G12            ; NO,KEEP PRINTING STRING
E6D6 C3          1480    RET
1481
P_MSG ENDP
1482
E6D7 20524F40     1483    F3A  DB ' ROM',13,10
E6DB 00
E6DC 0A
1484
E6DD
D_EOI PROC NEAR
E6DD 50          1485    PUSH  AX
E6DE B020        1486    MOV   AL,20H
E6E0 E620        1487    OUT  20H,AL
E6E2 58          1488    POP   AX
E6E3 CF          1489    IRET
1490
D_EOI ENDP
1491
1492
1493 ;--- INT 19 -----
1494 ; BOOT STRAP LOADER :
1495 ; IF A 5 1/4" DISKETTE DRIVE IS AVAILABLE ON THE SYSTEM, :
1496 ; TRACK 0, SECTOR 1 IS READ INTO THE BOOT LOCATION :
1497 ; (SEGMENT 0, OFFSET 7C00) AND CONTROL IS TRANSFERRED :
1498 ; THERE. :
1499 ; :
1500 ; IF THERE IS NO DISKETTE DRIVE, OR IF THERE IS A :
1501 ; HARDWARE ERROR CONTROL IS TRANSFERRED TO THE RESIDENT :
1502 ; BASIC ENTRY POINT. :
1503 ; :
1504 ; IPL ASSUMPTIONS: :
1505 ; 8255 PORT 60H BIT 0 = 1 IF IPL FROM DISKETTE :
1506 ;-----
1507 ASSUME CS:CODE,DS:ABS0

```

```

LOC OBJ          LINE  SOURCE

1508
1509          ;----- IPL WAS SUCCESSFUL
1510
E6E4            1511      H4:
E6E4 EA007C0000  1512          JMP      BOOT_LOCN
E6F2            1513          ORG      0E6F2H
E6F2            1514      BOOT_STRAP  PROC    NEAR
E6F2 FB         1515          STI             ; ENABLE INTERRUPTS
E6F3 2BC0       1516          SUB      AX,AX
E6F5 8ED8       1517          MOV      DS,AX
1518
1519          ;----- RESET DISKETTE PARAMETER TABLE VECTOR
1520
E6F7 C7067800C7EF 1521          MOV      WORD PTR DISK_POINTER,OFFSET DISK_BASE
E6FD 8C0E7A00     1522          MOV      WORD PTR DISK_POINTER+2,CS
E701 A11004       1523          MOV      AX,DATA_WORD[OFFSET EQUIP_FLAG] ; GET THE EQUIPMENT SWITCHES
E704 A801         1524          TEST     AL,1             ; ISOLATE IPL SENSE SWITCH
E706 741E         1525          JZ       H3             ; GO TO CASSETTE BASIC ENTRY POINT
1526
1527          ;----- MUST LOAD SYSTEM FROM DISKETTE -- CX HAS RETRY COUNT
1528
E708 B90400       1529          MOV      CX,4             ; SET RETRY COUNT
E70B            1530          H1:             ; IPL_SYSTEM
E70B 51          1531          PUSH     CX             ; SAVE RETRY COUNT
E70C B400         1532          MOV      AH,0           ; RESET THE DISKETTE SYSTEM
E70E CD13         1533          INT     13H           ; DISKETTE_IO
E710 720F         1534          JC      H2             ; IF ERROR, TRY AGAIN
E712 B80102       1535          MOV      AX,201H        ; READ IN THE SINGLE SECTOR
E715 2BD2         1536          SUB      DX,DX
E717 8EC2         1537          MOV      ES,DX
E719 BB007C       1538          MOV      BX,OFFSET BOOT_LOCN
E71C B90100       1539          MOV      CX,1           ; SECTOR 1, TRACK 0
E71F CD13         1540          INT     13H           ; DISKETTE_IO
E721 59           1541          H2:     POP      CX           ; RECOVER RETRY COUNT
E722 73C0         1542          JNC     H4             ; CF SET BY UNSUCCESSFUL READ
E724 E2E5         1543          LOOP   H1             ; DO IT FOR RETRY TIMES
1544
1545          ;----- UNABLE TO IPL FROM THE DISKETTE
1546
E726            1547          H3:             ; CASSETTE_JUMP:
E726 CD18         1548          INT     18H           ; USE INTERRUPT VECTOR TO GET TO BASIC
1549          BOOT_STRAP  ENDP
1550
1551          ;-----INT 14-----
1552          ; RS232_IO
1553          ; THIS ROUTINE PROVIDES BYTE STREAM I/O TO THE COMMUNICATIONS
1554          ; PORT ACCORDING TO THE PARAMETERS:
1555          ; (AH)=0 INITIALIZE THE COMMUNICATIONS PORT
1556          ; (AL) HAS PARAMETERS FOR INITIALIZATION
1557          ;
1558          ; 7      6      5      4      3      2      1      0
1559          ; ----- BAUD RATE -- -PARITY-- STOPBIT --WORD LENGTH--
1560          ; 000 - 110          X0 - NONE      0 - 1    10 - 7 BITS
1561          ; 001 - 150          01 - ODD      1 - 2    11 - 8 BITS
1562          ; 010 - 300          11 - EVEN
1563          ; 011 - 600
1564          ; 100 - 1200
1565          ; 101 - 2400
1566          ; 110 - 4800
1567          ; 111 - 9600
1568          ;
1569          ; ON RETURN, CONDITIONS SET AS IN CALL TO COMMO STATUS (AH=3)
1570          ; (AH)=1 SEND THE CHARACTER IN (AL) OVER THE COMMO LINE
1571          ; (AL) REGISTER IS PRESERVED
1572          ; ON EXIT, BIT 7 OF AH IS SET IF THE ROUTINE WAS UNABLE
1573          ; TO TRANSMIT THE BYTE OF DATA OVER THE LINE.
1574          ; IF BIT 7 OF AH IS NOT SET, THE REMAINDER OF AH
1575          ; IS SET AS IN A STATUS REQUEST, REFLECTING THE
1576          ; CURRENT STATUS OF THE LINE.
1577          ; (AH)=2 RECEIVE A CHARACTER IN (AL) FROM COMMO LINE BEFORE
1578          ; RETURNING TO CALLER
1579          ; ON EXIT, AH HAS THE CURRENT LINE STATUS, AS SET BY THE
1580          ; THE STATUS ROUTINE, EXCEPT THAT THE ONLY BITS
1581          ; LEFT ON ARE THE ERROR BITS (7,4,3,2,1)
1582          ; IF AH HAS BIT 7 ON (TIME OUT) THE REMAINING
1583          ; BITS ARE NOT PREDICTABLE.
1584          ; THUS, AH IS NON ZERO ONLY WHEN AN ERROR

```

```

LOC OBJ          LINE  SOURCE
1585             ;                OCCURRED.                ;
1586             ; (AH)=3 RETURN THE COMMO PORT STATUS IN (AX) ;
1587             ; AH CONTAINS THE LINE STATUS              ;
1588             ; BIT 7 = TIME OUT                          ;
1589             ; BIT 6 = TRANS SHIFT REGISTER EMPTY       ;
1590             ; BIT 5 = TRAN HOLDING REGISTER EMPTY       ;
1591             ; BIT 4 = BREAK DETECT                      ;
1592             ; BIT 3 = FRAMING ERROR                    ;
1593             ; BIT 2 = PARITY ERROR                    ;
1594             ; BIT 1 = OVERRUN ERROR                    ;
1595             ; BIT 0 = DATA READY                      ;
1596             ; AL CONTAINS THE MODEM STATUS              ;
1597             ; BIT 7 = RECEIVED LINE SIGNAL DETECT      ;
1598             ; BIT 6 = RING INDICATOR                   ;
1599             ; BIT 5 = DATA SET READY                  ;
1600             ; BIT 4 = CLEAR TO SEND                    ;
1601             ; BIT 3 = DELTA RECEIVE LINE SIGNAL DETECT  ;
1602             ; BIT 2 = TRAILING EDGE RING DETECTOR      ;
1603             ; BIT 1 = DELTA DATA SET READY            ;
1604             ; BIT 0 = DELTA CLEAR TO SEND              ;
1605             ;                                           ;
1606             ; (DX) = PARAMETER INDICATING WHICH RS232 CARD (0,1 ALLOWED) ;
1607             ;                                           ;
1608             ; DATA AREA RS232_BASE CONTAINS THE BASE ADDRESS OF THE 8250 ON THE ;
1609             ; CARD LOCATION 400H CONTAINS UP TO 4 RS232 ADDRESSES POSSIBLE ;
1610             ; DATA AREA LABEL RS232_TII_OUT (BYTE) CONTAINS OUTER LOOP COUNT ;
1611             ; VALUE FOR TIMEOUT (DEFAULT=1)              ;
1612             ; OUTPUT                                     ;
1613             ; AX MODIFIED ACCORDING TO PARMS OF CALL     ;
1614             ; ALL OTHERS UNCHANGED                      ;
1615             ;-----
1616             ; ASSUME CS:CODE,DS:DATA
1617             ; ORG 0E729H
E729             A1 LABEL WORD ; TABLE OF INIT VALUE
E729             DW 1047 ; 110 BAUD
E72B 0003       DW 768 ; 150
E72D 8001       DW 384 ; 300
E72F C000       DW 192 ; 600
E731 6000       DW 96 ; 1200
E733 3000       DW 48 ; 2400
E735 1800       DW 24 ; 4800
E737 0C00       DW 12 ; 9600
1627
E739             RS232_IO PROC FAR
1629
1630             ;----- VECTOR TO APPROPRIATE ROUTINE
1631
E739 FB         STI ; INTERRUPTS BACK ON
E73A 1E         PUSH DS ; SAVE SEGMENT
E73B 52         PUSH DX
E73C 56         PUSH SI
E73D 57         PUSH DI
E73E 51         PUSH CX
E73F 53         PUSH BX
E740 8BF2       MOV SI,DX ; RS232 VALUE TO SI
E742 8BFA       MOV DI,DX
E744 D1E6       SHL SI,1 ; WORD OFFSET
E746 E8F517     CALL DDS
E749 8B14       MOV DX,RS232_BASE[SI] ; GET BASE ADDRESS
E74B 0BD2       OR DX,DX ; TEST FOR 0 BASE ADDRESS
E74D 7413       JZ A3 ; RETURN
E74F 0AE4       OR AH,AH ; TEST FOR (AH)=0
E751 7416       JZ A4 ; COMMUN INIT
E753 FECC       DEC AH ; TEST FOR (AH)=1
E755 74A5       JZ A5 ; SEND AL
E757 FECC       DEC AH ; TEST FOR (AH)=2
E759 746A       JZ A12 ; RECEIVE INTO AL
E75B           A2:
E75B FECC       DEC AH ; TEST FOR (AH)=3
E75D 7503       JNZ A3
E75F E98300     JMP A18 ; COMMUNICATION STATUS
E762           A3:
E762 5B         POP BX ; RETURN FROM RS232
E763 59         POP CX
E764 5F         POP DI
E765 5E         POP SI
E766 5A         POP DX

```

LOC OBJ	LINE	SOURCE	
E767 1F	1662	POP DS	
E768 CF	1663	IRET	; RETURN TO CALLER, NO ACTION
	1664		
	1665	;----- INITIALIZE THE COMMUNICATIONS PORT	
	1666		
E769	1667	A4:	
E769 8AE0	1668	MOV AH,AL	; SAVE INIT PARMS IN AH
E76B 83C203	1669	ADD DX,3	; POINT TO 8250 CONTROL REGISTER
E76E B080	1670	MOV AL,80H	
E770 EE	1671	OUT DX,AL	; SET DLAB=1
	1672		
	1673	;----- DETERMINE BAUD RATE DIVISOR	
	1674		
E771 8AD4	1675	MOV DL,AH	; GET PARMS TO DL
E773 B104	1676	MOV CL,4	
E775 D2C2	1677	ROL DL,CL	
E777 81E20E00	1678	AND DX,0EH	; ISOLATE THEM
E77B BF29E7	1679	MOV DI,OFFSET A1	; BASE OF TABLE
E77E 03FA	1680	ADD DI,DX	; PUT INTO INDEX REGISTER
E780 8B14	1681	MOV DX,RS232_BASE[SI]	; POINT TO HIGH ORDER OF DIVISOR
E782 42	1682	INC DX	
E783 2E8A4501	1683	MOV AL,CS:[DI]+1	; GET HIGH ORDER OF DIVISOR
E787 EE	1684	OUT DX,AL	; SET MS OF DIV TO 0
E788 4A	1685	DEC DX	
E789 2E8A05	1686	MOV AL,CS:[DI]	; GET LOW ORDER OF DIVISOR
E78C EE	1687	OUT DX,AL	; SET LOW OF DIVISOR
E78D 83C203	1688	ADD DX,3	
E790 8AC4	1689	MOV AL,AH	; GET PARMS BACK
E792 241F	1690	AND AL,01FH	; STRIP OFF THE BAUD BITS
E794 EE	1691	OUT DX,AL	; LINE CONTROL TO 8 BITS
E795 4A	1692	DEC DX	
E796 4A	1693	DEC DX	
E797 B000	1694	MOV AL,0	
E799 EE	1695	OUT DX,AL	; INTERRUPT ENABLES ALL OFF
E79A EB49	1696	JMP SHORT A18	; COM_STATUS
	1697		
	1698	;----- SEND CHARACTER IN (AL) OVER COMMO LINE	
	1699		
E79C	1700	A5:	
E79C 50	1701	PUSH AX	; SAVE CHAR TO SEND
E79D 83C204	1702	ADD DX,4	; MODEM CONTROL REGISTER
E7A0 B003	1703	MOV AL,3	; DTR AND RTS
E7A2 EE	1704	OUT DX,AL	; DATA TERMINAL READY, REQUEST TO SEND
E7A3 42	1705	INC DX	; MODEM STATUS REGISTER
E7A4 42	1706	INC DX	
E7A5 B730	1707	MOV BH,30H	; DATA SET READY & CLEAR TO SEND
E7A7 E84800	1708	CALL WAIT_FOR_STATUS	; ARE BOTH TRUE
E7AA 7408	1709	JE A9	; YES, READY TO TRANSMIT CHAR
E7AC	1710	A7:	
E7AC 59	1711	POP CX	
E7AD 8AC1	1712	MOV AL,CL	; RELOAD DATA BYTE
E7AF	1713	A8:	
E7AF 80CC80	1714	OR AH,80H	; INDICATE TIME OUT
E7B2 EBAE	1715	JMP A3	; RETURN
E7B4	1716	A9:	; CLEAR_TO_SEND
E7B4 4A	1717	DEC DX	; LINE STATUS REGISTER
E7B5	1718	A10:	; WAIT_SEND
E7B5 B720	1719	MOV BH,20H	; IS TRANSMITTER READY
E7B7 E63800	1720	CALL WAIT_FOR_STATUS	; TEST FOR TRANSMITTER READY
E7BA 75F0	1721	JNZ A7	; RETURN WITH TIME OUT SET
E7BC	1722	A11:	; OUT_CHAR
E7BC 83EA05	1723	SUB DX,5	; DATA PORT
E7BF 59	1724	POP CX	; RECOVER IN CX TEMPORARILY
E7C0 8AC1	1725	MOV AL,CL	; MOVE CHAR TO AL FOR OUT, STATUS IN AH
E7C2 EE	1726	OUT DX,AL	; OUTPUT CHARACTER
E7C3 EB9D	1727	JMP A3	; RETURN
	1728		
	1729	;----- RECEIVE CHARACTER FROM COMMO LINE	
	1730		
E7C5	1731	A12:	
E7C5 83C204	1732	ADD DX,4	; MODEM CONTROL REGISTER
E7C8 B001	1733	MOV AL,1	; DATA TERMINAL READY
E7CA EE	1734	OUT DX,AL	
E7CB 42	1735	INC DX	; MODEM STATUS REGISTER
E7CC 42	1736	INC DX	
E7CD	1737	A13:	; WAIT_DSR
E7CD B720	1738	MOV BH,20H	; DATA SET READY

LOC	OBJ	LINE	SOURCE	
E7CF	E62000	1739	CALL	WAIT_FOR_STATUS ; TEST FOR DSR
E7D2	75DB	1740	JNZ	A8 ; RETURN WITH ERROR
E7D4		1741	A15:	DEC DX ; WAIT_DSR_END
E7D4	4A	1742	DEC	DX ; LINE STATUS REGISTER
E7D5		1743	A16:	MOV BH,1 ; WAIT_REC V
E7D5	B701	1744	MOV	BH,1 ; RECEIVE BUFFER FULL
E7D7	E61800	1745	CALL	WAIT_FOR_STATUS ; TEST FOR REC. BUFF. FULL
E7DA	75D3	1746	JNZ	A8 ; SET TIME OUT ERROR
E7DC		1747	A17:	AND AH,00011110B ; GET_CHAR
E7DC	80E41E	1748	AND	AH,00011110B ; TEST FOR ERR CONDITIONS ON REC V CHAR
E7DF	8B14	1749	MOV	DX,RS232_BASE[SI] ; DATA PORT
E7E1	EC	1750	IN	AL,DX ; GET CHARACTER FROM LINE
E7E2	E970FF	1751	JMP	A3 ; RETURN
		1752		
		1753		;----- COMMO PORT STATUS ROUTINE
		1754		
E7E5		1755	A18:	MOV DX,RS232_BASE[SI] ; CONTROL PORT
E7E5	8B14	1756	MOV	DX,RS232_BASE[SI] ; CONTROL PORT
E7E7	83C205	1757	ADD	DX,5 ; GET LINE CONTROL STATUS
E7EA	EC	1758	IN	AL,DX ; PUT IN AH FOR RETURN
E7EB	8AE0	1759	MOV	AH,AL ; POINT TO MODEM STATUS REGISTER
E7ED	42	1760	INC	DX ; GET MODEM CONTROL STATUS
E7EE	EC	1761	IN	AL,DX ; RETURN
E7EF	E970FF	1762	JMP	A3 ; RETURN
		1763		;-----
		1764		; WAIT FOR STATUS ROUTINE ;
		1765		;
		1766		; ENTRY: ;
		1767		; BH=STATUS BIT(S) TO LOOK FOR, ;
		1768		; DX=ADDR. OF STATUS REG ;
		1769		; EXIT: ;
		1770		; ZERO FLAG ON = STATUS FOUND ;
		1771		; ZERO FLAG OFF = TIMEOUT. ;
		1772		; AH=LAST STATUS READ ;
		1773		;-----
E7F2		1774	WAIT_FOR_STATUS PROC NEAR	
E7F2	8A5D7C	1775	MOV	BL,RS232_TIM_OUT[DI] ; LOAD OUTER LOOP COUNT
E7F5		1776	WFS0:	SUB CX,CX
E7F5	2BC9	1777	SUB	CX,CX
E7F7		1778	WFS1:	IN AL,DX ; GET STATUS
E7F7	EC	1779	IN	AL,DX ; GET STATUS
E7F8	8AE0	1780	MOV	AH,AL ; MOVE TO AH
E7FA	22C7	1781	AND	AL,BH ; ISOLATE BITS TO TEST
E7FC	3AC7	1782	CHP	AL,BH ; EXACTLY = TO MASK
E7FE	7408	1783	JE	WFS_END ; RETURN WITH ZERO FLAG ON
E800	E2F5	1784	LOOP	WFS1 ; TRY AGAIN
E802	FECB	1785	DEC	BL
E804	75EF	1786	JNZ	WFS0
E806	0AFF	1787	OR	BH,BH ; SET ZERO FLAG OFF
E808		1788	WFS_END:	RET
E808	C3	1789	RET	
		1790	WAIT_FOR_STATUS ENDP	
		1791	RS232_IO ENDP	
		1792		
		1793		;-----
		1794		; PRINT ADDRESS AND ERROR MESSAGE FOR ROM CHECKSUM ERRORS ;
		1795		;
		1796		;-----
E809		1796	ROM_ERR PROC NEAR	
E809	52	1797	PUSH	DX ; SAVE POINTER
E80A	50	1798	PUSH	AX
E80B	8CDA	1799	MOV	DX,DS ; GET ADDRESS POINTER
E80D	81FA00C8	1800	CHP	DX,0C800H
E811	7E13	1801	JLE	ROM_ERR_BEEP ; SPECIAL ERROR INDICATION
E813	8AC6	1802	MOV	AL,DH
E815	E80DFE	1803	CALL	XPC_BYTE ; DISPLAY ADDRESS
E818	8AC2	1804	MOV	AL,DL
E81A	E808FE	1805	CALL	XPC_BYTE
E81D	BED7E6	1806	MOV	SI,OFFSET F3A ; DISPLAY ERROR MSG
E820	E897FE	1807	CALL	P_MSG
E823		1808	ROM_ERR_END:	
E823	58	1809	POP	AX
E824	5A	1810	POP	DX
E825	C3	1811	RET	
E826		1812	ROM_ERR_BEEP:	
E826	8A0201	1813	MOV	DX,0102H ; BEEP 1 LONG, 2 SHORT
E829	E8A3FD	1814	CALL	ERR_BEEP
E82C	EBF5	1815	JMP	SHORT ROM_ERR_END

```

1016 ROM_ERR ENDP
1017
1018 ;----- INT 16 -----
1019 ; KEYBOARD I/O
1020 ; THESE ROUTINES PROVIDE KEYBOARD SUPPORT
1021 ; INPUT
1022 ; (AH)=0 READ THE NEXT ASCII CHARACTER STRUCK FROM THE KEYBOARD
1023 ; RETURN THE RESULT IN (AL), SCAN CODE IN (AH)
1024 ; (AH)=1 SET THE Z FLAG TO INDICATE IF AN ASCII CHARACTER IS
1025 ; AVAILABLE TO BE READ.
1026 ; (ZF)=1 -- NO CODE AVAILABLE
1027 ; (ZF)=0 -- CODE IS AVAILABLE
1028 ; IF ZF = 0, THE NEXT CHARACTER IN THE BUFFER TO BE READ
1029 ; IS IN AX, AND THE ENTRY REMAINS IN THE BUFFER
1030 ; (AH)=2 RETURN THE CURRENT SHIFT STATUS IN AL REGISTER
1031 ; THE BIT SETTINGS FOR THIS CODE ARE INDICATED IN THE
1032 ; THE EQUATES FOR KB_FLAG
1033 ; OUTPUT
1034 ; AS NOTED ABOVE, ONLY AX AND FLAGS CHANGED
1035 ; ALL REGISTERS PRESERVED
1036 ;-----
1037 ASSUME CS:CODE,DS:DATA
1038 ORG 0E82EH
E02E E82E 1039 KEYBOARD_IO PROC FAR
E02E FB 1040 STI ; INTERRUPTS BACK ON
E02F 1E 1041 PUSH DS ; SAVE CURRENT DS
E030 53 1042 PUSH BX ; SAVE BX TEMPORARILY
E031 E0A17 1043 CALL DDS
E034 0AE4 1044 OR AH,AH ; AH=0
E036 740A 1045 JZ K1 ; ASCII_READ
E038 FECC 1046 DEC AH ; AH=1
E03A 741E 1047 JZ K2 ; ASCII_STATUS
E03C FECC 1048 DEC AH ; AH=2
E03E 742B 1049 JZ K3 ; SHIFT_STATUS
E040 EB2C 1050 JMP SHORT INT10_END ; EXIT
1051
1052 ;----- READ THE KEY TO FIGURE OUT WHAT TO DO
1053
E042 1054 K1: ; ASCII READ
E042 FB 1055 STI ; INTERRUPTS BACK ON DURING LOOP
E043 90 1056 NOP ; ALLOW AN INTERRUPT TO OCCUR
E044 FA 1057 CLI ; INTERRUPTS BACK OFF
E045 8B1E1A00 1058 MOV BX,BUFFER_HEAD ; GET POINTER TO HEAD OF BUFFER
E049 3B1E1C00 1059 CMP BX,BUFFER_TAIL ; TEST END OF BUFFER
E04D 74F3 1060 JZ K1 ; LOOP UNTIL SOMETHING IN BUFFER
E04F 8B07 1061 MOV AX,[BX] ; GET SCAN CODE AND ASCII CODE
E051 E01D00 1062 CALL K4 ; MOVE POINTER TO NEXT POSITION
E054 891E1A00 1063 MOV BUFFER_HEAD,BX ; STORE VALUE IN VARIABLE
E056 EB14 1064 JMP SHORT INT10_END ; RETURN
1065
1066 ;----- ASCII STATUS
1067
E05A 1068 K2:
E05A FA 1069 CLI ; INTERRUPTS OFF
E05B 8B1E1A00 1070 MOV BX,BUFFER_HEAD ; GET HEAD POINTER
E05F 3B1E1C00 1071 CMP BX,BUFFER_TAIL ; IF EQUAL (Z=1) THEN NOTHING THERE
E063 8B07 1072 MOV AX,[BX]
E065 FB 1073 STI ; INTERRUPTS BACK ON
E066 5B 1074 POP BX ; RECOVER REGISTER
E067 1F 1075 POP DS ; RECOVER SEGMENT
E068 CA0200 1076 RET 2 ; THROW AWAY FLAGS
1077
1078 ;----- SHIFT STATUS
1079
E06B 1080 K3:
E06B A01700 1081 MOV AL,KB_FLAG ; GET THE SHIFT STATUS FLAGS
E06E 1082 INT10_END:
E06E 5B 1083 POP BX ; RECOVER REGISTER
E06F 1F 1084 POP DS ; RECOVER REGISTERS
E070 CF 1085 IRET ; RETURN TO CALLER
1086 KEYBOARD_IO ENDP
1087
1088 ;----- INCREMENT A BUFFER POINTER
1089
E071 1090 K4 PROC NEAR
E071 43 1091 INC BX ; MOVE TO NEXT WORD IN LIST
E072 43 1092 INC BX

```

LOC OBJ	LINE	SOURCE
E873 3B1E8200	1893	CMF BX,BUFFER_END ; AT END OF BUFFER?
E877 7504	1894	JNE K5 ; NO, CONTINUE
E879 8B1E8000	1895	MOV BX,BUFFER_START ; YES, RESET TO BUFFER BEGINNING
E87D	1896	K5:
E87D C3	1897	RET
	1898	K4 ENDP
	1899	
	1900	!----- TABLE OF SHIFT KEYS AND MASK VALUES
	1901	
E87E	1902	K6 LABEL BYTE
E87E 52	1903	DB INS_KEY ; INSERT KEY
E87F 3A	1904	DB CAPS_KEY,NUM_KEY,SCROLL_KEY,ALT_KEY,CTL_KEY
E880 45		
E881 46		
E882 38		
E883 10		
E884 2A	1905	DB LEFT_KEY,RIGHT_KEY
E885 36		
0008	1906	K6L EQU 9-K6
	1907	
	1908	!----- SHIFT_MASK_TABLE
	1909	
E886	1910	K7 LABEL BYTE
E886 80	1911	DB INS_SHIFT ; INSERT MODE SHIFT
E887 40	1912	DB CAPS_SHIFT,NUM_SHIFT,SCROLL_SHIFT,ALT_SHIFT,CTL_SHIFT
E888 20		
E889 10		
E88A 08		
E88B 04		
E88C 02	1913	DB LEFT_SHIFT,RIGHT_SHIFT
E88D 01		
	1914	
	1915	!----- SCAN CODE TABLES
	1916	
E88E 1B	1917	K8 DB 27,-1,0,-1,-1,-1,30,-1
E88F FF		
E890 00		
E891 FF		
E892 FF		
E893 FF		
E894 1E		
E895 FF		
E896 FF	1918	DB -1,-1,-1,31,-1,127,-1,17
E897 FF		
E898 FF		
E899 1F		
E89A FF		
E89B 7F		
E89C FF		
E89D 11		
E89E 17	1919	DB 23,5,18,20,25,21,9,15
E89F 05		
E8A0 12		
E8A1 14		
E8A2 19		
E8A3 15		
E8A4 09		
E8A5 0F		
E8A6 10	1920	DB 16,27,29,10,-1,1,19
E8A7 1B		
E8A8 10		
E8A9 0A		
E8AA FF		
E8AB 01		
E8AC 13		
E8AD 04	1921	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	1922	DB -1,-1,28,26,24,3,22,2
E8B7 FF		
E8B8 1C		

LOC OBJ	LINE	SOURCE		
E8B9 1A				
E8BA 18				
E8BB 03				
E8BC 16				
E8BD 02				
E8BE 0E	1923		DB	14,13,-1,-1,-1,-1,-1,-1
E8BF 0D				
E8C0 FF				
E8C1 FF				
E8C2 FF				
E8C3 FF				
E8C4 FF				
E8C5 FF				
E8C6 20	1924		DB	' ', -1
E8C7 FF				
E8C8	1925	;----- CTL TABLE SCAN		
E8C8 5E	1926	K9 LABEL BYTE		
E8C9 5F	1927		DB	94,95,96,97,98,99,100,101
E8CA 60				
E8CB 61				
E8CC 62				
E8CD 63				
E8CE 64				
E8CF 65				
E8D0 66	1928		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	1929		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	1930		DB	-1
E8E1	1931	;----- LC TABLE		
E8E1 1B	1932	K10 LABEL BYTE		
E8E2 31323334353637	1933		DB	01BH, '1234567890-', 08H, 09H
3839302D3D				
E8EE 08				
E8EF 09				
E8F0 71776572747975	1934		DB	'qwertyuiop[]', 0DH, -1, 'asdfghjkl;', 027H
696F705B5D				
E8FC 0D				
E8FD FF				
E8FE 6173646667686A				
686C3B				
E908 27				
E909 60	1935		DB	60H, -1, 5CH, 'zxcvbnm,./', -1, 'M', -1, ' '
E90A FF				
E90B 5C				
E90C 7A786376626E6D				
2C2E2F				
E916 FF				
E917 2A				
E918 FF				
E919 20				
E91A FF	1936		DB	-1
E91B	1937	;----- UC TABLE		
E91B 1B	1938	K11 LABEL BYTE		
E91C 21402324	1939		DB	27, '!@#\$', 37, 05EH, '&*()_+', 08H, 0
E920 25				
E921 5E				
E922 262A28295F2B				
E928 08				
E929 00				
E92A 51574552545955	1940		DB	'QWERTYUIOP()', 0DH, -1, 'ASDFGHJKL:'''
494F507B7D				

LOC OBJ	LINE	SOURCE
E936 0D		
E937 FF		
E938 4153444647404A		
4B4C3A22		
E943 7E	1941	DB 07EH,-1,' ZXCVBNM*>?','-1,0,-1','-1
E944 FF		
E945 7C5A584356424E		
4D3C3E3F		
E950 FF		
E951 00		
E952 FF		
E953 20		
E954 FF		
	1942	;------ UC TABLE SCAN
E955	1943	K12 LABEL BYTE
E955 54	1944	DB 84,85,86,87,88,89,90
E956 55		
E957 56		
E958 57		
E959 58		
E95A 59		
E95B 5A		
E95C 5B	1945	DB 91,92,93
E95D 5C		
E95E 5D		
	1946	;------ ALT TABLE SCAN
E95F	1947	K13 LABEL BYTE
E95F 68	1948	DB 104,105,106,107,108
E960 69		
E961 6A		
E962 6B		
E963 6C		
E964 6D	1949	DB 109,110,111,112,113
E965 6E		
E966 6F		
E967 70		
E968 71		
	1950	;------ NUM STATE TABLE
E969	1951	K14 LABEL BYTE
E969 3738392D343536	1952	DB '789-456+1230.'
2B313233302E		
	1953	;------ BASE CASE TABLE
E976	1954	K15 LABEL BYTE
E976 47	1955	DB 71,72,73,-1,75,-1,77
E977 48		
E978 49		
E979 FF		
E97A 4B		
E97B FF		
E97C 4D		
E97D FF	1956	DB -1,79,80,81,82,83
E97E 4F		
E97F 50		
E980 51		
E981 52		
E982 53		
	1957	
	1958	;------ KEYBOARD INTERRUPT ROUTINE
	1959	
E987	1960	ORG 0E987H
E987	1961	KB_INT PROC FAR
E987 FB	1962	STI ; ALLOW FURTHER INTERRUPTS
E988 50	1963	PUSH AX
E989 53	1964	PUSH BX
E98A 51	1965	PUSH CX
E98B 52	1966	PUSH DX
E98C 56	1967	PUSH SI
E98D 57	1968	PUSH DI
E98E 1E	1969	PUSH DS
E98F 06	1970	PUSH ES
E990 FC	1971	CLD ; FORWARD DIRECTION
E991 E8AA15	1972	CALL DDS
E994 E460	1973	IN AL,KB_DATA ; READ IN THE CHARACTER
E996 50	1974	PUSH AX ; SAVE IT
E997 E461	1975	IN AL,KB_CTL ; GET THE CONTROL PORT
E999 8AE0	1976	MOV AH,AL ; SAVE VALUE
E99B 0C80	1977	OR AL,80H ; RESET BIT FOR KEYBOARD

LOC OBJ	LINE	SOURCE
E9D0 E661	1978	OUT KB_CTL,AL
E9F0 86E0	1979	XCHG AH,AL ; GET BACK ORIGINAL CONTROL
E9A1 E661	1980	OUT KB_CTL,AL ; KB HAS BEEN RESET
E9A3 58	1981	POP AX ; RECOVER SCAN CODE
E9A4 8AE0	1982	MOV AH,AL ; SAVE SCAN CODE IN AH ALSO
	1983	
	1984	I----- TEST FOR OVERRUN SCAN CODE FROM KEYBOARD
	1985	
E9A6 3CFF	1986	CHP AL,0FFH ; IS THIS AN OVERRUN CHAR
E9A8 7503	1987	JNZ K16 ; NO, TEST FOR SHIFT KEY
E9AA E97A02	1988	JMP K62 ; BUFFER_FULL_BEEP
	1989	
	1990	I----- TEST FOR SHIFT KEYS
	1991	
E9AD	1992	K16: ; TEST_SHIFT
E9AD 247F	1993	AND AL,07FH ; TURN OFF THE BREAK BIT
E9AF 0E	1994	PUSH CS
E9B0 07	1995	POP ES ; ESTABLISH ADDRESS OF SHIFT TABLE
E9B1 BF7EE8	1996	MOV DI,OFFSET K6 ; SHIFT KEY TABLE
E9B4 B90B00	1997	MOV CX,K6L ; LENGTH
E9B7 F2	1998	REPNE SCASB ; LOOK THROUGH THE TABLE FOR A MATCH
E9B8 AE		
E9B9 8AC4	1999	MOV AL,AH ; RECOVER SCAN CODE
E9BB 7403	2000	JE K17 ; JUMP IF MATCH FOUND
E9BD E96500	2001	JMP K25 ; IF NO MATCH, THEN SHIFT NOT FOUND
	2002	
	2003	I----- SHIFT KEY FOUND
	2004	
E9C0 81EF7FE8	2005	K17: SUB DI,OFFSET K6+1 ; ADJUST PTR TO SCAN CODE HTCH
E9C4 2E8AA586E8	2006	MOV AH,CS:K7(DI) ; GET MASK INTO AH
E9C9 A880	2007	TEST AL,80H ; TEST FOR BREAK KEY
E9CB 7551	2008	JNZ K23 ; BREAK_SHIFT_FOUND
	2009	
	2010	I----- SHIFT MAKE FOUND, DETERMINE SET OR TOGGLE
	2011	
E9CD 80FC10	2012	CHP AH,SCROLL_SHIFT
E9DD 7307	2013	JAE K18 ; IF SCROLL SHIFT OR ABOVE, TOGGLE KEY
	2014	
	2015	I----- PLAIN SHIFT KEY, SET SHIFT ON
	2016	
E9D2 08261700	2017	OR KB_FLAG,AH ; TURN ON SHIFT BIT
E9D6 E98000	2018	JMP K26 ; INTERRUPT_RETURN
	2019	
	2020	I----- TOGGLED SHIFT KEY, TEST FOR 1ST MAKE OR NOT
	2021	
E9D9	2022	K18: ; SHIFT-TOGGLE
E9D9 F606170004	2023	TEST KB_FLAG, CTL_SHIFT ; CHECK CTL SHIFT STATE
E9DE 7565	2024	JNZ K25 ; JUMP IF CTL STATE
E9E0 3C52	2025	CHP AL, INS_KEY ; CHECK FOR INSERT KEY
E9E2 7522	2026	JNZ K22 ; JUMP IF NOT INSERT KEY
E9E4 F606170008	2027	TEST KB_FLAG, ALT_SHIFT ; CHECK FOR ALTERNATE SHIFT
E9E9 755A	2028	JNZ K25 ; JUMP IF ALTERNATE SHIFT
E9EB F606170020	2029	K19: TEST KB_FLAG, NUM_STATE ; CHECK FOR BASE STATE
E9F0 750D	2030	JNZ K21 ; JUMP IF NUM LOCK IS ON
E9F2 F606170003	2031	TEST KB_FLAG, LEFT_SHIFT+ RIGHT_SHIFT
E9F7 740D	2032	JZ K22 ; JUMP IF BASE STATE
	2033	
E9F9	2034	K20: ; NUMERIC ZERO, NOT INSERT KEY
E9F9 B83052	2035	MOV AX, 5230H ; PUT OUT AN ASCII ZERO
E9FC E9D601	2036	JMP K57 ; BUFFER_FILL
E9FF	2037	K21: ; MIGHT BE NUMERIC
E9FF F606170003	2038	TEST KB_FLAG, LEFT_SHIFT+ RIGHT_SHIFT
EA04 74F3	2039	JZ K20 ; JUMP NUMERIC, NOT INSERT
	2040	
EA06	2041	K22: ; SHIFT TOGGLE KEY HIT; PROCESS IT
EA06 84261800	2042	TEST AH,KB_FLAG_1 ; IS KEY ALREADY DEPRESSED
EA0A 754D	2043	JNZ K26 ; JUMP IF KEY ALREADY DEPRESSED
EA0C 08261800	2044	OR KB_FLAG_1,AH ; INDICATE THAT THE KEY IS DEPRESSED
EA10 30261700	2045	XOR KB_FLAG,AH ; TOGGLE THE SHIFT STATE
EA14 3C52	2046	CHP AL,INS_KEY ; TEST FOR 1ST MAKE OF INSERT KEY
EA16 7541	2047	JNE K26 ; JUMP IF NOT INSERT KEY
EA18 B80052	2048	MOV AX,INS_KEY*256 ; SET SCAN CODE INTO AH, 0 INTO AL
EA1B E9B701	2049	JMP K57 ; PUT INTO OUTPUT BUFFER
	2050	
	2051	I----- BREAK SHIFT FOUND
	2052	
EA1E	2053	K23: ; BREAK-SHIFT-FOUND

LOC OBJ	LINE	SOURCE	
EA1E 80FC10	2054	CMP AH,SCROLL_SHIFT	; IS THIS A TOGGLE KEY
EA21 731A	2055	JAE K26	; YES, HANDLE BREAK TOGGLE
EA23 F6D4	2056	NOT AH	; INVERT MASK
EA25 20261700	2057	AND KB_FLAG,AH	; TURN OFF SHIFT BIT
EA29 3CB8	2058	CMP AL,ALT_KEY+80H	; IS THIS ALTERNATE SHIFT RELEASE
EA2B 752C	2059	JNE K26	; INTERRUPT_RETURN
	2060		
	2061	;----- ALTERNATE SHIFT KEY RELEASED, GET THE VALUE INTO BUFFER	
	2062		
EA2D A01900	2063	MOV AL,ALT_INPUT	
EA30 B400	2064	MOV AH,0	; SCAN CODE OF 0
EA32 08261900	2065	MOV ALT_INPUT,AH	; ZERO OUT THE FIELD
EA36 3C00	2066	CMP AL,0	; WAS THE INPUT=0
EA38 741F	2067	JE K26	; INTERRUPT_RETURN
EA3A E9A101	2068	JMP K50	; IT WASH'T, SO PUT IN BUFFER
EA3D	2069	K24:	; BREAK-TOGGLE
EA30 F6D4	2070	NOT AH	; INVERT MASK
EA3F 20261800	2071	AND KB_FLAG_1,AH	; INDICATE NO LONGER DEPRESSED
EA43 EB14	2072	JMP SHORT K26	; INTERRUPT_RETURN
	2073		
	2074	;----- TEST FOR HOLD STATE	
	2075		
EA45	2076	K25:	; NO-SHIFT-FOUND
EA45 3C80	2077	CMP AL,80H	; TEST FOR BREAK KEY
EA47 7310	2078	JAE K26	; NOTHING FOR BREAK CHARS FROM HERE ON
EA49 F606180000	2079	TEST KB_FLAG_1,HOLD_STATE	; ARE WE IN HOLD STATE
EA4E 7417	2080	JZ K28	; BRANCH AROUND TEST IF NOT
EA50 3C45	2081	CMP AL,NUM_KEY	
EA52 7405	2082	JE K26	; CAN'T END HOLD ON NUM_LOCK
EA54 80261800F7	2083	AND KB_FLAG_1,NOT_HOLD_STATE	; TURN OFF THE HOLD STATE BIT
EA59	2084	K26:	; INTERRUPT_RETURN
EA59 FA	2085	CLI	; TURN OFF INTERRUPTS
EA5A B020	2086	MOV AL,EOI	; END OF INTERRUPT COMMAND
EA5C E620	2087	OUT 020H,AL	; SEND COMMAND TO INT CONTROL PORT
EA5E	2088	K27:	; INTERRUPT_RETURN-NO-EOI
EA5E 07	2089	POP ES	
EA5F 1F	2090	POP DS	
EA60 5F	2091	POP DI	
EA61 5E	2092	POP SI	
EA62 5A	2093	POP DX	
EA63 59	2094	POP CX	
EA64 5B	2095	POP BX	
EA65 58	2096	POP AX	; RESTORE STATE
EA66 CF	2097	IRET	; RETURN, INTERRUPTS BACK ON
	2098		; WITH FLAG CHANGE
	2099		
	2100	;----- NOT IN HOLD STATE, TEST FOR SPECIAL CHARS	
	2101		
EA67	2102	K28:	; NO-HOLD-STATE
EA67 F606170000	2103	TEST KB_FLAG,ALT_SHIFT	; ARE WE IN ALTERNATE SHIFT
EA6C 7503	2104	JNZ K29	; JUMP IF ALTERNATE SHIFT
EA6E E99100	2105	JMP K30	; JUMP IF NOT ALTERNATE
	2106		
	2107	;----- TEST FOR RESET KEY SEQUENCE (CTL ALT DEL)	
	2108		
EA71	2109	K29:	; TEST-RESET
EA71 F606170004	2110	TEST KB_FLAG,CTL_SHIFT	; ARE WE IN CONTROL SHIFT ALSO
EA76 7433	2111	JZ K31	; NO_RESET
EA78 3C53	2112	CMP AL,DEL_KEY	; SHIFT STATE IS THERE, TEST KEY
EA7A 752F	2113	JNE K31	; NO_RESET
	2114		
	2115	;----- CTL-ALT-DEL HAS BEEN FOUND, DO I/O CLEANUP	
	2116		
EA7C C70672003412	2117	MOV RESET_FLAG,1234H	; SET FLAG FOR RESET FUNCTION
EA82 EA5BE000F0	2118	JMP RESET	; JUMP TO POWER ON DIAGNOSTICS
	2119		
	2120	;----- ALT-INPUT-TABLE	
EA87	2121	K30 LABEL BYTE	
EA87 52	2122	DB 82,79,80,81,75,76,77	
EA88 4F			
EA89 50			
EA8A 51			
EA8B 4B			
EA8C 4C			
EA8D 4D			
EA8E 47	2123	DB 71,72,73	; 10 NUMBERS ON KEYPAD
EA8F 48			

LOC OBJ	LINE	SOURCE
EA90 49		
	2124	!----- SUPER-SHIFT-TABLE
EA91 10	2125	DB 16,17,18,19,20,21,22,23 ; A-Z TYPEWRITER CHARS
EA92 11		
EA93 12		
EA94 13		
EA95 14		
EA96 15		
EA97 16		
EA98 17		
EA99 18	2126	DB 24,25,30,31,32,33,34,35
EA9A 19		
EA9B 1E		
EA9C 1F		
EA9D 20		
EA9E 21		
EA9F 22		
EAA0 23		
EAA1 24	2127	DB 36,37,38,44,45,46,47,48
EAA2 25		
EAA3 26		
EAA4 2C		
EAA5 2D		
EAA6 2E		
EAA7 2F		
EAA8 30		
EAA9 31	2128	DB 49,50
EAAA 32		
	2129	
	2130	!----- IN ALTERNATE SHIFT, RESET NOT FOUND
	2131	
EAB8	2132	K31: ; NO-RESET
EAB8 3C39	2133	CHP AL,57 ; TEST FOR SPACE KEY
EAD 7505	2134	JNE K32 ; NOT THERE
EAAF B020	2135	MOV AL,' ' ; SET SPACE CHAR
EAB1 E92101	2136	JMP K57 ; BUFFER_FILL
	2137	
	2138	!----- LOOK FOR KEY PAD ENTRY
	2139	
EAB4	2140	K32: ; ALT-KEY-PAD
EAB4 BF87EA	2141	MOV DI,OFFSET K30 ; ALT-INPUT-TABLE
EAB7 B90A00	2142	MOV CX,10 ; LOOK FOR ENTRY USING KEYPAD
EAB8 F2	2143	REPNE SCASB ; LOOK FOR MATCH
EABB AE		
EABC 7512	2144	JNE K33 ; NO_ALT_KEYPAD
EABE 81EF08EA	2145	SUB DI,OFFSET K30+1 ; DI,NOW HAS ENTRY VALUE
EAC2 A01900	2146	MOV AL,ALT_INPUT ; GET THE CURRENT BYTE
EAC5 B40A	2147	MOV AH,10 ; MULTIPLY BY 10
EAC7 F6E4	2148	MUL AH
EAC9 03C7	2149	ADD AX,DI ; ADD IN THE LATEST ENTRY
EACB A21900	2150	MOV ALT_INPUT,AL ; STORE IT AWAY
EACE EB89	2151	JMP K26 ; THROW AWAY THAT KEYSTROKE
	2152	
	2153	!----- LOOK FOR SUPERSHIFT ENTRY
	2154	
EAD0	2155	K33: ; NO-ALT-KEYPAD
EAD0 C606190000	2156	MOV ALT_INPUT,0 ; ZERO ANY PREVIOUS ENTRY INTO INPUT
EAD5 B91A00	2157	MOV CX,26 ; DI,ES ALREADY POINTING
EAD8 F2	2158	REPNE SCASB ; LOOK FOR MATCH IN ALPHABET
EAD9 AE		
EADA 7505	2159	JNE K34 ; NOT FOUND, FUNCTION KEY OR OTHER
EADC B000	2160	MOV AL,0 ; ASCII CODE OF ZERO
EADE E9F400	2161	JMP K57 ; PUT IT IN THE BUFFER
	2162	
	2163	!----- LOOK FOR TOP ROW OF ALTERNATE SHIFT
	2164	
EAE1	2165	K34: ; ALT-TOP-ROW
EAE1 3C02	2166	CHP AL,2 ; KEY WITH '1' ON IT
EAE3 720C	2167	JB K35 ; NOT ONE OF INTERESTING KEYS
EAE5 3C0E	2168	CHP AL,14 ; IS IT IN THE REGION
EAE7 7308	2169	JAE K35 ; ALT-FUNCTION
EAE9 80C476	2170	ADD AH,118 ; CONVERT PSEDUDO SCAN CODE TO RANGE
EAEC B000	2171	MOV AL,0 ; INDICATE AS SUCH
EAE8 E9E400	2172	JMP K57 ; BUFFER_FILL
	2173	
	2174	!----- TRANSLATE ALTERNATE SHIFT PSEUDO SCAN CODES
	2175	

LOC OBJ	LINE	SOURCE	
EAF1	2176	K35:	; ALT-FUNCTION
EAF1 3C3B	2177	CMP AL,59	; TEST FOR IN TABLE
EAF3 7303	2178	JAE K37	; ALT-CONTINUE
EAF5	2179	K36:	; CLOSE-RETURN
EAF5 E961FF	2180	JMP K26	; IGNORE THE KEY
EAF8	2181	K37:	; ALT-CONTINUE
EAF8 3C47	2182	CMP AL,71	; IN KEYPAD REGION
EAF4 73F9	2183	JAE K36	; IF SO, IGNORE
EAF4 BB5FE9	2184	MOV BX,OFFSET K13	; ALT SHIFT PSEUDO SCAN TABLE
EAF5 E91B01	2185	JMP K63	; TRANSLATE THAT
	2186		
	2187	;-----	NOT IN ALTERNATE SHIFT
	2188		
EB02	2189	K38:	; NOT-ALT-SHIFT
EB02 F606170004	2190	TEST KB_FLAG,CTL_SHIFT	; ARE WE IN CONTROL SHIFT
EB07 7458	2191	JZ K44	; NOT-CTL-SHIFT
	2192		
	2193	;-----	CONTROL SHIFT, TEST SPECIAL CHARACTERS
	2194	;-----	TEST FOR BREAK AND PAUSE KEYS
	2195		
EB09 3C46	2196	CMP AL,SCROLL_KEY	; TEST FOR BREAK
EB0B 7518	2197	JNE K39	; NO-BREAK
EB0D 0B1E8000	2198	MOV BX,BUFFER_START	; RESET BUFFER TO EMPTY
EB11 891E1A00	2199	MOV BUFFER_HEAD,BX	
EB15 891E1C00	2200	MOV BUFFER_TAIL,BX	
EB19 C606710080	2201	MOV BIOS_BREAK,80H	; TURN ON BIOS_BREAK BIT
EB1E CD1B	2202	INT 1BH	; BREAK INTERRUPT VECTOR
EB20 2BC0	2203	SUB AX,AX	; PUT OUT DUMMY CHARACTER
EB22 E98000	2204	JMP K57	; BUFFER_FILL
EB25	2205	K39:	; NO-BREAK
EB25 3C45	2206	CMP AL,NUM_KEY	; LOOK FOR PAUSE KEY
EB27 7521	2207	JNE K41	; NO-PAUSE
EB29 800E180008	2208	OR KB_FLAG_1,HOLD_STATE	; TURN ON THE HOLD FLAG
EB2E B020	2209	MOV AL,EDI	; END OF INTERRUPT TO CONTROL PORT
EB30 E620	2210	OUT 020H,AL	; ALLOW FURTHER KEYSTROKE INTS
	2211		
	2212	;-----	DURING PAUSE INTERVAL, TURN CRT BACK ON
	2213		
EB32 803E490007	2214	CMP CRT_MODE,7	; IS THIS BLACK AND WHITE CARD
EB37 7407	2215	JE K40	; YES, NOTHING TO DO
EB39 BAD803	2216	MOV DX,0308H	; PORT FOR COLOR CARD
EB3C A06500	2217	MOV AL,CRT_MODE_SET	; GET THE VALUE OF THE CURRENT MODE
EB3F EE	2218	OUT DX,AL	; SET THE CRT MODE, SO THAT CRT IS ON
EB40	2219	K40:	; PAUSE-LOOP
EB40 F606180008	2220	TEST KB_FLAG_1,HOLD_STATE	
EB45 75F9	2221	JNZ K40	; LOOP UNTIL FLAG TURNED OFF
EB47 E914FF	2222	JMP K27	; INTERRUPT_RETURN_NO_EOI
EB4A	2223	K41:	; NO-PAUSE
	2224		
	2225	;-----	TEST SPECIAL CASE KEY 55
	2226		
EB4A 3C37	2227	CMP AL,55	
EB4C 7506	2228	JNE K42	; NOT-KEY-55
EB4E B80072	2229	MOV AX,114*256	; START/STOP PRINTING SWITCH
EB51 E98100	2230	JMP K57	; BUFFER_FILL
	2231		
	2232	;-----	SET UP TO TRANSLATE CONTROL SHIFT
	2233		
EB54	2234	K42:	; NOT-KEY-55
EB54 B88EE8	2235	MOV BX,OFFSET K8	; SET UP TO TRANSLATE CTL
EB57 3C3B	2236	CMP AL,59	; IS IT IN TABLE
	2237		; CTL-TABLE-TRANSLATE
EB59 7276	2238	JB K56	; YES, GO TRANSLATE CHAR
EB5B	2239	K43:	; CTL-TABLE-TRANSLATE
EB5B B8C8E8	2240	MOV BX,OFFSET K9	; CTL TABLE SCAN
EB5E E9BC00	2241	JMP K63	; TRANSLATE_SCAN
	2242		
	2243	;-----	NOT IN CONTROL SHIFT
	2244		
EB61	2245	K44:	; NOT-CTL-SHIFT
EB61 3C47	2246	CMP AL,71	; TEST FOR KEYPAD REGION
EB63 732C	2247	JAE K48	; HANDLE KEYPAD REGION
EB65 F606170003	2248	TEST KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT	
EB6A 745A	2249	JZ K54	; TEST FOR SHIFT STATE
	2250		
	2251	;-----	UPPER CASE, HANDLE SPECIAL CASES
	2252		

LOC OBJ	LINE	SOURCE
EB6C 3C0F	2253	CHP AL,15 ; BACK TAB KEY
EB6E 7505	2254	JNE K45 ; NOT-BACK-TAB
EB70 B000F	2255	MOV AX,15*256 ; SET PSEUDO SCAN CODE
EB73 EB60	2256	JMP SHORT K57 ; BUFFER_FILL
EB75	2257	K45: ; NOT-BACK-TAB
EB75 3C37	2258	CHP AL,55 ; PRINT SCREEN KEY
EB77 7509	2259	JNE K46 ; NOT-PRINT-SCREEN
	2260	
	2261	;----- ISSUE INTERRUPT TO INDICATE PRINT SCREEN FUNCTION
	2262	
EB79 B020	2263	MOV AL,EOI ; END OF CURRENT INTERRUPT
EB7B E620	2264	OUT 020H,AL ; SO FURTHER THINGS CAN HAPPEN
EB7D CD05	2265	INT 5H ; ISSUE PRINT SCREEN INTERRUPT
EB7F E9DCE	2266	JMP K27 ; GO BACK WITHOUT EOI OCCURRING
EB82	2267	K46: ; NOT-PRINT-SCREEN
EB82 3C3B	2268	CHP AL,59 ; FUNCTION KEYS
EB84 7206	2269	JB K47 ; NOT-UPPER-FUNCTION
EB86 BB55E9	2270	MOV BX,OFFSET K12 ; UPPER CASE PSEUDO SCAN CODES
EB89 E99100	2271	JMP K63 ; TRANSLATE_SCAN
EB8C	2272	K47: ; NOT-UPPER-FUNCTION
EB8C BB1BE9	2273	MOV BX,OFFSET K11 ; POINT TO UPPER CASE TABLE
EB8F EB40	2274	JMP SHORT K56 ; OK, TRANSLATE THE CHAR
	2275	
	2276	;----- KEYPAD KEYS, MUST TEST NUM LOCK FOR DETERMINATION
	2277	
EB91	2278	K48: ; KEYPAD-REGION
EB91 F06170020	2279	TEST KB_FLAG,NUM_STATE ; ARE WE IN NUM_LOCK
EB96 7520	2280	JNZ K52 ; TEST FOR SURE
EB98 F06170003	2281	TEST KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT ; ARE WE IN SHIFT STATE
EB9D 7520	2282	JNZ K53 ; IF SHIFTED, REALLY NUM STATE
	2283	
	2284	;----- BASE CASE FOR KEYPAD
	2285	
EB9F	2286	K49: ; BASE-CASE
EB9F 3C4A	2287	CHP AL,74 ; SPECIAL CASE FOR A COUPLE OF KEYS
EBA1 740B	2288	JE K50 ; MINUS
EBA3 3C4E	2289	CHP AL,78
EBA5 740C	2290	JE K51
EBA7 2C47	2291	SUB AL,71 ; CONVERT ORIGIN
EBA9 BB76E9	2292	MOV BX,OFFSET K15 ; BASE CASE TABLE
EBAC EB71	2293	JMP SHORT K64 ; CONVERT TO PSEUDO SCAN
EBAE	2294	K50: ;
EBAE B02D4A	2295	MOV AX,74*256+'-' ; MINUS
EBB1 EB22	2296	JMP SHORT K57 ; BUFFER_FILL
EBB3	2297	K51: ;
EBB3 B02B4E	2298	MOV AX,78*256+'+' ; PLUS
EBB6 EB10	2299	JMP SHORT K57 ; BUFFER_FILL
	2300	
	2301	;----- MIGHT BE NUM LOCK, TEST SHIFT STATUS
	2302	
EBB8	2303	K52: ; ALMOST-NUM-STATE
EBB8 F06170003	2304	TEST KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT
EBB8 75E0	2305	JNZ K49 ; SHIFTED TEMP OUT OF NUM STATE
EBBF	2306	K53: ; REALLY_NUM_STATE
EBBF 2C46	2307	SUB AL,70 ; CONVERT ORIGIN
EBC1 BB69E9	2308	MOV BX,OFFSET K14 ; NUM STATE TABLE
EBC4 EB0B	2309	JMP SHORT K56 ; TRANSLATE_CHAR
	2310	
	2311	;----- PLAIN OLD LOWER CASE
	2312	
EBC6	2313	K54: ; NOT-SHIFT
EBC6 3C3B	2314	CHP AL,59 ; TEST FOR FUNCTION KEYS
EBC8 7204	2315	JB K55 ; NOT-LOWER-FUNCTION
EBCA B000	2316	MOV AL,0 ; SCAN CODE IN AH ALREADY
EBC8 EB07	2317	JMP SHORT K57 ; BUFFER_FILL
EBC8	2318	K55: ; NOT-LOWER-FUNCTION
EBC8 BBE1E8	2319	MOV BX,OFFSET K10 ; LC TABLE
	2320	
	2321	;----- TRANSLATE THE CHARACTER
	2322	
EBD1	2323	K56: ; TRANSLATE-CHAR
EBD1 FEC8	2324	DEC AL ; CONVERT ORIGIN
EBD3 2ED7	2325	XLAT CS:K11 ; CONVERT THE SCAN CODE TO ASCII
	2326	
	2327	;----- PUT CHARACTER INTO BUFFER
	2328	
EBD5	2329	K57: ; BUFFER-FILL

LOC OBJ	LINE	SOURCE	
EBD5 3CFF	2330	CMP AL,-1	; IS THIS AN IGNORE CHAR
EBD7 741F	2331	JE K59	; YES, DO NOTHING WITH IT
EBD9 80FCFF	2332	CMP AH,-1	; LOOK FOR -1 PSEUDO SCAN
EBDC 741A	2333	JE K59	; NEAR_INTERRUPT_RETURN
	2334		
	2335	I----- HANDLE THE CAPS LOCK PROBLEM	
	2336		
EBDE	2337	K58:	; BUFFER-FILL-NOTEST
EBDE F06170040	2338	TEST KB_FLAG,CAPS_STATE	; ARE WE IN CAPS LOCK STATE
EBE3 7420	2339	JZ K61	; SKIP IF NOT
	2340		
	2341	I----- IN CAPS LOCK STATE	
	2342		
EBE5 F06170003	2343	TEST KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT	; TEST FOR SHIFT STATE
EBEA 740F	2344	JZ K60	; IF NOT SHIFT, CONVERT LOWER TO UPPER
	2345		
	2346	I----- CONVERT ANY UPPER CASE TO LOWER CASE	
	2347		
EBEC 3C41	2348	CMP AL,'A'	; FIND OUT IF ALPHABETIC
EBEE 7215	2349	JB K61	; NOT_CAPS_STATE
EBF0 3C5A	2350	CMP AL,'Z'	
EBF2 7711	2351	JA K61	; NOT_CAPS_STATE
EBF4 0420	2352	ADD AL,'a'-'A'	; CONVERT TO LOWER CASE
EBF6 EB0D	2353	JMP SHORT K61	; NOT_CAPS_STATE
EBF8	2354	K59:	; NEAR_INTERRUPT_RETURN
EBF8 E95EFE	2355	JMP K26	; INTERRUPT_RETURN
	2356		
	2357	I----- CONVERT ANY LOWER CASE TO UPPER CASE	
	2358		
EBFB	2359	K60:	; LOWER-TO-UPPER
EBFB 3C61	2360	CMP AL,'a'	; FIND OUT IF ALPHABETIC
EBFD 7206	2361	JB K61	; NOT_CAPS_STATE
EBFF 3C7A	2362	CMP AL,'z'	
EC01 7702	2363	JA K61	; NOT_CAPS_STATE
EC03 2C20	2364	SUB AL,'a'-'A'	; CONVERT TO UPPER CASE
EC05	2365	K61:	; NOT-CAPS-STATE
EC05 8B1EC00	2366	MOV BX,BUFFER_TAIL	; GET THE END POINTER TO THE BUFFER
EC09 8BF3	2367	MOV SI,BX	; SAVE THE VALUE
EC0B E863FC	2368	CALL K4	; ADVANCE THE TAIL
EC0E 3B1E1A00	2369	CMP BX,BUFFER_HEAD	; HAS THE BUFFER WRAPPED AROUND
EC12 7413	2370	JE K62	; BUFFER_FULL_BEEP
EC14 8904	2371	MOV [SI],AX	; STORE THE VALUE
EC16 891E1C00	2372	MOV BUFFER_TAIL,BX	; MOVE THE POINTER UP
EC1A E93CFE	2373	JMP K26	; INTERRUPT_RETURN
	2374		
	2375	I----- TRANSLATE SCAN FOR PSEUDO SCAN CODES	
	2376		
EC1D	2377	K63:	; TRANSLATE-SCAN
EC1D 2C3B	2378	SUB AL,59	; CONVERT ORIGIN TO FUNCTION KEYS
EC1F	2379	K64:	; TRANSLATE-SCAN-ORGD
EC1F 2ED7	2380	XLAT CS:K9	; CTL TABLE SCAN
EC21 8AE0	2381	MOV AH,AL	; PUT VALUE INTO AH
EC23 B000	2382	MOV AL,0	; ZERO ASCII CODE
EC25 EBAE	2383	JMP K57	; PUT IT INTO THE BUFFER
	2384		
	2385	KB_INT	ENDP
	2386		
	2387	I----- BUFFER IS FULL, SOUND THE BEEPER	
	2388		
EC27	2389	K62:	; BUFFER-FULL-BEEP
EC27 B020	2390	MOV AL,EOI	; END OF INTERRUPT COMMAND
EC29 E620	2391	OUT 20H,AL	; SEND COMMAND TO INT CONTROL PORT
EC2B BB8000	2392	MOV BX,080H	; NUMBER OF CYCLES FOR 1/12 SECOND TONE
EC2E E461	2393	IN AL,KB_CTL	; GET CONTROL INFORMATION
EC30 50	2394	PUSH AX	; SAVE
EC31	2395	K65:	; BEEP-CYCLE
EC31 24FC	2396	AND AL,0FCH	; TURN OFF TIMER GATE AND SPEAKER DATA
EC33 E661	2397	OUT KB_CTL,AL	; OUTPUT TO CONTROL
EC35 B94800	2398	MOV CX,48H	; HALF CYCLE TIME FOR TONE
EC38	2399	K66:	
EC38 E2FE	2400	LOOP K66	; SPEAKER OFF
EC3A 0C02	2401	OR AL,2	; TURN ON SPEAKER BIT
EC3C E661	2402	OUT KB_CTL,AL	; OUTPUT TO CONTROL
EC3E B94800	2403	MOV CX,48H	; SET UP COUNT
EC41	2404	K67:	
EC41 E2FE	2405	LODP K67	; ANOTHER HALF CYCLE
EC43 4B	2406	DEC BX	; TOTAL TIME COUNT


```

LOC OBJ          LINE  SOURCE
EC44 75EB        2407          JNZ    K65          ; DO ANOTHER CYCLE
EC46 58          2408          POP    AX           ; RECOVER CONTROL
EC47 E661        2409          OUT    KB_CTL,AL   ; OUTPUT THE CONTROL
EC49 E912FE      2410          JMP    K27
2411          ;-----
2412          ;   ROS CHECKSUM SUBROUTINE   ;
2413          ;-----
EC4C          2414  ROS_CHECKSUM  PROC    NEAR          ; NEXT_ROS_MODULE
EC4C B90020      2415          MOV    CX,8192     ; NUMBER OF BYTES TO ADD
EC4F          2416  ROS_CHECKSUM_CNT: ; ENTRY FOR OPTIONAL ROS TEST
EC4F 32C0         2417          XOR    AL,AL
EC51          2418  C26:
EC51 0207        2419          ADD    AL,DS:[BX]
EC53 43         2420          INC    BX          ; POINT TO NEXT BYTE
EC54 E2FB       2421          LOOP  C26         ; ADD ALL BYTES IN ROS MODULE
EC56 0AC0       2422          OR    AL,AL       ; SUM = 0?
EC58 C3        2423          RET
2424          ROS_CHECKSUM  ENDP
2425
2426          ;-- INT 13 -----
2427          ; DISKETTE I/O
2428          ;   THIS INTERFACE PROVIDES ACCESS TO THE 5 1/4" DISKETTE DRIVES
2429          ; INPUT
2430          ;   (AH)=0  RESET DISKETTE SYSTEM
2431          ;   HARD RESET TO NEC, PREPARE COMMAND, RECAL REQUIRED
2432          ;   ON ALL DRIVES
2433          ;   (AH)=1  READ THE STATUS OF THE SYSTEM INTO (AL)
2434          ;   DISKETTE_STATUS FROM LAST OPERATION IS USED
2435          ;
2436          ; REGISTERS FOR READ/WRITE/VERIFY/FORMAT
2437          ;   (DL) - DRIVE NUMBER (0-3 ALLOWED, VALUE CHECKED)
2438          ;   (DH) - HEAD NUMBER (0-1 ALLOWED, NOT VALUE CHECKED)
2439          ;   (CH) - TRACK NUMBER (0-39, NOT VALUE CHECKED)
2440          ;   (CL) - SECTOR NUMBER (1-8, NOT VALUE CHECKED,
2441          ;   NOT USED FOR FORMAT)
2442          ;   (AL) - NUMBER OF SECTORS ( MAX = 8, NOT VALUE CHECKED, NOT USED
2443          ;   FOR FORMAT)
2444          ;   (ES:BX) - ADDRESS OF BUFFER ( NOT REQUIRED FOR VERIFY)
2445          ;
2446          ;   (AH)=2  READ THE DESIRED SECTORS INTO MEMORY
2447          ;   (AH)=3  WRITE THE DESIRED SECTORS FROM MEMORY
2448          ;   (AH)=4  VERIFY THE DESIRED SECTORS
2449          ;   (AH)=5  FORMAT THE DESIRED TRACK
2450          ;   FOR THE FORMAT OPERATION, THE BUFFER POINTER (ES:BX)
2451          ;   MUST POINT TO THE COLLECTION OF DESIRED ADDRESS FIELDS
2452          ;   FOR THE TRACK.  EACH FIELD IS COMPOSED OF 4 BYTES,
2453          ;   (C,H,R,N), WHERE C = TRACK NUMBER, H=HEAD NUMBER,
2454          ;   R = SECTOR NUMBER, N= NUMBER OF BYTES PER SECTOR
2455          ;   (00=128, 01=256, 02=512, 03=1024).  THERE MUST BE ONE
2456          ;   ENTRY FOR EVERY SECTOR ON THE TRACK.  THIS INFORMATION
2457          ;   IS USED TO FIND THE REQUESTED SECTOR DURING READ/WRITE
2458          ;   ACCESS.
2459          ;
2460          ; DATA VARIABLE -- DISK_POINTER
2461          ;   DOUBLE WORD POINTER TO THE CURRENT SET OF DISKETTE PARAMETERS
2462          ; OUTPUT
2463          ;   AH = STATUS OF OPERATION
2464          ;   STATUS BITS ARE DEFINED IN THE EQUATES FOR
2465          ;   DISKETTE_STATUS VARIABLE IN THE DATA SEGMENT OF THIS
2466          ;   MODULE.
2467          ;   CY = 0  SUCCESSFUL OPERATION (AH=0 ON RETURN)
2468          ;   CY = 1  FAILED OPERATION (AH HAS ERROR REASON)
2469          ;   FOR READ/WRITE/VERIFY
2470          ;   DS,BX,DX,CH,CL PRESERVED
2471          ;   AL = NUMBER OF SECTORS ACTUALLY READ
2472          ;   ***** AL MAY NOT BE CORRECT IF TIME OUT ERROR OCCURS
2473          ;   NOTE:  IF AN ERROR IS REPORTED BY THE DISKETTE CODE, THE
2474          ;   APPROPRIATE ACTION IS TO RESET THE DISKETTE, THEN RETRY
2475          ;   THE OPERATION.  ON READ ACCESSES, NO MOTOR START DELAY
2476          ;   IS TAKEN, SO THAT THREE RETRIES ARE REQUIRED ON READS
2477          ;   TO ENSURE THAT THE PROBLEM IS NOT DUE TO MOTOR
2478          ;   START-UP.
2479          ;-----
2480          ASSUME  CS:CODE,DS:DATA,ES:DATA
EC59          2481          ORG    0EC59H
EC59          2482  DISKETTE_IO  PROC    FAR
EC59 FB        2483          STI          ; INTERRUPTS BACK ON

```

LOC OBJ	LINE	SOURCE	
EC5A 53	2484	PUSH BX	; SAVE ADDRESS
EC5B 51	2485	PUSH CX	
EC5C 1E	2486	PUSH DS	; SAVE SEGMENT REGISTER VALUE
EC5D 56	2487	PUSH SI	; SAVE ALL REGISTERS DURING OPERATION
EC5E 57	2488	PUSH DI	
EC5F 55	2489	PUSH BP	
EC60 52	2490	PUSH DX	
EC61 8BEC	2491	MOV BP,SP	; SET UP POINTER TO HEAD PARM
EC63 E8D812	2492	CALL DOS	
EC66 E81C00	2493	CALL J1	; CALL THE REST TO ENSURE DS RESTORED
EC69 BB0400	2494	MOV BX,4	; GET THE MOTOR WAIT PARAMETER
EC6C E8FD01	2495	CALL GET_PARM	
EC6F 88264000	2496	MOV MOTOR_COUNT,AH	; SET THE TIMER COUNT FOR THE MOTOR
EC73 8A264100	2497	MOV AH,DISKETTE_STATUS	; GET STATUS OF OPERATION
EC77 80FCD1	2498	CHP AH,1	; SET THE CARRY FLAG TO INDICATE
EC7A F5	2499	CMC	; SUCCESS OR FAILURE
EC7B 5A	2500	POP DX	; RESTORE ALL REGISTERS
EC7C 5D	2501	POP BP	
EC7D 5F	2502	PDP DI	
EC7E 5E	2503	PDP SI	
EC7F 1F	2504	POP DS	
EC80 59	2505	POP CX	
EC81 5B	2506	POP BX	; RECOVER ADDRESS
EC82 CA0200	2507	RET 2	; THROW AWAY SAVED FLAGS
	2508	DISKETTE_IO	ENDP
	2509		
EC85	2510	J1 PROC NEAR	
EC85 8AF0	2511	MOV DH,AL	; SAVE # SECTORS IN DH
EC87 80263F007F	2512	AND MOTOR_STATUS,07FH	; INDICATE A READ OPERATION
EC8C 0AE4	2513	OR AH,AH	; AH=0
EC8E 7427	2514	JZ DISK_RESET	
EC90 FECC	2515	DEC AH	; AH=1
EC92 7473	2516	JZ DISK_STATUS	
EC94 C606410000	2517	MOV DISKETTE_STATUS,0	; RESET THE STATUS INDICATOR
EC99 80FA04	2518	CHP DL,4	; TEST FOR DRIVE IN 0-3 RANGE
EC9C 7313	2519	JAE J3	; ERROR IF ABOVE
EC9E FECC	2520	DEC AH	; AH=2
ECA0 7469	2521	JZ DISK_READ	
ECA2 FECC	2522	DEC AH	; AH=3
ECA4 7503	2523	JNZ J2	; TEST_DISK_VERF
ECA6 E99500	2524	JMP DISK_WRITE	
ECA9	2525	J2:	; TEST_DISK_VERF
ECA9 FECC	2526	DEC AH	; AH=4
ECAB 7467	2527	JZ DISK_VERF	
ECAD FECC	2528	DEC AH	; AH=5
ECAF 7467	2529	JZ DISK_FORMAT	
ECB1	2530	J3:	; BAD_COMMAND
ECB1 C606410001	2531	MOV DISKETTE_STATUS,BAD_CMD	; ERROR CODE, NO SECTORS TRANSFERRED
ECB6 C3	2532	RET	; UNDEFINED OPERATION
	2533	J1	ENDP
	2534		
	2535	;	----- RESET THE DISKETTE SYSTEM
	2536		
ECB7	2537	DISK_RESET PROC NEAR	
ECB7 BAF203	2538	MOV DX,03F2H	; ADAPTER CONTROL PORT
ECBA FA	2539	CLI	; NO INTERRUPTS
ECBB A03F0D	2540	MOV AL,MOTOR_STATUS	; WHICH MOTOR IS ON
ECBE B104	2541	MOV CL,4	; SHIFT COUNT
ECC0 D2E0	2542	SAL AL,CL	; MOVE MOTOR VALUE TO HIGH NYBBLE
ECC2 A820	2543	TEST AL, 20H	; SELECT CORRESPONDING DRIVE
ECC4 750C	2544	JNZ J5	; JUMP IF MOTOR ONE IS ON
ECC6 A840	2545	TEST AL, 40H	
ECC8 7506	2546	JNZ J4	; JUMP IF MOTOR TWO IS ON
ECCA A880	2547	TEST AL, 80H	
ECCC 7406	2548	JZ J6	; JUMP IF MOTOR ZERO IS ON
ECCE FEC0	2549	INC AL	
ECD0	2550	J4:	
ECD0 FEC0	2551	INC AL	
ECD2	2552	J5:	
ECD2 FEC0	2553	INC AL	
ECD4	2554	J6:	
ECD4 0C08	2555	OR AL,8	; TURN ON INTERRUPT ENABLE
ECDE EE	2556	OUT DX,AL	; RESET THE ADAPTER
ECD7 C6063E0000	2557	MOV SEEK_STATUS,0	; SET REGAL REQUIRED ON ALL DRIVES
ECDC C606410000	2558	MOV DISKETTE_STATUS,0	; SET OK STATUS FOR DISKETTE
ECDE 0CD4	2559	OR AL,4	; TURN OFF RESET
EE3 EE	2560	OUT DX,AL	; TURN OFF THE RESET

LOC OBJ	LINE	SOURCE	
ECE4 FB	2561	STI	; REENABLE THE INTERRUPTS
ECE5 E82A02	2562	CALL	CHK_STAT_2 ; DO SENSE INTERRUPT STATUS
	2563		; FOLLOWING RESET
ECE8 A04200	2564	MOV	AL,NEC_STATUS ; IGNORE ERROR RETURN AND DO DMN TEST
ECEB 3CC0	2565	CHP	AL,0C0H ; TEST FOR DRIVE READY TRANSITION
ECED 7406	2566	JZ	J7 ; EVERYTHING OK
ECFE 800E410020	2567	OR	DISKETTE_STATUS,BAD_NEC ; SET ERROR CODE
ECF4 C3	2568	RET	
	2569		
	2570		;----- SEND SPECIFY COMMAND TO NEC
	2571		
ECF5	2572	J7:	; DRIVE_READY
ECF5 B403	2573	MOV	AH,03H ; SPECIFY COMMAND
ECF7 E84701	2574	CALL	NEC_OUTPUT ; OUTPUT THE COMMAND
ECFA B80100	2575	MOV	BX,1 ; FIRST BYTE PARM IN BLOCK
ECFD E86C01	2576	CALL	GET_PARM ; TO THE NEC CONTROLLER
ED00 B80300	2577	MOV	BX,3 ; SECOND BYTE PARM IN BLOCK
ED03 E86601	2578	CALL	GET_PARM ; TO THE NEC CONTROLLER
ED06	2579	J8:	; RESET_RET
ED06 C3	2580	RET	; RETURN TO CALLER
	2581	DISK_RESET	ENDP
	2582		
	2583		;----- DISKETTE STATUS ROUTINE
	2584		
	2585	DISK_STATUS	PROC NEAR
ED07	2586	MOV	AL,DISKETTE_STATUS
ED07 A04100	2587	RET	
ED0A C3	2588	DISK_STATUS	ENDP
	2589		
	2590		;----- DISKETTE READ
	2591		
ED0B	2592	DISK_READ	PROC NEAR
ED0B B046	2593	MOV	AL,046H ; READ COMMAND FOR DMA
ED0D	2594	J9:	; DISK_READ_CONT
ED0D E88801	2595	CALL	DMA_SETUP ; SET UP THE DMA
ED10 B4E6	2596	MOV	AH,0E6H ; SET UP RD COMMAND FOR NEC CONTROLLER
ED12 EB36	2597	JMP	SHORT RM_OPN ; GO DO THE OPERATION
	2598	DISK_READ	ENDP
	2599		
	2600		;----- DISKETTE VERIFY
	2601		
ED14	2602	DISK_VERF	PROC NEAR
ED14 B042	2603	MOV	AL,042H ; VERIFY COMMAND FOR DMA
ED16 BFF5	2604	JMP	J9 ; DO AS IF DISK READ
	2605	DISK_VERF	ENDP
	2606		
	2607		;----- DISKETTE FORMAT
	2608		
ED18	2609	DISK_FORMAT	PROC NEAR
ED18 800E3F0080	2610	OR	MOTOR_STATUS,80H ; INDICATE WRITE OPERATION
ED1D B04A	2611	MOV	AL,04AH ; WILL WRITE TO THE DISKETTE
ED1F E8A601	2612	CALL	DMA_SETUP ; SET UP THE DMA
ED22 B44D	2613	MOV	AH,04DH ; ESTABLISH THE FORMAT COMMAND
ED24 EB24	2614	JMP	SHORT RM_OPN ; DO THE OPERATION
ED26	2615	J10:	; CONTINUATION OF RM_OPN FOR FMT
ED26 B80700	2616	MOV	BX,7 ; GET THE
ED29 E84001	2617	CALL	GET_PARM ; BYTES/SECTOR VALUE TO NEC
ED2C B80900	2618	MOV	BX,9 ; GET THE
ED2F E83A01	2619	CALL	GET_PARM ; SECTORS/TRACK VALUE TO NEC
ED32 B80F00	2620	MOV	BX,15 ; GET THE
ED35 E83401	2621	CALL	GET_PARM ; GAP LENGTH VALUE TO NEC
ED38 B81100	2622	MOV	BX,17 ; GET THE FILLER BYTE
ED3B E9AB00	2623	JMP	J16 ; TO THE CONTROLLER
	2624	DISK_FORMAT	ENDP
	2625		
	2626		;----- DISKETTE WRITE ROUTINE
	2627		
ED3E	2628	DISK_WRITE	PROC NEAR
ED3E 800E3F0080	2629	OR	MOTOR_STATUS,80H ; INDICATE WRITE OPERATION
ED43 B04A	2630	MOV	AL,04AH ; DMA WRITE COMMAND
ED45 E88001	2631	CALL	DMA_SETUP
ED48 B4C5	2632	MOV	AH,0C5H ; NEC COMMAND TO WRITE TO DISKETTE
	2633	DISK_WRITE	ENDP
	2634		
	2635		;----- ALLOW WRITE ROUTINE TO FALL INTO RM_OPN
	2636		
	2637		;-----

LOC OBJ	LINE	SOURCE
	2638	; RM_OPN :
	2639	; THIS ROUTINE PERFORMS THE READ/WRITE/VERIFY OPERATION :
	2640	;
	2641	-----
ED4A	2641	RM_OPN PROC NEAR
ED4A 7308	2642	JNC J11 ; TEST FOR DMA ERROR
ED4C C606410009	2643	MOV DISKETTE_STATUS,DMA_BOUNDARY ; SET ERROR
ED51 B000	2644	MOV AL,0 ; NO SECTORS TRANSFERRED
ED53 C3	2645	RET ; RETURN TO MAIN ROUTINE
ED54	2646	J11: ; DO_RM_OPN
ED54 50	2647	PUSH AX ; SAVE THE COMMAND
	2648	
	2649	;----- TURN ON THE MOTOR AND SELECT THE DRIVE
	2650	
ED55 51	2651	PUSH CX ; SAVE THE T/S PARMS
ED56 8ACA	2652	MOV CL,DL ; GET DRIVE NUMBER AS SHIFT COUNT
ED58 B001	2653	MOV AL,1 ; MASK FOR DETERMINING MOTOR BIT
ED5A D2E0	2654	SAL AL,CL ; SHIFT THE MASK BIT
ED5C FA	2655	CLI ; NO INTERRUPTS WHILE DETERMINING
	2656	; MOTOR STATUS
ED5D C6064000FF	2657	MOV MOTOR_COUNT,0FFH ; SET LARGE COUNT DURING OPERATION
ED62 84063F00	2658	TEST AL,MOTOR_STATUS ; TEST THAT MOTOR FOR OPERATING
ED66 7531	2659	JNZ J14 ; IF RUNNING, SKIP THE WAIT
ED68 80263F00F0	2660	AND MOTOR_STATUS,0F0H ; TURN OFF ALL MOTOR BITS
ED6D 08063F00	2661	OR MOTOR_STATUS,AL ; TURN ON THE CURRENT MOTOR
ED71 FB	2662	STI ; INTERRUPTS BACK ON
ED72 B010	2663	MOV AL,10H ; MASK BIT
ED74 D2E0	2664	SAL AL,CL ; DEVELOP BIT MASK FOR MOTOR ENABLE
ED76 0AC2	2665	OR AL,DL ; GET DRIVE SELECT BITS IN
ED78 0C0C	2666	OR AL,0CH ; NO RESET, ENABLE DMA/INT
ED7A 52	2667	PUSH DX ; SAVE REG
ED7B BAF203	2668	MOV DX,03F2H ; CONTROL PORT ADDRESS
ED7E EE	2669	OUT DX,AL
ED7F 5A	2670	POP DX ; RECOVER REGISTERS
	2671	
	2672	;----- WAIT FOR MOTOR IF WRITE OPERATION
	2673	
ED80 F6063F00B0	2674	TEST MOTOR_STATUS,80H ; IS THIS A WRITE
ED85 7412	2675	JZ J14 ; NO, CONTINUE WITHOUT WAIT
ED87 BB1400	2676	MOV BX,20 ; GET THE MOTOR WAIT
ED8A E8DF00	2677	CALL GET_PARM ; PARAMETER
ED8D 0AE4	2678	OR AH,AH ; TEST FOR NO WAIT
ED8F	2679	J12: ; TEST_WAIT_TIME
ED8F 7408	2680	JZ J14 ; EXIT WITH TIME EXPIRED
ED91 2BC9	2681	SUB CX,CX ; SET UP 1/8 SECOND LOOP TIME
ED93	2682	J13:
ED93 E2FE	2683	LOOP J13 ; WAIT FOR THE REQUIRED TIME
ED95 FECC	2684	DEC AH ; DECREMENT TIME VALUE
ED97 EBF6	2685	JMP J12 ; ARE WE DONE YET
ED99	2686	J14: ; MOTOR_RUNNING
ED99 FB	2687	STI ; INTERRUPTS BACK ON FOR BYPASS WAIT
ED9A 59	2688	POP CX
	2689	
	2690	;----- DO THE SEEK OPERATION
	2691	
ED9B E8DF00	2692	CALL SEEK ; MOVE TO CORRECT TRACK
ED9E 58	2693	POP AX ; RECOVER COMMAND
ED9F 8AFC	2694	MOV BH,AH ; SAVE COMMAND IN BH
EDA1 B600	2695	MOV DH,0 ; SET NO SECTORS READ IN CASE OF ERROR
EDA3 724B	2696	JC J17 ; IF ERROR, THEN EXIT AFTER MOTOR OFF
EDA5 BEF0ED90	2697	MOV SI,OFFSET J17 ; DUMMY RETURN ON STACK FOR NEC_OUTPUT
EDA9 56	2698	PUSH SI ; SO THAT IT WILL RETURN TO MOTOR OFF
	2699	; LOCATION
	2700	
	2701	;----- SEND OUT THE PARAMETERS TO THE CONTROLLER
	2702	
EDAA E89400	2703	CALL NEC_OUTPUT ; OUTPUT THE OPERATION COMMAND
EDAD 8A6601	2704	MOV AH,[BP+1] ; GET THE CURRENT HEAD NUMBER
EDB0 D0E4	2705	SAL AH,1 ; MOVE IT TO BIT 2
EDB2 D0E4	2706	SAL AH,1
EDB4 80E404	2707	AND AH,4 ; ISOLATE THAT BIT
EDB7 0AE2	2708	OR AH,DL ; OR IN THE DRIVE NUMBER
EDB9 E88500	2709	CALL NEC_OUTPUT
	2710	
	2711	;----- TEST FOR FORMAT COMMAND
	2712	
EDBC 80FF4D	2713	CMPSI BH,04DH ; IS THIS A FORMAT OPERATION
EDBF 7503	2714	JNE J15 ; NO. CONTINUE WITH R/W/V

LOC	OBJ	LINE	SOURCE
EDC1	E962FF	2715	JMP J10 ; IF SO, HANDLE SPECIAL
EDC4		2716	J15:
EDC4	8AE5	2717	MOV AH,CH ; CYLINDER NUMBER
EDC6	E87800	2718	CALL NEC_OUTPUT
EDC9	8A6601	2719	MOV AH,[BP+1] ; HEAD NUMBER FROM STACK
EDCC	E87200	2720	CALL NEC_OUTPUT
EDCF	8AE1	2721	MOV AH,CL ; SECTOR NUMBER
EDD1	E86D00	2722	CALL NEC_OUTPUT
EDD4	BB0700	2723	MOV BX,7 ; BYTES/SECTOR PARM FROM BLOCK
EDD7	E89200	2724	CALL GET_PARM ; TO THE NEC
EDDA	BB0900	2725	MOV BX,9 ; EOT PARM FROM BLOCK
EDDD	E88C00	2726	CALL GET_PARM ; TO THE NEC
EDE0	BB0B00	2727	MOV BX,11 ; GAP LENGTH PARM FROM BLOCK
EDE3	E88600	2728	CALL GET_PARM ; TO THE NEC
EDE6	BB0D00	2729	MOV BX,13 ; DTL PARM FROM BLOCK
EDE9		2730	J16: ; RM_OPN_FINISH
EDE9	E88000	2731	CALL GET_PARM ; TO THE NEC
EDEC	5E	2732	POP SI ; CAN NOW DISCARD THAT DUMMY
		2733	; RETURN ADDRESS
		2734	
		2735	;----- LET THE OPERATION HAPPEN
		2736	
EDED	E84301	2737	CALL WAIT_INT ; WAIT FOR THE INTERRUPT
EDF0		2738	J17: ; MOTOR_OFF
EDF0	7245	2739	JC J21 ; LOOK FOR ERROR
EDF2	E87401	2740	CALL RESULTS ; GET THE NEC STATUS
EDF5	723F	2741	JC J20 ; LOOK FOR ERROR
		2742	
		2743	;----- CHECK THE RESULTS RETURNED BY THE CONTROLLER
		2744	
EDF7	FC	2745	CLD ; SET THE CORRECT DIRECTION
EDF8	BE4200	2746	MOV SI,OFFSET NEC_STATUS ; POINT TO STATUS FIELD
EDFB	AC	2747	LODS NEC_STATUS ; GET ST0
EDFC	24C0	2748	AND AL,0C0H ; TEST FOR NORMAL TERMINATION
EDFE	743B	2749	JZ J22 ; OPN_OK
EE00	3C40	2750	CMPL AL,040H ; TEST FOR ABNORMAL TERMINATION
EE02	7529	2751	JNZ J18 ; NOT ABNORMAL, BAD NEC
		2752	
		2753	;----- ABNORMAL TERMINATION, FIND OUT WHY
		2754	
EE04	AC	2755	LODS NEC_STATUS ; GET ST1
EE05	D0E0	2756	SAL AL,1 ; TEST FOR EOT FOUND
EE07	B404	2757	MOV AH,RECORD_NOT_FND
EE09	7224	2758	JC J19 ; RM_FAIL
EE0B	D0E0	2759	SAL AL,1
EE0D	D0E0	2760	SAL AL,1 ; TEST FOR CRC ERROR
EE0F	B410	2761	MOV AH,BAD_CRC
EE11	721C	2762	JC J19 ; RM_FAIL
EE13	D0E0	2763	SAL AL,1 ; TEST FOR DMA OVERRUN
EE15	B408	2764	MOV AH,BAD_DMA
EE17	7216	2765	JC J19 ; RM_FAIL
EE19	D0E0	2766	SAL AL,1
EE1B	D0E0	2767	SAL AL,1 ; TEST FOR RECORD NOT FOUND
EE1D	B404	2768	MOV AH,RECORD_NOT_FND
EE1F	720E	2769	JC J19 ; RM_FAIL
EE21	D0E0	2770	SAL AL,1
EE23	B403	2771	MOV AH,WRITE_PROTECT ; TEST FOR WRITE_PROTECT
EE25	7208	2772	JC J19 ; RM_FAIL
EE27	D0E0	2773	SAL AL,1 ; TEST MISSING ADDRESS MARK
EE29	B402	2774	MOV AH,BAD_ADDR_MARK
EE2B	7202	2775	JC J19 ; RM_FAIL
		2776	
		2777	;----- NEC MUST HAVE FAILED
		2778	
EE2D		2779	J18: ; RW-NEC-FAIL
EE2D	B420	2780	MOV AH,BAD_NEC
EE2F		2781	J19: ; RW-FAIL
EE2F	08264100	2782	OR DISKETTE_STATUS,AH
EE33	E87901	2783	CALL HLP_TRANS ; HOW MANY WERE REALLY TRANSFERRED
EE36		2784	J20: ; RM_ERR
EE36	C3	2785	RET ; RETURN TO CALLER
EE37		2786	J21: ; RM_ERR_RES
EE37	E82F01	2787	CALL RESULTS ; FLUSH THE RESULTS BUFFER
EE3A	C3	2788	RET
		2789	
		2790	;----- OPERATION WAS SUCCESSFUL
		2791	

```

LOC OBJ          LINE  SOURCE
EE3B             2792  J22:                               ; OPN_OK
EE3B E87001     2793          CALL  NUM_TRANS                ; HOW MANY GOT MOVED
EE3E 32E4       2794          XOR   AH,AH                      ; NO ERRORS
EE40 C3         2795          RET
                2796  RM_OPN  ENDP
                2797  |-----|
                2798  | NEC_OUTPUT                               ;
                2799  | THIS ROUTINE SENDS A BYTE TO THE NEC CONTROLLER AFTER TESTING ;
                2800  | FOR CORRECT DIRECTION AND CONTROLLER READY THIS ROUTINE WILL ;
                2801  | TIME OUT IF THE BYTE IS NOT ACCEPTED WITHIN A REASONABLE ;
                2802  | AMOUNT OF TIME, SETTING THE DISKETTE STATUS ON COMPLETION. ;
                2803  | INPUT                                       ;
                2804  | (AH)  BYTE TO BE OUTPUT                 ;
                2805  | OUTPUT                                       ;
                2806  | CY = 0  SUCCESS                          ;
                2807  | CY = 1  FAILURE -- DISKETTE STATUS UPDATED ;
                2808  | IF A FAILURE HAS OCCURRED, THE RETURN IS MADE ONE LEVEL ;
                2809  | HIGHER THAN THE CALLER OF NEC_OUTPUT. ;
                2810  | THIS REMOVES THE REQUIREMENT OF TESTING AFTER EVERY ;
                2811  | CALL OF NEC_OUTPUT. ;
                2812  | (AL) DESTROYED ;
                2813  |-----|
EE41             2814  NEC_OUTPUT  PROC  NEAR
EE41 52          2815          PUSH  DX                       ; SAVE REGISTERS
EE42 51          2816          PUSH  CX
EE43 BAF403     2817          MOV   DX,03F4H                ; STATUS PORT
EE46 33C9       2818          XOR   CX,CX                ; COUNT FOR TIME OUT
EE48            2819  J23:
EE48 EC         2820          IN   AL,DX                ; GET STATUS
EE49 A840       2821          TEST  AL,040H                ; TEST DIRECTION BIT
EE4B 740C       2822          JZ   J25                    ; DIRECTION OK
EE4D E2F9       2823          LOOP J23
EE4F            2824  J24:
EE4F 800E41000  2825          OR   DISKETTE_STATUS,TIME_OUT
EE54 59         2826          POP   CX
EE55 5A         2827          POP   DX                       ; SET ERROR CODE AND RESTORE REGS
EE56 58         2828          POP   AX                       ; DISCARD THE RETURN ADDRESS
EE57 F9         2829          STC                               ; INDICATE ERROR TO CALLER
EE58 C3         2830          RET
EE59            2831  J25:
EE59 33C9       2832          XOR   CX,CX                ; RESET THE COUNT
EE5B            2833  J26:
EE5B EC         2834          IN   AL,DX                ; GET THE STATUS
EE5C A880       2835          TEST  AL,080H                ; IS IT READY
EE5E 7504       2836          JNZ  J27                    ; YES, GO OUTPUT
EE60 E2F9       2837          LOOP J26                    ; COUNT DOWN AND TRY AGAIN
EE62 EBEB       2838          JMP  J24                    ; ERROR CONDITION
EE64            2839  J27:
EE64 8AC4       2840          MOV   AL,AH                ; GET BYTE TO OUTPUT
EE66 B2F5       2841          MOV   DL,0F5H                ; DATA PORT (3F5)
EE68 EE         2842          OUT  DX,AL                ; OUTPUT THE BYTE
EE69 59         2843          POP   CX                       ; RECOVER REGISTERS
EE6A 5A         2844          POP   DX
EE6B C3         2845          RET                               ; CY = 0 FROM TEST INSTRUCTION
                2846  NEC_OUTPUT  ENDP
                2847  |-----|
                2848  | GET_PARM                               ;
                2849  | THIS ROUTINE FETCHES THE INDEXED POINTER FROM THE DISK_BASE ;
                2850  | BLOCK POINTED AT BY THE DATA VARIABLE DISK_POINTER. A BYTE FROM ;
                2851  | THAT TABLE IS THEN MOVED INTO AH, THE INDEX OF THAT BYTE BEING ;
                2852  | THE PARM IN BX ;
                2853  | ENTRY -- ;
                2854  | BX = INDEX OF BYTE TO BE FETCHED * 2 ;
                2855  | IF THE LOW BIT OF BX IS ON, THE BYTE IS IMMEDIATELY OUTPUT ;
                2856  | TO THE NEC CONTROLLER ;
                2857  | EXIT -- ;
                2858  | AH = THAT BYTE FROM BLOCK ;
                2859  |-----|
EE6C             2860  GET_PARM  PROC  NEAR
EE6C 1E         2861          PUSH  DS                       ; SAVE SEGMENT
EE6D 2BC0       2862          SUB   AX,AX                       ; ZERO TO AX
EE6F 8ED8       2863          MOV   DS,AX
                2864          ASSUME DS:ABS0
EE71 C5367800  2865          LDS  SI,DISK_POINTER            ; POINT TO BLOCK
EE75 D1EB       2866          SHR  BX,1                       ; DIVIDE BX BY 2, AND SET FLAG
                2867          ; FOR EXIT
EE77 8A20       2868          MOV  AH,[SI+BX]                ; GET THE WORD

```

```

LOC OBJ          LINE  SOURCE
EE79 1F          2869          POP      DS          ; RESTORE SEGMENT
                2870          ASSUME  DS:DATA
EE7A 72C5        2871          JC       NEC_OUTPUT  ; IF FLAG SET, OUTPUT TO CONTROLLER
EE7C C3          2872          RET          ; RETURN TO CALLER
                2873          GET_PARM  ENDP
                2874          ;-----
                2875          ; SEEK
                2876          ; THIS ROUTINE WILL MOVE THE HEAD ON THE NAMED DRIVE TO THE
                2877          ; NAMED TRACK. IF THE DRIVE HAS NOT BEEN ACCESSED SINCE THE
                2878          ; DRIVE RESEY COMMAND WAS ISSUED, THE DRIVE WILL BE RECALIBRATED.
                2879          ; INPUT
                2880          ; (DL) = DRIVE TO SEEK ON
                2881          ; (CH) = TRACK TO SEEK TO
                2882          ; OUTPUT
                2883          ; CY = 0 SUCCESS
                2884          ; CY = 1 FAILURE -- DISKETTE_STATUS SET ACCORDINGLY
                2885          ; (AX) DESTROYED
                2886          ;-----
EE7D             2887          SEEK     PROC     NEAR
EE7D B001        2888          MOV     AL,1          ; ESTABLISH MASK FOR RECAL TEST
EE7F 51          2889          PUSH   CX            ; SAVE INPUT VALUES
EE80 8ACA        2890          MOV     CL,DL         ; GET DRIVE VALUE INTO CL
EE82 D2C0        2891          ROL    AL,CL         ; SHIFT IT BY THE DRIVE VALUE
EE84 59          2892          POP     CX            ; RECOVER TRACK VALUE
EE85 84063E00    2893          TEST   AL,SEEK_STATUS ; TEST FOR RECAL REQUIRED
EE89 7513        2894          JNZ    J28           ; NO_RECAL
EE8B 08063E00    2895          OR     SEEK_STATUS,AL ; TURN ON THE NO RECAL BIT IN FLAG
EE8F B407        2896          MOV     AH,07H        ; RECALIBRATE COMMAND
EE91 EBADFF      2897          CALL   NEC_OUTPUT
EE94 8AE2        2898          MOV     AH,DL
EE96 EBA8FF      2899          CALL   NEC_OUTPUT    ; OUTPUT THE DRIVE NUMBER
EE99 E07600      2900          CALL   CHK_STAT_2    ; GET THE INTERRUPT AND SENSE INT STATUS
EE9C 7229        2901          JC     J32           ; SEEK_ERROR
                2902
                2903          ;----- DRIVE IS IN SYNCH WITH CONTROLLER, SEEK TO TRACK
                2904
EE9E             2905          J28:
EE9E B40F        2906          MOV     AH,0FH        ; SEEK COMMAND TO NEC
EEA0 E89EFF      2907          CALL   NEC_OUTPUT
EEA3 8AE2        2908          MOV     AH,DL         ; DRIVE NUMBER
EEA5 E899FF      2909          CALL   NEC_OUTPUT
EEA8 8AE5        2910          MOV     AH,CH         ; TRACK NUMBER
EEAA E894FF      2911          CALL   NEC_OUTPUT
EEAD E86200      2912          CALL   CHK_STAT_2    ; GET ENDING INTERRUPT AND
                2913          ; SENSE STATUS
                2914
                2915          ;----- WAIT FOR HEAD SETTLE
                2916
EEB0 9C          2917          PUSHF
EEB1 BB1200      2918          MOV     BX,18         ; SAVE STATUS FLAGS
EEB4 E8B5FF      2919          CALL   GET_PARM      ; GET HEAD SETTLE PARAMETER
EEB7 51          2920          PUSH   CX            ; SAVE REGISTER
EEB8             2921          J29:                ; HEAD SETTLE
EEB8 B92602      2922          MOV     CX,550        ; 1 MS LOOP
EEBB 0AE4        2923          OR     AH,AH         ; TEST FOR TIME EXPIRED
EEBD 7406        2924          JZ     J31           ;
EEBF             2925          J30:
EEBF E2FE        2926          LOOP  J30           ; DELAY FOR 1 MS
EEC1 FECC        2927          DEC   AH             ; DECREMENT THE COUNT
EEC3 EBF3        2928          JMP   J29           ; DO IT SOME MORE
EEC5             2929          J31:
EEC5 59          2930          POP     CX            ; RECOVER STATE
EEC6 90          2931          POPF
EEC7             2932          J32:                ; SEEK_ERROR
EEC7 C3          2933          RET          ; RETURN TO CALLER
                2934          SEEK     ENDP
                2935          ;-----
                2936          ; DMA_SETUP
                2937          ; THIS ROUTINE SETS UP THE DMA FOR READ/WRITE/VERIFY OPERATIONS.
                2938          ; INPUT
                2939          ; (AL) = MODE BYTE FOR THE DMA
                2940          ; (ES:BX) - ADDRESS TO READ/WRITE THE DATA
                2941          ; OUTPUT
                2942          ; (AX) DESTROYED
                2943          ;-----
EEC8             2944          DMA_SETUP PROC NEAR
EEC8 51          2945          PUSH   CX            ; SAVE THE REGISTER

```

LOC OBJ	LINE	SOURCE	
EEC9 FA	2946	CLI	; NO MORE INTERRUPTS
EECA E60C	2947	OUT DMA+12,AL	; SET THE FIRST/LAST F/F
EECC 50	2948	PUSH AX	
EECD 58	2949	POP AX	
EECE E60B	2950	OUT DMA+11,AL	; OUTPUT THE MODE BYTE
EED0 8CC0	2951	MOV AX,ES	; GET THE ES VALUE
EED2 B104	2952	MOV CL,4	; SHIFT COUNT
EED4 D3C0	2953	ROL AX,CL	; ROTATE LEFT
EED6 8AE8	2954	MOV CH,AL	; GET HIGHEST NYBLE OF ES TO CH
EED8 24F0	2955	AND AL,0F0H	; ZERO THE LOW NYBBLE FROM SEGMENT
EEDA 03C3	2956	ADD AX,BX	; TEST FOR CARRY FROM ADDITION
EEDC 7302	2957	JNC J33	
EEDE FEC5	2958	INC CH	; CARRY MEANS HIGH 4 BITS MUST BE INC
EEEE	2959	J33:	
EEEE 50	2960	PUSH AX	; SAVE START ADDRESS
EEE1 E604	2961	OUT DMA+4,AL	; OUTPUT LOW ADDRESS
EEE3 8AC4	2962	MOV AL,AH	
EEE5 E604	2963	OUT DMA+4,AL	; OUTPUT HIGH ADDRESS
EEE7 8AC5	2964	MOV AL,CH	; GET HIGH 4 BITS
EEE9 240F	2965	AND AL,0FH	
EEEB E681	2966	OUT 081H,AL	; OUTPUT THE HIGH 4 BITS TO ; THE PAGE REGISTER
	2967		
	2968		
	2969	I----- DETERMINE COUNT	
	2970		
EEEE 8AE6	2971	MOV AH,DH	; NUMBER OF SECTORS
EEEF 2AC0	2972	SUB AL,AL	; TIMES 256 INTO AX
EEF1 D1E8	2973	SHR AX,1	; SECTORS * 128 INTO AX
EEF3 50	2974	PUSH AX	
EEF4 BB0600	2975	MOV BX,6	; GET THE BYTES/SECTOR PARM
EEF7 E672FF	2976	CALL GET_PARM	
EEFA 8ACC	2977	MOV CL,AH	; USE AS SHIFT COUNT (0=128, 1=256 ETC)
EEFC 58	2978	POP AX	
EEFD D3E0	2979	SHL AX,CL	; MULTIPLY BY CORRECT AMOUNT
EEFF 48	2980	DEC AX	; -1 FOR DMA VALUE
EF00 50	2981	PUSH AX	; SAVE COUNT VALUE
EF01 E605	2982	OUT DMA+5,AL	; LOW BYTE OF COUNT
EF03 8AC4	2983	MOV AL,AH	
EF05 E605	2984	OUT DMA+5,AL	; HIGH BYTE OF COUNT
EF07 FB	2985	STI	; INTERRUPTS BACK ON
EF08 59	2986	POP CX	; RECOVER COUNT VALUE
EF09 58	2987	POP AX	; RECOVER ADDRESS VALUE
EF0A 03C1	2988	ADD AX,CX	; ADD, TEST FOR 64K OVERFLOW
EF0C 59	2989	POP CX	; RECOVER REGISTER
EF0D B002	2990	MOV AL,2	; MODE FOR 8237
EF0F E60A	2991	OUT DMA+10,AL	; INITIALIZE THE DISKETTE CHANNEL
EF11 C3	2992	RET	; RETURN TO CALLER,
	2993		; CFL SET BY ABOVE IF ERROR
	2994	DMA_SETUP	ENDP
	2995		
	2996	; CHK_STAT_2	:
	2997	; THIS ROUTINE HANDLES THE INTERRUPT RECEIVED AFTER A	:
	2998	; RECALIBRATE, SEEK, OR RESET TO THE ADAPTER.	:
	2999	; THE INTERRUPT IS WAITED FOR, THE INTERRUPT STATUS SENSED,	:
	3000	; AND THE RESULT RETURNED TO THE CALLER.	:
	3001	; INPUT	:
	3002	; NONE	:
	3003	; OUTPUT	:
	3004	; CY = 0 SUCCESS	:
	3005	; CY = 1 FAILURE -- ERROR IS IN DISKETTE_STATUS	:
	3006	; (AX) DESTROYED	:
	3007	;	:
	3008	CHK_STAT_2	PROC NEAR
EF12	3009	CALL WAIT_INT	; WAIT FOR THE INTERRUPT
EF12 E81E00	3010	JC J34	; IF ERROR, RETURN IT
EF15 7214	3011	MOV AH,08H	; SENSE INTERRUPT STATUS COMMAND
EF17 B408	3012	CALL NEC_OUTPUT	
EF19 E825FF	3013	CALL RESULTS	; READ IN THE RESULTS
EF1C E84A00	3014	JC J34	; CHK2_RETURN
EF1F 720A	3015	MOV AL,NEC_STATUS	; GET THE FIRST STATUS BYTE
EF21 A04200	3016	AND AL,060H	; ISOLATE THE BITS
EF24 2460	3017	CHP AL,060H	; TEST FOR CORRECT VALUE
EF26 3C60	3018	JZ J35	; IF ERROR, GO MARK IT
EF28 7402	3019	CLC	; GOOD RETURN
EF2A F8	3020	J34:	
EF2B	3021	RET	; RETURN TO CALLER
EF2B C3	3022	J35:	; CHK2_ERROR
EF2C			


```

LOC OBJ          LINE SOURCE
EF2C 800E410040 3023          OR   DISKETTE_STATUS,BAD_SEEK
EF31 F9          3024          STC          ; ERROR RETURN CODE
EF32 C3          3025          RET
3026          CHK_STAT_2      ENDP
3027          ;-----
3028          ; WAIT INT          :
3029          ; THIS ROUTINE WAITS FOR AN INTERRUPT TO OCCUR. A TIME OUT :
3030          ; ROUTINE TAKES PLACE DURING THE WAIT, SO THAT AN ERROR MAY BE :
3031          ; RETURNED IF THE DRIVE IS NOT READY.          :
3032          ; INPUT          :
3033          ; NONE          :
3034          ; OUTPUT          :
3035          ; CY = 0 SUCCESS          :
3036          ; CY = 1 FAILURE -- DISKETTE_STATUS IS SET ACCORDINGLY          :
3037          ; (AX) DESTROYED          :
3038          ;-----
EF33          3039          WAIT_INT      PROC      NEAR
EF33 FB          3040          STI          ; TURN ON INTERRUPTS, JUST IN CASE
EF34 53          3041          PUSH      BX
EF35 51          3042          PUSH      CX          ; SAVE REGISTERS
EF36 B302        3043          MOV      BL,2          ; CLEAR THE COUNTERS
EF38 33C9        3044          XOR      CX,CX          ; FOR 2 SECOND WAIT
EF3A          3045          J36:
EF3A F6063E0080 3046          TEST     SEEK_STATUS,INT_FLAG ; TEST FOR INTERRUPT OCCURRING
EF3F 750C        3047          JNZ     J37
EF41 E2F7        3048          LOOP    J36          ; COUNT DOWN WHILE WAITING
EF43 FECB        3049          DEC     BL          ; SECOND LEVEL COUNTER
EF45 75F3        3050          JNZ     J36
EF47 800E410080 3051          OR      DISKETTE_STATUS,TIME_OUT ; NOTHING HAPPENED
EF4C F9          3052          STC          ; ERROR RETURN
EF4D          3053          J37:
EF4D 9C          3054          PUSHF         ; SAVE CURRENT CARRY
EF4E 80263E007F 3055          AND     SEEK_STATUS,NOT_INT_FLAG ; TURN OFF INTERRUPT FLAG
EF53 9D          3056          POPF        ; RECOVER CARRY
EF54 59          3057          POP      CX
EF55 5B          3058          POP      BX          ; RECOVER REGISTERS
EF56 C3          3059          RET          ; GOOD RETURN CODE COMES
3060          ; FROM TEST INST
3061          WAIT_INT      ENDP
3062          ;-----
3063          ; DISK_INT          :
3064          ; THIS ROUTINE HANDLES THE DISKETTE INTERRUPT          :
3065          ; INPUT          :
3066          ; NONE          :
3067          ; OUTPUT          :
3068          ; THE INTERRUPT FLAG IS SET IS SEEK_STATUS          :
3069          ;-----
EF57          3070          ORG      0EF57H
EF57          3071          DISK_INT      PROC      FAR
EF57 FB          3072          STI          ; RE ENABLE INTERRUPTS
EF58 1E          3073          PUSH      DS
EF59 50          3074          PUSH      AX
EF5A E8E10F      3075          CALL     DDS
EF5D 800E3E0080 3076          OR      SEEK_STATUS,INT_FLAG
EF62 B020        3077          MOV     AL,20H          ; END OF INTERRUPT MARKER
EF64 E620        3078          OUT     20H,AL          ; INTERRUPT CONTROL PORT
EF66 58          3079          POP     AX
EF67 1F          3080          POP     DS          ; RECOVER SYSTEM
EF68 CF          3081          IRET         ; RETURN FROM INTERRUPT
3082          DISK_INT      ENDP
3083          ;-----
3084          ; RESULTS          :
3085          ; THIS ROUTINE WILL READ ANYTHING THAT THE NEC CONTROLLER HAS :
3086          ; TO SAY FOLLOWING AN INTERRUPT.          :
3087          ; INPUT          :
3088          ; NONE          :
3089          ; OUTPUT          :
3090          ; CY = 0 SUCCESSFUL TRANSFER          :
3091          ; CY = 1 FAILURE -- TIME OUT IN WAITING FOR STATUS          :
3092          ; NEC_STATUS AREA HAS STATUS BYTE LOADED INTO IT          :
3093          ; (AH) DESTROYED          :
3094          ;-----
EF69          3095          RESULTS      PROC      NEAR
EF69 FC          3096          CLD
EF6A BF4200      3097          MOV     DI,OFFSET_NEC_STATUS ; POINTER TO DATA AREA
EF6D 51          3098          PUSH     CX          ; SAVE COUNTER
EF6E 52          3099          PUSH     DX

```

LOC OBJ	LINE	SOURCE	
EF6F 53	3100	PUSH BX	
EF70 8307	3101	MOV BL,7	; MAX STATUS BYTES
	3102		
	3103	;----- WAIT FOR REQUEST FOR MASTER	
	3104		
EF72	3105	J38:	; INPUT_LOOP
EF72 33C9	3106	XOR CX,CX	; COUNTER
EF74 BAF403	3107	MOV DX,03F4H	; STATUS PORT
EF77	3108	J39:	; WAIT FOR MASTER
EF77 EC	3109	IN AL,DX	; GET STATUS
EF78 A880	3110	TEST AL,080H	; MASTER READY
EF7A 750C	3111	JNZ J40A	; TEST_DIR
EF7C E2F9	3112	LOOP J39	; WAIT_MASTER
EF7E 800E410080	3113	OR DISKETTE_STATUS,TIME_OUT	
EF83	3114	J40:	; RESULTS_ERROR
EF83 F9	3115	STC	; SET ERROR RETURN
EF84 5B	3116	POP BX	
EF85 5A	3117	POP DX	
EF86 59	3118	POP CX	
EF87 C3	3119	RET	
	3120		
	3121	;----- TEST THE DIRECTION BIT	
	3122		
EF88	3123	J40A:	
EF88 EC	3124	IN AL,DX	; GET STATUS REG AGAIN
EF89 A840	3125	TEST AL,040H	; TEST DIRECTION BIT
EF8B 7507	3126	JNZ J42	; OK TO READ STATUS
EF8D	3127	J41:	; NEC_FAIL
EF8D 800E410020	3128	OR DISKETTE_STATUS,BAD_NEC	
EF92 EBEF	3129	JMP J40	; RESULTS_ERROR
	3130		
	3131	;----- READ IN THE STATUS	
	3132		
EF94	3133	J42:	; INPUT_STAT
EF94 42	3134	INC DX	; POINT AT DATA PORT
EF95 EC	3135	IN AL,DX	; GET THE DATA
EF96 8805	3136	MOV [DI],AL	; STORE THE BYTE
EF98 47	3137	INC DI	; INCREMENT THE POINTER
EF99 B90A00	3138	MOV CX,10	; LOOP TO KILL TIME FOR NEC
EF9C E2FE	3139	J43: LOOP J43	
EF9E 4A	3140	DEC DX	; POINT AT STATUS PORT
EF9F EC	3141	IN AL,DX	; GET STATUS
EFA0 A810	3142	TEST AL,010H	; TEST FOR NEC STILL BUSY
EFA2 7406	3143	JZ J44	; RESULTS DONE
EFA4 FECB	3144	DEC BL	; DECREMENT THE STATUS COUNTER
EFA6 75CA	3145	JNZ J38	; GO BACK FOR MORE
EFA8 EBE3	3146	JMP J41	; CHIP HAS FAILED
	3147		
	3148	;----- RESULT OPERATION IS DONE	
	3149		
EFAA	3150	J44:	
EFAA 5B	3151	POP BX	
EFA8 5A	3152	POP DX	
EFA8 59	3153	POP CX	; RECOVER REGISTERS
EFA8 C3	3154	RET	; GOOD RETURN CODE FROM TEST INST
	3155	;-----	
	3156	; NUM_TRANS	:
	3157	; THIS ROUTINE CALCULATES THE NUMBER OF SECTORS THAT	:
	3158	; WERE ACTUALLY TRANSFERRED TO/FROM THE DISKETTE	:
	3159	; INPUT	:
	3160	; (CH) = CYLINDER OF OPERATION	:
	3161	; (CL) = START SECTOR OF OPERATION	:
	3162	; OUTPUT	:
	3163	; (AL) = NUMBER ACTUALLY TRANSFERRED	:
	3164	; NO OTHER REGISTERS MODIFIED	:
	3165	;-----	
EFAE	3166	NUM_TRANS PROC NEAR	
EFAE A04500	3167	MOV AL,NEC_STATUS+3	; GET CYLINDER ENDED UP ON
EFB1 3AC5	3168	CHP AL,CH	; SAME AS WE STARTED
EFB3 A04700	3169	MOV AL,NEC_STATUS+5	; GET ENDING SECTOR
EFB6 740A	3170	JZ J45	; IF ON SAME CYL, THEN NO ADJUST
EFB8 BB0800	3171	MOV BX,8	
EFBB E8AEFE	3172	CALL GET_PARAM	; GET EOT VALUE
EFBE 8AC4	3173	MOV AL,AH	; INTO AL
EFC0 FEC0	3174	INC AL	; USE EOT+1 FOR CALCULATION
EFC2	3175	J45:	
EFC2 2AC1	3176	SUB AL,CL	; SUBTRACT START FROM END

```

LOC OBJ          LINE    SOURCE
EFC4 C3          3177          RET
                 3178      MUM_TRANS      ENDP
                 3179      RESULTS ENDP
                 3180
                 3181      ;-----
                 3181      ; DISK_BASE
                 3182      ;
                 3182      ; THIS IS THE SET OF PARAMETERS REQUIRED FOR DISKETTE OPERATION.
                 3183      ; THEY ARE POINTED AT BY THE DATA VARIABLE DISK_POINTER. TO
                 3184      ; MODIFY THE PARAMETERS, BUILD ANOTHER PARAMETER BLOCK AND POINT
                 3185      ; DISK_POINTER TO IT.
                 3186      ;-----
EFC7             3187      ORG      0EFC7H
EFC7             3188      DISK_BASE LABEL BYTE
EFC7 CF          3189      DB      11001111B      ; SRT=C, HD UNLOAD=0F - 1ST SPECIFY BYTE
EFC8 02          3190      DB      2          ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
EFC9 25          3191      DB      MOTOR_WAIT    ; WAIT AFTER OPN TIL MOTOR OFF
EFCA 02          3192      DB      2          ; 512 BYTES/SECTOR
EFCB 08          3193      DB      8          ; EOT ( LAST SECTOR ON TRACK)
EFCC 2A          3194      DB      02AH      ; GAP LENGTH
EFCD FF          3195      DB      0FFH      ; DTL
EFCE 50          3196      DB      050H      ; GAP LENGTH FOR FORMAT
EFCF F6          3197      DB      0F6H      ; FILL BYTE FOR FORMAT
EFD0 19          3198      DB      25         ; HEAD SETTLE TIME (MILLISECONDS)
EFD1 04          3199      DB      4          ; MOTOR START TIME (1/8 SECONDS)
                 3200
                 3201      ;--- INT 17 ---
                 3202      ; PRINTER_IO
                 3203      ; THIS ROUTINE PROVIDES COMMUNICATION WITH THE PRINTER
                 3204      ; INPUT
                 3205      ; (AH)=0 PRINT THE CHARACTER IN (AL)
                 3206      ; ON RETURN, AH=1 IF CHARACTER COULD NOT BE PRINTED
                 3207      ; (TIME OUT). OTHER BITS SET AS ON NORMAL STATUS CALL
                 3208      ; (AH)=1 INITIALIZE THE PRINTER PORT
                 3209      ; RETURNS WITH (AH) SET WITH PRINTER STATUS
                 3210      ; (AH)=2 READ THE PRINTER STATUS INTO (AH)
                 3211      ; 7 6 5 4 3 2-1 0
                 3212      ; | | | | | | | TIME OUT
                 3213      ; | | | | | | | UNUSED
                 3214      ; | | | | | | | 1 = I/O ERROR
                 3215      ; | | | | | | | 1 = SELECTED
                 3216      ; | | | | | | | 1 = OUT OF PAPER
                 3217      ; | | | | | | | 1 = ACKNOWLEDGE
                 3218      ; | | | | | | | 1 = NOT BUSY
                 3219      ;
                 3220      ; (DX) = PRINTER TO BE USED (0,1,2) CORRESPONDING TO ACTUAL
                 3221      ; VALUES IN PRINTER_BASE AREA
                 3222      ;
                 3223      ; DATA AREA PRINTER_BASE CONTAINS THE BASE ADDRESS OF THE PRINTER
                 3224      ; CARD(S) AVAILABLE (LOCATED AT BEGINNING OF DATA SEGMENT,
                 3225      ; 400H ABSOLUTE, 3 WORDS)
                 3226      ;
                 3227      ; DATA AREA PRINT_TIM_OUT (BYTE) MAY BE CHANGED TO CAUSE DIFFERENT
                 3228      ; TIME-OUT WAITS. DEFAULT=20
                 3229      ;
                 3230      ; REGISTERS AH IS MODIFIED
                 3231      ; ALL OTHERS UNCHANGED
                 3232      ;-----
EFD2             3233      ASSUME CS:CODE,DS:DATA
EFD2             3234      ORG      0EFD2H
EFD2             3235      PRINTER_IO PROC FAR
EFD2 FB          3236      STI          ; INTERRUPTS BACK ON
EFD3 1E          3237      PUSH     DS      ; SAVE SEGMENT
EFD4 52          3238      PUSH     DX
EFD5 56          3239      PUSH     SI
EFD6 51          3240      PUSH     CX
EFD7 53          3241      PUSH     BX
EFD8 E8630F      3242      CALL    D0S
EFD8 8BF2        3243      MOV     SI,DX      ; GET PRINTER PARM
EFD0 8A5C78      3244      MOV     BL,PRINT TIM OUT(SI) ; LOAD TIME-OUT PARM
EFD0 D1E6        3245      SHL     SI,1      ; WORD OFFSET INTO TABLE
EFE2 8B5408      3246      MOV     DX,PRINTER_BASE(SI) ; GET BASE ADDRESS FOR PRINTER CARD
EFE5 0B02        3247      OR     DX,DX      ; TEST DX FOR ZERO,
                 3248      ; INDICATING NO PRINTER
EFE7 740C        3249      JZ     B1          ; RETURN
EFE9 0AE4        3250      OR     AH,AH      ; TEST FOR (AH)=0
EFEB 740E        3251      JZ     B2          ; PRINT_AL
EFED FECC        3252      DEC     AH        ; TEST FOR (AH)=1
EFEF 743F        3253      JZ     B8          ; INIT_PRT

```

LOC OBJ	LINE	SOURCE	
EFF1 FECC	3254	DEC AH	; TEST FOR (AH)=2
EFF3 7428	3255	JZ B5	; PRINTER STATUS
EFF5	3256	B1:	; RETURN
EFF5 5B	3257	POP BX	
EFF6 59	3258	POP CX	
EFF7 5E	3259	POP SI	; RECOVER REGISTERS
EFF8 5A	3260	POP DX	; RECOVER REGISTERS
EFF9 1F	3261	POP DS	
EFFA CF	3262	IRET	
	3263		
	3264	;----- PRINT THE CHARACTER IN (AL)	
	3265		
EFFB	3266	B2:	
EFFB 50	3267	PUSH AX	; SAVE VALUE TO PRINT
EFFC EE	3268	OUT DX,AL	; OUTPUT CHAR TO PORT
EFFD 42	3269	INC DX	; POINT TO STATUS PORT
EFFE	3270	B3:	
EFFE 2BC9	3271	SUB CX,CX	; WAIT_BUSY
F000	3272	B3_1:	
F000 EC	3273	IN AL,DX	; GET STATUS
F001 8AE0	3274	MOV AH,AL	; STATUS TO AH ALSO
F003 A880	3275	TEST AL,80H	; IS THE PRINTER CURRENTLY BUSY
F005 750E	3276	JNZ B4	; OUT_STROBE
F007 E2F7	3277	LOOP B3_1	; TRY AGAIN
F009 FECB	3278	DEC BL	; DROP LOOP COUNT
F00B 75F1	3279	JNZ B3	; GO TILL TIMEOUT ENDS
F00D 80CC01	3280	OR AH,1	; SET ERROR FLAG
F010 80E4F9	3281	AND AH,0F9H	; TURN OFF THE OTHER BITS
F013 EB13	3282	JMP SHORT B7	; RETURN WITH ERROR FLAG SET
F015	3283	B4:	; OUT_STROBE
F015 B00D	3284	MOV AL,0DH	; SET THE STROBE HIGH
F017 42	3285	INC DX	; STROBE IS BIT 0 OF PORT C OF 8255
F018 EE	3286	OUT DX,AL	
F019 B00C	3287	MOV AL,0CH	; SET THE STROBE LOW
F01B EE	3288	OUT DX,AL	
F01C 58	3289	POP AX	; RECOVER THE OUTPUT CHAR
	3290		
	3291	;----- PRINTER STATUS	
	3292		
F01D	3293	B5:	
F01D 50	3294	PUSH AX	; SAVE AL REG
F01E	3295	B6:	
F01E 8B540B	3296	MOV DX,PRINTER_BASE[SI]	
F021 42	3297	INC DX	
F022 EC	3298	IN AL,DX	; GET PRINTER STATUS
F023 8AE0	3299	MOV AH,AL	
F025 80E4F8	3300	AND AH,0F8H	; TURN OFF UNUSED BITS
F028	3301	B7:	; STATUS_SET
F028 5A	3302	POP DX	; RECOVER AL REG
F029 8AC2	3303	MOV AL,DL	; GET CHARACTER INTO AL
F02B 80F448	3304	XOR AH,48H	; FLIP A COUPLE OF BITS
F02E EBC5	3305	JMP B1	; RETURN FROM ROUTINE
	3306		
	3307	;----- INITIALIZE THE PRINTER PORT	
	3308		
F030	3309	B8:	
F030 50	3310	PUSH AX	; SAVE AL
F031 42	3311	INC DX	; POINT TO OUTPUT PORT
F032 42	3312	INC DX	
F033 B008	3313	MOV AL,8	; SET INIT LINE LOW
F035 EE	3314	OUT DX,AL	
F036 B8E803	3315	MOV AX,1000	
F039	3316	B9:	; INIT_LOOP
F039 48	3317	DEC AX	; LOOP FOR RESET TO TAKE
F03A 75FD	3318	JNZ B9	; INIT_LOOP
F03C B00C	3319	MOV AL,0CH	; NO INTERRUPTS, NON AUTO LF,
	3320		; INIT HIGH
F03E EE	3321	OUT DX,AL	
F03F EBDD	3322	JMP B6	; PRT_STATUS_1
	3323	PRINTER_IO	ENDP
	3324		
F041 62E1	3325	C2 DW C24	; RETURN ADDRESS FOR DUMMY STACK
	3326		
	3327	;--- INT 10 -----	
	3328	; VIDEO_IO	:
	3329	; THESE ROUTINES PROVIDE THE CRT INTERFACE	:
	3330	; THE FOLLOWING FUNCTIONS ARE PROVIDED:	:

LOC OBJ	LINE	SOURCE
	3331	;(AH)=0 SET MODE (AL) CONTAINS MODE VALUE ;
	3332	;(AL)=0 40X25 BW (POWER ON DEFAULT) ;
	3333	;(AL)=1 40X25 COLOR ;
	3334	;(AL)=2 80X25 BW ;
	3335	;(AL)=3 80X25 COLOR ;
	3336	; GRAPHICS MODES ;
	3337	;(AL)=4 320X200 COLOR ;
	3338	;(AL)=5 320X200 BW ;
	3339	;(AL)=6 640X200 BW ;
	3340	; CRT MODE=7 80X25 B&W CARD (USED INTERNAL TO VIDEO ONLY) ;
	3341	; *** NOTE BW MODES OPERATE SAME AS COLOR MODES, BUT ;
	3342	; COLOR BURST IS NOT ENABLED ;
	3343	;(AH)=1 SET CURSOR TYPE ;
	3344	;(CH) = BITS 4-0 = START LINE FOR CURSOR ;
	3345	; ** HARDWARE WILL ALWAYS CAUSE BLIN ;
	3346	; ** SETTING BIT 5 OR 6 WILL CAUSE ERRATIC ;
	3347	; BLINKING OR NO CURSOR AT ALL ;
	3348	;(CL) = BITS 4-0 = END LINE FOR CURSOR ;
	3349	;(AH)=2 SET CURSOR POSITION ;
	3350	;(DH,DL) = ROW,COLUMN (0,0) IS UPPER LEFT ;
	3351	;(BH) = PAGE NUMBER (MUST BE 0 FOR GRAPHICS MODES) ;
	3352	;(AH)=3 READ CURSOR POSITION ;
	3353	;(BH) = PAGE NUMBER (MUST BE 0 FOR GRAPHICS MODES) ;
	3354	; ON EXIT (DH,DL) = ROW,COLUMN OF CURRENT CURSOR ;
	3355	; (CH,CL) = CURSOR MODE CURRENTLY SET ;
	3356	;(AH)=4 READ LIGHT PEN POSITION ;
	3357	; ON EXIT: ;
	3358	;(AH) = 0 -- LIGHT PEN SWITCH NOT DOWN/HOT TRIGGERED ;
	3359	;(AH) = 1 -- VALID LIGHT PEN VALUE IN REGISTERS ;
	3360	;(DH,DL) = ROW,COLUMN OF CHARACTER LP POSN ;
	3361	;(CH) = RASTER LINE (0-199) ;
	3362	;(BX) = PIXEL COLUMN (0-319,639) ;
	3363	;(AH)=5 SELECT ACTIVE DISPLAY PAGE (VALID ONLY FOR ALPHA MODES) ;
	3364	;(AL)=NEW PAGE VAL (0-7 FOR MODES 0&1, 0-3 FOR MODES 2&3);
	3365	;(AH)=6 SCROLL ACTIVE PAGE UP ;
	3366	;(AL) = NUMBER OF LINES, INPUT LINES BLANKED AT BOTTOM ;
	3367	; OF WINDOW ;
	3368	; AL = 0 MEANS BLANK ENTIRE WINDOW ;
	3369	;(CH,CL) = ROW,COLUMN OF UPPER LEFT CORNER OF SCROLL ;
	3370	;(DH,DL) = ROW,COLUMN OF LOWER RIGHT CORNER OF SCROLL ;
	3371	;(BH) = ATTRIBUTE TO BE USED ON BLANK LINE ;
	3372	;(AH)=7 SCROLL ACTIVE PAGE DOWN ;
	3373	;(AL) = NUMBER OF LINES, INPUT LINES BLANKED AT TOP ;
	3374	; OF WINDOW ;
	3375	; AL = 0 MEANS BLANK ENTIRE WINDOW ;
	3376	;(CH,CL) = ROW,COLUMN OF UPPER LEFT CORNER OF SCROLL ;
	3377	;(DH,DL) = ROW,COLUMN OF LOWER RIGHT CORNER OF SCROLL ;
	3378	;(BH) = ATTRIBUTE TO BE USED ON BLANK LINE ;
	3379	; ;
	3380	; CHARACTER HANDLING ROUTINES ;
	3381	; ;
	3382	;(AH) = 8 READ ATTRIBUTE/CHARACTER AT CURRENT CURSOR POSITION ;
	3383	;(BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY) ;
	3384	; ON EXIT: ;
	3385	;(AL) = CHAR READ ;
	3386	;(AH) = ATTRIBUTE OF CHARACTER READ (ALPHA MODES ONLY) ;
	3387	;(AH) = 9 WRITE ATTRIBUTE/CHARACTER AT CURRENT CURSOR POSITION ;
	3388	;(BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY) ;
	3389	;(CX) = COUNT OF CHARACTERS TO WRITE ;
	3390	;(AL) = CHAR TO WRITE ;
	3391	;(BL) = ATTRIBUTE OF CHARACTER (ALPHA)/COLOR OF CHAR ;
	3392	; (GRAPHICS) ;
	3393	; SEE NOTE ON WRITE DOT FOR BIT 7 OF BL = 1. ;
	3394	;(AH) = 10 WRITE CHARACTER ONLY AT CURRENT CURSOR POSITION ;
	3395	;(BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY) ;
	3396	;(CX) = COUNT OF CHARACTERS TO WRITE ;
	3397	;(AL) = CHAR TO WRITE ;
	3398	; FOR READ/WRITE CHARACTER INTERFACE WHILE IN GRAPHICS MODE, THE ;
	3399	; CHARACTERS ARE FORMED FROM A CHARACTER GENERATOR IMAGE ;
	3400	; MAINTAINED IN THE SYSTEM ROM. ONLY THE 1ST 128 CHARS ;
	3401	; ARE CONTAINED THERE. TO READ/WRITE THE SECOND 128 ;
	3402	; CHARS, THE USER MUST INITIALIZE THE POINTER AT ;
	3403	; INTERRUPT 1FH (LOCATION 0007CH) TO POINT TO THE 1K BYTE ;
	3404	; TABLE CONTAINING THE CODE POINTS FOR THE SECOND ;
	3405	; 128 CHARS (128-255). ;
	3406	; FOR WRITE CHARACTER INTERFACE IN GRAPHICS MODE, THE REPLICATION ;
	3407	; FACTOR CONTAINED IN (CX) ON ENTRY WILL PRODUCE VALID ;

```

3408 | RESULTS ONLY FOR CHARACTERS CONTAINED ON THE SAME ROW. |
3409 | CONTINUATION TO SUCCEEDING LINES WILL NOT PRODUCE |
3410 | CORRECTLY. |
3411 | |
3412 | GRAPHICS INTERFACE |
3413 | (AH) = 11 SET COLOR PALETTE |
3414 | (BH) = PALETTE COLOR ID BEING SET (0-127) |
3415 | (BL) = COLOR VALUE TO BE USED WITH THAT COLOR ID |
3416 | NOTE: FOR THE CURRENT COLOR CARD, THIS ENTRY POINT |
3417 | HAS MEANING ONLY FOR 320X200 GRAPHICS. |
3418 | COLOR ID = 0 SELECTS THE BACKGROUND COLOR (0-15); |
3419 | COLOR ID = 1 SELECTS THE PALETTE TO BE USED: |
3420 | 0 = GREEN(1)/RED(2)/YELLOW(3) |
3421 | 1 = CYAN(1)/MAGENTA(2)/WHITE(3) |
3422 | IN 40X25 OR 80X25 ALPHA MODES, THE VALUE SET |
3423 | FOR PALETTE COLOR 0 INDICATES THE |
3424 | BORDER COLOR TO BE USED (VALUES 0-31, |
3425 | WHERE 16-31 SELECT THE HIGH INTENSITY |
3426 | BACKGROUND SET. |
3427 | (AH) = 12 WRITE DOT |
3428 | (DX) = ROW NUMBER |
3429 | (CX) = COLUMN NUMBER |
3430 | (AL) = COLOR VALUE |
3431 | IF BIT 7 OF AL = 1, THEN THE COLOR VALUE IS |
3432 | EXCLUSIVE OR'D WITH THE CURRENT CONTENTS OF |
3433 | THE DOT |
3434 | (AH) = 13 READ DOT |
3435 | (DX) = ROW NUMBER |
3436 | (CX) = COLUMN NUMBER |
3437 | (AL) RETURNS THE DOT READ |
3438 | |
3439 | ASCII TELETYPE ROUTINE FOR OUTPUT |
3440 | |
3441 | (AH) = 14 WRITE TELETYPE TO ACTIVE PAGE |
3442 | (AL) = CHAR TO WRITE |
3443 | (BL) = FOREGROUND COLOR IN GRAPHICS MODE |
3444 | NOTE -- SCREEN WIDTH IS CONTROLLED BY PREVIOUS MODE SET |
3445 | |
3446 | (AH) = 15 CURRENT VIDEO STATE |
3447 | RETURNS THE CURRENT VIDEO STATE |
3448 | (AL) = MODE CURRENTLY SET ( SEE AH=0 FOR EXPLANATION) |
3449 | (AH) = NUMBER OF CHARACTER COLUMNS ON SCREEN |
3450 | (BH) = CURRENT ACTIVE DISPLAY PAGE |
3451 | |
3452 | CS,SS,DS,ES,BX,CX,DX PRESERVED DURING CALL |
3453 | ALL OTHERS DESTROYED |
3454 | ----- |
3455 | ASSUME CS:CODE,DS:DATA,ES:VIDEO_RAM |
3456 | ORG 0F045H |
F045 M1 LABEL WORD ; TABLE OF ROUTINES WITHIN VIDEO I/O
F045 FCF0 3458 DW OFFSET SET_MODE
F047 CDF1 3459 DW OFFSET SET_CTYPE
F049 EEF1 3460 DW OFFSET SET_CPOS
F04B 39F2 3461 DW OFFSET READ_CURSOR
F04D 9CF7 3462 DW OFFSET READ_LPEN
F04F 17F2 3463 DW OFFSET ACT_DISP_PAGE
F051 96F2 3464 DW OFFSET SCROLL_UP
F053 38F3 3465 DW OFFSET SCROLL_DOWN
F055 74F3 3466 DW OFFSET READ_AC_CURRENT
F057 B9F3 3467 DW OFFSET WRITE_AC_CURRENT
F059 ECF3 3468 DW OFFSET WRITE_C_CURRENT
F05B 4EF2 3469 DW OFFSET SET_COLOR
F05D 2FF4 3470 DW OFFSET WRITE_DOT
F05F 1EF4 3471 DW OFFSET READ_DOT
F061 18F7 3472 DW OFFSET WRITE_TTY
F063 74F2 3473 DW OFFSET VIDEO_STATE
0020 M1L EQU 4-H1
3475 |
3476 | ORG 0F065H |
F065 VIDEO_IO PROC NEAR
F065 FB 3478 STI ; INTERRUPTS BACK ON
F066 FC 3479 CLD ; SET DIRECTION FORWARD
F067 06 3480 PUSH ES
F068 1E 3481 PUSH DS ; SAVE SEGMENT REGISTERS
F069 52 3482 PUSH DX
F06A 51 3483 PUSH CX
F06B 53 3484 PUSH BX

```

LOC OBJ	LINE	SOURCE
F06C 56	3485	PUSH SI
F06D 57	3486	PUSH DI
F06E 50	3487	PUSH AX ; SAVE AX VALUE
F06F 8AC4	3488	MOV AL,AH ; GET INTO LOW BYTE
F071 32E4	3489	XOR AH,AH ; ZERO TO HIGH BYTE
F073 D1E0	3490	SAL AX,1 ; *2 FOR TABLE LOOKUP
F075 8BF0	3491	MOV SI,AX ; PUT INTO SI FOR BRANCH
F077 3D2000	3492	CMF AX,H1L ; TEST FOR WITHIN RANGE
F07A 7204	3493	JB H2 ; BRANCH AROUND BRANCH
F07C 58	3494	POP AX ; THROW AWAY THE PARAMETER
F07D E94501	3495	JMP VIDEO_RETURN ; DO NOTHING IF NOT IN RANGE
F080	3496	M2:
F080 E8B80E	3497	CALL DDS
F083 B800B8	3498	MOV AX,0B000H ; SEGMENT FOR COLOR CARD
F086 8B3E1000	3499	MOV DI,EQUIP_FLAG ; GET EQUIPMENT SETTING
F08A 81E73000	3500	AND DI,30H ; ISOLATE CRT SWITCHES
F08E 83FF30	3501	CMF DI,30H ; IS SETTING FOR BW CARD?
F091 7502	3502	JNE H3
F093 B4B0	3503	MOV AH,0B00H ; SEGMENT FOR BW CARD
F095	3504	M3:
F095 8EC0	3505	MOV ES,AX ; SET UP TO POINT AT VIDEO RAM AREAS
F097 58	3506	POP AX ; RECOVER VALUE
F098 8A264900	3507	MOV AH,CRT_MODE ; GET CURRENT MODE INTO AH
F09C 2EFA445F0	3508	JMP WORD PTR CS:[SI+OFFSET M1]
	3509	VIDEO_ID ENDP
	3510	;------
	3511	; SET_MODE :
	3512	; THIS ROUTINE INITIALIZES THE ATTACHMENT TO :
	3513	; THE SELECTED MODE. THE SCREEN IS BLANKED. :
	3514	; INPUT :
	3515	; (AL) = MODE SELECTED (RANGE 0-9) :
	3516	; OUTPUT :
	3517	; NONE :
	3518	;------
	3519	
	3520	;----- TABLES FOR USE IN SETTING OF MODE
	3521	
F0A4	3522	ORG 0F0A4H
F0A4	3523	VIDEO_PARMS LABEL BYTE
	3524	;----- INIT_TABLE
F0A4 38	3525	DB 38H,28H,2DH,0AH,1FH,6,19H ; SET UP FOR 40X25
F0A5 28		
F0A6 2D		
F0A7 0A		
F0A8 1F		
F0A9 06		
F0AA 19		
F0AB 1C	3526	DB 1CH,2,7,6,7
F0AC 02		
F0AD 07		
F0AE 06		
F0AF 07		
F0B0 00	3527	DB 0,0,0,0
F0B1 00		
F0B2 00		
F0B3 00		
0010	3528	M4 EQU 9-VIDEO_PARMS
	3529	
F0B4 71	3530	DB 71H,50H,5AH,0AH,1FH,6,19H ; SET UP FOR 80X25
F0B5 50		
F0B6 5A		
F0B7 0A		
F0B8 1F		
F0B9 06		
F0BA 19		
F0BB 1C	3531	DB 1CH,2,7,6,7
F0BC 02		
F0BD 07		
F0BE 06		
F0BF 07		
F0C0 00	3532	DB 0,0,0,0
F0C1 00		
F0C2 00		
F0C3 00		
	3533	
F0C4 38	3534	DB 38H,28H,2DH,0AH,7FH,6,64H ; SET UP FOR GRAPHICS
F0C5 28		

LOC OBJ	LINE	SOURCE	
F0C6 2D			
F0C7 0A			
F0C8 7F			
F0C9 06			
F0CA 64			
F0CB 70	3535	DB	70H,2,1,6,7
F0CC 02			
F0CD 01			
F0CE 06			
F0CF 07			
F0D0 00	3536	DB	0,0,0,0
F0D1 00			
F0D2 00			
F0D3 00			
	3537		
F0D4 61	3538	DB	61H,50H,52H,0FH,19H,6,19H ; SET UP FOR 80X25 B&H CARD
F0D5 50			
F0D6 52			
F0D7 0F			
F0D8 19			
F0D9 06			
F0DA 19			
F0DB 19	3539	DB	19H,2,0DH,0BH,0CH
F0DC 02			
F0DD 0D			
F0DE 0B			
F0DF 0C			
F0E0 00	3540	DB	0,0,0,0
F0E1 00			
F0E2 00			
F0E3 00			
	3541		
F0E4	3542	M5	LABEL WORD ; TABLE OF REGEN LENGTHS
F0E4 0008	3543	DM	2048 ; 40X25
F0E6 0010	3544	DM	4096 ; 80X25
F0E8 0040	3545	DM	16384 ; GRAPHICS
F0EA 0040	3546	DM	16384
	3547		
	3548]------ COLUMNS
	3549		
F0EC	3550	M6	LABEL BYTE
F0EC 28	3551	DB	40,40,80,80,40,40,80,80
F0ED 28			
F0EE 50			
F0EF 50			
F0F0 28			
F0F1 28			
F0F2 50			
F0F3 50			
	3552		
	3553]------ C_REG_TAB
	3554		
F0F4	3555	M7	LABEL BYTE ; TABLE OF MODE SETS
F0F4 2C	3556	DB	2CH,28H,2DH,29H,2AH,2EH,1EH,29H
F0F5 28			
F0F6 2D			
F0F7 29			
F0F8 2A			
F0F9 2E			
F0FA 1E			
F0FB 29			
	3557		
F0FC	3558	SET_MODE	PROC NEAR
F0FC BAD403	3559	MOV	DX,03D4H ; ADDRESS OF COLOR CARD
F0FF B300	3560	MOV	BL,0 ; MODE SET FOR COLOR CARD
F101 83FF30	3561	CMF	DI,30H ; IS BW CARD INSTALLED
F104 7506	3562	JNE	M8 ; OK WITH COLOR
F106 B007	3563	MOV	AL,7 ; INDICATE BW CARD MODE
F108 B2B4	3564	MOV	DI,0B4H ; ADDRESS OF BW CARD (384)
F10A FEC3	3565	INC	BL ; MODE SET FOR BW CARD
F10C	3566	M8:	
F10C 8AE0	3567	MOV	AH,AL ; SAVE MODE IN AH
F10E A24900	3568	MOV	CRT_MODE,AL ; SAVE IN GLOBAL VARIABLE
F111 89166300	3569	MOV	ADDR_6045,DX ; SAVE ADDRESS OF BASE
F115 1E	3570	PUSH	DS ; SAVE POINTER TO DATA SEGMENT
F116 50	3571	PUSH	AX ; SAVE MODE
F117 52	3572	PUSH	DX ; SAVE OUTPUT PORT VALUE

LOC OBJ	LINE	SOURCE	
F110 03C204	3573	ADD DX,4	; POINT TO CONTROL REGISTER
F11B 8AC3	3574	MOV AL,BL	; GET MODE SET FOR CARD
F11D EE	3575	OUT DX,AL	; RESET VIDEO
F11E 5A	3576	POP DX	; BACK TO BASE REGISTER
F11F 2BC0	3577	SUB AX,AX	; SET UP FOR ABSO SEGMENT
F121 8ED8	3578	MOV DS,AX	; ESTABLISH VECTOR TABLE ADDRESSING
	3579	ASSUME DS:ABSO	
F123 C51E7400	3580	LDS BX,PARM_PTR	; GET POINTER TO VIDEO PARMS
F127 50	3581	POP AX	; RECOVER PARMS
	3582	ASSUME DS:CODE	
F128 B91000	3583	MOV CX,M4	; LENGTH OF EACH ROW OF TABLE
F12B 80FC02	3584	CHP AH,2	; DETERMINE WHICH ONE TO USE
F12E 7210	3585	JC M9	; MODE IS 0 OR 1
F130 03D9	3586	ADD BX,CX	; MOVE TO NEXT ROW OF INIT TABLE
F132 80FC04	3587	CHP AH,4	
F135 7209	3588	JC M9	; MODE IS 2 OR 3
F137 03D9	3589	ADD BX,CX	; MOVE TO GRAPHICS ROW OF INIT_TABLE
F139 80FC07	3590	CHP AH,7	
F13C 7202	3591	JC M9	; MODE IS 4,5, OR 6
F13E 03D9	3592	ADD BX,CX	; MOVE TO BW CARD ROW OF INIT_TABLE
	3593		
	3594	;----- BX POINTS TO CORRECT ROW OF INITIALIZATION TABLE	
	3595		
F140	3596	M9:	; OUT_INIT
F140 50	3597	PUSH AX	; SAVE MODE IN AH
F141 32E4	3598	XOR AH,AH	; AH WILL SERVE AS REGISTER
	3599		; NUMBER DURING LOOP
	3600		
	3601	;----- LOOP THROUGH TABLE, OUTPUTTING REG ADDRESS, THEN VALUE FROM TABLE	
	3602		
F143	3603	M10:	; INIT LOOP
F143 8AC4	3604	MOV AL,AH	; GET 6845 REGISTER NUMBER
F145 EE	3605	OUT DX,AL	
F146 42	3606	INC DX	; POINT TO DATA PORT
F147 FEC4	3607	INC AH	; NEXT REGISTER VALUE
F149 8A07	3608	MOV AL,[BX]	; GET TABLE VALUE
F14B EE	3609	OUT DX,AL	; OUT TO CHIP
F14C 43	3610	INC BX	; NEXT IN TABLE
F14D 4A	3611	DEC DX	; BACK TO POINTER REGISTER
F14E E2F3	3612	LOOP M10	; DO THE WHOLE TABLE
F150 50	3613	POP AX	; GET MODE BACK
F151 1F	3614	POP DS	; RECOVER SEGMENT VALUE
	3615	ASSUME DS:DATA	
	3616		
	3617	;----- FILL REGEN AREA WITH BLANK	
	3618		
F152 33FF	3619	XOR DI,DI	; SET UP POINTER FOR REGEN
F154 893E4E00	3620	MOV CRT_START,DI	; START ADDRESS SAVED IN GLOBAL
F158 C606620000	3621	MOV ACTIVE_PAGE,0	; SET PAGE VALUE
F15D B90020	3622	MOV CX,8192	; NUMBER OF WORDS IN COLOR CARD
F160 80FC04	3623	CHP AH,4	; TEST FOR GRAPHICS
F163 720B	3624	JC M12	; NO_GRAPHICS_INIT
F165 80FC07	3625	CHP AH,7	; TEST FOR BW CARD
F168 7404	3626	JE M11	; BW_CARD_INIT
F16A 33C0	3627	XOR AX,AX	; FILL FOR GRAPHICS MODE
F16C EB05	3628	JMP SHORT M13	; CLEAR_BUFFER
F16E	3629	M11:	; BW_CARD_INIT
F16E B508	3630	MOV CH,08H	; BUFFER SIZE ON BW CARD
F170	3631	M12:	; NO_GRAPHICS_INIT
F170 B82007	3632	MOV AX,' '+7*256	; FILL CHAR FOR ALPHA
F173	3633	M13:	; CLEAR_BUFFER
F173 F3	3634	REP STOSW	; FILL THE REGEN BUFFER WITH BLANKS
F174 AB			
	3635		
	3636	;----- ENABLE VIDEO AND CORRECT PORT SETTING	
	3637		
F175 C70660000706	3638	MOV CURSOR_MODE,607H	; SET CURRENT CURSOR MODE
F17B A04900	3639	MOV AL,CRT_MODE	; GET THE MODE
F17E 32E4	3640	XOR AH,AH	; INTO AX REGISTER
F180 8BF0	3641	MOV SI,AX	; TABLE POINTER, INDEXED BY MODE
F182 8B166300	3642	MOV DX,ADDR_6845	; PREPARE TO OUTPUT TO
	3643		; VIDEO ENABLE PORT
F186 03C204	3644	ADD DX,4	
F189 2E8A84F4F0	3645	MOV AL,CS:[SI+OFFSET M7]	
F18E EE	3646	OUT DX,AL	; SET VIDEO ENABLE PORT
F18F A26500	3647	MOV CRT_MODE_SET,AL	; SAVE THAT VALUE
	3648		

```

LOC OBJ          LINE  SOURCE
3649             |----- DETERMINE NUMBER OF COLUMNS, BOTH FOR ENTIRE DISPLAY
3650             |----- AND THE NUMBER TO BE USED FOR TTY INTERFACE
3651
F192 2E8A84ECF0 3652             MOV     AL,CS:[SI + OFFSET M6]
F197 32E4        3653             XOR     AH,AH
F199 A34A00      3654             MOV     CRT_COLS,AX             ; NUMBER OF COLUMNS IN THIS SCREEN
3655
3656             |----- SET CURSOR POSITIONS
3657
F19C 81E60E00    3658             AND     SI,0EH                 ; WORD OFFSET INTO CLEAR LENGTH TABLE
F1A0 2E8B8CE4F0 3659             MOV     CX,CS:[SI + OFFSET M5] ; LENGTH TO CLEAR
F1A5 890E4C00    3660             MOV     CRT_LEN,CX           ; SAVE LENGTH OF CRT -- NOT USED FOR BW
F1A9 B90800      3661             MOV     CX,8                 ; CLEAR ALL CURSOR POSITIONS
F1AC BF5000      3662             MOV     DI,OFFSET CURSOR_POSH
F1AF 1E          3663             PUSH   DS                    ; ESTABLISH SEGMENT
F1B0 07          3664             POP     ES                    ; ADDRESSING
F1B1 33C0        3665             XOR     AX,AX
F1B3 F3          3666             REP     STOSW                ; FILL WITH ZEROES
F1B4 AB
3667
3668             |----- SET UP OVERSCAN REGISTER
3669
F1B5 42          3670             INC     DX                    ; SET OVERSCAN PORT TO A DEFAULT
F1B6 B030        3671             MOV     AL,30H               ; VALUE OF 30H FOR ALL MODES
3672             ; EXCEPT 640X200
F1B8 803E490006 3673             CMP     CRT_MODE,6           ; SEE IF THE MODE IS 640X200 BW
F1BD 7502        3674             JNZ     M14                  ; IF IT ISNT 640X200, THEN GOTO REGULAR
F1BF B03F        3675             MOV     AL,3FH               ; IF IT IS 640X200, THEN PUT IN 3FH
F1C1
M14:            3676
F1C1 EE          3677             OUT     DX,AL                ; OUTPUT THE CORRECT VALUE TO 3D9 PORT
F1C2 A26600      3678             MOV     CRT_PALETTE,AL       ; SAVE THE VALUE FOR FUTURE USE
3679
3680             |----- NORMAL RETURN FROM ALL VIDEO RETURNS
3681
F1C5             VIDEO_RETURN:
F1C5 5F          3683             POP     DI
F1C6 5E          3684             POP     SI
F1C7 5B          3685             POP     BX
F1C8             M15: ; VIDEO_RETURN_C
F1C8 59          3687             POP     CX
F1C9 5A          3688             POP     DX
F1CA 1F          3689             POP     DS
F1CB 07          3690             POP     ES                    ; RECOVER SEGMENTS
F1CC CF          3691             IRET                    ; ALL DONE
3692
3693             SET_MODE ENDP
3694
3695             |-----
3696             | SET_CTYPE
3697             | THIS ROUTINE SETS THE CURSOR VALUE
3698             | INPUT
3699             | (CX) HAS CURSOR VALUE CH-START LINE, CL-STOP LINE
3700             | OUTPUT
3701             | NONE
3702             |-----
F1CD             SET_CTYPE PROC NEAR
F1CD B40A        3702             MOV     AH,10                ; 6845 REGISTER FOR CURSOR SET
F1CF 890E6000    3703             MOV     CURSOR_MODE,CX       ; SAVE IN DATA AREA
F1D3 E80200      3704             CALL   M16                   ; OUTPUT CX REG
F1D6 EBED        3705             JMP     VIDEO_RETURN
3706
3707             |----- THIS ROUTINE OUTPUTS THE CX REGISTER TO THE 6845 REGS NAMED IN AH
3708
F1D8             M16:
F1D8 8B166300    3710             MOV     DX,ADDR_6845         ; ADDRESS REGISTER
F1DC 8AC4        3711             MOV     AL,AH                ; GET VALUE
F1DE EE          3712             OUT     DX,AL                ; REGISTER SET
F1DF 42          3713             INC     DX                    ; DATA REGISTER
F1E0 8AC5        3714             MOV     AL,CH                ; DATA
F1E2 EE          3715             OUT     DX,AL
F1E3 4A          3716             DEC     DX
F1E4 8AC4        3717             MOV     AL,AH
F1E6 FEC0        3718             INC     AL                    ; POINT TO OTHER DATA REGISTER
F1E8 EE          3719             OUT     DX,AL                ; SET FOR SECOND REGISTER
F1E9 42          3720             INC     DX
F1EA 8AC1        3721             MOV     AL,CL                ; SECOND DATA VALUE
F1EC EE          3722             OUT     DX,AL
F1ED C3          3723             RET                            ; ALL DONE
3724             SET_CTYPE ENDP

```

LOC OBJ

LINE SOURCE

```

3725 |-----|
3726 | SET_CPOS                                :
3727 | THIS ROUTINE SETS THE CURRENT CURSOR   :
3728 | POSITION TO THE NEW X-Y VALUES PASSED  :
3729 | INPUT                                    :
3730 | DX - ROW,COLUMN OF NEW CURSOR         :
3731 | BH - DISPLAY PAGE OF CURSOR           :
3732 | OUTPUT                                    :
3733 | CURSOR IS SET AT 6845 IF DISPLAY PAGE  :
3734 | IS CURRENT DISPLAY                     :
3735 |-----|
F1EE 3736 SET_CPOS      PROC   NEAR
F1EE 8ACF 3737      MOV    CL,BH
F1F0 32ED 3738      XOR    CH,CH          ; ESTABLISH LOOP COUNT
F1F2 DIE1 3739      SAL    CX,1          ; WORD OFFSET
F1F4 8BF1 3740      MOV    SI,CX          ; USE INDEX REGISTER
F1F6 895450 3741      MOV    [SI+OFFSET CURSOR_POSN],DX ; SAVE THE POINTER
F1F9 383E6200 3742      CMP    ACTIVE_PAGE,BH
F1FD 7505 3743      JNZ    M17          ; SET_CPOS_RETURN
F1FF 8BC2 3744      MOV    AX,DX          ; GET ROW/COLUMN TO AX
F201 E00200 3745      CALL   M18          ; CURSOR_SET
F204 3746      M17:          ; SET_CPOS_RETURN
F204 EBBF 3747      JMP    VIDEO_RETURN
3748      SET_CPOS      ENDP
3749
3750 |----- SET CURSOR POSITION, AX HAS ROW/COLUMN FOR CURSOR
3751
F206 3752      M18      PROC   NEAR
F206 E87C00 3753      CALL   POSITION          ; DETERMINE LOCATION IN REGEN BUFFER
F209 8BC8 3754      MOV    CX,AX
F20B 030E4E00 3755      ADD    CX,CRT_START      ; ADD IN THE START ADDR FOR THIS PAGE
F20F D1F9 3756      SAR    CX,1          ; DIVIDE BY 2 FOR CHAR ONLY COUNT
F211 B40E 3757      MOV    AH,14          ; REGISTER NUMBER FOR CURSOR
F213 E8C2FF 3758      CALL   M16          ; OUTPUT THE VALUE TO THE 6845
F216 C3 3759      RET
3760      M18      ENDP
3761 |-----|
3762 | ACT_DISP_PAGE                          :
3763 | THIS ROUTINE SETS THE ACTIVE DISPLAY PAGE, ALLOWING THE :
3764 | FULL USE OF THE RAM SET ASIDE FOR THE VIDEO ATTACHMENT :
3765 | INPUT                                    :
3766 | AL HAS THE NEW ACTIVE DISPLAY PAGE     :
3767 | OUTPUT                                    :
3768 | THE 6845 IS RESET TO DISPLAY THAT PAGE :
3769 |-----|
F217 3770      ACT_DISP_PAGE  PROC   NEAR
F217 A26200 3771      MOV    ACTIVE_PAGE,AL    ; SAVE ACTIVE PAGE VALUE
F21A 880E4C00 3772      MOV    CX,CRT_LEN      ; GET SAVED LENGTH OF REGEN BUFFER
F21E 98 3773      CBW                    ; CONVERT AL TO WORD
F21F 50 3774      PUSH   AX              ; SAVE PAGE VALUE
F220 F7E1 3775      MUL    CX              ; DISPLAY PAGE TIMES REGEN LENGTH
F222 A34E00 3776      MOV    CRT_START,AX     ; SAVE START ADDRESS FOR
3777      ; LATER REQUIREMENTS
F225 8BC8 3778      MOV    CX,AX          ; START ADDRESS TO CX
F227 D1F9 3779      SAR    CX,1          ; DIVIDE BY 2 FOR 6845 HANDLING
F229 B40C 3780      MOV    AH,12          ; 6845 REGISTER FOR START ADDRESS
F22B E0AAFF 3781      CALL   M16
F22E 5B 3782      POP    BX              ; RECOVER PAGE VALUE
F22F DIE3 3783      SAL    BX,1          ; *2 FOR WORD OFFSET
F231 8B4750 3784      MOV    AX,[BX + OFFSET CURSOR_POSN] ; GET CURSOR FOR THIS PAGE
F234 E8CFFF 3785      CALL   M18          ; SET THE CURSOR POSITION
F237 EB8C 3786      JMP    SHORT VIDEO_RETURN
3787      ACT_DISP_PAGE  ENDP
3788 |-----|
3789 | READ_CURSOR                             :
3790 | THIS ROUTINE READS THE CURRENT CURSOR VALUE FROM THE :
3791 | 6845, FORMATS IT, AND SENDS IT BACK TO THE CALLER   :
3792 | INPUT                                    :
3793 | BH - PAGE OF CURSOR                         :
3794 | OUTPUT                                    :
3795 | DX - ROW, COLUMN OF THE CURRENT CURSOR POSITION :
3796 | CX - CURRENT CURSOR MODE                   :
3797 |-----|
F239 3798      READ_CURSOR  PROC   NEAR
F239 8ADF 3799      MOV    BL,BH
F23B 32FF 3800      XOR    BH,BH
F23D DIE3 3801      SAL    BX,1          ; WORD OFFSET

```

LOC OBJ	LINE	SOURCE
F23F 0B5750	3802	MOV DX,[BX+OFFSET CURSOR_POSN]
F242 0B0E6000	3803	MOV CX,CURSOR_MODE
F246 5F	3804	POP DI
F247 5E	3805	POP SI
F248 5B	3806	POP BX
F249 58	3807	POP AX ; DISCARD SAVED CX AND DX
F24A 58	3808	POP AX
F24B 1F	3809	POP DS
F24C 07	3810	POP ES
F24D CF	3811	IRET
	3812	READ_CURSOR ENDP
	3813	;
	3814	-----
	3814	; SET COLOR ;
	3815	; THIS ROUTINE WILL ESTABLISH THE BACKGROUND COLOR, THE OVERSCAN ;
	3816	; COLOR, AND THE FOREGROUND COLOR SET FOR MEDIUM RESOLUTION ;
	3817	; GRAPHICS ;
	3818	; INPUT ;
	3819	; (BH) HAS COLOR ID ;
	3820	; IF BH=0, THE BACKGROUND COLOR VALUE IS SET ;
	3821	; FROM THE LOW BITS OF BL (0-31) ;
	3822	; IF BH=1, THE PALETTE SELECTION IS MADE ;
	3823	; BASED ON THE LOW BIT OF BL: ;
	3824	; ;
	3825	; 0=GREEN, RED, YELLOW FOR COLORS 1,2,3 ;
	3826	; 1=BLUE, CYAN, MAGENTA FOR COLORS 1,2,3 ;
	3827	; (BL) HAS THE COLOR VALUE TO BE USED ;
	3828	; OUTPUT ;
	3828	; THE COLOR SELECTION IS UPDATED ;
	3829	-----
	3830	SET_COLOR PROC NEAR
F24E	3831	MOV DX,ADDR_6845 ; I/O PORT FOR PALETTE
F24E 0B166300	3832	ADD DX,5 ; OVERSCAN PORT
F252 83C205	3833	MOV AL,CRT_PALETTE ; GET THE CURRENT PALETTE VALUE
F255 A06600	3834	OR BH,BH ; IS THIS COLOR 0?
F258 0AFF	3835	JNZ H20 ; OUTPUT COLOR 1
F25A 750E	3836	
	3837	;---- HANDLE COLOR 0 BY SETTING THE BACKGROUND COLOR
	3838	
F25C 24E0	3839	AND AL,0E0H ; TURN OFF LOW 5 BITS OF CURRENT
F25E 80E31F	3840	AND BL,01FH ; TURN OFF HIGH 3 BITS OF INPUT VALUE
F261 0AC3	3841	OR AL,BL ; PUT VALUE INTO REGISTER
F263	3842	M19: ; OUTPUT THE PALETTE
F263 EE	3843	OUT DX,AL ; OUTPUT COLOR SELECTION TO 3D9 PORT
F264 A26600	3844	MOV CRT_PALETTE,AL ; SAVE THE COLOR VALUE
F267 E95BFF	3845	JMP VIDEO_RETURN
	3846	
	3847	;---- HANDLE COLOR 1 BY SELECTING THE PALETTE TO BE USED
	3848	
F26A	3849	M20:
F26A 24DF	3850	AND AL,0DFH ; TURN OFF PALETTE SELECT BIT
F26C D0EB	3851	SHR BL,1 ; TEST THE LOW ORDER BIT OF BL
F26E 73F3	3852	JNC M19 ; ALREADY DONE
F270 0C20	3853	OR AL,20H ; TURN ON PALETTE SELECT BIT
F272 EBFF	3854	JMP M19 ; GO DO IT
	3855	SET_COLOR ENDP
	3856	;
	3857	-----
	3857	; VIDEO STATE ;
	3858	; RETURNS THE CURRENT VIDEO STATE IN AX ;
	3859	; AH = NUMBER OF COLUMNS ON THE SCREEN ;
	3860	; AL = CURRENT VIDEO MODE ;
	3861	; BH = CURRENT ACTIVE PAGE ;
	3862	-----
	3863	VIDEO_STATE PROC NEAR
F274	3864	MOV AH,BYTE PTR CRT_COLS ; GET NUMBER OF COLUMNS
F274 8A264A00	3865	MOV AL,CRT_MODE ; CURRENT MODE
F278 A04900	3866	MOV BH,ACTIVE_PAGE ; GET CURRENT ACTIVE PAGE
F27B 8A3E6200	3867	POP DI ; RECOVER REGISTERS
F27F 5F	3868	POP SI
F280 5E	3869	POP CX ; DISCARD SAVED BX
F281 59	3870	JMP M15 ; RETURN TO CALLER
F282 E943FF	3870	
	3871	VIDEO_STATE ENDP
	3872	;
	3873	-----
	3873	; POSITION ;
	3874	; THIS SERVICE ROUTINE CALCULATES THE REGEN ;
	3875	; BUFFER ADDRESS OF A CHARACTER IN THE ALPHA MODE ;
	3876	; INPUT ;
	3877	; AX = ROW, COLUMN POSITION ;
	3878	; OUTPUT ;

LOC OBJ

LINE SOURCE

```

3879 ; AX = OFFSET OF CHAR POSITION IN REGEN BUFFER ;
3880 ;-----
F285 3881 POSITION PROC NEAR
F285 53 3882 PUSH BX ; SAVE REGISTER
F286 8BD8 3883 MOV BX,AX
F288 8AC4 3884 MOV AL,AH ; ROWS TO AL
F28A F6264A00 3885 MUL BYTE PTR CRT_COLS ; DETERMINE BYTES TO ROW
F28E 32FF 3886 XOR BH,BH
F290 03C3 3887 ADD AX,BX ; ADD IN COLUMN VALUE
F292 D1E0 3888 SAL AX,1 ; * 2 FOR ATTRIBUTE BYTES
F294 5B 3889 POP BX
F295 C3 3890 RET
3891 POSITION ENDP
3892 ;-----
3893 ; SCROLL UP ;
3894 ; THIS ROUTINE MOVES A BLOCK OF CHARACTERS UP ;
3895 ; ON THE SCREEN ;
3896 ; INPUT ;
3897 ; (AH) = CURRENT CRT MODE ;
3898 ; (AL) = NUMBER OF ROWS TO SCROLL ;
3899 ; (CX) = ROW/COLUMN OF UPPER LEFT CORNER ;
3900 ; (DX) = ROW/COLUMN OF LOWER RIGHT CORNER ;
3901 ; (BH) = ATTRIBUTE TO BE USED ON BLANKED LINE ;
3902 ; (DS) = DATA SEGMENT ;
3903 ; (ES) = REGEN BUFFER SEGMENT ;
3904 ; OUTPUT ;
3905 ; NONE -- THE REGEN BUFFER IS MODIFIED ;
3906 ;-----
3907 ASSUME CS:CODE,DS:DATA,ES:DATA
3908 SCROLL_UP PROC NEAR
F296 3909 MOV BL,AL ; SAVE LINE COUNT IN BL
F296 8AD8 3910 CMP AH,4 ; TEST FOR GRAPHICS MODE
F298 80FC04 3911 JC N1 ; HANDLE SEPARATELY
F298 7208 3912 CMP AH,7 ; TEST FOR BW CARD
F29D 80FC07 3913 JE N1
F2A0 7403 3914 JMP GRAPHICS_UP
F2A2 E9F001 3915 N1: ; UP_CONTINUE
F2A5 3916 PUSH BX ; SAVE FILL ATTRIBUTE IN BH
F2A5 53 3917 MOV AX,CX ; UPPER LEFT POSITION
F2A6 8BC1 3918 CALL SCROLL_POSITION ; DO SETUP FOR SCROLL
F2A8 E83700 3919 JZ N7 ; BLANK_FIELD
F2AB 7431 3920 ADD SI,AX ; FROM ADDRESS
F2AD 03F0 3921 MOV AH,DH ; # ROWS IN BLOCK
F2AF 8AE6 3922 SUB AH,BL ; # ROWS TO BE MOVED
F2B1 2AE3 3923 N2: ; ROW_LOOP
F2B3 3924 CALL N10 ; MOVE ONE ROW
F2B3 E87200 3925 ADD SI,BP
F2B6 03F5 3926 ADD DI,BP ; POINT TO NEXT LINE IN BLOCK
F2B8 03FD 3927 DEC AH ; COUNT OF LINES TO MOVE
F2BA FECC 3928 JNZ N2 ; ROW_LOOP
F2BC 75F5 3929 N3: ; CLEAR_ENTRY
F2BE 3930 POP AX ; RECOVER ATTRIBUTE IN AH
F2BE 58 3931 MOV AL,' ' ; FILL WITH BLANKS
F2BF B020 3932 N4: ; CLEAR_LOOP
F2C1 3933 CALL N11 ; CLEAR THE ROW
F2C1 E86D00 3934 ADD DI,BP ; POINT TO NEXT LINE
F2C4 03FD 3935 DEC BL ; COUNTER OF LINES TO SCROLL
F2C6 FECB 3936 JNZ N4 ; CLEAR_LOOP
F2C8 75F7 3937 N5: ; SCROLL_END
F2CA 3938 CALL DDS
F2CA E8710C 3939 CMP CRT_MODE,7 ; IS THIS THE BLACK AND WHITE CARD
F2CD 803E490007 3940 JE N6 ; IF SO, SKIP THE MODE RESET
F2D2 7407 3941 MOV AL,CRT_MODE_SET ; GET THE VALUE OF THE MODE SET
F2D4 A06500 3942 MOV DX,03D8H ; ALWAYS SET COLOR CARD PORT
F2D7 BAD803 3943 OUT DX,AL
F2DA EE 3944 N6: ; VIDEO_RET_HERE
F2DB 3945 JMP VIDEO_RETURN
F2DB E9E7FE 3946 N7: ; BLANK_FIELD
F2DE 3947 MOV BL,DH ; GET ROW COUNT
F2DE 8ADE 3948 JMP N3 ; GO CLEAR THAT AREA
F2E0 E8DC 3949 SCROLL_UP ENDP
3950
3951 ;---- HANDLE COMMON SCROLL SET UP HERE
3952
3953 SCROLL_POSITION PROC NEAR
F2E2 3954 CMP CRT_MODE,2 ; TEST FOR SPECIAL CASE HERE
F2E2 803E490002 3955 JB N9 ; HAVE TO HANDLE 80X25 SEPARATELY
F2E7 7218

```

```

LOC OBJ          LINE  SOURCE

F2E9 803E490003 3956      CHP   CRT_MODE,3
F2EE 7711        3957      JA    N9
3958
3959      1----- 80X25 COLOR CARD SCROLL
3960
F2F0 52         3961      PUSH  DX
F2F1 BADA03     3962      MOV  DX,3DAH          ; GUARANTEED TO BE COLOR CARD HERE
F2F4 50         3963      PUSH  AX
F2F5          3964      N8:   ; WAIT_DISP_ENABLE
F2F5 EC         3965      IN   AL,DX           ; GET PORT
F2F6 A808     3966      TEST AL,8            ; WAIT FOR VERTICAL RETRACE
F2F8 74FB     3967      JZ   N8              ; WAIT_DISP_ENABLE
F2FA B025     3968      MOV  AL,25H
F2FC B2D8     3969      MOV  DL,0DBH         ; DX=308
F2FE EE       3970      OUT  DX,AL           ; TURN OFF VIDEO
F2FF 58       3971      POP  AX              ; DURING VERTICAL RETRACE
F300 5A       3972      POP  DX
F301          3973      N9:
F301 E881FF   3974      CALL POSITION          ; CONVERT TO REGEN POINTER
F304 03064E00 3975      ADD  AX,CRT_START    ; OFFSET OF ACTIVE PAGE
F308 8BF8     3976      MOV  DI,AX           ; TO ADDRESS FOR SCROLL
F30A 8BF0     3977      MOV  SI,AX           ; FROM ADDRESS FOR SCROLL
F30C 2BD1     3978      SUB  DX,CX           ; DX = #ROWS, #COLS IN BLOCK
F30E FEC6     3979      INC  DH
F310 FEC2     3980      INC  DL              ; INCREMENT FOR 0 ORIGIN
F312 32ED     3981      XOR  CH,CH           ; SET HIGH BYTE OF COUNT TO ZERO
F314 8B2E4A00 3982      MOV  BP,CRT_COLS    ; GET NUMBER OF COLUMNS IN DISPLAY
F318 03ED     3983      ADD  BP,BP           ; TIMES 2 FOR ATTRIBUTE BYTE
F31A 8AC3     3984      MOV  AL,BL           ; GET LINE COUNT
F31C F6264A00 3985      MUL  BYTE PTR CRT_COLS ; DETERMINE OFFSET TO FROM ADDRESS
F320 03C0     3986      ADD  AX,AX           ; *2 FOR ATTRIBUTE BYTE
F322 06       3987      PUSH ES              ; ESTABLISH ADDRESSING TO REGEN BUFFER
F323 1F       3988      POP  DS              ; FOR BOTH POINTERS
F324 80FB00   3989      CMP  BL,0            ; 0 SCROLL MEANS BLANK FIELD
F327 C3       3990      RET                  ; RETURN WITH FLAGS SET
3991      SCROLL_POSITION ENDP
3992
3993      1----- MOVE_ROW
3994
F328          3995      N10  PROC  NEAR
F328 8ACA     3996      MOV  CL,DL           ; GET # OF COLS TO MOVE
F32A 56       3997      PUSH SI
F32B 57       3998      PUSH DI              ; SAVE START ADDRESS
F32C F3       3999      REP  MOVSW           ; MOVE THAT LINE ON SCREEN
F32D A5
F32E 5F       4000      POP  DI
F32F 5E       4001      POP  SI              ; RECOVER ADDRESSES
F330 C3       4002      RET
4003      N10  ENDP
4004
4005      1----- CLEAR_ROW
4006
F331          4007      N11  PROC  NEAR
F331 8ACA     4008      MOV  CL,DL           ; GET # COLUMNS TO CLEAR
F333 57       4009      PUSH DI
F334 F3       4010      REP  STOSW           ; STORE THE FILL CHARACTER
F335 AB
F336 5F       4011      POP  DI
F337 C3       4012      RET
4013      N11  ENDP
4014
4015      ;-----
4015      ; SCROLL_DOWN :
4016      ; THIS ROUTINE MOVES THE CHARACTERS WITHIN A :
4017      ; DEFINED BLOCK DOWN ON THE SCREEN, FILLING THE :
4018      ; TOP LINES WITH A DEFINED CHARACTER :
4019      ; INPUT :
4020      ; (AH) = CURRENT CRT MODE :
4021      ; (AL) = NUMBER OF LINES TO SCROLL :
4022      ; (CX) = UPPER LEFT CORNER OF REGION :
4023      ; (DX) = LOWER RIGHT CORNER OF REGION :
4024      ; (BH) = FILL CHARACTER :
4025      ; (DS) = DATA SEGMENT :
4026      ; (ES) = REGEN SEGMENT :
4027      ; OUTPUT :
4028      ; NONE -- SCREEN IS SCROLLED :
4029      ;-----
F338          4030      SCROLL_DOWN  PROC  NEAR

```

LOC OBJ	LINE	SOURCE	
F338 FD	4031	STD	; DIRECTION FOR SCROLL DOWN
F339 8A08	4032	MOV BL,AL	; LINE COUNT TO BL
F33B 80FC04	4033	CMF AH,4	; TEST FOR GRAPHICS
F33E 7208	4034	JC N12	
F340 80FC07	4035	CMF AH,7	; TEST FOR BW CARD
F343 7403	4036	JE N12	
F345 E9A601	4037	JMP GRAPHICS_DOWN	
F348	4038	N12:	; CONTINUE DOWN
F348 53	4039	PUSH BX	; SAVE ATTRIBUTE IN BX
F349 8BC2	4040	MOV AX,DX	; LOWER RIGHT CORNER
F34B E89AFF	4041	CALL SCROLL_POSITION	; GET REGEN LOCATION
F34E 7420	4042	JZ N16	
F350 2BF0	4043	SUB SI,AX	; SI IS FROM ADDRESS
F352 8AE6	4044	MOV AH,DX	; GET TOTAL # ROWS
F354 2AE3	4045	SUB AH,BL	; COUNT TO MOVE IN SCROLL
F356	4046	N13:	
F356 E8CFFF	4047	CALL N10	; MOVE ONE ROW
F359 2BF5	4048	SUB SI,BP	
F35B 2BFD	4049	SUB DI,BP	
F35D FECC	4050	DEC AH	
F35F 75F5	4051	JNZ N13	
F361	4052	N14:	
F361 58	4053	POP AX	; RECOVER ATTRIBUTE IN AH
F362 B020	4054	MOV AL,' '	
F364	4055	N15:	
F364 E8CAFF	4056	CALL N11	; CLEAR ONE ROW
F367 2BFD	4057	SUB DI,BP	; GO TO NEXT ROW
F369 FECC	4058	DEC BL	
F36B 75F7	4059	JNZ N15	
F36D E95AFF	4060	JMP N5	; SCROLL_END
F370	4061	N16:	
F370 8ADE	4062	MOV BL,DX	
F372 EBED	4063	JMP N14	
	4064	SCROLL_DOWN	ENDP
	4065	;	-----
	4066	; READ_AC_CURRENT	:
	4067	; THIS ROUTINE READS THE ATTRIBUTE AND CHARACTER	:
	4068	; AT THE CURRENT CURSOR POSITION AND RETURNS THEM	:
	4069	; TO THE CALLER	:
	4070	;INPUT	:
	4071	; (AH) = CURRENT CRT MODE	:
	4072	; (BH) = DISPLAY PAGE (ALPHA MODES ONLY)	:
	4073	; (DS) = DATA SEGMENT	:
	4074	; (ES) = REGEN SEGMENT	:
	4075	;OUTPUT	:
	4076	; (AL) = CHAR READ	:
	4077	; (AH) = ATTRIBUTE READ	:
	4078	;	-----
	4079	ASSUME CS:CODE,DS:DATA,ES:DATA	
F374	4080	READ_AC_CURRENT PROC NEAR	
F374 80FC04	4081	CMF AH,4	; IS THIS GRAPHICS
F377 7208	4082	JC P1	
F379 80FC07	4083	CMF AH,7	; IS THIS BW CARD
F37C 7403	4084	JE P1	
F37E E9A802	4085	JMP GRAPHICS_READ	
F381	4086	P1:	; READ_AC_CONTINUE
F381 E81A00	4087	CALL FIND_POSITION	
F384 8BF3	4088	MOV SI,BX	; ESTABLISH ADDRESSING IN SI
	4089		
	4090	;	-----
	4091	;	WAIT FOR HORIZONTAL RETRACE
F386 8B166300	4092	MOV DX,ADDR_6845	; GET BASE ADDRESS
F38A 83C206	4093	ADD DX,6	; POINT AT STATUS PORT
F38D 06	4094	PUSH ES	
F38E 1F	4095	POP DS	; GET SEGMENT FOR QUICK ACCESS
F38F	4096	P2:	; WAIT FOR RETRACE LOW
F38F EC	4097	IN AL,DX	; GET STATUS
F390 A801	4098	TEST AL,1	; IS HORZ RETRACE LOW
F392 75FB	4099	JNZ P2	; WAIT UNTIL IT IS
F394 FA	4100	CLI	; NO MORE INTERRUPTS
F395	4101	P3:	; WAIT FOR RETRACE HIGH
F395 EC	4102	IN AL,DX	; GET STATUS
F396 A801	4103	TEST AL,1	; IS IT HIGH
F398 74FB	4104	JZ P3	; WAIT UNTIL IT IS
F39A AD	4105	LODSW	; GET THE CHAR/ATTR
F39B E927FE	4106	JMP VIDEO_RETURN	
	4107	READ_AC_CURRENT ENDP	

LOC OBJ	LINE	SOURCE
	4108	
F39E	4109	FIND_POSITION PROC NEAR
F39E 8ACF	4110	MOV CL,BH ; DISPLAY PAGE TO CX
F3A0 32ED	4111	XOR CH,CH
F3A2 8BF1	4112	MOV SI,CX ; MOVE TO SI FOR INDEX
F3A4 D1E6	4113	SAL SI,1 ; # 2 FOR WORD OFFSET
F3A6 084450	4114	MOV AX,[SI+ OFFSET CURSOR_POSN] ; GET ROW/COLUMN OF THAT PAGE
F3A9 330B	4115	XOR BX,BX ; SET START ADDRESS TO ZERO
F3AB E306	4116	JCXZ P5 ; NO_PAGE
F3AD	4117	P6: ; PAGE_LOOP
F3AD 031E4C00	4118	ADD BX,CRT_LEN ; LENGTH OF BUFFER
F3B1 E2FA	4119	LOOP P6
F3B3	4120	P5: ; NO_PAGE
F3B3 E0CFFE	4121	CALL POSITION ; DETERMINE LOCATION IN REGEN
F3B6 0308	4122	ADD BX,AX ; ADD TO START OF REGEN
F3B8 C3	4123	RET
	4124	FIND_POSITION ENDP
	4125	};-----
	4126	; WRITE_AC_CURRENT ;
	4127	; THIS ROUTINE WRITES THE ATTRIBUTE ;
	4128	; AND CHARACTER AT THE CURRENT CURSOR ;
	4129	; POSITION ;
	4130	; INPUT ;
	4131	; (AH) = CURRENT CRT MODE ;
	4132	; (BH) = DISPLAY PAGE ;
	4133	; (CX) = COUNT OF CHARACTERS TO WRITE ;
	4134	; (AL) = CHAR TO WRITE ;
	4135	; (BL) = ATTRIBUTE OF CHAR TO WRITE ;
	4136	; (DS) = DATA SEGMENT ;
	4137	; (ES) = REGEN SEGMENT ;
	4138	; OUTPUT ;
	4139	; NONE ;
	4140	};-----
F3B9	4141	WRITE_AC_CURRENT PROC NEAR
F3B9 80FC04	4142	CHP AH,4 ; IS THIS GRAPHICS
F3BC 7208	4143	JC P6
F3BE 80FC07	4144	CHP AH,7 ; IS THIS BW CARD
F3C1 7403	4145	JE P6
F3C3 E9B201	4146	JMP GRAPHICS_WRITE
F3C6	4147	P6: ; WRITE_AC_CONTINUE
F3C6 8AE3	4148	MOV AH,BL ; GET ATTRIBUTE TO AH
F3C8 50	4149	PUSH AX ; SAVE ON STACK
F3C9 51	4150	PUSH CX ; SAVE WRITE COUNT
F3CA E8D1FF	4151	CALL FIND_POSITION
F3CD 88FB	4152	MOV DI,BX ; ADDRESS TO DI REGISTER
F3CF 59	4153	POP CX ; WRITE COUNT
F3D0 5B	4154	POP BX ; CHARACTER IN BX REG
F3D1	4155	P7: ; WRITE_LOOP
	4156	
	4157	};---- WAIT FOR HORIZONTAL RETRACE
	4158	
F3D1 8B166300	4159	MOV DX,ADDR_6845 ; GET BASE ADDRESS
F3D5 83C206	4160	ADD DX,6 ; POINT AT STATUS PORT
F3D8	4161	P8: ;
F3D8 EC	4162	IN AL,DX ; GET STATUS
F3D9 A801	4163	TEST AL,1 ; IS IT LOW
F3DB 75FB	4164	JNZ P8 ; WAIT UNTIL IT IS
F3DD FA	4165	CLI ; NO MORE INTERRUPTS
F3DE	4166	P9: ;
F3DE EC	4167	IN AL,DX ; GET STATUS
F3DF A801	4168	TEST AL,1 ; IS IT HIGH
F3E1 74FB	4169	JZ P9 ; WAIT UNTIL IT IS
F3E3 8BC3	4170	MOV AX,BX ; RECOVER THE CHAR/ATTR
F3E5 AB	4171	STOSH ; PUT THE CHAR/ATTR
F3E6 FB	4172	STI ; INTERRUPTS BACK ON
F3E7 E2E8	4173	LOOP P7 ; AS MANY TIMES AS REQUESTED
F3E9 E9D9FD	4174	JMP VIDEO_RETURN
	4175	WRITE_AC_CURRENT ENDP
	4176	};-----
	4177	; WRITE_C_CURRENT ;
	4178	; THIS ROUTINE WRITES THE CHARACTER AT ;
	4179	; THE CURRENT CURSOR POSITION, ATTRIBUTE ;
	4180	; UNCHANGED ;
	4181	; INPUT ;
	4182	; (AH) = CURRENT CRT MODE ;
	4183	; (BH) = DISPLAY PAGE ;
	4184	; (CX) = COUNT OF CHARACTERS TO WRITE ;

LOC OBJ

LINE SOURCE

```

4185 ; (AL) = CHAR TO WRITE ;
4186 ; (DS) = DATA SEGMENT ;
4187 ; (ES) = REGEN SEGMENT ;
4188 ; OUTPUT ;
4189 ; NONE ;
4190 ;-----
F3EC 4191 WRITE_C_CURRENT PROC NEAR
F3EC 80FC04 4192 CMP AH,4 ; IS THIS GRAPHICS
F3EF 7208 4193 JC P10
F3F1 80FC07 4194 CMP AH,7 ; IS THIS BM CARD
F3F4 7403 4195 JE P10
F3F6 E97F01 4196 JHP GRAPHICS_WRITE
F3F9 4197 P10:
F3F9 50 4198 PUSH AX ; SAVE ON STACK
F3FA 51 4199 PUSH CX ; SAVE WRITE COUNT
F3FB E8A0FF 4200 CALL FIND_POSITION
F3FE 8BFB 4201 MOV DI,BX ; ADDRESS TO DI
F400 59 4202 POP CX ; WRITE COUNT
F401 5B 4203 POP BX ; BL HAS CHAR TO WRITE
F402 4204 P11: ; WRITE_LOOP
4205
4206 ;----- WAIT FOR HORIZONTAL RETRACE
4207
F402 8B166300 4208 MOV DX,ADDR_6845 ; GET BASE ADDRESS
F406 83C206 4209 ADD DX,6 ; POINT AT STATUS PORT
F409 4210 P12:
F409 EC 4211 IN AL,DX ; GET STATUS
F40A A801 4212 TEST AL,1 ; IS IT LOW
F40C 75FB 4213 JNZ P12 ; WAIT UNTIL IT IS
F40E FA 4214 CLI ; NO MORE INTERRUPTS
F40F 4215 P13:
F40F EC 4216 IN AL,DX ; GET STATUS
F410 A801 4217 TEST AL,1 ; IS IT HIGH
F412 74FB 4218 JZ P13 ; WAIT UNTIL IT IS
F414 8AC3 4219 MOV AL,BL ; RECOVER CHAR
F416 AA 4220 STOSB ; PUT THE CHAR/ATTR
F417 FB 4221 STI ; INTERRUPTS BACK ON
F418 47 4222 INC DI ; BUMP POINTER PAST ATTRIBUTE
F419 E2E7 4223 LOOP P11 ; AS MANY TIMES AS REQUESTED
F41B E9A7FD 4224 JMP VIDEO_RETURN
4225 WRITE_C_CURRENT ENDP
4226 ;-----
4227 ; READ DOT -- WRITE DOT ;
4228 ; THESE ROUTINES WILL WRITE A DOT, OR READ THE DOT AT ;
4229 ; THE INDICATED LOCATION ;
4230 ; ENTRY -- ;
4231 ; DX = ROW ( 0-199) (THE ACTUAL VALUE DEPENDS ON THE MODE) ;
4232 ; CX = COLUMN ( 0-639) ( THE VALUES ARE NOT RANGE CHECKED ) ;
4233 ; AL = DOT VALUE TO WRITE (1,2 OR 4 BITS DEPENDING ON MODE, ;
4234 ; REQ'D FOR WRITE DOT ONLY, RIGHT JUSTIFIED) ;
4235 ; BIT 7 OF AL=1 INDICATES XOR THE VALUE INTO THE LOCATION ;
4236 ; DS = DATA SEGMENT ;
4237 ; ES = REGEN SEGMENT ;
4238 ;
4239 ; EXIT ;
4240 ; AL = DOT VALUE READ, RIGHT JUSTIFIED, READ ONLY ;
4241 ;-----
4242 ASSUME CS:CODE,DS:DATA,ES:DATA
F41E 4243 READ_DOT PROC NEAR
F41E E83100 4244 CALL R3 ; DETERMINE BYTE POSITION OF DOT
F421 268A04 4245 MOV AL,ES:[SI] ; GET THE BYTE
F424 22C4 4246 AND AL,AH ; MASK OFF THE OTHER BITS IN THE BYTE
F426 D2E0 4247 SHL AL,CL ; LEFT JUSTIFY THE VALUE
F428 BACE 4248 MOV CL,DH ; GET NUMBER OF BITS IN RESULT
F42A D2C0 4249 ROL AL,CL ; RIGHT JUSTIFY THE RESULT
F42C E996FD 4250 JHP VIDEO_RETURN ; RETURN FROM VIDEO IO
4251 READ_DOT ENDP
4252
F42F 4253 WRITE_DOT PROC NEAR
F42F 50 4254 PUSH AX ; SAVE DOT VALUE
F430 50 4255 PUSH AX ; TWICE
F431 E81E00 4256 CALL R3 ; DETERMINE BYTE POSITION OF THE DOT
F434 D2E8 4257 SHR AL,CL ; SHIFT TO SET UP THE BITS FOR OUTPUT
F436 22C4 4258 AND AL,AH ; STRIP OFF THE OTHER BITS
F438 268A0C 4259 MOV CL,ES:[SI] ; GET THE CURRENT BYTE
F43B 5B 4260 POP BX ; RECOVER XOR FLAG
F43C F6C380 4261 TEST BL,80H ; IS IT ON

```

LOC OBJ	LINE	SOURCE
F43F 750D	4262	JNZ R2 ; YES, XOR THE DOT
F441 F6D4	4263	NOT AH ; SET THE MASK TO REMOVE THE
F443 22CC	4264	AND CL,AH ; INDICATED BITS
F445 0AC1	4265	OR AL,CL ; OR IN THE NEW VALUE OF THOSE BITS
F447	4266	R1: ; FINISH_DOT
F447 268804	4267	MOV ES:[SI],AL ; RESTORE THE BYTE IN MEMORY
F44A 58	4268	POP AX
F44B E977FD	4269	JMP VIDEO_RETURN ; RETURN FROM VIDEO IO
F44E	4270	R2: ; XOR_DOT
F44E 32C1	4271	XOR AL,CL ; EXCLUSIVE OR THE DOT
F450 EBF5	4272	JMP R1 ; FINISH UP THE WRITING
	4273	WRITE_DOT ENDP
	4274	;
	4275	; THIS SUBROUTINE DETERMINES THE REGEN BYTE LOCATION :
	4276	; OF THE INDICATED ROW COLUMN VALUE IN GRAPHICS MODE. :
	4277	; ENTRY -- :
	4278	; DX = ROW VALUE (0-199) :
	4279	; CX = COLUMN VALUE (0-639) :
	4280	; EXIT -- :
	4281	; SI = OFFSET INTO REGEN BUFFER FOR BYTE OF INTEREST :
	4282	; AH = MASK TO STRIP OFF THE BITS OF INTEREST :
	4283	; CL = BITS TO SHIFT TO RIGHT JUSTIFY THE MASK IN AH :
	4284	; DH = # BITS IN RESULT :
	4285	;
F452	4286	R3 PROC NEAR
F452 53	4287	PUSH BX ; SAVE BX DURING OPERATION
F453 50	4288	PUSH AX ; WILL SAVE AL DURING OPERATION
	4289	
	4290	;----- DETERMINE 1ST BYTE IN IDICATED ROW BY MULTIPLYING ROW VALUE BY 40
	4291	;----- (LOW BIT OF ROW DETERMINES EVEN/ODD, 80 BYTES/ROW
	4292	
F454 B028	4293	MOV AL,40
F456 52	4294	PUSH DX ; SAVE ROW VALUE
F457 80E2FE	4295	AND DL,0FEH ; STRIP OFF ODD/EVEN BIT
F45A F6E2	4296	MUL DL ; AX HAS ADDRESS OF 1ST BYTE
	4297	; OF INDICATED ROW
F45C 5A	4298	POP DX ; RECOVER IT
F45D F6C201	4299	TEST DL,1 ; TEST FOR EVEN/ODD
F460 7403	4300	JZ R4 ; JUMP IF EVEN ROW
F462 050020	4301	ADD AX,2000H ; OFFSET TO LOCATION OF ODD ROWS
F465	4302	R4: ; EVEN_ROW
F465 8BF0	4303	MOV SI,AX ; MOVE POINTER TO SI
F467 58	4304	POP AX ; RECOVER AL VALUE
F468 8BD1	4305	MOV DX,CX ; COLUMN VALUE TO DX
	4306	
	4307	;----- DETERMINE GRAPHICS MODE CURRENTLY IN EFFECT
	4308	
	4309	;
	4310	; SET UP THE REGISTERS ACCORDING TO THE MODE :
	4311	; CH = MASK FOR LOW OF COLUMN ADDRESS (7/3 FOR HIGH/MED RES) :
	4312	; CL = # OF ADDRESS BITS IN COLUMN VALUE (3/2 FOR H/M) :
	4313	; BL = MASK TO SELECT BITS FROM POINTED BYTE (80H/C0H FOR H/M) :
	4314	; BH = NUMBER OF VALID BITS IN POINTED BYTE (1/2 FOR H/M) :
	4315	;
	4316	
F46A BBC002	4317	MOV BX,2C0H
F46D B90203	4318	MOV CX,302H ; SET PARMS FOR MED RES
F470 803E490006	4319	CHP CRT_MODE,6
F475 7206	4320	JC R5 ; HANDLE IF MED ARES
F477 BB8001	4321	MOV BX,180H
F47A B90307	4322	MOV CX,703H ; SET PARMS FOR HIGH RES
	4323	
	4324	;----- DETERMINE BIT OFFSET IN BYTE FROM COLUMN MASK
	4325	
F47D	4326	R5: ;
F47D 22EA	4327	AND CH,DL ; ADDRESS OF PEL WITHIN BYTE TO CH
	4328	
	4329	;----- DETERMINE BYTE OFFSET FOR THIS LOCATION IN COLUMN
	4330	
F47F D3EA	4331	SHR DX,CL ; SHIFT BY CORRECT AMOUNT
F481 03F2	4332	ADD SI,DX ; INCREMENT THE POINTER
F483 8AF7	4333	MOV DH,BH ; GET THE # OF BITS IN RESULT TO DH
	4334	
	4335	;----- MULTIPLY BH (VALID BITS IN BYTE) BY CH (BIT OFFSET)
	4336	
F485 2AC9	4337	SUB CL,CL ; ZERO INTO STORAGE LOCATION
F487	4338	R6: ;

LOC OBJ	LINE	SOURCE
F487 D0C8	4339	ROR AL,1 ; LEFT JUSTIFY THE VALUE
	4340	; IM AL (FOR WRITE)
F489 02CD	4341	ADD CL,CH ; ADD IN THE BIT OFFSET VALUE
F48B FECF	4342	DEC BH ; LOOP CONTROL
F48D 75F8	4343	JNZ R6 ; ON EXIT, CL HAS SHIFT COUNT
	4344	; TO RESTORE BITS
F48F 8AE3	4345	MOV AH,BL ; GET MASK TO AH
F491 D2EC	4346	SHR AH,CL ; MOVE THE MASK TO CORRECT LOCATION
F493 5B	4347	POP BX ; RECOVER REG
F494 C3	4348	RET ; RETURN WITH EVERYTHING SET UP
	4349	R3
	4350	ENDP
	4351	-----
	4351	; SCROLL UP ;
	4352	; THIS ROUTINE SCROLLS UP THE INFORMATION ON THE CRT ;
	4353	; ENTRY ;
	4354	; CH,CL = UPPER LEFT CORNER OF REGION TO SCROLL ;
	4355	; DH,DL = LOWER RIGHT CORNER OF REGION TO SCROLL ;
	4356	; BOTH OF THE ABOVE ARE IN CHARACTER POSITIONS ;
	4357	; BH = FILL VALUE FOR BLANKED LINES ;
	4358	; AL = # LINES TO SCROLL (AL=0 MEANS BLANK THE ENTIRE ;
	4359	; FIELD) ;
	4360	; DS = DATA SEGMENT ;
	4361	; ES = REGEN SEGMENT ;
	4362	; EXIT ;
	4363	; NOTHING, THE SCREEN IS SCROLLED ;
	4364	-----
F495	4365	GRAPHICS_UP PROC NEAR
F495 8AD8	4366	MOV BL,AL ; SAVE LINE COUNT IN BL
F497 8BC1	4367	MOV AX,CX ; GET UPPER LEFT POSITION INTO AX REG
	4368	
	4369	;----- USE CHARACTER SUBROUTINE FOR POSITIONING
	4370	;----- ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE
	4371	
F499 E86902	4372	CALL GRAPH_POSN
F49C 8BF8	4373	MOV DI,AX ; SAVE RESULT AS DESTINATION ADDRESS
	4374	
	4375	;----- DETERMINE SIZE OF WINDOW
	4376	
F49E 2BD1	4377	SUB DX,CX
F4A0 81C20101	4378	ADD DX,101H ; ADJUST VALUES
F4A4 D0E6	4379	SAL DH,1 ; MULTIPLY # ROWS BY 4
	4380	; SINCE 8 VERT DOTS/CHAR
F4A6 D0E6	4381	SAL DH,1 ; AND EVEN/ODD ROWS
	4382	
	4383	;----- DETERMINE CRT MODE
	4384	
F4A8 803E490006	4385	CHP CRT_MODE,6 ; TEST FOR MEDIUM RES
F4AD 7304	4386	JNC R7 ; FIND_SOURCE
	4387	
	4388	;----- MEDIUM RES UP
	4389	
F4AF D0E2	4390	SAL DL,1 ; # COLUMNS = 2, SINCE 2 BYTES/CHAR
F4B1 D1E7	4391	SAL DI,1 ; OFFSET #2 SINCE 2 BYTES/CHAR
	4392	
	4393	;----- DETERMINE THE SOURCE ADDRESS IN THE BUFFER
	4394	
F4B3	4395	R7: ; FIND_SOURCE
F4B3 06	4396	PUSH ES ; GET SEGMENTS BOTH POINTING TO REGEN
F4B4 1F	4397	POP DS
F4B5 2AED	4398	SUB CH,CH ; ZERO TO HIGH OF COUNT REG
F4B7 D0E3	4399	SAL BL,1 ; MULTIPLY NUMBER OF LINES BY 4
F4B9 D0E3	4400	SAL BL,1
F4BB 742D	4401	JZ R11 ; IF ZERO, THEN BLANK ENTIRE FIELD
F4BD 8AC3	4402	MOV AL,BL ; GET NUMBER OF LINES IN AL
F4BF B450	4403	MOV AH,80 ; 80 BYTES/ROW
F4C1 F6E4	4404	MUL AH ; DETERMINE OFFSET TO SOURCE
F4C3 8BF7	4405	MOV SI,DI ; SET UP SOURCE
F4C5 03F0	4406	ADD SI,AX ; ADD IN OFFSET TO IT
F4C7 8AE6	4407	MOV AH,DH ; NUMBER OF ROWS IN FIELD
F4C9 2AE3	4408	SUB AH,BL ; DETERMINE NUMBER TO MOVE
	4409	
	4410	;----- LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD FIELDS
	4411	
F4CB	4412	R8: ; ROW_LOOP
F4CB E88000	4413	CALL R17 ; MOVE ONE ROW
F4CE 81EEB01F	4414	SUB SI,2000H-80 ; MOVE TO NEXT ROW
F4D2 81EFB01F	4415	SUB DI,2000H-80

LOC OBJ	LINE	SOURCE
F4D6 FECC	4416	DEC AH ; NUMBER OF ROWS TO MOVE
F4D8 75F1	4417	JNZ R8 ; CONTINUE TILL ALL MOVED
	4418	
	4419	;----- FILL IN THE VACATED LINE(S)
	4420	
F4DA	4421	R9: ; CLEAR_ENTRY
F4DA 8AC7	4422	MOV AL,BH ; ATTRIBUTE TO FILL WITH
F4DC	4423	R10:
F4DC E8800	4424	CALL R10 ; CLEAR THAT ROW
F4DF 81EFB01F	4425	SUB DI,200DH-80 ; POINT TO NEXT LINE
F4E3 FECD	4426	DEC BL ; NUMBER OF LINES TO FILL
F4E5 75F5	4427	JNZ R10 ; CLEAR_LOOP
F4E7 E9DBFC	4428	JMP VIDEO_RETURN ; EVERYTHING DONE
F4EA	4429	R11: ; BLANK_FIELD
F4EA 8ADE	4430	MOV BL,DH ; SET BLANK COUNT TO
	4431	; EVERYTHING IN FIELD
F4EC EBEC	4432	JMP R9 ; CLEAR THE FIELD
	4433	GRAPHICS_UP ENDP
	4434	;-----
	4435	; SCROLL DOWN :
	4436	; THIS ROUTINE SCROLLS DOWN THE INFORMATION ON THE CRT :
	4437	; ENTRY :
	4438	; CH,CL = UPPER LEFT CORNER OF REGION TO SCROLL :
	4439	; DH,DL = LOWER RIGHT CORNER OF REGION TO SCROLL :
	4440	; BOTH OF THE ABOVE ARE IN CHARACTER POSITIONS :
	4441	; BH = FILL VALUE FOR BLANKED LINES :
	4442	; AL = # LINES TO SCROLL (AL=0 MEANS BLANK THE ENTIRE :
	4443	; FIELD) :
	4444	; DS = DATA SEGMENT :
	4445	; ES = REGEN SEGMENT :
	4446	; EXIT :
	4447	; NOTHING, THE SCREEN IS SCROLLED :
	4448	;-----
F4EE	4449	GRAPHICS_DOWN PROC NEAR
F4EE FD	4450	STD ; SET DIRECTION
F4EF 8AD8	4451	MOV BL,AL ; SAVE LINE COUNT IN BL
F4F1 8BC2	4452	MOV AX,DX ; GET LOWER RIGHT POSITION INTO AX REG
	4453	
	4454	;----- USE CHARACTER SUBROUTINE FOR POSITIONING
	4455	;----- ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE
	4456	
F4F3 E80F02	4457	CALL GRAPH_POSN
F4F6 8BF8	4458	MOV DI,AX ; SAVE RESULT AS DESTINATION ADDRESS
	4459	
	4460	;----- DETERMINE SIZE OF WINDOW
	4461	
F4F8 2BD1	4462	SUB DX,CX
F4FA 81C20101	4463	ADD DX,101H ; ADJUST VALUES
F4FE D0E6	4464	SAL DH,1 ; MULTIPLY # ROWS BY 4
	4465	; SINCE 8 VERT DOTS/CHAR
F500 D0E6	4466	SAL DH,1 ; AND EVEN/OOD ROWS
	4467	
	4468	;----- DETERMINE CRT MODE
	4469	
F502 803E490006	4470	CMP CRT_MODE,6 ; TEST FOR MEDIUM RES
F507 7305	4471	JNC R12 ; FIND_SOURCE_DOWN
	4472	
	4473	;----- MEDIUM RES DOWN
	4474	
F509 D0E2	4475	SAL DL,1 ; # COLUMNS * 2, SINCE
	4476	; 2 BYTES/CHAR (OFFSET OK)
F50B D1E7	4477	SAL DI,1 ; OFFSET #2 SINCE 2 BYTES/CHAR
F50D 47	4478	INC DI ; POINT TO LAST BYTE
	4479	
	4480	;----- DETERMINE THE SOURCE ADDRESS IN THE BUFFER
	4481	
F50E	4482	R12: ; FIND_SOURCE_DOWN
F50E 06	4483	PUSH ES ; BOTH SEGMENTS TO REGEN
F50F 1F	4484	POP DS
F510 2AED	4485	SUB CH,CX ; ZERO TO HIGH OF COUNT REG
F512 81C7F000	4486	ADD DI,240 ; POINT TO LAST ROW OF PIXELS
F516 D0E3	4487	SAL BL,1 ; MULTIPLY NUMBER OF LINES BY 4
F518 D0E3	4488	SAL BL,1
F51A 742E	4489	JZ R16 ; IF ZERO, THEN BLANK ENTIRE FIELD
F51C 8AC3	4490	MOV AL,BL ; GET NUMBER OF LINES IN AL
F51E B450	4491	MOV AH,80 ; 80 BYTES/ROW
F520 F6E4	4492	MUL AH ; DETERMINE OFFSET TO SOURCE

LOC OBJ	LINE	SOURCE
F522 8BF7	4493	MOV SI,DI ; SET UP SOURCE
F524 2BF0	4494	SUB SI,AX ; SUBTRACT THE OFFSET
F526 8AE6	4495	MOV AH,DH ; NUMBER OF ROWS IN FIELD
F528 2AE3	4496	SUB AH,BL ; DETERMINE NUMBER TO MOVE
	4497	
	4498	
	4499	I----- LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD FIELDS
	4500	
F52A	R13:	; ROM_LOOP_DOWN
F52A E82100	4501	CALL R17 ; MOVE ONE ROW
F52D 81EE5020	4502	SUB SI,2000H+80 ; MOVE TO NEXT ROW
F531 81EF5020	4503	SUB DI,2000H+80
F535 FECC	4504	DEC AH ; NUMBER OF ROWS TO MOVE
F537 75F1	4505	JNZ R13 ; CONTINUE TILL ALL MOVED
	4506	
	4507	
	4508	I----- FILL IN THE VACATED LINE(S)
	4509	
F539	R14:	; CLEAR_ENTRY_DOWN
F539 8AC7	4510	MOV AL,BH ; ATTRIBUTE TO FILL WITH
F53B	R15:	; CLEAR_LOOP_DOWN
F53B E82900	4512	CALL R18 ; CLEAR A ROW
F53E 81EF5020	4513	SUB DI,2000H+80 ; POINT TO NEXT LINE
F542 FECC	4514	DEC BL ; NUMBER OF LINES TO FILL
F544 75F5	4515	JNZ R15 ; CLEAR_LOOP_DOWN
F546 FC	4516	CLD ; RESET THE DIRECTION FLAG
F547 E97BFC	4517	JMP VIDEO_RETURN ; EVERYTHING DONE
F54A	R16:	; BLANK_FIELD_DOWN
F54A 8ADE	4519	MOV BL,DH ; SET BLANK COUNT TO
	4520	; EVERYTHING IN FIELD
F54C EBEB	4521	JMP R14 ; CLEAR THE FIELD
	4522	GRAPHICS_DOWN ENDP
	4523	
	4524	I----- ROUTINE TO MOVE ONE ROW OF INFORMATION
	4525	
F54E	R17	PROC NEAR
F54E 8ACA	4527	MOV CL,DL ; NUMBER OF BYTES IN THE ROW
F550 56	4528	PUSH SI
F551 57	4529	PUSH DI ; SAVE POINTERS
F552 F3	4530	REP MOVSB ; MOVE THE EVEN FIELD
F553 A4		
F554 5F	4531	POP DI
F555 5E	4532	POP SI
F556 81C60020	4533	ADD SI,2000H
F55A 81C70020	4534	ADD DI,2000H ; POINT TO THE ODD FIELD
F55E 56	4535	PUSH SI
F55F 57	4536	PUSH DI ; SAVE THE POINTERS
F560 8ACA	4537	MOV CL,DL ; COUNT BACK
F562 F3	4538	REP MOVSB ; MOVE THE ODD FIELD
F563 A4		
F564 5F	4539	POP DI
F565 5E	4540	POP SI ; POINTERS BACK
F566 C3	4541	RET ; RETURN TO CALLER
	4542	R17 ENDP
	4543	
	4544	I----- CLEAR A SINGLE ROW
	4545	
F567	R18	PROC NEAR
F567 8ACA	4547	MOV CL,DL ; NUMBER OF BYTES IN FIELD
F569 57	4548	PUSH DI ; SAVE POINTER
F56A F3	4549	REP STOSB ; STORE THE NEW VALUE
F56B AA		
F56C 5F	4550	POP DI ; POINTER BACK
F56D 81C70020	4551	ADD DI,2000H ; POINT TO ODD FIELD
F571 57	4552	PUSH DI
F572 8ACA	4553	MOV CL,DL
F574 F3	4554	REP STOSB ; FILL THE ODD FILED
F575 AA		
F576 5F	4555	POP DI
F577 C3	4556	RET ; RETURN TO CALLER
	4557	R18 ENDP
	4558	
	4559	; GRAPHICS WRITE ;
	4560	; THIS ROUTINE WRITES THE ASCII CHARACTER TO THE ;
	4561	; CURRENT POSITION ON THE SCREEN. ;
	4562	; ENTRY ;
	4563	; AL = CHARACTER TO WRITE ;
	4564	; BL = COLOR ATTRIBUTE TO BE USED FOR FOREGROUND COLOR ;
	4565	; IF BIT 7 IS SET, THE CHAR IS XOR'D INTO THE REGEN ;

```

4566 ; BUFFER ( 0 IS USED FOR THE BACKGROUND COLOR ) ;
4567 ; CX = NUMBER OF CHARS TO WRITE ;
4568 ; DS = DATA SEGMENT ;
4569 ; ES = REGEN SEGMENT ;
4570 ; EXIT ;
4571 ; NOTHING IS RETURNED ;
4572 ; ;
4573 ; GRAPHICS READ ;
4574 ; THIS ROUTINE READS THE ASCII CHARACTER AT THE CURRENT ;
4575 ; CURSOR POSITION ON THE SCREEN BY MATCHING THE DOTS ON ;
4576 ; THE SCREEN TO THE CHARACTER GENERATOR CODE POINTS ;
4577 ; ENTRY ;
4578 ; NONE ( 0 IS ASSUMED AS THE BACKGROUND COLOR ;
4579 ; EXIT ;
4580 ; AL = CHARACTER READ AT THAT POSITION ( 0 RETURNED IF ;
4581 ; NONE FOUND) ;
4582 ; ;
4583 ; FOR BOTH ROUTINES, THE IMAGES USED TO FORM CHARS ARE ;
4584 ; CONTAINED IN ROM FOR THE 1ST 128 CHARS. TO ACCESS CHARS ;
4585 ; IN THE SECOND HALF, THE USER MUST INITIALIZE THE VECTOR AT ;
4586 ; INTERRUPT 1FH (LOCATION 0007CH) TO POINT TO THE USER ;
4587 ; SUPPLIED TABLE OF GRAPHIC IMAGES (8x8 BOXES). ;
4588 ; FAILURE TO DO SO WILL CAUSE IN STRANGE RESULTS ;
4589 ;-----
4590 ASSUME CS:CODE,DS:DATA,ES:DATA
F578 4591 GRAPHICS_WRITE PROC NEAR
F578 B400 4592 MOV AH,0 ; ZERO TO HIGH OF CODE POINT
F57A 50 4593 PUSH AX ; SAVE CODE POINT VALUE
4594
4595 ;----- DETERMINE POSITION IN REGEN BUFFER TO PUT CODE POINTS
4596
F57B E80401 4597 CALL S26 ; FIND LOCATION IN REGEN BUFFER
F57E 8BF8 4598 MOV DI,AX ; REGEN POINTER IN DI
4599
4600 ;----- DETERMINE REGION TO GET CODE POINTS FROM
4601
F580 58 4602 POP AX ; RECOVER CODE POINT
F581 3C80 4603 CMP AL,80H ; IS IT IN SECOND HALF
F583 7306 4604 JAE S1 ; YES
4605
4606 ;----- IMAGE IS IN FIRST HALF, CONTAINED IN ROM
4607
F585 BE6EFA 4608 MOV SI,0FA6EH ; CRT_CHAR_GEN (OFFSET OF IMAGES)
F588 0E 4609 PUSH CS ; SAVE SEGMENT ON STACK
F589 EB0F 4610 JMP SHORT S2 ; DETERMINE_MODE
4611
4612 ;----- IMAGE IS IN SECOND HALF, IN USER RAM
4613
S1: 4614 ; EXTEND_CHAR
F58B 2C80 4615 SUB AL,80H ; ZERO ORIGIN FOR SECOND HALF
F58D 1E 4616 PUSH DS ; SAVE DATA POINTER
F58E 2BF6 4617 SUB SI,SI
F590 8EDE 4618 MOV DS,SI ; ESTABLISH VECTOR ADDRESSING
4619 ASSUME DS:ABS0
F592 C5367C00 4620 LDS SI,EXT_PTR ; GET THE OFFSET OF THE TABLE
F596 8CDA 4621 MOV DX,DS ; GET THE SEGMENT OF THE TABLE
4622 ASSUME DS:DATA
F598 1F 4623 POP DS ; RECOVER DATA SEGMENT
F599 52 4624 PUSH DX ; SAVE TABLE SEGMENT ON STACK
4625
4626 ;----- DETERMINE GRAPHICS MODE IN OPERATION
4627
F59A 4628 S2: ; DETERMINE_MODE
F59A D1E0 4629 SAL AX,1 ; MULTIPLY CODE POINT
F59C D1E0 4630 SAL AX,1 ; VALUE BY 8
F59E D1E0 4631 SAL AX,1
F5A0 03F0 4632 ADD SI,AX ; SI HAS OFFSET OF DESIRED CODES
F5A2 803E490006 4633 CMP CRT_MODE,6
F5A7 1F 4634 POP DS ; RECOVER TABLE POINTER SEGMENT
F5A8 722C 4635 JC S7 ; TEST FOR MEDIUM RESOLUTION MODE
4636
4637 ;----- HIGH RESOLUTION MODE
4638
F5AA 4639 S3: ; HIGH_CHAR
F5AA 57 4640 PUSH DI ; SAVE REGEN POINTER
F5AB 56 4641 PUSH SI ; SAVE CODE POINTER
F5AC B604 4642 MOV DH,4 ; NUMBER OF TIMES THROUGH LOOP

```

LOC OBJ	LINE	SOURCE	
F5AE	4643	S4:	
F5AE AC	4644	LODSB	; GET BYTE FROM CODE POINTS
F5AF F6C380	4645	TEST BL,80H	; SHOULD WE USE THE FUNCTION
F5B2 7516	4646	JNZ S6	; TO PUT CHAR IN
F5B4 AA	4647	STOSB	; STORE IN REGEN BUFFER
F5B5 AC	4648	LODSB	
F5B6	4649		
F5B6 26885FF1F	4650	S5: MOV ES:[DI+2000H-1],AL	; STORE IN SECOND HALF
F5BB 83C74F	4651	ADD DI,79	; MOVE TO NEXT ROW IN REGEN
F5BE FECE	4652	DEC DH	; DONE WITH LOOP
F5C0 75EC	4653	JNZ S4	
F5C2 5E	4654	POP SI	
F5C3 5F	4655	POP DI	; RECOVER REGEN POINTER
F5C4 47	4656	INC DI	; POINT TO NEXT CHAR POSITION
F5C5 E2E3	4657	LOOP S3	; MORE CHARS TO WRITE
F5C7 E9FBFB	4658	JMP VIDEO_RETURN	
F5CA	4659		
F5CA 263205	4660	S6: XOR AL,ES:[DI]	; EXCLUSIVE OR WITH CURRENT
F5CD AA	4661	STOSB	; STORE THE CODE POINT
F5CE AC	4662	LODSB	; AGAIN FOR ODD FIELD
F5CF 263285FF1F	4663	XOR AL,ES:[DI+2000H-1]	
F5D4 EBE0	4664	JMP S5	; BACK TO MAINSTREAM
	4665		
	4666	;----- MEDIUM RESOLUTION WRITE	
	4667		
F5D6	4668	S7:	; MED_RES_WRITE
F5D6 8AD3	4669	MOV DL,BL	; SAVE HIGH COLOR BIT
F5D8 D1E7	4670	SAL DI,1	; OFFSET*2 SINCE 2 BYTES/CHAR
F5DA E8D100	4671	CALL S19	; EXPAND BL TO FULL WORD OF COLOR
F5DD	4672		; MED_CHAR
F5DD 57	4673	S8: PUSH DI	; SAVE REGEN POINTER
F5DE 56	4674	PUSH SI	; SAVE THE CODE POINTER
F5DF B604	4675	MOV DH,4	; NUMBER OF LOOPS
F5E1	4676		
F5E1 AC	4677	S9: LODSB	; GET CODE POINT
F5E2 E8DE00	4678	CALL S21	; DOUBLE UP ALL THE BITS
F5E5 23C3	4679	AND AX,BX	; CONVERT THEM TO FOREGROUND
	4680		; COLOR (0 BACK)
F5E7 F6C280	4681	TEST DL,80H	; IS THIS XOR FUNCTION
F5EA 7407	4682	JZ S10	; NO, STORE IT IN AS IT IS
F5EC 263225	4683	XOR AH,ES:[DI]	; DO FUNCTION WITH HALF
F5EF 26324501	4684	XOR AL,ES:[DI+1]	; AND WITH OTHER HALF
F5F3	4685		
F5F3 268825	4686	S10: MOV ES:[DI],AH	; STORE FIRST BYTE
F5F6 26884501	4687	MOV ES:[DI+1],AL	; STORE SECOND BYTE
F5FA AC	4688	LODSB	; GET CODE POINT
F5FB E8C500	4689	CALL S21	
F5FE 23C3	4690	AND AX,BX	; CONVERT TO COLOR
F600 F6C280	4691	TEST DL,80H	; AGAIN, IS THIS XOR FUNCTION
F603 740A	4692	JZ S11	; NO, JUST STORE THE VALUES
F605 2632A50020	4693	XOR AH,ES:[DI+2000H]	; FUNCTION WITH FIRST HALF
F60A 2632851020	4694	XOR AL,ES:[DI+2001H]	; AND WITH SECOND HALF
F60F	4695		
F60F 2688A50020	4696	S11: MOV ES:[DI+2000H],AH	
F614 2688851020	4697	MOV ES:[DI+2000H+1],AL	; STORE IN SECOND PORTION OF BUFFER
F619 83C750	4698	ADD DI,80	; POINT TO NEXT LOCATION
F61C FECE	4699	DEC DH	
F61E 75C1	4700	JNZ S9	; KEEP GOING
F620 5E	4701	POP SI	; RECOVER CODE POINTER
F621 5F	4702	POP DI	; RECOVER REGEN POINTER
F622 47	4703	INC DI	; POINT TO NEXT CHAR POSITION
F623 47	4704	INC DI	
F624 E2B7	4705	LOOP S8	; MORE TO WRITE
F626 E99CFB	4706	JMP VIDEO_RETURN	
	4707	GRAPHICS_WRITE ENDP	
	4708	;-----	
	4709	; GRAPHICS_READ :	
	4710	;-----	
F629	4711	GRAPHICS_READ PROC NEAR	
F629 E8D600	4712	CALL S26	; CONVERTED TO OFFSET IN REGEN
F62C 8BF0	4713	MOV SI,AX	; SAVE IN SI
F62E 83EC08	4714	SUB SP,8	; ALLOCATE SPACE TO SAVE THE
	4715		; READ CODE POINT
F631 8BEC	4716	MOV BP,SP	; POINTER TO SAVE AREA
	4717		
	4718	;----- DETERMINE GRAPHICS MODES	
	4719		

LOC OBJ	LINE	SOURCE	
F633 803E490006	4720	CHP	CRT_MODE,6
F638 06	4721	PUSH	ES
F639 1F	4722	POP	DS ; POINT TO REGEN SEGMENT
F63A 721A	4723	JC	S13 ; MEDIUM RESOLUTION
	4724		
	4725	;----- HIGH RESOLUTION READ	
	4726		
	4727	;----- GET VALUES FROM REGEN BUFFER AND CONVERT TO CODE POINT	
	4728		
F63C B604	4729	MOV	DH,4 ; NUMBER OF PASSES
F63E	4730	S12:	
F63E 8A04	4731	MOV	AL,[SI] ; GET FIRST BYTE
F640 884600	4732	MOV	[BP],AL ; SAVE IN STORAGE AREA
F643 45	4733	INC	BP ; NEXT LOCATION
F644 8A840020	4734	MOV	AL,[SI+2000H] ; GET LOWER REGION BYTE
F648 884600	4735	MOV	[BP],AL ; ADJUST AND STORE
F64B 45	4736	INC	BP
F64C 83C650	4737	ADD	SI,80 ; POINTER INTO REGEN
F64F FECE	4738	DEC	DH ; LOOP CONTROL
F651 75EB	4739	JNZ	S12 ; DO IT SOME MORE
F653 EB1790	4740	JMP	S15 ; GO MATCH THE SAVED CODE POINTS
	4741		
	4742	;----- MEDIUM RESOLUTION READ	
	4743		
F656	4744	S13:	; MED_RES_READ
F656 D1E6	4745	SAL	SI,1 ; OFFSET*2 SINCE 2 BYTES/CHAR
F65B B604	4746	MOV	DH,4 ; NUMBER OF PASSES
F65A	4747	S14:	
F65A E88800	4748	CALL	S23 ; GET PAIR BYTES FROM REGEN
	4749		; INTO SINGLE SAVE
F65D 81C60020	4750	ADD	SI,2000H ; GO TO LOWER REGION
F661 E88100	4751	CALL	S23 ; GET THIS PAIR INTO SAVE
F664 81EEB01F	4752	SUB	SI,2000H-80 ; ADJUST POINTER BACK INTO UPPER
F668 FECE	4753	DEC	DH
F66A 75EE	4754	JNZ	S14 ; KEEP GOING UNTIL ALL 8 DONE
	4755		
	4756	;----- SAVE AREA HAS CHARACTER IN IT, MATCH IT	
	4757		
F66C	4758	S15:	; FIND_CHAR
F66C BF6EFA90	4759	MOV	DI,OFFSET CRT_CHAR_GEN ; ESTABLISH ADDRESSING
F670 0E	4760	PUSH	CS
F671 07	4761	POP	ES ; CODE POINTS IN CS
F672 83ED08	4762	SUB	BP,8 ; ADJUST POINTER TO BEGINNING
	4763		; OF SAVE AREA
F675 8BF5	4764	MOV	SI,BP
F677 FC	4765	CLD	; ENSURE DIRECTION
F678 B000	4766	MOV	AL,0 ; CURRENT CODE POINT BEING MATCHED
F67A	4767	S16:	
F67A 16	4768	PUSH	SS ; ESTABLISH ADDRESSING TO STACK
F67B 1F	4769	POP	DS ; FOR THE STRING COMPARE
F67C BA8000	4770	MOV	DX,128 ; NUMBER TO TEST AGAINST
F67F	4771	S17:	
F67F 56	4772	PUSH	SI ; SAVE SAVE AREA POINTER
F680 57	4773	PUSH	DI ; SAVE CODE POINTER
F681 B90800	4774	MOV	CX,8 ; NUMBER OF BYTES TO MATCH
F684 F3	4775	REPE	CMPSB ; COMPARE THE 8 BYTES
F685 A6			
F686 5F	4776	POP	DI ; RECOVER THE POINTERS
F687 5E	4777	POP	SI
F688 741E	4778	JZ	S18 ; IF ZERO FLAG SET, THEN MATCH OCCURRED
F68A FE00	4779	INC	AL ; NO MATCH, MOVE ON TO NEXT
F68C 83C708	4780	ADD	DI,8 ; NEXT CODE POINT
F68F 4A	4781	DEC	CX ; LOOP CONTROL
F690 75ED	4782	JNZ	S17 ; DO ALL OF THEM
	4783		
	4784	;----- CHAR NOT MATCHED, MIGHT BE IN USER SUPPLIED SECOND HALF	
	4785		
F692 3C00	4786	CHP	AL,0 ; AL <> 0 IF ONLY 1ST HALF SCANNED
F694 7412	4787	JE	S18 ; IF = 0, THEN ALL HAS BEEN SCANNED
F696 2BC0	4788	SUB	AX,AX
F698 8ED8	4789	MOV	DS,AX ; ESTABLISH ADDRESSING TO VECTOR
	4790	ASSUME	DS:ABS0
F69A C43E7C00	4791	LES	DI,EXT_PTR ; GET POINTER
F69E 8CC0	4792	MOV	AX,ES ; SEE IF THE POINTER REALLY EXISTS
F6A0 0BC7	4793	OR	AX,DI ; IF ALL 0, THEN DOESN'T EXIST
F6A2 7404	4794	JZ	S18 ; NO SENSE LOOKING
F6A4 B080	4795	MOV	AL,128 ; ORIGIN FOR SECOND HALF

LOC OBJ	LINE	SOURCE
F6A6 EBD2	4796	JMP S16 ; GO BACK AND TRY FOR IT
	4797	ASSUME DS:DATA
	4798	
	4799	;----- CHARACTER IS FOUND (AL=0 IF NOT FOUND)
	4800	
F6A8	4801	S18:
F6A8 83C408	4802	ADD SP,8 ; READJUST THE STACK, THROW AWAY SAVE
F6A8 E917FB	4803	JMP VIDEO_RETURN ; ALL DONE
	4804	GRAPHICS_READ ENDP
	4805	;-----
	4806	; EXPAND_MED_COLOR :
	4807	; THIS ROUTINE EXPANDS THE LOW 2 BITS IN BL TO :
	4808	; FILL THE ENTIRE BX REGISTER :
	4809	; ENTRY :
	4810	; BL = COLOR TO BE USED (LOW 2 BITS) :
	4811	; EXIT :
	4812	; BX = COLOR TO BE USED (8 REPLICATIONS OF THE :
	4813	; 2 COLOR BITS) :
	4814	;-----
F6AE	4815	S19 PROC NEAR
F6AE 80E303	4816	AND BL,3 ; ISOLATE THE COLOR BITS
F6B1 8AC3	4817	MOV AL,BL ; COPY TO AL
F6B3 51	4818	PUSH CX ; SAVE REGISTER
F6B4 B90300	4819	MOV CX,3 ; NUMBER OF TIMES TO DO THIS
F6B7	4820	S20:
F6B7 D0E0	4821	SAL AL,1
F6B9 D0E0	4822	SAL AL,1 ; LEFT SHIFT BY 2
F6BB 0AD8	4823	OR BL,AL ; ANOTHER COLOR VERSION INTO BL
F6BD E2F8	4824	LOOP S20 ; FILL ALL OF BL
F6BF 8AFB	4825	MOV BH,BL ; FILL UPPER PORTION
F6C1 59	4826	POP CX ; REGISTER BACK
F6C2 C3	4827	RET ; ALL DONE
	4828	S19 ENDP
	4829	;-----
	4830	; EXPAND_BYTE :
	4831	; THIS ROUTINE TAKES THE BYTE IN AL AND DOUBLES :
	4832	; ALL OF THE BITS, TURNING THE 8 BITS INTO :
	4833	; 16 BITS. THE RESULT IS LEFT IN AX :
	4834	;-----
F6C3	4835	S21 PROC NEAR
F6C3 52	4836	PUSH DX ; SAVE REGISTERS
F6C4 51	4837	PUSH CX
F6C5 53	4838	PUSH BX
F6C6 2BD2	4839	SUB DX,DX ; RESULT REGISTER
F6C8 B90100	4840	MOV CX,1 ; MASK REGISTER
F6CB	4841	S22:
F6CB 8BD8	4842	MOV BX,AX ; BASE INTO TEMP
F6CD 23D9	4843	AND BX,CX ; USE MASK TO EXTRACT A BIT
F6CF 0BD3	4844	OR DX,BX ; PUT INTO RESULT REGISTER
F6D1 D1E0	4845	SHL AX,1
F6D3 D1E1	4846	SHL CX,1 ; SHIFT BASE AND MASK BY 1
F6D5 8BD8	4847	MOV BX,AX ; BASE TO TEMP
F6D7 23D9	4848	AND BX,CX ; EXTRACT THE SAME BIT
F6D9 0BD3	4849	OR DX,BX ; PUT INTO RESULT
F6DB D1E1	4850	SHL CX,1 ; SHIFT ONLY MASK NOW,
	4851	; MOVING TO NEXT BASE
F6DD 73EC	4852	JNC S22 ; USE MASK BIT COMING OUT TO TERMINATE
F6DF 8BC2	4853	MOV AX,DX ; RESULT TO PARM REGISTER
F6E1 5B	4854	POP BX
F6E2 59	4855	POP CX ; RECOVER REGISTERS
F6E3 5A	4856	POP DX
F6E4 C3	4857	RET ; ALL DONE
	4858	S21 ENDP
	4859	;-----
	4860	; MED_READ_BYTE :
	4861	; THIS ROUTINE WILL TAKE 2 BYTES FROM THE REGEN :
	4862	; BUFFER, COMPARE AGAINST THE CURRENT FOREGROUND :
	4863	; COLOR, AND PLACE THE CORRESPONDING ON/OFF BIT :
	4864	; PATTERN INTO THE CURRENT POSITION IN THE SAVE :
	4865	; AREA :
	4866	; ENTRY :
	4867	; SI,DS = POINTER TO REGEN AREA OF INTEREST :
	4868	; BX = EXPANDED FOREGROUND COLOR :
	4869	; BP = POINTER TO SAVE AREA :
	4870	; EXIT :
	4871	; BP IS INCREMENT AFTER SAVE :
	4872	;-----

LOC OBJ	LINE	SOURCE
F6E5	4873	S23 PROC NEAR
F6E5 8A24	4874	MOV AH,[SI] ; GET FIRST BYTE
F6E7 8A4401	4875	MOV AL,[SI+1] ; GET SECOND BYTE
F6EA B900C0	4876	MOV CX,0C000H ; 2 BIT MASK TO TEST THE ENTRIES
F6ED B200	4877	MOV DL,0 ; RESULT REGISTER
F6EF	4878	
F6EF 85C1	4879	S24: TEST AX,CX ; IS THIS SECTION BACKGROUND?
F6F1 F8	4880	CLC ; CLEAR CARRY IN HOPES THAT IT IS
F6F2 7401	4881	JZ S25 ; IF ZERO, IT IS BACKGROUND
F6F4 F9	4882	STC ; WASN'T, SO SET CARRY
F6F5 D0D2	4883	S25: RCL DL,1 ; MOVE THAT BIT INTO THE RESULT
F6F7 D1E9	4884	SHR CX,1
F6F9 D1E9	4885	SHR CX,1 ; MOVE THE MASK TO THE RIGHT BY 2 BITS
F6FB 73F2	4886	JNC S24 ; DO IT AGAIN IF MASK DIDN'T FALL OUT
F6FD 885600	4887	MOV [BP],DL ; STORE RESULT IN SAVE AREA
F700 45	4888	INC BP ; ADJUST POINTER
F701 C3	4889	RET ; ALL DONE
	4890	S23 ENDP
	4891	-----
	4892	; V4_POSITION :
	4893	; THIS ROUTINE TAKES THE CURSOR POSITION :
	4894	; CONTAINED IN THE MEMORY LOCATION, AND :
	4895	; CONVERTS IT INTO AN OFFSET INTO THE :
	4896	; REGEN BUFFER, ASSUMING ONE BYTE/CHAR. :
	4897	; FOR MEDIUM RESOLUTION GRAPHICS, :
	4898	; THE NUMBER MUST BE DOUBLED. :
	4899	; ENTRY :
	4900	; NO REGISTERS, MEMORY LOCATION :
	4901	; CURSOR_POSN IS USED :
	4902	; EXIT :
	4903	; AX CONTAINS OFFSET INTO REGEN BUFFER :
	4904	-----
F702	4905	S26 PROC NEAR
F702 A15000	4906	MOV AX,CURSOR_POSN ; GET CURRENT CURSOR
F705	4907	GRAPH_POSN LABEL NEAR
F705 53	4908	PUSH BX ; SAVE REGISTER
F706 8B08	4909	MOV BX,AX ; SAVE A COPY OF CURRENT CURSOR
F708 8AC4	4910	MOV AL,AH ; GET ROWS TO AL
F70A F6264A00	4911	MUL BYTE PTR CRT_COLS ; MULTIPLY BY BYTES/COLUMN
F70E D1E0	4912	SHL AX,1 ; MULTIPLY * 4 SINCE 4 ROWS/BYTE
F710 D1E0	4913	SHL AX,1
F712 2AFF	4914	SUB BH,BH ; ISOLATE COLUMN VALUE
F714 03C3	4915	ADD AX,BX ; DETERMINE OFFSET
F716 5B	4916	POP BX ; RECOVER POINTER
F717 C3	4917	RET ; ALL DONE
	4918	S26 ENDP
	4919	-----
	4920	; WRITE_TTY :
	4921	; THIS INTERFACE PROVIDES A TELETYPE LIKE INTERFACE TO THE VIDEO :
	4922	; CARD. THE INPUT CHARACTER IS WRITTEN TO THE CURRENT CURSOR :
	4923	; POSITION, AND THE CURSOR IS MOVED TO THE NEXT POSITION. IF THE :
	4924	; CURSOR LEAVES THE LAST COLUMN OF THE FIELD, THE COLUMN IS SET :
	4925	; TO ZERO, AND THE ROW VALUE IS INCREMENTED. IF THE ROW VALUE :
	4926	; LEAVES THE FIELD, THE CURSOR IS PLACED ON THE LAST ROW, FIRST :
	4927	; COLUMN, AND THE ENTIRE SCREEN IS SCROLLED UP ONE LINE. WHEN :
	4928	; THE SCREEN IS SCROLLED UP, THE ATTRIBUTE FOR FILLING THE NEWLY :
	4929	; BLANKED LINE IS READ FROM THE CURSOR POSITION ON THE PREVIOUS :
	4930	; LINE BEFORE THE SCROLL, IN CHARACTER MODE. IN GRAPHICS MODE, :
	4931	; THE 0 COLOR IS USED. :
	4932	; ENTRY :
	4933	; (AH) = CURRENT CRT MODE :
	4934	; (AL) = CHARACTER TO BE WRITTEN :
	4935	; NOTE THAT BACK SPACE, CAR RET, BELL AND LINE FEED ARE HANDLED :
	4936	; AS COMMANDS RATHER THAN AS DISPLAYABLE GRAPHICS :
	4937	; (BL) = FOREGROUND COLOR FOR CHAR WRITE IF CURRENTLY IN A :
	4938	; GRAPHICS MODE :
	4939	; EXIT :
	4940	; ALL REGISTERS SAVED :
	4941	-----
	4942	ASSUME CS:CODE,DS:DATA
F718	4943	WRITE_TTY PROC NEAR
F718 50	4944	PUSH AX ; SAVE REGISTERS
F719 50	4945	PUSH AX ; SAVE CHAR TO WRITE
F71A B403	4946	MOV AH,3
F71C 8A3E6200	4947	MOV BH,ACTIVE_PAGE ; GET THE CURRENT ACTIVE PAGE
F720 C010	4948	INT 10H ; READ THE CURRENT CURSOR POSITION
F722 58	4949	POP AX ; RECOVER CHAR

LOC OBJ	LINE	SOURCE
	4950	
	4951	;----- DX NOW HAS THE CURRENT CURSOR POSITION
	4952	
F723 3C08	4953	CMP AL,8 ; IS IT A BACKSPACE
F725 7452	4954	JE U8 ; BACK_SPACE
F727 3C0D	4955	CMP AL,0DH ; IS IT CARRIAGE RETURN
F729 7457	4956	JE U9 ; CAR_RET
F72B 3C0A	4957	CMP AL,0AH ; IS IT A LINE FEED
F72D 7457	4958	JE U10 ; LINE_FEED
F72F 3C07	4959	CMP AL,07H ; IS IT A BELL
F731 745A	4960	JE U11 ; BELL
	4961	
	4962	;----- WRITE THE CHAR TO THE SCREEN
	4963	
	4964	
F733 B40A	4965	MOV AH,10 ; WRITE CHAR ONLY
F735 B9010D	4966	MOV CX,1 ; ONLY ONE CHAR
F738 CD10	4967	INT 10H ; WRITE THE CHAR
	4968	
	4969	;----- POSITION THE CURSOR FOR NEXT CHAR
	4970	
F73A FEC2	4971	INC DL
F73C 3A164A0D	4972	CMP DL,BYTE PTR CRT_COLS ; TEST FOR COLUMN OVERFLOW
F740 7533	4973	JNZ U7 ; SET_CURSOR
F742 B200	4974	MOV DL,0 ; COLUMN FOR CURSOR
F744 80FE18	4975	CMP DH,24
F747 752A	4976	JNZ U6 ; SET_CURSOR_INC
	4977	
	4978	;----- SCROLL REQUIRED
	4979	
F749	4980	U1:
F749 B402	4981	MOV AH,2
F74B CD10	4982	INT 10H ; SET THE CURSOR
	4983	
	4984	;----- DETERMINE VALUE TO FILL WITH DURING SCROLL
	4985	
F74D A0490D	4986	MOV AL,CRT_MODE ; GET THE CURRENT MODE
F750 3C04	4987	CMP AL,4
F752 7206	4988	JC U2 ; READ-CURSOR
F754 3C07	4989	CMP AL,7
F756 B70D	4990	MOV BH,0 ; FILL WITH BACKGROUND
F758 7506	4991	JNE U3 ; SCROLL-UP
F75A	4992	U2:
F75A B408	4993	MOV AH,8
F75C CD10	4994	INT 10H ; READ CHAR/ATTR AT CURRENT CURSOR
F75E 8AFC	4995	MOV BH,AH ; STORE IN BH
F760	4996	U3:
F760 B80106	4997	MOV AX,601H ; SCROLL ONE LINE
F763 2BC9	4998	SUB CX,CX ; UPPER LEFT CORNER
F765 B618	4999	MOV DH,24 ; LOWER RIGHT ROW
F767 8A164A0D	5000	MOV DL,BYTE PTR CRT_COLS ; LOWER RIGHT COLUMN
F76B FECA	5001	DEC DL
F76D	5002	U4:
F76D CD10	5003	INT 10H ; VIDEO-CALL-RETURN
F76F	5004	U5:
F76F 58	5005	POP AX ; SCROLL UP THE SCREEN
F770 E952FA	5006	JMP VIDEO_RETURN ; TTY-RETURN
F773	5007	U6:
F773 FEC6	5008	INC DH ; RESTORE THE CHARACTER
F775	5009	JMP VIDEO_RETURN ; RETURN TO CALLER
F775 B402	5010	U7:
F777 EBF4	5011	MOV AH,2 ; SET-CURSOR-INC
	5012	JMP U4 ; SET-CURSOR
	5013	;----- BACK SPACE FOUND
	5014	
F779	5015	U8:
F779 80FA0D	5016	CMP DL,0 ; ALREADY AT END OF LINE
F77C 74F7	5017	JE U7 ; SET_CURSOR
F77E FECA	5018	DEC DL ; NO -- JUST MOVE IT BACK
F780 EBF3	5019	JMP U7 ; SET_CURSOR
	5020	
	5021	;----- CARRIAGE RETURN FOUND
	5022	
F782	5023	U9:
F782 B200	5024	MOV DL,0 ; MOVE TO FIRST COLUMN
F784 EBEF	5025	JMP U7 ; SET_CURSOR
	5026	

LOC OBJ	LINE	SOURCE	
	5027	i----- LINE FEED FOUND	
	5028		
F786	5029	U10:	
F786 80FE18	5030	CMP DH,24	; BOTTOM OF SCREEN
F789 75E8	5031	JNE U6	; YES, SCROLL THE SCREEN
F78B EBBC	5032	JMP U1	; NO, JUST SET THE CURSOR
	5033		
	5034	i----- BELL FOUND	
	5035		
F78D	5036	U11:	
F78D B302	5037	MOV BL,2	; SET UP COUNT FOR BEEP
F78F E871EE	5038	CALL BEEP	; SOUND THE POD BELL
F792 EB0B	5039	JMP US	; TTY_RETURN
	5040	WRITE_TTY ENDP	
	5041	i-----	
	5042	; LIGHT PEN	:
	5043	; THIS ROUTINE TESTS THE LIGHT PEN SWITCH AND THE LIGHT	:
	5044	; PEN TRIGGER. IF BOTH ARE SET, THE LOCATION OF THE LIGHT	:
	5045	; PEN IS DETERMINED. OTHERWISE, A RETURN WITH NO	:
	5046	; INFORMATION IS MADE.	:
	5047	; ON EXIT	:
	5048	; (AH) = 0 IF NO LIGHT PEN INFORMATION IS AVAILABLE	:
	5049	; BX,CX,DX ARE DESTROYED	:
	5050	; (AH) = 1 IF LIGHT PEN IS AVAILABLE	:
	5051	; (DH,DL) = ROW,COLUMN OF CURRENT LIGHT PEN	:
	5052	; POSITION	:
	5053	; (CH) = RASTER POSITION	:
	5054	; (BX) = BEST GUESS AT PIXEL HORIZONTAL POSITION	:
	5055	i-----	
	5056	ASSUME CS:CODE,DS:DATA	
	5057	i----- SUBTRACT_TABLE	
F794	5058	V1 LABEL BYTE	
F794 03	5059	DB 3,3,5,5,3,3,3,4 ;	
F795 03			
F796 05			
F797 05			
F798 03			
F799 03			
F79A 03			
F79B 04			
F79C			
	5060	READ_LPEN PROC NEAR	
	5061		
	5062	i----- WAIT FOR LIGHT PEN TO BE DEPRESSED	
	5063		
F79C B400	5064	MOV AH,0	; SET NO LIGHT PEN RETURN CODE
F79E 8B166300	5065	MOV DX,ADDR_6845	; GET BASE ADDRESS OF 6845
F7A2 83C206	5066	ADD DX,6	; POINT TO STATUS REGISTER
F7A5 EC	5067	IN AL,DX	; GET STATUS REGISTER
F7A6 A804	5068	TEST AL,4	; TEST LIGHT PEN SWITCH
F7A8 757E	5069	JNZ V6	; NOT SET, RETURN
	5070		
	5071	i----- NOW TEST FOR LIGHT PEN TRIGGER	
	5072		
F7AA A802	5073	TEST AL,2	; TEST LIGHT PEN TRIGGER
F7AC 7503	5074	JNZ V7A	; RETURN WITHOUT RESETTING TRIGGER
F7AE E9B100	5075	JMP V7	
	5076		
	5077	i----- TRIGGER HAS BEEN SET, READ THE VALUE IN	
	5078		
F7B1	5079	V7A:	
F7B1 B410	5080	MOV AH,16	; LIGHT PEN REGISTERS ON 6845
	5081		
	5082	i----- INPUT REGS POINTED TO BY AH, AND CONVERT TO ROW COLUMN IN DX	
	5083		
F7B3 8B166300	5084	MOV DX,ADDR_6845	; ADDRESS REGISTER FOR 6845
F7B7 8AC4	5085	MOV AL,AH	; REGISTER TO READ
F7B9 EE	5086	OUT DX,AL	; SET IT UP
F7BA 42	5087	INC DX	; DATA REGISTER
F7BB EC	5088	IN AL,DX	; GET THE VALUE
F7BC 8AE8	5089	MOV CH,AL	; SAVE IN CX
F7BE 4A	5090	DEC DX	; ADDRESS REGISTER
F7BF FEC4	5091	INC AH	
F7C1 8AC4	5092	MOV AL,AH	; SECOND DATA REGISTER
F7C3 EE	5093	OUT DX,AL	
F7C4 42	5094	INC DX	; POINT TO DATA REGISTER
F7C5 EC	5095	IN AL,DX	; GET SECOND DATA VALUE
F7C6 8AE5	5096	MOV AH,CH	; AX HAS INPUT VALUE

LOC OBJ	LINE	SOURCE	
	5097		
	5098	1----- AX HAS THE VALUE READ IN FROM THE 6845	
	5099		
F7C8 8A1E4900	5100	MOV BL,CRT_MODE	
F7CC 2AFF	5101	SUB BH,BH	; MODE VALUE TO BX
F7CE 2E8A9F94F7	5102	MOV BL,CS:V1(BX)	; DETERMINE AMOUNT TO SUBTRACT
F7D3 2BC3	5103	SUB AX,BX	; TAKE IT AWAY
F7D5 8B1E4E00	5104	MOV BX,CRT_START	
F7D9 D1EB	5105	SHR BX,1	
F7DB 2BC3	5106	SUB AX,BX	
F7DD 7902	5107	JNS V2	; IF POSITIVE, DETERMINE MODE
F7DF 2BC0	5108	SUB AX,AX	; <0 PLAYS AS 0
	5109		
	5110	1----- DETERMINE MODE OF OPERATION	
	5111		
F7E1	5112	V2:	; DETERMINE_MODE
F7E1 B103	5113	MOV CL,3	; SET #8 SHIFT COUNT
F7E3 803E490004	5114	CMF CRT_MODE,4	; DETERMINE IF GRAPHICS OR ALPHA
F7E8 722A	5115	JB V4	; ALPHA_PEN
F7EA 803E490007	5116	CMF CRT_MODE,7	
F7EF 7423	5117	JE V4	; ALPHA_PEN
	5118		
	5119	1----- GRAPHICS MODE	
	5120		
F7F1 B228	5121	MOV DL,40	; DIVISOR FOR GRAPHICS
F7F3 F6F2	5122	DIV DL	; DETERMINE ROW(AL) AND COLUMN(AH)
	5123		; AL RANGE 0-99, AH RANGE 0-39
	5124		
	5125	1----- DETERMINE GRAPHIC ROW POSITION	
	5126		
F7F5 8AE8	5127	MOV CH,AL	; SAVE ROW VALUE IN CH
F7F7 02ED	5128	ADD CH,CH	; #2 FOR EVEN/ODD FIELD
F7F9 8ADC	5129	MOV BL,AH	; COLUMN VALUE TO BX
F7FB 2AFF	5130	SUB BH,BH	; MULTIPLY BY 8 FOR MEDIUM RES
F7FD 803E490006	5131	CMF CRT_MODE,6	; DETERMINE MEDIUM OR HIGH RES
F802 7504	5132	JNE V3	; NOT_HIGH_RES
F804 B104	5133	MOV CL,4	; SHIFT VALUE FOR HIGH RES
F806 D0E4	5134	SAL AH,1	; COLUMN VALUE TIMES 2 FOR HIGH RES
F808	5135	V3:	; NOT_HIGH_RES
F808 D3E3	5136	SHL BX,CL	; MULTIPLY *16 FOR HIGH RES
	5137		
	5138	1----- DETERMINE ALPHA CHAR POSITION	
	5139		
F80A 8AD4	5140	MOV DL,AH	; COLUMN VALUE FOR RETURN
F80C 8AF0	5141	MOV DH,AL	; ROW VALUE
F80E D0EE	5142	SHR DH,1	; DIVIDE BY 4
F810 D0EE	5143	SHR DH,1	; FOR VALUE IN 0-24 RANGE
F812 EB12	5144	JMP SHORT V5	; LIGHT_PEN_RETURN_SET
	5145		
	5146	1----- ALPHA MODE ON LIGHT PEN	
	5147		
F814	5148	V4:	; ALPHA_PEN
F814 F6364A00	5149	DIV BYTE PTR CRT_COLS	; DETERMINE ROW,COLUMN VALUE
F818 8AF0	5150	MOV DH,AL	; ROWS TO DH
F81A 8AD4	5151	MOV DL,AH	; COLS TO DL
F81C D2E0	5152	SAL AL,CL	; MULTIPLY ROWS * 8
F81E 8AE8	5153	MOV CH,AL	; GET RASTER VALUE TO RETURN REG
F820 8ADC	5154	MOV BL,AH	; COLUMN VALUE
F822 32FF	5155	XOR BH,BH	; TO BX
F824 D3E3	5156	SAL BX,CL	
F826	5157	V5:	; LIGHT_PEN_RETURN_SET
F826 B401	5158	MOV AH,1	; INDICATE EVERTHING SET
F828	5159	V6:	; LIGHT_PEN_RETURN
F828 52	5160	PUSH DX	; SAVE RETURN VALUE (IN CASE)
F829 8B166300	5161	MOV DX,ADDR_6845	; GET BASE ADDRESS
F82D 83C207	5162	ADD DX,7	; POINT TO RESET PARAM
F830 EE	5163	OUT DX,AL	; ADDRESS, NOT DATA, IS IMPORTANT
F831 5A	5164	POP DX	; RECOVER VALUE
F832	5165	V7:	; RETURN_NO_RESET
F832 5F	5166	POP DI	
F833 5E	5167	POP SI	
F834 1F	5168	POP DS	; DISCARD SAVED BX,CX,DX
F835 1F	5169	POP DS	
F836 1F	5170	POP DS	
	5171		
F837 1F	5172	POP DS	
F838 07	5173	POP ES	

```

LOC OBJ          LINE   SOURCE
F039 CF          5174          IRET
5175          READ_LPEN      ENDP
5176
5177          |--- INT 12 -----
5178          | MEMORY_SIZE_DET
5179          | THIS ROUTINE DETERMINES THE AMOUNT OF MEMORY IN THE SYSTEM
5180          | AS REPRESENTED BY THE SWITCHES ON THE PLANAR. NOTE THAT THE
5181          | SYSTEM MAY NOT BE ABLE TO USE I/O MEMORY UNLESS THERE IS A FULL
5182          | COMPLEMENT OF 64K BYTES ON THE PLANAR.
5183          | INPUT
5184          | NO REGISTERS
5185          | THE MEMORY_SIZE VARIABLE IS SET DURING POWER ON DIAGNOSTICS
5186          | ACCORDING TO THE FOLLOWING HARDWARE ASSUMPTIONS:
5187          | PORT 60 BITS 3,2 = 00 - 16K BASE RAM
5188          |                               01 - 32K BASE RAM
5189          |                               10 - 48K BASE RAM
5190          |                               11 - 64K BASE RAM
5191          | PORT 62 BITS 3-0 INDICATE AMOUNT OF I/O RAM IN 32K INCREMENTS
5192          | E.G., 0000 - NO RAM IN I/O CHANNEL
5193          |                               0010 - 64K RAM IN I/O CHANNEL, ETC.
5194          | OUTPUT
5195          | (AX) = NUMBER OF CONTIGUOUS 1K BLOCKS OF MEMORY
5196          |-----
5197          | ASSUME CS:CODE,DS:DATA
5198          | ORG 0F841H
5199          MEMORY_SIZE_DET PROC FAR
5200          STI                                ; INTERRUPTS BACK ON
5201          PUSH DS                            ; SAVE SEGMENT
5202          CALL DDS
5203          MOV AX,MEMORY_SIZE                ; GET VALUE
5204          POP DS                             ; RECOVER SEGMENT
5205          IRET                              ; RETURN TO CALLER
5206          MEMORY_SIZE_DET ENDP
5207
5208          |--- INT 11 -----
5209          | EQUIPMENT DETERMINATION
5210          | THIS ROUTINE ATTEMPTS TO DETERMINE WHAT OPTIONAL
5211          | DEVICES ARE ATTACHED TO THE SYSTEM.
5212          | INPUT
5213          | NO REGISTERS
5214          | THE EQUIP_FLAG VARIABLE IS SET DURING THE POWER ON
5215          | DIAGNOSTICS USING THE FOLLOWING HARDWARE ASSUMPTIONS:
5216          | PORT 60 = LOW ORDER BYTE OF EQUIPMENT
5217          | PORT 3FA = INTERRUPT ID REGISTER OF 8250
5218          | BITS 7-3 ARE ALWAYS 0
5219          | PORT 378 = OUTPUT PORT OF PRINTER -- 8255 PORT THAT
5220          | CAN BE READ AS WELL AS WRITTEN
5221          | OUTPUT
5222          | (AX) IS SET, BIT SIGNIFICANT, TO INDICATE ATTACHED I/O
5223          | BIT 15,14 = NUMBER OF PRINTERS ATTACHED
5224          | BIT 13 NOT USED
5225          | BIT 12 = GAME I/O ATTACHED
5226          | BIT 11,10,9 = NUMBER OF RS232 CARDS ATTACHED
5227          | BIT 8 UNUSED
5228          | BIT 7,6 = NUMBER OF DISKETTE DRIVES
5229          | 00=1, 01=2, 10=3, 11=4 ONLY IF BIT 0 = 1
5230          | BIT 5,4 = INITIAL VIDEO MODE
5231          | 00 - UNUSED
5232          | 01 - 40X25 BW USING COLOR CARD
5233          | 10 - 80X25 BW USING COLOR CARD
5234          | 11 - 80X25 BW USING BW CARD
5235          | BIT 3,2 = PLANAR RAM SIZE (00=16K,01=32K,10=48K,11=64K)
5236          | BIT 1 NOT USED
5237          | BIT 0 = IPL FROM DISKETTE -- THIS BIT INDICATES THAT
5238          | THERE ARE DISKETTE DRIVES ON THE SYSTEM
5239          |
5240          | NO OTHER REGISTERS AFFECTED
5241          |-----
5242          | ASSUME CS:CODE,DS:DATA
5243          | ORG 0F84DH
5244          EQUIPMENT PROC FAR
5245          STI                                ; INTERRUPTS BACK ON
5246          PUSH DS                            ; SAVE SEGMENT REGISTER
5247          CALL DDS
5248          MOV AX,EQUIP_FLAG                ; GET THE CURRENT SETTINGS
5249          POP DS                             ; RECOVER SEGMENT
5250          IRET                              ; RETURN TO CALLER

```

```

LOC OBJ          LINE   SOURCE
5251  EQUIPMENT      ENDP
5252
5253  ;--- INT 15 -----
5254  ; CASSETTE I/O
5255  ; (AH) = 0  TURN CASSETTE MOTOR ON
5256  ; (AH) = 1  TURN CASSETTE MOTOR OFF
5257  ; (AH) = 2  READ 1 OR MORE 256 BYTE BLOCKS FROM CASSETTE
5258  ; (ES,BX) = POINTER TO DATA BUFFER
5259  ; (CX) = COUNT OF BYTES TO READ
5260  ; ON EXIT
5261  ; (ES,BX) = POINTER TO LAST BYTE READ + 1
5262  ; (DX) = COUNT OF BYTES ACTUALLY READ
5263  ; (CY) = 0 IF NO ERROR OCCURRED
5264  ; = 1 IF ERROR OCCURRED
5265  ; (AH) = ERROR RETURN IF (CY)= 1
5266  ; = 01 IF CRC ERROR WAS DETECTED
5267  ; = 02 IF DATA TRANSITIONS ARE LOST
5268  ; = 04 IF NO DATA WAS FOUND
5269  ; (AH) = 3  WRITE 1 OR MORE 256 BYTE BLOCKS TO CASSETTE
5270  ; (ES,BX) = POINTER TO DATA BUFFER
5271  ; (CX) = COUNT OF BYTES TO WRITE
5272  ; ON EXIT
5273  ; (EX,BX) = POINTER TO LAST BYTE WRITTEN + 1
5274  ; (CX) = 0
5275  ; (AH) = ANY OTHER THAN ABOVE VALUES CAUSES (CY)= 1
5276  ; AND (AH)= 80 TO BE RETURNED (INVALID COMMAND).
5277  ;-----
5278  ASSUME DS:DATA,ES:NOTHING,SS:NOTHING,CS:CODE
F859  F859          5279  ORG 0F859H
F859  F859 FB       5280  CASSETTE_IO PROC FAR
F85A  F85A 1E       5281  STI ; INTERRUPTS BACK ON
F85B  F85B E8E006  5282  PUSH DS ; ESTABLISH ADDRESSING TO DATA
F85E  F85E 802671007F 5283  CALL DDS
F863  F863 E80400  5284  AND BIOS_BREAK, 7FH ; MAKE SURE BREAK FLAG IS OFF
F866  F866 1F       5285  CALL M1 ; CASSETTE_IO_CONT
F867  F867 CA0200  5286  POP DS
5287  RET 2 ; INTERRUPT RETURN
F86A  F86A         5288  CASSETTE_IO ENDP
5289  M1 PROC NEAR
5290  ;-----
5291  ; PURPOSE:
5292  ; TO CALL APPROPRIATE ROUTINE DEPENDING ON REG AH
5293  ;
5294  ; AH ROUTINE
5295  ;-----
5296  ; 0 MOTOR ON
5297  ; 1 MOTOR OFF
5298  ; 2 READ CASSETTE BLOCK
5299  ; 3 WRITE CASSETTE BLOCK
5300  ;-----
F86A  F86A 0AE4     5301  OR AH,AH ; TURN ON MOTOR?
F86C  F86C 7413     5302  JZ MOTOR_ON ; YES, DO IT
F86E  F86E FECC     5303  DEC AH ; TURN OFF MOTOR?
F870  F870 7418     5304  JZ MOTOR_OFF ; YES, DO IT
F872  F872 FECC     5305  DEC AH ; READ CASSETTE BLOCK?
F874  F874 1A       5306  JZ READ_BLOCK ; YES, DO IT
F876  F876 FECC     5307  DEC AH ; WRITE CASSETTE BLOCK?
F878  F878 7503     5308  JNZ M2 ; NOT_DEFINED
F87A  F87A E92401  5309  JMP WRITE_BLOCK ; YES, DO IT
F87D  F87D         5310  M2: ; COMMAND NOT DEFINED
F87D  F87D B480     5311  MOV AH,080H ; ERROR, UNDEFINED OPERATION
F87F  F87F F9       5312  STC ; ERROR FLAG
F880  F880 C3       5313  RET
5314  M1 ENDP
F881  F881         5315  MOTOR_ON PROC NEAR
5316  ;-----
5317  ; PURPOSE:
5318  ; TO TURN ON CASSETTE MOTOR
5319  ;-----
F881  F881 E461     5320  IN AL,PORT_B ; READ CASSETTE OUTPUT
F883  F883 24F7     5321  AND AL,NOT 08H ; CLEAR BIT TO TURN ON MOTOR
F885  F885         5322  M3:
F885  F885 E661     5323  OUT PORT_B,AL ; WRITE IT OUT
F887  F887 2AE4     5324  SUB AH,AH ; CLEAR AH
F889  F889 C3       5325  RET
5326  MOTOR_ON ENDP
F88A  F88A         5327  MOTOR_OFF PROC NEAR

```

LOC OBJ

LINE SOURCE

```

5328 ;-----
5329 ; PURPOSE: ;
5330 ; TO TURN CASSETTE MOTOR OFF ;
5331 ;-----
F88A E461 5332 IN AL,PORT_B ; READ CASSETTE OUTPUT
F88C 0C08 5333 OR AL,08H ; SET BIT TO TURN OFF
F88E EBF5 5334 JMP W3 ; WRITE IT, CLEAR ERROR, RETURN
5335 MOTOR_OFF ENDP
F890 5336 READ_BLOCK PROC NEAR
5337 ;-----
5338 ; PURPOSE: ;
5339 ; TO READ 1 OR MORE 256 BYTE BLOCKS FROM CASSETTE ;
5340 ; ;
5341 ; ON ENTRY: ;
5342 ; ES IS SEGMENT FOR MEMORY BUFFER (FOR COMPACT CODE) ;
5343 ; BX POINTS TO START OF MEMORY BUFFER ;
5344 ; CX CONTAINS NUMBER OF BYTES TO READ ;
5345 ; ON EXIT: ;
5346 ; BX POINTS 1 BYTE PAST LAST BYTE PUT IN MEM ;
5347 ; CX CONTAINS DECREMENTED BYTE COUNT ;
5348 ; DX CONTAINS NUMBER OF BYTES ACTUALLY READ ;
5349 ; ;
5350 ; CARRY FLAG IS CLEAR IF NO ERROR DETECTED ;
5351 ; CARRY FLAG IS SET IF CRC ERROR DETECTED ;
5352 ;-----
F890 53 5353 PUSH BX ; SAVE BX
F891 51 5354 PUSH CX ; SAVE CX
F892 56 5355 PUSH SI ; SAVE SI
F893 BE0700 5356 MOV SI, 7 ; SET UP RETRY COUNT FOR LEADER
F894 E8F01 5357 CALL BEGIN_OP ; BEGIN BY STARTING MOTOR
F899 5358 W4: ; SEARCH FOR LEADER
F899 E462 5359 IN AL,PORT_C ; GET INITIAL VALUE
F89B 2410 5360 AND AL,010H ; MASK OFF EXTRANEOUS BITS
F89D A26B00 5361 MOV LAST_VAL,AL ; SAVE IN LOC LAST_VAL
F8A0 BA7A3F 5362 MOV DX,16250 ; # OF TRANSITIONS TO LOOK FOR
F8A3 5363 M5: ; WAIT_FOR_EDGE
F8A3 F066710080 5364 TEST BIOS_BREAK, 80H ; CHECK FOR BREAK KEY
F8A8 7503 5365 JNZ W6A ; JUMP IF NO BREAK KEY
5366 ; JUMP IF BREAK KEY HIT
F8AA 5367 W6:
F8AA 4A 5368 DEC DX
F8AB 7503 5369 JNZ W7 ; JUMP IF BEGINNING OF LEADER
F8AD 5370 W6A:
F8AD E98400 5371 JMP W17 ; JUMP IF NO LEADER FOUND
F8B0 5372 W7:
F8B0 E8C600 5373 CALL READ_HALF_BIT ; IGNORE FIRST EDGE
F8B3 E3EE 5374 JCXZ W5 ; JUMP IF NO EDGE DETECTED
F8B5 BA7803 5375 MOV DX,0378H ; CHECK FOR HALF BITS
F8B6 B90002 5376 MOV CX,200H ; MUST HAVE AT LEAST THIS MANY ONE SIZE
5377 ; PULSES BEFORE CHCKNG FOR SYNC BIT (0)
F8BB E421 5378 IN AL, 021H ; INTERRUPT MASK REGISTER
F8BD 0C01 5379 OR AL,1 ; DISABLE TIMER INTERRUPTS
F8BF E621 5380 OUT 021H, AL
F8C1 5381 W8: ; SEARCH-LDR
F8C1 F066710080 5382 TEST BIOS_BREAK, 80H ; CHECK FOR BREAK KEY
F8C6 756C 5383 JNZ W17 ; JUMP IF BREAK KEY HIT
F8C8 51 5384 PUSH CX ; SAVE REG CX
F8C9 E8AD00 5385 CALL READ_HALF_BIT ; GET PULSE WIDTH
F8CC 0BC9 5386 OR CX, CX ; CHECK FOR TRANSITION
F8CE 59 5387 POP CX ; RESTORE ONE BIT COUNTER
F8CF 74C8 5388 JZ W4 ; JUMP IF NO TRANSITION
F8D1 38D3 5389 CMP DX,BX ; CHECK PULSE WIDTH
F8D3 E304 5390 JCXZ W9 ; IF CX=0 THEN HE CAN LOOK
5391 ; FOR SYNC BIT (0)
F8D5 73C2 5392 JNC W4 ; JUMP IF ZERO BIT (NOT GOOD LEADER)
F8D7 E2E8 5393 LOOP W8 ; DEC CX AND READ ANOTHER HALF ONE BIT
F8D9 5394 W9: ; F1ND-SYNC
F8D9 72E6 5395 JC W8 ; JUMP IF ONE BIT (STILL LEADER)
5396
5397 ;----- A SYNCH BIT HAS BEEN FOUND. READ SYN CHARACTER:
5398
F8DB E89B00 5399 CALL READ_HALF_BIT ; SKIP OTHER HALF OF SYNC BIT (0)
F8DE E86A00 5400 CALL READ_BYTE ; READ SYN BYTE
F8E1 3C16 5401 CMP AL, 16H ; SYNCHRONIZATION CHARACTER
F8E3 7549 5402 JNE W16 ; JUMP IF BAD LEADER FOUND.
5403
5404 ;----- GOOD CRC SO READ DATA BLOCK(S)

```


LOC OBJ	LINE	SOURCE	
	5405		
F8E5 5E	5406	POP SI	; RESTORE REGS
F8E6 59	5407	POP CX	
F8E7 5B	5408	POP BX	
	5409		-----
	5410	; READ 1 OR MORE 256 BYTE BLOCKS FROM CASSETTE	:
	5411	;	:
	5412	; ON ENTRY:	:
	5413	; ES IS SEGMENT FOR MEMORY BUFFER (FOR COMPACT CODE)	:
	5414	; BX POINTS TO START OF MEMORY BUFFER	:
	5415	; CX CONTAINS NUMBER OF BYTES TO READ	:
	5416	; ON EXIT:	:
	5417	; BX POINTS 1 BYTE PAST LAST BYTE PUT IN MEM	:
	5418	; CX CONTAINS DECREMENTED BYTE COUNT	:
	5419	; DX CONTAINS NUMBER OF BYTES ACTUALLY READ	:
	5420	;	-----
F8E8 51	5421	PUSH CX	; SAVE BYTE COUNT
F8E9	5422	W10:	; COME HERE BEFORE EACH
	5423		; 256 BYTE BLOCK IS READ
F8E9 C7066900FFFF	5424	MOV CRC_REG,0FFFFH	; INIT CRC REG
F8EF BA0001	5425	MOV DX,256	; SET DX TO DATA BLOCK SIZE
F8F2	5426	W11:	; RD_BLK
F8F2 F606710080	5427	TEST BIOS_BREAK, 80H	; CHECK FOR BREAK KEY
F8F7 7523	5428	JNZ W13	; JUMP IF BREAK KEY HIT
F8F9 E84F00	5429	CALL READ_BYTE	; READ BYTE FROM CASSETTE
F8FC 721E	5430	JC W13	; CY SET INDICATES NO DATA TRANSITIONS
F8FE E305	5431	JCXZ W12	; IF WE'VE ALREADY REACHED
	5432		; END OF MEMORY BUFFER
	5433		; SKIP REST OF BLOCK
F900 268807	5434	MOV ES:[BX],AL	; STORE DATA BYTE AT BYTE PTR
F903 43	5435	INC BX	; INC BUFFER PTR
F904 49	5436	DEC CX	; DEC BYTE COUNTER
F905	5437	W12:	; LOOP UNTIL DATA BLOCK HAS BEEN
	5438		; READ FROM CASSETTE.
F905 4A	5439	DEC DX	; DEC BLOCK CNT
F906 7FEA	5440	JG W11	; RD_BLK
F908 EB4000	5441	CALL READ_BYTE	; NOW READ TWO CRC BYTES
F90B E83000	5442	CALL READ_BYTE	
F90E 2AE4	5443	SUB AH,AH	; CLEAR AH
F910 813E69000FID	5444	CMC CRC_REG,100FH	; IS THE CRC CORRECT
F916 7506	5445	JNE W14	; IF NOT EQUAL CRC IS BAD
F918 E306	5446	JCXZ W15	; IF BYTE COUNT IS ZERO
	5447		; THEN WE HAVE READ ENOUGH
	5448		; SO WE WILL EXIT
F91A EB0D	5449	JMP W10	; STILL MORE, SO READ ANOTHER BLOCK
F91C	5450	W13:	; MISSING-DATA
	5451		; NO DATA TRANSITIONS SO
F91C B401	5452	MOV AH,01H	; SET AH=02 TO INDICATE
	5453		; DATA TIMEOUT
F91E	5454	W14:	; BAD-CRC
F91E FEC4	5455	INC AH	; EXIT EARLY ON ERROR
	5456		; SET AH=01 TO INDICATE CRC ERROR
F920	5457	W15:	; RD-BLK-EX
F920 5A	5458	POP DX	; CALCULATE COUNT OF
F921 2B01	5459	SUB DX,CX	; DATA BYTES ACTUALLY READ
	5460		; RETURN COUNT IN REG DX
F923 50	5461	PUSH AX	; SAVE AX (RET CODE)
F924 F6C490	5462	TEST AH, 90H	; CHECK FOR ERRORS
F927 7513	5463	JNZ W18	; JUMP IF ERROR DETECTED
F929 E81F00	5464	CALL READ_BYTE	; READ TRAILER
F92C EB0E	5465	JMP SHORT W18	; SKIP TO TURN OFF MOTOR
F92E	5466	W16:	; BAD-LEADER
F92E 4E	5467	DEC SI	; CHECK RETRIES
F92F 7403	5468	JZ W17	; JUMP IF TOO MANY RETRIES
F931 E965FF	5469	JMP W4	; JUMP IF NOT TOO MANY RETRIES
F934	5470	W17:	; NO VALID DATA FOUND
	5471		
	5472	;---- NO DATA FROM CASSETTE ERROR, I.E. TIMEOUT	
	5473		
F934 5E	5474	POP SI	; RESTORE REGS
F935 59	5475	POP CX	; RESTORE REGS
F936 5B	5476	POP BX	
F937 2B02	5477	SUB DX,DX	; ZERO NUMBER OF BYTES READ
F939 B404	5478	MOV AH,04H	; TIME OUT ERROR (NO LEADER)
F93B 50	5479	PUSH AX	
F93C	5480	W18:	; HOT-OFF

LOC OBJ

LINE SOURCE

```

F93C E421      5481      IN      AL, 021H      ; RE_ENABLE INTERRUPTS
F93E 24FE      5482      AND     AL, 0FFH- 1
F940 E621      5483      OUT     021H, AL
F942 E845FF    5484      CALL   MOTOR_OFF    ; TURN OFF MOTOR
F945 58        5485      POP     AX           ; RESTORE RETURN CODE
F946 80FC01    5486      CHP    AH,01H       ; SET CARRY IF ERROR (AH>0)
F949 F5        5487      CMC
F94A C3        5488      RET           ; FINISHED
5489      READ_BLOCK      ENDP
5490      ;-----
5491      ; PURPOSE:      :
5492      ; TO READ A BYTE FROM CASSETTE :
5493      ; ON EXIT:      :
5494      ; REG AL CONTAINS READ DATA BYTE :
5495      ;-----
F94B          5496      READ_BYTE      PROC   NEAR
F94B 53        5497      PUSH    BX           ; SAVE REGS BX,CX
F94C 51        5498      PUSH    CX
F94D B108     5499      MOV     CL,8H       ; SET BIT COUNTER FOR 8 BITS
F94F          5500      W19:          ; BYTE-ASM
F94F 51        5501      PUSH    CX           ; SAVE CX
5502      ;-----
5503      ; READ DATA BIT FROM CASSETTE :
5504      ;-----
F950 E82600    5505      CALL   READ_HALF_BIT ; READ ONE PULSE
F953 E320     5506      JCXZ   W21          ; IF CX=0 THEN TIMEOUT
5507      ; BECAUSE OF NO DATA TRANSITIONS
F955 53        5508      PUSH    BX           ; SAVE 1ST HALF BIT'S
5509      ; PULSE WIDTH (IN BX)
F956 E82000    5510      CALL   READ_HALF_BIT ; READ COMPLEMENTARY PULSE
F959 58        5511      POP     AX           ; COMPUTE DATA BIT
F95A E319     5512      JCXZ   W21          ; IF CX=0 THEN TIMEOUT DUE TO
5513      ; NO DATA TRANSITIONS
F95C 0308     5514      ADD     BX,AX        ; PERIOD
F95E 81FBF006  5515      CHP    BX, 06F0H    ; CHECK FOR ZERO BIT
F962 F5        5516      CMC     ; CARRY IS SET IF ONE BIT
F963 9F        5517      LAHF   ; SAVE CARRY IN AH
F964 59        5518      POP     CX           ; RESTORE CX
5519      ; NOTE:
5520      ; MS BIT OF BYTE IS READ FIRST.
5521      ; REG CH IS SHIFTED LEFT WITH
5522      ; CARRY BEING INSERTED INTO LS
5523      ; BIT OF CH.
5524      ; AFTER ALL 8 BITS HAVE BEEN
5525      ; READ, THE MS BIT OF THE DATA BYTE
5526      ; WILL BE IN THE MS BIT OF REG CH
F965 D0D5     5527      RCL    CH,1         ; ROTATE REG CH LEFT WITH CARRY TO
5528      ; LS BIT OF REG CH
F967 9E        5529      SAHF   ; RESTORE CARRY FOR CRC ROUTINE
F968 E8D900    5530      CALL   CRC_GEN      ; GENERATE CRC FOR BIT
F96B FEC9     5531      DEC     CL           ; LOOP TILL ALL 8 BITS OF DATA
5532      ; ASSEMBLED IN REG CH
F96D 75E0     5533      JNZ    W19          ; BYTE_ASM
F96F 8AC5     5534      MOV     AL,CH       ; RETURN DATA BYTE IN REG AL
F971 F8        5535      CLC
F972          5536      W20:          ; RD-BYT-EX
F972 59        5537      POP     CX           ; RESTORE REGS CX,BX
F973 5B        5538      POP     BX
F974 C3        5539      RET           ; FINISHED
F975          5540      W21:          ; NO-DATA
F975 59        5541      POP     CX           ; RESTORE CX
F976 F9        5542      STC     ; INDICATE ERROR
F977 EBF9     5543      JMP     W20          ; RD_BYT_EX
5544      READ_BYTE      ENDP
5545      ;-----
5546      ; PURPOSE:      :
5547      ; TO COMPUTE TIME TILL NEXT DATA :
5548      ; TRANSITION (EDGE) :
5549      ; ON ENTRY:      :
5550      ; EDGE_CNT CONTAINS LAST EDGE COUNT :
5551      ; ON EXIT:      :
5552      ; AX CONTAINS OLD LAST EDGE COUNT :
5553      ; BX CONTAINS PULSE WIDTH (HALF BIT) :
5554      ;-----
F979          5555      READ_HALF_BIT  PROC   NEAR
F979 B96400     5556      MOV     CX, 100      ; SET TIME TO WAIT FOR BIT
F97C 8A268000  5557      MOV     AH, LAST_VAL ; GET PRESENT INPUT VALUE

```

LOC OBJ	LINE	SOURCE	
F980	5558	M22:	; RD-H-BIT
F980 E462	5559	IN AL,PORT_C	; INPUT DATA BIT
F982 2410	5560	AND AL,010H	; MASK OFF EXTRANEIOUS BITS
F984 3AC4	5561	CMP AL,AH	; SAME AS BEFORE?
F986 E1F8	5562	LOOPE M22	; LOOP TILL IT CHANGES
F988 A26800	5563	MOV LAST_VAL,AL	; UPDATE LAST_VAL WITH NEW VALUE
F98B B000	5564	MOV AL,0	; READ TIMER'S COUNTER COMMAND
F98D E643	5565	OUT TIM_CTL,AL	; LATCH COUNTER
F98F 8B1E6700	5566	MOV BX,EDGE_CNT	; BX GETS LAST EDGE COUNT
F993 E440	5567	IN AL,TIMER0	; GET LS BYTE
F995 8AE0	5568	MOV AH,AL	; SAVE IN AH
F997 E440	5569	IN AL,TIMER0	; GET HS BYTE
F999 86C4	5570	XCHG AL,AH	; XCHG AL,AH
F99B 28D8	5571	SUB BX,AX	; SET BX EQUAL TO HALF BIT PERIOD
F99D A36700	5572	MOV EDGE_CNT,AX	; UPDATE EDGE COUNT;
F9A0 C3	5573	RET	
	5574	READ_HALF_BIT ENDP	
	5575	;	
	5576	;	-----
	5577	; PURPOSE	;
	5577	; WRITE 1 OR MORE 256 BYTE BLOCKS TO CASSETTE.	;
	5578	; THE DATA IS PADDED TO FILL OUT THE LAST 256 BYTE BLOCK.	;
	5579	; ON ENTRY:	;
	5580	; BX POINTS TO MEMORY BUFFER ADDRESS	;
	5581	; CX CONTAINS NUMBER OF BYTES TO WRITE	;
	5582	; ON EXIT:	;
	5583	; BX POINTS 1 BYTE PAST LAST BYTE WRITTEN TO CASSETTE	;
	5584	; CX IS ZERO	;
	5585	;	-----
F9A1	5586	WRITE_BLOCK PROC NEAR	
F9A1 53	5587	PUSH BX	
F9A2 51	5588	PUSH CX	
F9A3 E461	5589	IN AL,PORT_B	; DISABLE SPEAKER
F9A5 24FD	5590	AND AL,NOT 02H	
F9A7 0C01	5591	OR AL, 01H	; ENABLE TIMER
F9A9 E661	5592	OUT PORT_B,AL	
F9AB B0B6	5593	MOV AL,0B6H	; SET UP TIMER -- MODE 3 SQUARE WAVE
F9AD E643	5594	OUT TIM_CTL,AL	
F9AF E8A600	5595	CALL BEGIN_OP	; START MOTOR AND DELAY
F9B2 B8A004	5596	MOV AX,1184	; SET NORMAL BIT SIZE
F9B5 E88500	5597	CALL M31	; SET_TIMER
F9B8 B90008	5598	MOV CX,0800H	; SET CX FOR LEADER BYTE COUNT
F9BB	5599	M23:	; WRITE LEADER
F9BB F9	5600	STC	; WRITE ONE BITS
F9BC E86800	5601	CALL WRITE_BIT	
F9BF E2FA	5602	LOOP M23	; LOOP 'TIL LEADER IS WRITTEN
F9C1 F8	5603	CLC	; WRITE SYNC BIT (0)
F9C2 E86200	5604	CALL WRITE_BIT	
F9C5 59	5605	POP CX	; RESTORE REGS CX,BX
F9C6 5B	5606	POP BX	
F9C7 B016	5607	MOV AL,16H	; WRITE SYN CHARACTER
F9C9 E84400	5608	CALL WRITE_BYTE	
	5609	;	-----
	5610	; PURPOSE	;
	5611	; WRITE 1 OR MORE 256 BYTE BLOCKS TO CASSETTE	;
	5612	; ON ENTRY:	;
	5613	; BX POINTS TO MEMORY BUFFER ADDRESS	;
	5614	; CONTAINS NUMBER OF BYTES TO WRITE	;
	5615	; ON EXIT:	;
	5616	; BX POINTS 1 BYTE PAST LAST BYTE WRITTEN TO CASSETTE	;
	5617	; CX IS ZERO	;
	5618	;	-----
F9CC	5619	NR_BLOCK:	
F9CC C706690FFFFF	5620	MOV CRC_REG,OFFFH	; INIT CRC
F9D2 BA0001	5621	MOV DX,256	; FOR 256 BYTES
F9D5	5622	M24:	; NR-BLK
F9D5 268A07	5623	MOV AL,ES:[BX]	; READ BYTE FROM MEM
F9D8 E83500	5624	CALL WRITE_BYTE	; WRITE IT TO CASSETTE
F9DB E302	5625	JCXZ M25	; UNLESS CX=0, ADVANCE PTRS & DEC COUNT
F9DD 43	5626	INC BX	; INC BUFFER POINTER
F9DE 49	5627	DEC CX	; DEC BYTE COUNTER
F9DF	5628	M25:	; SKIP-ADV
F9DF 4A	5629	DEC DX	; DEC BLOCK CNT
F9E0 7FF3	5630	JG M24	; LOOP TILL 256 BYTE BLOCK
	5631	;	; IS WRITTEN TO TAPE
	5632	;	-----
	5633	; WRITE CRC	;
	5634	; WRITE 1'S COMPLEMENT OF CRC REG TO CASSETTE	;

LOC OBJ	LINE	SOURCE
	5635	; WHICH IS CHECKED FOR CORRECTNESS WHEN THE BLOCK IS READ :
	5636	; REG AX IS MODIFIED :
	5637	-----
F9E2 A16900	5638	MOV AX,CRC_REG ; WRITE THE ONE'S COMPLEMENT OF THE
	5639	; TWO BYTE CRC TO TAPE
F9E5 F7D0	5640	NOT AX ; FOR 1'S COMPLEMENT
F9E7 50	5641	PUSH AX ; SAVE IT
F9E8 86E0	5642	XCHG AH,AL ; WRITE MS BYTE FIRST
F9EA E82300	5643	CALL WRITE_BYTE ; WRITE IT
F9ED 58	5644	POP AX ; GET IT BACK
F9EE E81F00	5645	CALL WRITE_BYTE ; NOW WRITE LS BYTE
F9F1 08C9	5646	OR CX,CX ; IS BYTE COUNT EXHAUSTED?
F9F3 75D7	5647	JNZ WR_BLOCK ; JUMP IF NOT DONE YET
F9F5 51	5648	PUSH CX ; SAVE REG CX
F9F6 B92000	5649	MOV CX, 32 ; WRITE OUT TRAILER BITS
F9F9	5650	M26: ; TRAIL-LOOP
F9F9 F9	5651	STC
F9FA E82A00	5652	CALL WRITE_BIT
F9FD E2FA	5653	LOOP M26 ; WRITE UNTIL TRAILER WRITTEN
F9FF 59	5654	POP CX ; RESTORE REG CX
FA00 B0B0	5655	MOV AL, 0B0H ; TURN TIMER2 OFF
FA02 E643	5656	OUT T1H_CTL, AL
FA04 B80100	5657	MOV AX, 1
FA07 E83300	5658	CALL W31 ; SET_TIMER
FA0A E87DFE	5659	CALL MOTOR_OFF ; TURN MOTOR OFF
FA0D 2BC0	5660	SUB AX,AX ; NO ERRORS REPORTED ON WRITE OP
FA0F C3	5661	RET ; FINISHED
	5662	WRITE_BLOCK ENDP
	5663	-----
	5664	; WRITE A BYTE TO CASSETTE. :
	5665	; BYTE TO WRITE IS IN REG AL. :
	5666	-----
FA10	5667	WRITE_BYTE PROC NEAR
FA10 51	5668	PUSH CX ; SAVE REGS CX,AX
FA11 50	5669	PUSH AX
FA12 8AE8	5670	MOV CH,AL ; AL=BYTE TO WRITE.
	5671	; (MS BIT WRITTEN FIRST)
FA14 B108	5672	MOV CL,8 ; FOR 8 DATA BITS IN BYTE.
	5673	; NOTE: TWO EDGES PER BIT
FA16	5674	M27: ; DISASSEMBLE THE DATA BIT
FA16 D0D5	5675	RCL CH,1 ; ROTATE MS BIT INTO CARRY
FA18 9C	5676	PUSHF ; SAVE FLAGS.
	5677	; NOTE: DATA BIT IS IN CARRY
FA19 E80B00	5678	CALL WRITE_BIT ; WRITE DATA BIT
FA1C 9D	5679	POPF ; RESTORE CARRY FOR CRC CALC
FA1D E82400	5680	CALL CRC_GEN ; COMPUTE CRC ON DATA BIT
FA20 FEC9	5681	DEC CL ; LOOP TILL ALL 8 BITS DONE
FA22 75F2	5682	JNZ M27 ; JUMP IF NOT DONE YET
FA24 58	5683	POP AX ; RESTORE REGS AX,CX
FA25 59	5684	POP CX
FA26 C3	5685	RET ; WE ARE FINISHED
	5686	WRITE_BYTE ENDP
	5687	-----
	5688	; PURPOSE: :
	5689	; TO WRITE A DATA BIT TO CASSETTE :
	5690	; CARRY FLAG CONTAINS DATA BIT :
	5691	; I.E. IF SET DATA BIT IS A ONE :
	5692	; IF CLEAR DATA BIT IS A ZERO :
	5693	; :
	5694	; NOTE: TWO EDGES ARE WRITTEN PER BIT :
	5695	; ONE BIT HAS 500 USEC BETWEEN EDGES :
	5696	; FOR A 1000 USEC PERIOD (1 MILLISEC) :
	5697	; :
	5698	; ZERO BIT HAS 250 USEC BETWEEN EDGES :
	5699	; FOR A 500 USEC PERIOD (.5 MILLISEC) :
	5700	; CARRY FLAG IS DATA BIT :
	5701	-----
FA27	5702	WRITE_BIT PROC NEAR
	5703	; ASSUME IT'S A '1'
FA27 B8A004	5704	MOV AX,1184 ; SET AX TO NOMINAL ONE SIZE
FA2A 7203	5705	JC M28 ; JUMP IF ONE BIT
FA2C B85002	5706	MOV AX,592 ; NO, SET TO NOMINAL ZERO SIZE
FA2F	5707	M28: ; WRITE BIT-AX
FA2F 50	5708	PUSH AX ; WRITE BIT WITH PERIOD EQ TO VALUE AX
FA30	5709	M29: ;
FA30 E462	5710	IN AL,PORT_C ; INPUT TIMER_0 OUTPUT
FA32 2420	5711	AND AL,020H

```

LOC OBJ          LINE  SOURCE

FA34 74FA        5712      JZ      W29          ; LOOP TILL HIGH
FA36             5713      M30:
FA36 E462        5714      IN      AL,PORT_C   ; NOW WAIT TILL TIMER'S OUTPUT IS LOW
FA38 2420        5715      AND    AL,020H
FA3A 75FA        5716      JNZ    W30
5717
5718          ; RELOAD TIMER WITH PERIOD
FA3C 58          5719      POP    AX          ; FOR NEXT DATA BIT
FA3D             5720      M31:             ; RESTORE PERIOD COUNT
FA3D E642        5721      OUT    042H, AL    ; SET TIMER
FA3F 0AC4        5722      MOV    AL, AH      ; SET LOW BYTE OF TIMER 2
FA41 E642        5723      OUT    042H, AL    ; SET HIGH BYTE OF TIMER 2
FA43 C3          5724      RET
5725      WRITE_BIT      ENDP
5726      ;-----
5727      ; UPDATE CRC REGISTER WITH NEXT DATA BIT      :
5728      ; CRC IS USED TO DETECT READ ERRORS           :
5729      ; ASSUMES DATA BIT IS IN CARRY                :
5730      ; :
5731      ; REG AX IS MODIFIED                           :
5732      ; FLAGS ARE MODIFIED                           :
5733      ;-----
FA44             5734      CRC_GEN      PROC   NEAR
FA44 A16900       5735      MOV    AX,CRC_REG
5736
5737          ; THE FOLLOWING INSTRUCTIONS
5738          ; WILL SET THE OVERFLOW FLAG
5739          ; IF CARRY AND MS BIT OF CRC
5739          ; ARE UNEQUAL
FA47 D1D8        5740      RCR    AX,1
FA49 D1D0        5741      RCL    AX,1
FA4B F8          5742      CLC
FA4C 7104        5743      JND    W32         ; CLEAR CARRY
5744          ; SKIP IF NO OVERFLOW
5745          ; IF DATA BIT XORED WITH
5746          ; CRC REG BIT 15 IS ONE
FA4E 351008      5746      XOR    AX,0810H   ; THEN XOR CRC REG WITH 0801H
FA51 F9          5747      STC          ; SET CARRY
FA52             5748      M32:
FA52 D1D0        5749      RCL    AX,1       ; ROTATE CARRY (DATA BIT)
5750          ; INTO CRC REG
FA54 A36900      5751      MOV    CRC_REG,AX ; UPDATE CRC_REG
FA57 C3          5752      RET          ; FINISHED
5753      CRC_GEN      ENDP
5754
FA58             5755      BEGIN_OP    PROC   NEAR ; START TAPE AND DELAY
FA58 E826FE      5756      CALL   MOTOR_ON ;TURN ON MOTOR
FA5B B342        5757      MOV    BL,42H    ;DELAY FOR TAPE DRIVE
5758          ;TO GET UP TO SPEED (1/2 SEC)
FA5D             5759      M33:
FA5D B90007      5760      MOV    CX,700H  ;INNER LOOP= APPROX. 10 MILLISEC
FA60 E2FE        5761      M34: LOOP W34
FA62 FECB        5762      DEC    BL
FA64 75F7        5763      JNZ    W33
FA66 C3          5764      RET
5765      BEGIN_OP    ENDP
5766
FA67 20323031    5767      EI      DB      ' 201',13,10
FA6B 0D
FA6C 0A
5768
5769      ;-----
5770      ; CHARACTER GENERATOR GRAPHICS FOR 320X200 AND 640X200 GRAPHICS
5771      ;-----
FA6E             5772      ORG    0FA6EH
FA6E             5773      CRT_CHAR_GEN LABEL BYTE
FA6E 0000000000000000 5774      DB      000H,000H,000H,000H,000H,000H,000H,000H ; D_00
FA76 7E81A581BD99817E 5775      DB      07EH,081H,0A5H,081H,0BDH,099H,081H,07EH ; D_01
FA7E 7EFFDBFFC3E7FFF7E 5776      DB      07EH,0FFH,0DBH,0FFH,0C3H,0E7H,0FFH,07EH ; D_02
FA86 6CFEFEFE7C381000 5777      DB      06CH,0FEH,0FEH,0FEH,07CH,038H,010H,000H ; D_03
FA8E 10387CFE7C381000 5778      DB      010H,038H,07CH,0FEH,07CH,038H,010H,000H ; D_04
FA96 387C38FEFE7C387C 5779      DB      038H,07CH,038H,0FEH,0FEH,07CH,038H,07CH ; D_05
FA9E 1010387CFE7C387C 5780      DB      010H,010H,038H,07CH,0FEH,07CH,038H,07CH ; D_06
FAA6 0000183C3C180000 5781      DB      000H,000H,018H,03CH,03CH,018H,000H,000H ; D_07
FAAE FFFF7C3C3E7FFF 5782      DB      0FFH,0FFH,0E7H,0C3H,0C3H,0E7H,0FFH,0FFH ; D_08
FAB6 003C664242663C00 5783      DB      000H,03CH,066H,042H,042H,066H,03CH,000H ; D_09
FABE FFC3998BDB99C3FF 5784      DB      0FFH,0C3H,099H,0BDH,0BDH,099H,0C3H,0FFH ; D_0A
FAC6 0F0707DCCCCC78 5785      DB      00FH,007H,00FH,07DH,0CCH,0CCH,0CCH,078H ; D_0B
FACE 3C6666663C187E18 5786      DB      03CH,066H,066H,066H,03CH,018H,07EH,018H ; D_0C

```

LOC OBJ	LINE	SOURCE
FAD6	3F333F303070F0E0	5787 DB 03FH,033H,03FH,030H,030H,070H,0F0H,0E0H ; D_00
FADE	7F637F63637E6C00	5788 DB 07FH,063H,07FH,063H,063H,067H,056H,0C0H ; D_0E
FAE6	995A3ACE7E73C5A99	5789 DB 099H,05AH,03CH,0E7H,0E7H,03CH,05AH,099H ; D_0F
FAEE	80E0F8FE78E08000	5790 DB 080H,0E0H,0F8H,0FEH,0F8H,0E0H,080H,000H ; D_10
FAFE	020E3EFE3E0E0200	5791 DB 002H,00EH,03EH,0FEH,03EH,00EH,002H,000H ; D_11
FAFE	183C7E1818187E3C18	5792 DB 018H,03CH,07EH,018H,018H,07EH,03CH,018H ; D_12
FB06	6666666666006600	5793 DB 066H,066H,066H,066H,066H,000H,066H,000H ; D_13
FB0E	7FD8DB781B181B00	5794 DB 07FH,0DBH,0DBH,078H,018H,018H,018H,000H ; D_14
FB16	3E3386C638C7E000	5795 DB 03EH,063H,038H,06CH,06CH,038H,0CCH,078H ; D_15
FB1E	000000007E7E7E00	5796 DB 000H,000H,000H,000H,07EH,07EH,0CCH,000H ; D_16
FB26	183C7E187E3C18FF	5797 DB 018H,03CH,07EH,018H,07EH,03CH,018H,0FFH ; D_17
FB2E	183C7E1818181800	5798 DB 018H,03CH,07EH,018H,018H,018H,018H,000H ; D_18
FB36	181818187E3C1800	5799 DB 018H,018H,018H,018H,07EH,03CH,018H,000H ; D_19
FB3E	00180CFE0C180000	5800 DB 000H,018H,00CH,0FEH,00CH,018H,000H,000H ; D_1A
FB46	003060FE60300000	5801 DB 000H,030H,060H,0FEH,060H,030H,000H,000H ; D_1B
FB4E	0000CC0C0FE00000	5802 DB 000H,000H,0C0H,0C0H,0C0H,0FEH,000H,000H ; D_1C
FB56	002466FF6E240000	5803 DB 000H,024H,066H,0FFH,066H,024H,000H,000H ; D_1D
FB5E	00183C7EFFFF0000	5804 DB 000H,018H,03CH,07EH,0FFH,0F8H,000H,000H ; D_1E
FB66	00FFFF7E3C180000	5805 DB 000H,0FFH,0FFH,07EH,03CH,018H,000H,000H ; D_1F
FB6E	0000000000000000	5806 DB 000H,000H,000H,000H,000H,000H,000H,000H ; SP_D_20
FB76	3078783030003000	5807 DB 030H,078H,078H,030H,030H,000H,030H,000H ; I_D_21
FB7E	6C6C6C0000000000	5808 DB 06CH,06CH,06CH,000H,000H,000H,000H,000H ; " D_22
FB86	6C6CFE6CFE6C6C00	5809 DB 06CH,06CH,0FEH,06CH,0FEH,06CH,06CH,000H ; # D_23
FB8E	307CC0780CF83000	5810 DB 030H,07CH,0C0H,078H,0C0H,0F8H,030H,000H ; # D_24
FB96	00C6C1830666C000	5811 DB 000H,0C6H,0CCH,018H,030H,066H,0C6H,000H ; PER_CENT_D_25
FB9E	386C3876DCC76000	5812 DB 038H,06CH,038H,076H,076H,0DCH,0CCH,076H,000H ; # D_26
FBAA	6060C00000000000	5813 DB 060H,060H,0C0H,000H,000H,000H,000H,000H ; ' D_27
FBAE	1830606060301800	5814 DB 018H,030H,060H,060H,060H,030H,018H,000H ; (D_28
FBBA	6030181818306000	5815 DB 060H,030H,018H,018H,018H,030H,060H,000H ; I_D_29
FBBE	00663CF3C6400000	5816 DB 000H,066H,03CH,0FFH,03CH,066H,000H,000H ; # D_2A
FBCE	003030FC30300000	5817 DB 000H,030H,030H,0FCH,030H,030H,000H,000H ; + D_2B
FBCE	0000000000303060	5818 DB 000H,000H,000H,000H,000H,030H,030H,066H ; - D_2C
FBDE	000000F0C0000000	5819 DB 000H,000H,000H,0FCH,000H,030H,000H,000H ; - D_2D
FBDE	0000000000303000	5820 DB 000H,000H,000H,000H,000H,030H,030H,000H ; - D_2E
FBEE	060C183060C08000	5821 DB 066H,00CH,018H,030H,060H,0CCH,000H,000H ; / D_2F
FBEE	7CC6CEDEF6E67C00	5822 DB 07CH,0C6H,0CEH,0DEH,0F6H,0E6H,07CH,000H ; 0 D_30
FBF6	3070303030FC0000	5823 DB 030H,070H,030H,030H,030H,030H,0FCH,000H ; 1 D_31
FBFE	78CC0C3860CFFC00	5824 DB 078H,0CCH,0C0H,038H,060H,0CCH,0FCH,000H ; 2 D_32
FC06	78CC0C380CC78000	5825 DB 078H,0CCH,0C0H,038H,0CCH,0CCH,078H,000H ; 3 D_33
FC0E	1C36C6CCFE0C1E00	5826 DB 01CH,03CH,06CH,0CCH,0FEH,00CH,01EH,000H ; 4 D_34
FC16	FFC0F80C0CC78000	5827 DB 0FCH,0C0H,0F8H,00CH,0CCH,0CCH,078H,000H ; 5 D_35
FC1E	3860C0F8CC0C7800	5828 DB 038H,060H,0C0H,0F8H,0CCH,0CCH,030H,000H ; 6 D_36
FC26	FCC0C18303030000	5829 DB 0FCH,0CCH,00CH,018H,030H,030H,030H,000H ; 7 D_37
FC2E	78CC0C78CC0C7800	5830 DB 078H,0CCH,0CCH,078H,0CCH,0CCH,078H,000H ; 8 D_38
FC36	78CC0C7C0C187000	5831 DB 078H,0CCH,0CCH,07CH,0CCH,018H,070H,000H ; 9 D_39
FC3E	0030300000303000	5832 DB 000H,030H,030H,000H,000H,030H,030H,000H ; 1 D_3A
FC46	0030300000303060	5833 DB 000H,030H,030H,000H,000H,030H,030H,060H ; 1 D_3B
FC4E	183060C060301800	5834 DB 018H,030H,060H,0C0H,060H,030H,018H,000H ; 1 D_3C
FC56	0000F0C000F0C000	5835 DB 000H,000H,0FCH,000H,000H,0FCH,000H,000H ; - D_3D
FC5E	6030180C18306000	5836 DB 060H,030H,018H,00CH,018H,030H,060H,000H ; 1 D_3E
FC66	78CC0C1830003000	5837 DB 078H,0CCH,00CH,018H,030H,000H,030H,000H ; 1 D_3F
FC6E	7CC6DEDEE0C78000	5838 DB 07CH,0C6H,0DEH,0DEH,0DEH,0CCH,078H,000H ; 2 D_40
FC76	3078CC0CCFFCCCC000	5839 DB 030H,078H,0CCH,0CCH,0FCH,0CCH,0CCH,078H,000H ; A D_41
FC7E	FC66667C6666FC00	5840 DB 0FCH,066H,066H,07CH,066H,066H,0FCH,000H ; B D_42
FC86	3C66C0C0C0663C00	5841 DB 03CH,066H,0C0H,0C0H,0C0H,066H,03CH,000H ; C D_43
FC8E	F86C666666CF8000	5842 DB 0F8H,06CH,066H,066H,066H,06CH,0F8H,000H ; D D_44
FC96	FE6268786862F000	5843 DB 0FEH,062H,068H,078H,068H,062H,0FEH,000H ; E D_45
FC9E	FE6268786860F000	5844 DB 0FEH,062H,068H,078H,068H,060H,0F0H,000H ; F D_46
FCAA	3C66C0C0CE663E00	5845 DB 03CH,066H,0C0H,0C0H,0CEH,066H,03EH,000H ; G D_47
FCAE	CCCCCCCCFFCCCC0000	5846 DB 0CCH,0CCH,0CCH,0FCH,0CCH,0CCH,0CCH,000H ; H D_48
FCB6	7830303030307800	5847 DB 078H,030H,030H,030H,030H,030H,078H,000H ; I D_49
FCBE	1E0C0C0CCCC0C78000	5848 DB 01EH,00CH,00CH,00CH,0CCH,0CCH,0CCH,078H,000H ; J D_4A
FCCE	E66666786C6E6E00	5849 DB 066H,066H,06CH,078H,06CH,066H,0E6H,000H ; K D_4B
FCCE	F0606060626CF000	5850 DB 0F0H,060H,060H,060H,062H,066H,0FEH,000H ; L D_4C
FCDE	C6EEFFED6C6C6C00	5851 DB 0C6H,0EEH,0FEH,0FEH,0D6H,0C6H,0C6H,000H ; R D_4D
FCDE	C6E6F4DEECC6C600	5852 DB 0C6H,0E6H,0F6H,0DEH,0CEH,0C6H,0C6H,000H ; N D_4E
FC6E	386CC6C6C6C38000	5853 DB 038H,06CH,0C6H,0C6H,0C6H,06CH,038H,000H ; O D_4F
FCEE	FC66667C606F0000	5854 DB 0FCH,066H,066H,07CH,060H,060H,0F0H,000H ; P D_50
FCF6	78CCCC0C781C0000	5855 DB 078H,0CCH,0CCH,0CCH,0DCH,0DCH,078H,01CH,000H ; Q D_51
FCFE	FC66667C66C6E600	5856 DB 0FCH,066H,066H,07CH,06CH,066H,0E6H,0CCH ; R D_52
FD06	78CC0E701CC78000	5857 DB 078H,0CCH,0E0H,070H,01CH,0CCH,078H,000H ; S D_53
FD0E	FB430303030307800	5858 DB 0FCH,0B4H,030H,030H,030H,030H,078H,000H ; T D_54
FD16	CCCCCCCCCCCCFC0000	5859 DB 0CCH,0CCH,0CCH,0CCH,0CCH,0CCH,0FCH,000H ; U D_55
FD1E	CCCCCCCC78300000	5860 DB 0CCH,0CCH,0CCH,0CCH,0CCH,078H,030H,000H ; V D_56
FD26	C6C6A6D6FE6E0C00	5861 DB 0C6H,0C6H,0C6H,0D6H,0FEH,0EEH,0C6H,000H ; W D_57
FD2E	C6C6A638386CC000	5862 DB 0C6H,0C6H,06CH,038H,038H,06CH,0C6H,000H ; X D_58
FD36	7CCCC078303C7800	5863 DB 0CCH,0CCH,0CCH,078H,030H,030H,078H,000H ; Y D_59

```

FD3E FEC68C183266FE00 5864 DB OFEH,0C6H,08CH,018H,032H,066H,0FEH,000H ; Z D_5A
FD46 7860606060607800 5865 DB 078H,060H,060H,060H,060H,060H,078H,000H ; I D_5B
FD4E C06030180C060200 5866 DB 0C0H,060H,030H,018H,00CH,006H,002H,000H ; BACKSLASH D_5C
FD56 7818181818187800 5867 DB 078H,018H,018H,018H,018H,018H,078H,000H ; J D_5D
FD5E 10386CC600000000 5868 DB 010H,038H,06CH,0C6H,000H,000H,000H,000H ; CIRCUMFLEX D_5E
FD66 000000000000000FF 5869 DB 000H,000H,000H,000H,000H,000H,000H,0FFH ; _ D_5F
FD6E 3030180000000000 5870 DB 030H,030H,018H,000H,000H,000H,000H,000H ; ' D_60
FD76 0000780C7CCC7600 5871 DB 000H,000H,078H,00CH,07CH,0CCH,076H,000H ; LOWER CASE A D_61
FD7E E060607C6666DC00 5872 DB 0E0H,060H,060H,07CH,066H,066H,066H,000H ; L.C. B D_62
FD86 000078CC0CC7800 5873 DB 000H,000H,078H,0CCH,0C0H,0CCH,078H,000H ; L.C. C D_63
FD8E 1C0C0C7CCC7600 5874 DB 01CH,00CH,00CH,07CH,0CCH,0CCH,078H,000H ; L.C. D D_64
FD96 000078CC0CC7800 5875 DB 000H,000H,078H,0CCH,0FCH,0C0H,078H,000H ; L.C. E D_65
FD9E 386C60F06060F000 5876 DB 038H,06CH,060H,0F0H,060H,060H,0F0H,000H ; L.C. F D_66
FDA6 000076CCC7C0CF8 5877 DB 000H,000H,076H,0CCH,0CCH,07CH,0CCH,0F6H ; L.C. G D_67
FDAE E0606C766666E600 5878 DB 0E0H,060H,06CH,076H,066H,066H,0E6H,000H ; L.C. H D_68
FDB6 3000703030307800 5879 DB 030H,000H,070H,030H,030H,030H,078H,000H ; L.C. I D_69
FDBE 0C000C0C00000000 5880 DB 00CH,000H,00CH,00CH,00CH,00CH,00CH,078H ; L.C. J D_6A
FDC6 E060666C786CE600 5881 DB 0E0H,060H,066H,06CH,078H,06CH,066H,000H ; L.C. K D_6B
FDD6 7030303030307800 5882 DB 070H,030H,030H,030H,030H,030H,078H,000H ; L.C. L D_6C
FDE0 0000C0FEFFED6C60 5883 DB 000H,000H,0CCH,0FEH,0FEH,0D6H,0C6H,000H ; L.C. M D_6D
FDE2 0000F8CCCC0000 5884 DB 000H,000H,0F8H,0CCH,0CCH,0CCH,0CCH,000H ; L.C. N D_6E
FDE6 000078CCC7C07800 5885 DB 000H,000H,078H,0CCH,0CCH,0CCH,078H,000H ; L.C. O D_6F
FDEE 0000C6667C60F0 5886 DB 000H,000H,0DCH,066H,066H,07CH,060H,0F0H ; L.C. P D_70
FDF6 000076CCC7C0C1E 5887 DB 000H,000H,076H,0CCH,0CCH,07CH,00CH,01EH ; L.C. Q D_71
FDFE 0000C76666F000 5888 DB 000H,000H,0DCH,076H,066H,060H,0F0H,000H ; L.C. R D_72
FE06 00007C0780CF800 5889 DB 000H,000H,07CH,0C0H,078H,00CH,0F8H,000H ; L.C. S D_73
FE0E 10307C3030341800 5890 DB 010H,030H,07CH,030H,030H,034H,018H,000H ; L.C. T D_74
FE16 0000CCCC00007600 5891 DB 000H,000H,0CCH,0CCH,0CCH,0CCH,076H,000H ; L.C. U D_75
FE1E 0000CCCC000078300 5892 DB 000H,000H,0CCH,0CCH,0CCH,078H,030H,000H ; L.C. V D_76
FE26 0000C6D6FEFF6C00 5893 DB 000H,000H,0C6H,0D6H,0FEH,0FEH,06CH,000H ; L.C. W D_77
FE2E 0000C66C386CE600 5894 DB 000H,000H,0C6H,06CH,06CH,038H,06CH,000H ; L.C. X D_78
FE36 0000CCCC00007C0CF8 5895 DB 000H,000H,0CCH,0CCH,0CCH,07CH,0CCH,0F6H ; L.C. Y D_79
FE3E 0000F983064FC00 5896 DB 000H,000H,0FCH,098H,030H,064H,0FCH,000H ; L.C. Z D_7A
FE46 1C3030E030301C00 5897 DB 01CH,030H,030H,0E0H,030H,030H,01CH,000H ; C D_7B
FE4E 1818180018181800 5898 DB 018H,018H,018H,000H,018H,018H,018H,000H ; | D_7C
FE56 E030301C3030E000 5899 DB 0E0H,030H,030H,01CH,030H,030H,0E0H,000H ; } D_7D
FE5E 76DC000000000000 5900 DB 076H,0DCH,000H,000H,000H,000H,000H,000H ; TILDE D_7E
FE66 0010386CC6C6FE00 5901 DB 000H,010H,038H,06CH,0C6H,0C6H,0FEH,000H ; DELTA D_7F
5902
5903 ;--- INT 1A -----
5904 ; TIME_OF_DAY :
5905 ; THIS ROUTINE ALLOWS THE CLOCK TO BE SET/READ :
5906 ; :
5907 ; INPUT :
5908 ; (AH) = 0 READ THE CURRENT CLOCK SETTING :
5909 ; RETURNS CX = HIGH PORTION OF COUNT :
5910 ; DX = LOW PORTION OF COUNT :
5911 ; AL = 0 IF TIMER HAS NOT PASSED :
5912 ; 24 HOURS SINCE LAST READ :
5913 ; <0 IF ON ANOTHER DAY :
5914 ; (AH) = 1 SET THE CURRENT CLOCK :
5915 ; CX = HIGH PORTION OF COUNT :
5916 ; DX = LOW PORTION OF COUNT :
5917 ; NOTE: COUNTS OCCUR AT THE RATE OF :
5918 ; 1193180/65536 COUNTS/SEC :
5919 ; (OR ABOUT 18.2 PER SECOND -- SEE EQUATES BELOW) :
5920 ;-----
5921 ASSUME CS:CODE,DS:DATA
5922 ORG OFE6EH
FE6E TIME_OF_DAY PROC FAR ;
FE6E STI ; INTERRUPTS BACK ON
FE6F PUSH DS ; SAVE SEGMENT
FE70 CALL DDS
FE73 OR AH,AH ; AH=0
FE75 JZ T2 ; READ_TIME
FE77 FECC 5929 DEC AH ; AH=1
FE79 7416 5930 JZ T3 ; SET_TIME
FE7B T1: ; TOD_RETURN
FE7B STI ; INTERRUPTS BACK ON
FE7C IF 5933 POP DS ; RECOVER SEGMENT
FE7D CF 5934 IRET ; RETURN TO CALLER
FE7E T2: ; READ_TIME
FE7E FA 5936 CLI ; NO TIMER INTERRUPTS WHILE READING
FE7F A07000 5937 MOV AL,TIMER_OFL
FE82 C060700000 5938 MOV TIMER_OFL,0 ; GET OVERFLOW, AND RESET THE FLAG
FE87 8B0E6E00 5939 MOV CX,TIMER_HIGH
FE8B 8B166C00 5940 MOV DX,TIMER_LOW

```

```

LOC OBJ          LINE  SOURCE
FE8F EBEA        5941          JMP     T1          ; TOO_RETURN
FE91              5942      T3:          ; SET_TIME
FE91 FA          5943          CLI          ; NO INTERRUPTS WHILE WRITING
FE92 8916C00     5944          MOV     TIMER_LOW,DX
FE96 890E6E00    5945          MOV     TIMER_HIGH,CX ; SET THE TIME
FE9A C606700000  5946          MOV     TIMER_OFL,0   ; RESET OVERFLOW
FE9F EBD4        5947          JMP     T1          ; TOO_RETURN
5948          TIME_OF_DAY      ENDP
5949
5950          ;-----
5951          ; THIS ROUTINE HANDLES THE TIMER INTERRUPT FROM      :
5952          ; CHANNEL 0 OF THE 8253 TIMER. INPUT FREQUENCY      :
5953          ; IS 1.19318 MHZ AND THE DIVISOR IS 65536, RESULTING :
5954          ; IN APPROX. 18.2 INTERRUPTS EVERY SECOND.        :
5955          ; :
5956          ; THE INTERRUPT HANDLER MAINTAINS A COUNT OF INTERRUPTS :
5957          ; SINCE POWER ON TIME, WHICH MAY BE USED TO ESTABLISH :
5958          ; TIME OF DAY.                                       :
5959          ; THE INTERRUPT HANDLER ALSO DECREMENTS THE MOTOR   :
5960          ; CONTROL COUNT OF THE DISKETTE, AND WHEN IT EXPIRES, :
5961          ; WILL TURN OFF THE DISKETTE MOTOR, AND RESET THE  :
5962          ; MOTOR RUNNING FLAGS.                               :
5963          ; THE INTERRUPT HANDLER WILL ALSO INVOKE A USER ROUTINE :
5964          ; THROUGH INTERRUPT ICH AT EVERY TIME TICK. THE USER :
5965          ; MUST CODE A ROUTINE AND PLACE THE CORRECT ADDRESS IN :
5966          ; THE VECTOR TABLE.                               :
5967          ;-----
FEA5              5968          ORG     OFEASH
FEA5              5969          TIMER_INT  PROC     FAR
FEA5 FB          5970          STI          ; INTERRUPTS BACK ON
FEA6 1E          5971          PUSH   DS
FEA7 50          5972          PUSH   AX
FEA8 52          5973          PUSH   DX      ; SAVE MACHINE STATE
FEA9 E89200      5974          CALL   DDS
FEAC FF066C00    5975          INC     TIMER_LOW   ; INCREMENT TIME
FEB0 7504        5976          JNZ    T4          ; TEST_DAY
FEB2 FF066E00    5977          INC     TIMER_HIGH  ; INCREMENT HIGH WORD OF TIME
FEB6              5978          T4:          ; TEST_DAY
FEB6 833FE0018   5979          CMP     TIMER_HIGH,018H ; TEST FOR COUNT EQUALING 24 HOURS
FEBB 7515        5980          JNZ    T5          ; DISKETTE_CTL
FEBD 813E6C00B000 5981          CMP     TIMER_LOW,0B0H
FEC3 750D        5982          JNZ    T5          ; DISKETTE_CTL
5983
5984          ;----- TIMER HAS GONE 24 HOURS
5985
FEC5 2BC0        5986          SUB     AX,AX
FEC7 A36E00      5987          MOV     TIMER_HIGH,AX
FECA A36C00      5988          MOV     TIMER_LOW,AX
FECD C606700001  5989          MOV     TIMER_OFL,1
5990
5991          ;----- TEST FOR DISKETTE TIME OUT
5992
FED2              5993          T5:          ; DISKETTE_CTL
FED2 FE0E4000    5994          DEC     MOTOR_COUNT
FED6 750B        5995          JNZ    T6          ; RETURN IF COUNT NOT OUT
FED8 80263F00F0  5996          AND     MOTOR_STATUS,0F0H ; TURN OFF MOTOR RUNNING BITS
FEDD B00C        5997          MOV     AL,0CH
FEDF BAF203      5998          MOV     DX,03F2H    ; FDC CTL PORT
FEE2 EE          5999          OUT    DX,AL       ; TURN OFF THE MOTOR
FEE3              6000          T6:          ; TIMER_RET:
FEE3 CD1C        6001          INT    ICH         ; TRANSFER CONTROL TO A USER ROUTINE
FEE5 B020        6002          MOV     AL,EOI
FEE7 E620        6003          OUT    020H,AL     ; END OF INTERRUPT TO 8259
FEE9 5A          6004          POP    DX
FEEA 58          6005          POP    AX
FEEB 1F          6006          POP    DS          ; RESET MACHINE STATE
FEEC CF          6007          IRET           ; RETURN FROM INTERRUPT
6008          TIMER_INT      ENDP
6009
FEED 31383031    6010          F3B    DB     '1801',13,10
FEF1 0D
FEF2 0A
6011
6012          ;-----
6013          ; THESE ARE THE VECTORS WHICH ARE MOVED INTO      :
6014          ; THE 8086 INTERRUPT AREA DURING POWER ON.      :
6015          ; ONLY THE OFFSETS ARE DISPLAYED HERE, CODE SEGMENT :

```



```

LOC OBJ          LINE    SOURCE

6016 ;           WILL BE ADDED FOR ALL OF THEM, EXCEPT WHERE NOTED :
6017 |-----|
6018           ASSUME CS:CODE
FEF3           6019     ORG     OFFF3H
FEF3           6020     VECTOR_TABLE LABEL WORD ; VECTOR TABLE FOR MOVE TO INTERRUPTS
FEF3 ASFE      6021     DW     OFFSET TIMER_INT ; INTERRUPT 8
FEF5 87E9      6022     DW     OFFSET KB_INT ; INTERRUPT 9
FEF7 DDE6      6023     DW     OFFSET D_EOI ; INTERRUPT A
FEF9 DDE6      6024     DW     OFFSET D_EOI ; INTERRUPT B
FEFB DDE6      6025     DW     OFFSET D_EOI ; INTERRUPT C
FEFD DDE6      6026     DW     OFFSET D_EOI ; INTERRUPT D
FEFF 57EF      6027     DW     OFFSET DISK_INT ; INTERRUPT E
FF01 DDE6      6028     DW     OFFSET D_EOI ; INTERRUPT F
FF03 65F0      6029     DW     OFFSET VIDEO_IO ; INTERRUPT 10H
FF05 4DF8      6030     DW     OFFSET EQUIPMENT ; INTERRUPT 11H
FF07 41F8      6031     DW     OFFSET MEMORY_SIZE_DET ; INTERRUPT 12H
FF09 59EC      6032     DW     OFFSET DISKETTE_IO ; INTERRUPT 13H
FF0B 39EF      6033     DW     OFFSET RS232_IO ; INTERRUPT 14H
FF0D 59F8      6034     DW     OFFSET CASSETTE_IO ; INTERRUPT 15H
FF0F 2EE8      6035     DW     OFFSET KEYBOARD_IO ; INTERRUPT 16H
FF11 D2EF      6036     DW     OFFSET PRINTER_IO ; INTERRUPT 17H
6037
FF13 0000      6038     DW     00000H ; INTERRUPT 18H
6039 ;         DW     0F600H ; MUST BE INSERTED INTO TABLE LATER
6040
FF15 F2E6      6041     DW     OFFSET BOOT_STRAP ; INTERRUPT 19H
FF17 6EFE      6042     DW     TIME_OF_DAY ; INTERRUPT 1AH -- TIME OF DAY
FF19 53FF      6043     DW     DUMMY_RETURN ; INTERRUPT 1BH -- KEYBOARD BREAK ADDR
FF1B 53FF      6044     DW     DUMMY_RETURN ; INTERRUPT 1C -- TIMER BREAK ADDR
FF1D A4F0      6045     DW     VIDEO_PARMS ; INTERRUPT 1D -- VIDEO PARAMETERS
FF1F C7EF      6046     DW     OFFSET DISK_BASE ; INTERRUPT 1E -- DISK PARMS
FF21 0000      6047     DW     0 ; INTERRUPT 1F -- POINTER TO VIDED EXT
6048
FF23 50415249545920 6049     D2     DB     'PARITY CHECK 1',13,10
      43484543482031
FF31 0D
FF32 0A
FF33 20333031      6050     F1     DB     ' 301',13,10
FF37 0D
FF38 0A
FF39 313331      6051     F2     DB     '131',13,10
FF3C 0D
FF3D 0A
6052
FF3E           6053     DDS    PROC    NEAR
FF3E 50           6054     PUSH  AX ; SAVE AX
FF3F B84000        6055     MOV   AX,DATA
FF42 8ED8          6056     MOV   DS,AX ; SET DATA SEGMENT
FF44 58           6057     POP  AX ; RESTORE AX
FF45 C3           6058     RET
6059     DDS    ENDP
6060
6061 |-----|
6062 ;         TEMPORARY INTERRUPT SERVICE ROUTINE :
6063 |-----|
FF47           6064     ORG     OFF47H
FF47           6065     D11    PROC    NEAR
FF47 B401          6066     MOV   AH,1
FF49 50           6067     PUSH  AX ; SAVE REG AX CONTENTS
FF4A B0FF        6068     MOV   AL,OFFH ; MASK ALL INTERRUPTS OFF
FF4C E621        6069     OUT  INTA01,AL
FF4E B020        6070     MOV   AL,E0I
FF50 E620        6071     OUT  INTA00,AL
FF52 58          6072     POP  AX ; RESTORE REG AX CONTENTS
FF53           6073     DUMMY_RETURN: ; NEED IRET FOR VECTOR TABLE
FF53 CF          6074     IRET
6075     D11    ENDP
6076
6077 |-- INT 5 -----|
6078 ;         THIS LOGIC WILL BE INVOKED BY INTERRUPT 05H TO PRINT THE :
6079 ;         SCREEN. THE CURSOR POSITION AT THE TIME THIS ROUTINE IS INVOKED :
6080 ;         WILL BE SAVED AND RESTORED UPON COMPLETION. THE ROUTINE IS :
6081 ;         INTENDED TO RUN WITH INTERRUPTS ENABLED. IF A SUBSEQUENT :
6082 ;         'PRINT SCREEN' KEY IS DEPRESSED DURING THE TIME THIS ROUTINE :
6083 ;         IS PRINTING IT WILL BE IGNORED. :
6084 ;         ADDRESS 50:0 CONTAINS THE STATUS OF THE PRINT SCREEN: :
6085 ;

```

```

6086 ; 50:0 =0 EITHER PRINT SCREEN HAS NOT BEEN CALLED ;
6087 ; OR UPON RETURN FROM A CALL THIS INDICATES ;
6088 ; A SUCCESSFUL OPERATION. ;
6089 ; =1 PRINT SCREEN IS IN PROGRESS ;
6090 ; =255 ERROR ENCOUNTERED DURING PRINTING ;
6091 ; -----
6092 ASSUME CS:CODE,DS:XXDATA
6093 ORG OFF54H
6094 PRINT_SCREEN PROC FAR
6095 STI ; MUST RUN WITH INTERRUPTS ENABLED
6096 PUSH DS ; MUST USE 50:0 FOR DATA AREA STORAGE
6097 PUSH AX
6098 PUSH BX
6099 PUSH CX ; WILL USE THIS LATER FOR CURSOR LIMITS
6100 PUSH DX ; WILL HOLD CURRENT CURSOR POSITION
6101 MOV AX,XXDATA ; HEX 50
6102 MOV DS,AX
6103 CMP STATUS_BYTE,1 ; SEE IF PRINT ALREADY IN PROGRESS
6104 JZ EXIT ; JUMP IF PRINT ALREADY IN PROGRESS
6105 MOV STATUS_BYTE,1 ; INDICATE PRINT NOW IN PROGRESS
6106 MOV AH,15 ; WILL REQUEST THE CURRENT SCREEN MODE
6107 INT 10H ; [AL]=MODE
6108 ; [AH]=NUMBER COLUMNS/LINE
6109 ; [BH]=VISUAL PAGE
6110 ; -----
6111 ; AT THIS POINT WE KNOW THE COLUMNS/LINE ARE IN ;
6112 ; [AX] AND THE PAGE IF APPLICABLE IS IN [BH]. THE STACK ;
6113 ; HAS DS,AX,BX,CX,DX PUSHED. [AL] HAS VIDEO MODE ;
6114 ; -----
6115 MOV CL,AH ; WILL MAKE USE OF [CX] REGISTER TO
6116 MOV CH,25 ; CONTROL ROW & COLUMNS
6117 CALL CRLF ; CARRIAGE RETURN LINE FEED ROUTINE
6118 PUSH CX ; SAVE SCREEN BOUNDS
6119 MOV AH,3 ; WILL NOW READ THE CURSOR.
6120 INT 10H ; AND PRESERVE THE POSITION
6121 POP CX ; RECALL SCREEN BOUNDS
6122 PUSH DX ; RECALL [BH]=VISUAL PAGE
6123 XOR DX,DX ; WILL SET CURSOR POSITION TO [0,0]
6124 ; -----
6125 ; THE LOOP FROM PRI10 TO THE INSTRUCTION PRIOR TO PRI20 ;
6126 ; IS THE LOOP TO READ EACH CURSOR POSITION FROM THE ;
6127 ; SCREEN AND PRINT. ;
6128 ; -----
6129
6130 MOV AH,2 ; TO INDICATE CURSOR SET REQUEST
6131 INT 10H ; NEW CURSOR POSITION ESTABLISHED
6132 MOV AH,8 ; TO INDICATE READ CHARACTER
6133 INT 10H ; CHARACTER NOW IN [AL]
6134 OR AL,AL ; SEE IF VALID CHAR
6135 JNZ PRI15 ; JUMP IF VALID CHAR
6136 MOV AL,' ' ; MAKE A BLANK
6137
6138 MOV AH,DL ; TO INDICATE PRINT CHAR IN [AL]
6139 XOR DX,DX ; PRINT THE CHARACTER
6140 XOR AH,AH ; RECALL CURSOR POSITION
6141 INT 17H ; RECALL CURSOR POSITION
6142 POP DX ; TEST FOR PRINTER ERROR
6143 TEST AH,25H ; JUMP IF ERROR DETECTED
6144 JNZ ERR10 ; ADVANCE TO NEXT COLUMN
6145 INC DL ; SEE IF AT END OF LINE
6146 CMP CL,DL ; IF NOT PROCEED
6147 JNZ PRI10 ; BACK TO COLUMN 0
6148 XOR DL,DL ; [AH]=0
6149 MOV AH,DL ; [AH]=0
6150 PUSH DX ; SAVE NEW CURSOR POSITION
6151 CALL CRLF ; LINE FEED CARRIAGE RETURN
6152 POP DX ; RECALL CURSOR POSITION
6153 INC DH ; ADVANCE TO NEXT LINE
6154 CMP CH,DH ; FINISHED?
6155 JNZ PRI10 ; IF NOT CONTINUE
6156
6157 POP DX ; RECALL CURSOR POSITION
6158 MOV AH,2 ; TO INDICATE CURSOR SET REQUEST
6159 INT 10H ; CURSOR POSITION RESTORED
6160 MOV STATUS_BYTE,0 ; INDICATE FINISHED
6161 JMP SHORT EXIT ; EXIT THE ROUTINE
6162
ERR10:

```

LOC OBJ	LINE	SOURCE	
FFBB 5A	6163	POP DX	; GET CURSOR POSITION
FFBC B402	6164	MOV AH,2	; TO REQUEST CURSOR SET
FFBE CD10	6165	INT 10H	; CURSOR POSITION RESTORED
FFC0	6166	ERR20:	
FFC0 C060000FF	6167	MOV STATUS_BYTE,0FFH	; INDICATE ERROR
FFC5	6168	EXIT:	
FFC5 5A	6169	POP DX	; RESTORE ALL THE REGISTERS USED
FFC6 59	6170	POP CX	
FFC7 5B	6171	POP BX	
FFC8 58	6172	POP AX	
FFC9 1F	6173	POP DS	
FFCA CF	6174	IRET	
	6175	PRINT_SCREEN ENDP	
	6176		
	6177	;----- CARRIAGE RETURN, LINE FEED SUBROUTINE	
	6178		
FFCB	6179	CRLF PROC NEAR	
FFCB 33D2	6180	XOR DX,DX	; PRINTER 0
FFCD 32E4	6181	XOR AH,AH	; WILL NOW SEND INITIAL LF,CR
	6182		; TO PRINTER
FFCF B00A	6183	MOV AL,12Q	; LF
FFD1 CD17	6184	INT 17H	; SEND THE LINE FEED
FFD3 32E4	6185	XOR AH,AH	; NOW FOR THE CR
FFD5 B00D	6186	MOV AL,15Q	; CR
FFD7 CD17	6187	INT 17H	; SEND THE CARRIAGE RETURN
FFD9 C3	6188	RET	
	6189	CRLF ENDP	
	6190		
FFDA 50415249545920 434845434B2032	6191	D1 DB 'PARITY CHECK 2',13,10	
FFE8 0D			
FFE9 0A			
FFEA 363031	6192	F3 DB '601',13,10	
FFED 0D			
FFEE 0A			
	6193		
----	6194	CODE ENDS	
	6195		
	6196	;-----	
	6197	; POWER ON RESET VECTOR :	
	6198	;-----	
----	6199	VECTOR SEGMENT AT 0FFFFH	
	6200		
	6201	;----- POWER ON RESET	
	6202		
0000 EA5BE000F0	6203	JMP RESET	
	6204		
0005 31302F3232F38 32	6205	DB '10/27/82'	; RELEASE MARKER
----	6206	VECTOR ENDS	
	6207	END	

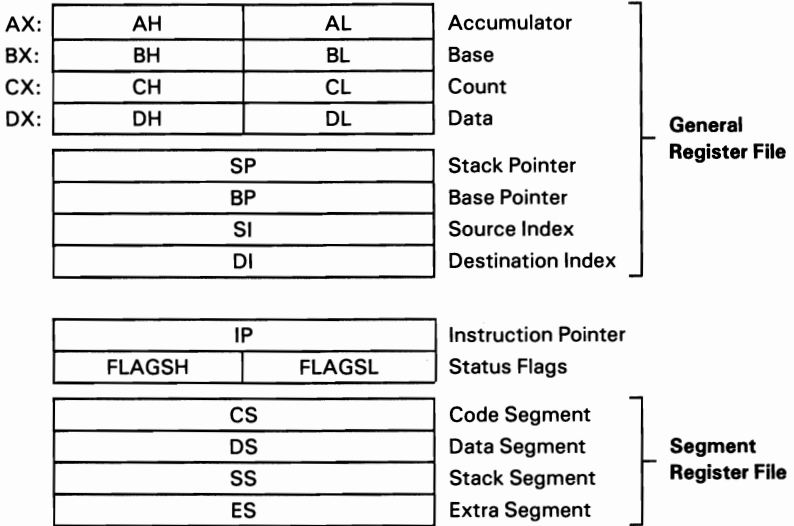
SECTION 6. INSTRUCTION SET

Contents

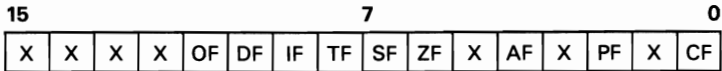
8088 Register Model	6-3
Operand Summary	6-4
Second Instruction Byte Summary	6-4
Memory Segmentation Model	6-5
Use of Segment Override	6-5
Data Transfer	6-6
Arithmetic	6-8
Logic	6-10
String Manipulation	6-11
Control Transfer	6-12
8088 Conditional Transfer Operations	6-15
Processor Control	6-16
8087 Extensions to the 8088 Instruction Set	6-17
Data Transfer	6-17
Comparison	6-19
Arithmetic	6-19
Transcendental	6-21

Constants **6-21**
Processor Control **6-22**
8088 Instruction Set Matrix **6-25**
Instruction Set Index **6-27**

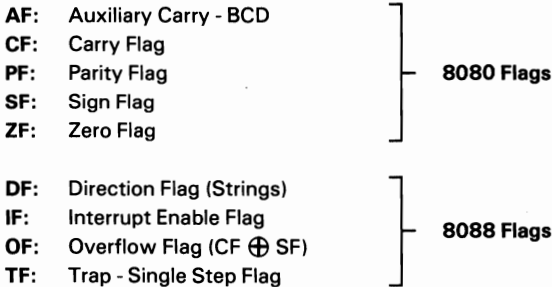
8088 Register Model



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



x = Don't Care



Operand Summary

“reg field Bit Assignments:

16-Bit [w = 1]	8-Bit [w = 0]	Segment
000 AX	000 AL	00 ES
001 CX	001 CL	01 CS
010 DX	010 DL	10 SS
011 BX	011 BL	11 DS
100 SP	100 AH	
101 BP	101 CH	
110 SI	110 DH	
111 DI	111 BH	

Second Instruction Byte Summary

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

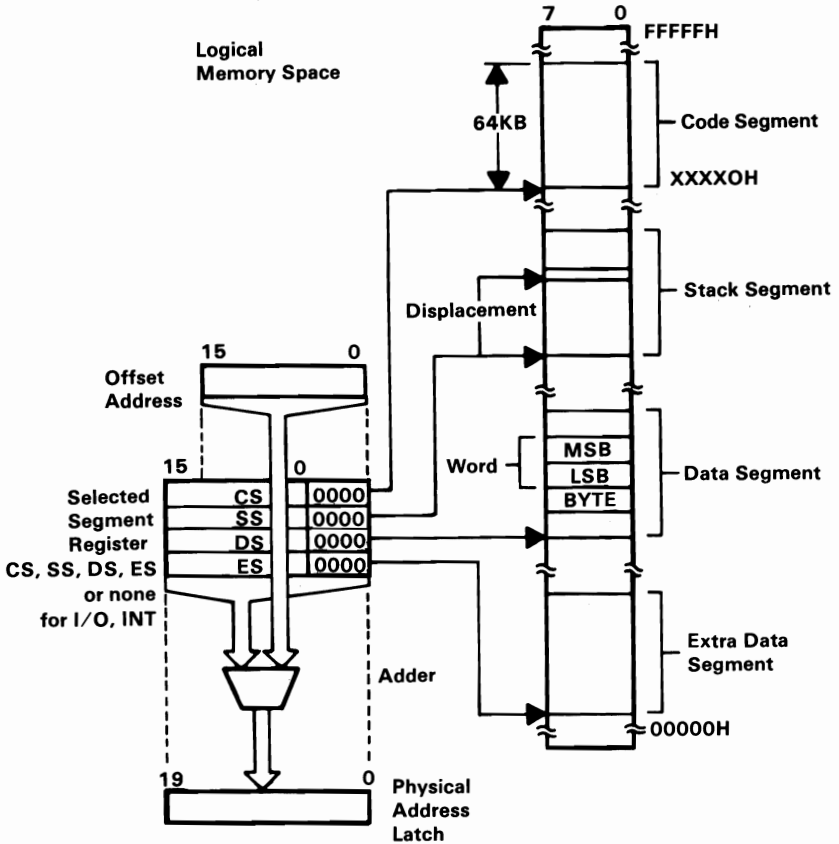
mod	Displacement
00	DISP = 0*, disp-low and disp-high are absent
01	DISP = disp-low sign-extended to 16-bits, disp-high is absent
10	DISP = disp-high: disp-low
11	r/m is treated as a “reg” field

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*
	111	(BX) + DISP

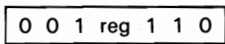
DISP follows 2nd byte of instruction (before data if required).

*except if mod = 00 and r/m = 110 then EA = disp-high: disp-low.

Memory Segmentation Modal



Segment Override Prefix



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

MOV = Move

Register/memory to/from register

1 0 0 0 1 0 d w	mod reg r/m
-----------------	-------------

Immediate to register/memory

1 1 0 0 0 1 1 w	mod 0 0 0 r/m	data	data if w = 1
-----------------	---------------	------	---------------

Immediate to register

1 0 1 1 w reg	data	data if w = 1
---------------	------	---------------

Memory to accumulator

1 0 1 0 0 0 0 w	addr-low	addr-high
-----------------	----------	-----------

Accumulator to memory

1 0 1 0 0 0 1 w	addr-low	addr-high
-----------------	----------	-----------

Register/memory to segment register

1 0 0 0 1 1 1 0	mod 0 reg r/m
-----------------	---------------

Segment register to register/memory

1 0 0 0 1 1 0 0	mod 0 reg r/m
-----------------	---------------

PUSH = Push

Register/memory

1 1 1 1 1 1 1 1	mod 1 1 0 r/m
-----------------	---------------

Register

0 1 0 1 0 reg

Segment register

0 0 0 reg 1 1 0

Pop = Pop

Register/memory

1 0 0 0 1 1 1 1	mod 0 0 0 r/m
-----------------	---------------

Register

0 1 0 1 1 reg

Segment register

0 0 0 reg 1 1 1

XCHG = Exchange
Register/memory with register

1 0 0 0 0 1 1 w	mod	reg	r/m
-----------------	-----	-----	-----

Register with accumulator

1 0 0 1 0 reg

IN = Input to AL/AX from
Fixed port

1 1 1 0 0 1 0 w	port
-----------------	------

Variable port (DX)

1 1 1 0 1 1 0 w

OUT = Output from AL/AX to
Fixed port

1 1 1 0 0 1 1 w	port
-----------------	------

Variable port (DX)

1 1 1 0 1 1 0 w

XLAT = Translate byte to AL

1 1 0 1 0 1 1 1

LEA = Load EA to register

1 0 0 0 1 1 0 1	mod	reg	r/m
-----------------	-----	-----	-----

LDS = Load pointer to DS

1 1 0 0 0 1 0 1	mod	reg	r/m
-----------------	-----	-----	-----

LES = Load pointer to ES

1 1 0 0 0 1 0 0	mod	reg	r/m
-----------------	-----	-----	-----

LAHF = Load AH with flags

1 0 0 1 1 1 1 1

SAHF = Store AH into flags

1 0 0 1 1 1 1 0

PUSHF = Push flags

1 0 0 1 1 1 0 0

POPF = Pop flags

1 0 0 1 1 1 0 1

Arithmetic

ADD = Add

Register/memory with register to either

0 0 0 0 0 0 d w	mod	reg	r/m
-----------------	-----	-----	-----

Immediate to register/memory

1 0 0 0 0 0 s w	mod	0 0 0	r/m	data	data if s:w = 01
-----------------	-----	-------	-----	------	------------------

Immediate to accumulator

0 0 0 0 0 1 0 w	data	data if w = 1
-----------------	------	---------------

ADC = Add with carry

Register/memory with register to either

0 0 0 1 0 0 d w	mod	reg	r/m
-----------------	-----	-----	-----

Immediate to register/memory

1 0 0 0 0 0 s w	mod	0 1 0	r/m	data	data if s:w = 01
-----------------	-----	-------	-----	------	------------------

Immediate to accumulator

0 0 0 1 0 1 0 w	data	data if w = 1
-----------------	------	---------------

INC = Increment

Register/Memory

1 1 1 1 1 1 1 w	mod	0 0 0	r/m
-----------------	-----	-------	-----

Register

0 1 0 0 0 reg

AAA = ASCII adjust for add

0 0 1 1 0 1 1 1

DAA = Decimal adjust for add

0 0 1 0 0 1 1 1

SUB = Subtract

Register/memory and register to either

0 0 1 0 1 0 d w	mod	reg	r/m
-----------------	-----	-----	-----

Immediate from register/memory

1 0 0 0 0 0 s w	mod	1 0 1	r/m	data	data if s:w = 01
-----------------	-----	-------	-----	------	------------------

Immediate from accumulator

0 0 1 0 1 1 0 w	data	data if w = 1
-----------------	------	---------------

SBB = Subtract with borrow
Register/memory and register to either

0 0 0 1 1 0 d w	mod reg r/m
-----------------	-------------

Immediate from register/memory

1 0 0 0 0 0 s w	mod 0 1 1 r/m	data	data if s:w = 01
-----------------	---------------	------	------------------

Immediate from accumulator

0 0 0 1 1 1 0 w	data	data if w = 1
-----------------	------	---------------

DEC = Decrement
Register/memory

1 1 1 1 1 1 1 w	mod 0 0 1 r/m
-----------------	---------------

Register

0 1 0 0 1 reg

NEG = Change sign

1 1 1 1 0 1 1 w	mod 0 1 1 r/m
-----------------	---------------

CMP = Compare
Register/memory and register

0 0 1 1 1 0 d w	mod reg r/m
-----------------	-------------

Immediate with register/memory

1 0 0 0 0 0 s w	mod 1 1 1 r/m	data	data if s:w = 01
-----------------	---------------	------	------------------

Immediate with accumulator

0 0 1 1 1 1 0 w	data	data if w = 1
-----------------	------	---------------

AAS = ASCII adjust for subact

0 0 1 1 1 1 1 1

DAS = Decimal adjust for subact

0 0 1 0 1 1 1 1

MUL = Multiply (unsigned)

1 1 1 1 0 1 1 w	mod 1 0 0 r/m
-----------------	---------------

IMUL = Integer multiply (signed)

1 1 1 1 0 1 1 w	mod 1 0 1 r/m
-----------------	---------------

AAM = ASCII adjust for multiply

1 1 0 1 0 1 0 0	0 0 0 0 1 0 1 0
-----------------	-----------------

DIV = Divide (unsigned)

1 1 1 1 0 1 1 w	mod 1 1 0 r/m
-----------------	---------------

IDIV = Integer divide (signed)

1 1 1 1 0 1 1 w	mod 1 1 1 r/m
-----------------	---------------

AAD = ASCII adjust for divide

1 1 0 1 0 1 0 1	0 0 0 0 1 0 1 0
-----------------	-----------------

CBW = Convert byte to word

1 0 0 1 1 0 0 0

CWD = Convert word to double word

1 0 0 1 1 0 0 1

Logic

NOT = Invert

1 1 1 1 0 1 1 w	mod 0 1 0 r/m
-----------------	---------------

SHL/SAL = Shift logical/arithmetic left

1 1 0 1 0 0 v w	mod 1 0 0 r/m
-----------------	---------------

SHR = Shift logical right

1 1 0 1 0 0 v w	mod 1 0 1 r/m
-----------------	---------------

SAR = Shift arithmetic right

1 1 0 1 0 0 v w	mod 1 1 1 r/m
-----------------	---------------

ROL = Rotate left

1 1 0 1 0 0 v w	mod 0 0 0 r/m
-----------------	---------------

ROR = Rotate right

1 1 0 1 0 0 v w	mod 0 0 1 r/m
-----------------	---------------

RCL = Rotate through carry left

1 1 0 1 0 0 v w	mod 0 1 0 r/m
-----------------	---------------

RCR = Rotate through carry right

1 1 0 1 0 0 v w	mod 0 1 1 r/m
-----------------	---------------

AND = And

Register/memory and register to either

0 0 1 0 0 0 d w	mod reg r/m
-----------------	-------------

Immediate to register/memory

1 0 0 0 0 0 0 w	mod 1 0 0 r/m	data	data if w = 1
-----------------	---------------	------	---------------

Immediate to accumulator

0 0 1 0 0 1 0 w	data	data if w = 1
-----------------	------	---------------

6-10 Instruction Set

TEST = And function to flags, no result
Register/memory and register

1 0 0 0 0 1 0 w	mod	reg	r/m
-----------------	-----	-----	-----

Immediate data and register/memory

1 1 1 1 0 1 1 w	mod	0 0 0	r/m	data	data if w = 1
-----------------	-----	-------	-----	------	---------------

Immediate data and accumulator

1 0 1 0 1 0 0 w	data	data if w = 1
-----------------	------	---------------

OR = Or

Register/memory and register to either

0 0 0 0 1 0 d w	mod	reg	r/m
-----------------	-----	-----	-----

Immediate to register/memory

1 0 0 0 0 0 0 w	mod	0 0 1	r/m	data	data if w = 1
-----------------	-----	-------	-----	------	---------------

Immediate to accumulator

0 0 0 0 1 1 0 w	data	data if w = 1
-----------------	------	---------------

XOR = Exclusive or

Register/memory and register to either

0 0 1 1 0 0 d w	mod	reg	r/m
-----------------	-----	-----	-----

Immediate to register/memory

1 0 0 0 0 0 0 w	mod	1 1 0	r/m	data	data if w = 1
-----------------	-----	-------	-----	------	---------------

Immediate to accumulator

0 0 1 1 0 1 0 w	data	data if w = 1
-----------------	------	---------------

String Manipulation

REP = Repeat

1 1 1 1 0 0 1 z

MOVS = Move String

1 0 1 0 0 1 0 w

CMPS = Compare String

1 0 1 0 0 1 1 w

SCAS = Scan String

1 0 1 0 1 1 1 w

LODS = Load String

1 0 1 0 1 1 0 w

STOS = Store String

1 0 1 0 1 0 1 w

Control Transfer

CALL = Call

Direct within segment

1 1 1 0 1 0 0 0	disp-low	disp-high
-----------------	----------	-----------

Indirect within segment

1 1 1 1 1 1 1 1	mod 0 1 0 r/m
-----------------	---------------

Direct intersegment

1 0 0 1 1 0 1 0	offset-low	offset-high
-----------------	------------	-------------

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

Indirect intersegment

1 1 1 1 1 1 1 1	mod 0 1 1 r/m
-----------------	---------------

JMP = Unconditional Jump

Direct within segment

1 1 1 0 1 0 0 1	disp-low	disp-high
-----------------	----------	-----------

Direct within segment-short

1 1 1 0 1 0 1 1	disp
-----------------	------

Indirect within segment

1 1 1 1 1 1 1 1	mod 1 0 0 r/m
-----------------	---------------

Direct intersegment

1 1 1 0 1 0 1 0	offset-low	offset-high
-----------------	------------	-------------

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

Indirect intersegment

1 1 1 1 1 1 1 1	mod 1 0 1 r/m
-----------------	---------------

RET = Return from CALL
Within segment

1 1 0 0 0 0 1 1

Within segment adding immediate to SP

1 1 0 0 0 0 1 0	data-low	data-high
-----------------	----------	-----------

Intersegment

1 1 0 0 1 0 1 1

Intersegment, adding immediate to SP

1 1 0 0 0 0 1 0	data-low	data-high
-----------------	----------	-----------

JE/JZ = Jump on equal/zero

0 1 1 1 0 1 0 0	disp
-----------------	------

JL/JNGE = Jump on less/not greater or equal

0 1 1 1 1 1 0 0	disp
-----------------	------

JLE/JNG = Jump on less or equal/not greater

0 1 1 1 1 1 1 0	disp
-----------------	------

JB/JNAE = Jump on below/not above or equal

0 1 1 1 0 0 1 0	disp
-----------------	------

JBE/JNA = Jump on below or equal/not above

0 1 1 1 0 1 1 0	disp
-----------------	------

JP/JPE = Jump on parity/parity even

0 1 1 1 1 0 1 0	disp
-----------------	------

JO = Jump on overflow

0 1 1 1 0 0 0 0	disp
-----------------	------

JS = Jump on sign

0 1 1 1 1 0 0 0	disp
-----------------	------

JNE/JNZ = Jump on not equal/not zero

0 1 1 1 0 1 0 1	disp
-----------------	------

JNL/JGE = Jump on not less/greater or equal

0 1 1 1 1 1 0 1	disp
-----------------	------

JNLE/JG = Jump on not less or equal/greater

0 1 1 1 1 1 1 1	disp
-----------------	------

JNB/JAE = Jump on not below/above or equal

0 1 1 1 0 0 1 1	disp
-----------------	------

JNBE/JA = Jump on not below or equal/above

0 1 1 1 0 1 1 1	disp
-----------------	------

JNP/JPO = Jump on not parity/parity odd

0 1 1 1 1 0 1 1	disp
-----------------	------

JNO = Jump on not overflow

0 1 1 1 0 0 0 1	disp
-----------------	------

JNS = Jump on not sign

0 1 1 1 1 0 0 1	disp
-----------------	------

LOOP = Loop CX times

1 1 1 0 0 0 1 0	disp
-----------------	------

LOOPZ/LOOPE = Loop while zero/equal

1 1 1 0 0 0 0 1	disp
-----------------	------

LOOPNZ/LOOPNE = Loop while not zero/not equal

1 1 1 0 0 0 0 0	disp
-----------------	------

JCXZ = Jump on CX zero

1 1 1 0 0 0 1 1	disp
-----------------	------

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 less 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

1 1 0 0 1 1 0 1	type
-----------------	------

Type 3

1 1 0 0 1 1 0 0

INTO = Interrupt on overflow

1 1 0 0 1 1 1 0

IRET = Interrupt return

1 1 0 0 1 1 1 1

Processor Control

CLC = Clear carry

1 1 1 1 1 0 0 0

CMC = Complement carry

1 1 1 1 0 1 0 1

CLD = Clear direction

1 1 1 1 1 1 0 0

CLI = Clear interrupt

1 1 1 1 1 0 1 0

HLT = Halt

1 1 1 1 0 1 0 0

LOCK = Bus lock prefix

1 1 1 1 0 0 0 0

STC = Set carry

1 1 1 1 1 0 0 1

NOP = No operation

1 0 0 1 0 0 0 0

STD = Set direction

1 1 1 1 1 1 0 1

STI = Set interrupt

1 1 1 1 1 0 1 1

WAIT = Wait

1 0 0 1 1 0 1 1

ESC = Escape (to external device)

1 1 0 1 1 x x x mod x x x r/m

Footnotes:

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 then "count" = 1; if v = 1 then "count" in (CL)

x = don't care

z is used for some string primitives to compare with ZF FLAG

AL = 8-bit accumulator

AX = 16-bit accumulator

CX = Count register

DS = Data segment

DX = Variable port register

ES = Extra segment

Above/below refers to unsigned value

Greater = more positive;

Less = less positive (more negative) signed values

8087 Extensions to the 8088 Instruction Set

Data Transfer

FLD = Load
Integer/Real Memory to ST(0)

Escape	MF	1	mod	0	0	0	r/m	disp-low	disp-high
--------	----	---	-----	---	---	---	-----	----------	-----------

Long Integer Memory to ST(0)

Escape	1	1	1	mod	1	0	1	r/m	disp-low	disp-high
--------	---	---	---	-----	---	---	---	-----	----------	-----------

Temporary Real Memory to ST(0)

Escape	0	1	1	mod	1	0	1	r/m	disp-low	disp-high
--------	---	---	---	-----	---	---	---	-----	----------	-----------

BCD Memory to ST(0)

Escape	1	1	1	mod	1	0	0	r/m	disp-low	disp-high
--------	---	---	---	-----	---	---	---	-----	----------	-----------

ST(i) to ST(0)

Escape	0	0	1	1	1	0	0	0	ST(i)
--------	---	---	---	---	---	---	---	---	-------

FST = Store
ST(0) to Integer/Real Memory

Escape	MF	1	mod	0	1	0	r/m	disp-low	disp-high
--------	----	---	-----	---	---	---	-----	----------	-----------

ST(0) to ST(i)

Escape	1	0	1	1	1	0	1	0	ST(i)
--------	---	---	---	---	---	---	---	---	-------

FSTP = STORE AND POP

ST(0) to Integer/Real Memory

Escape	MF	1	mod	0	1	1	r/m	disp-low	disp-high
--------	----	---	-----	---	---	---	-----	----------	-----------

ST(0) to Long Integer Memory

Escape	1	1	1	mod	1	1	1	r/m	disp-low	disp-high
--------	---	---	---	-----	---	---	---	-----	----------	-----------

ST(0) to Temporary Real Memory

Escape	0	1	1	mod	1	1	1	r/m	disp-low	disp-high
--------	---	---	---	-----	---	---	---	-----	----------	-----------

ST(0) to BCD Memory

Escape	1	1	1	mod	1	1	0	r/m	disp-low	disp-high
--------	---	---	---	-----	---	---	---	-----	----------	-----------

ST(0) to ST(i)

Escape	1	0	1	1	1	0	1	1	ST(i)
--------	---	---	---	---	---	---	---	---	-------

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

Escape	0	0	1	1	1	0	0	1	ST(i)
--------	---	---	---	---	---	---	---	---	-------

Comparison

FCOM = Compare
Integer/Real Memory to ST(0)

Escape	MF	0	mod	0	1	0	r/m	disp-low	disp-high
--------	----	---	-----	---	---	---	-----	----------	-----------

ST(i) to ST(0)

Escape	0	0	0	1	1	0	1	0	ST(i)
--------	---	---	---	---	---	---	---	---	-------

FCOMP = Compare and Pop
Integer/Real Memory to ST(0)

Escape	MF	0	mod	0	1	1	r/m	disp-low	disp-high
--------	----	---	-----	---	---	---	-----	----------	-----------

ST(i) to ST(0)

Escape	0	0	0	1	1	0	1	1	ST(i)
--------	---	---	---	---	---	---	---	---	-------

FCOMPP = Compare ST(1) to ST(0) and Pop twice

Escape	1	1	0	1	1	0	1	1	0	0	1
--------	---	---	---	---	---	---	---	---	---	---	---

FTST = Test ST(0)

Escape	0	0	1	1	1	1	0	0	1	0	0
--------	---	---	---	---	---	---	---	---	---	---	---

FXAM = Examine ST(0)

Escape	0	0	1	1	1	1	0	0	1	0	1
--------	---	---	---	---	---	---	---	---	---	---	---

Arithmetic

FADD = Addition
Integer/Real Memory with ST(0)

Escape	MF	0	mod	0	0	0	r/m	disp-low	disp-high
--------	----	---	-----	---	---	---	-----	----------	-----------

ST(i) to ST(0)

Escape	d	P	0	1	1	0	0	0	ST(i)
--------	---	---	---	---	---	---	---	---	-------

FSUB = Subtraction
Integer/Real Memory with ST(0)

Escape	MF	0	mod	1	0	R	r/m	disp-low	disp-high
--------	----	---	-----	---	---	---	-----	----------	-----------

ST(i) to ST(0)

Escape	d	P	0	1	1	1	0	R	r/m
--------	---	---	---	---	---	---	---	---	-----

Arithmetic (Continued)

FMUL = Multiplication
Integer/Real Memory to ST(0)

Escape	MF	0	mod	0	0	1	r/m	disp-low	disp-high
--------	----	---	-----	---	---	---	-----	----------	-----------

ST(i) and ST(0)

Escape	d	P	0	1	1	0	0	1	r/m
--------	---	---	---	---	---	---	---	---	-----

FDIV = Division
Integer/Real Memory with ST(0)

Escape	MF	0	mod	1	1	R	r/m	disp-low	disp-high
--------	----	---	-----	---	---	---	-----	----------	-----------

ST(i) and ST(0)

Escape	d	P	0	1	1	0	0	1	r/m
--------	---	---	---	---	---	---	---	---	-----

FSQRT = Square Root of ST(0)

Escape	0	0	1	1	1	1	1	1	0	1	0
--------	---	---	---	---	---	---	---	---	---	---	---

FSCALE = Scale ST(0) by ST(1)

Escape	0	0	1	1	1	1	1	1	1	0	1
--------	---	---	---	---	---	---	---	---	---	---	---

FPREM = Partial Remainder of ST(0) ÷ ST(1)

Escape	0	0	1	1	1	1	1	1	0	0	0
--------	---	---	---	---	---	---	---	---	---	---	---

FRNDINT = Round ST(0) to Integer

Escape	0	0	1	1	1	1	1	1	1	0	0
--------	---	---	---	---	---	---	---	---	---	---	---

FXTRACT = Extract Components of ST(0)

Escape	0	0	1	1	1	1	1	0	1	0	0
--------	---	---	---	---	---	---	---	---	---	---	---

FABS = Absolute Value of ST(0)

Escape	0	0	1	1	1	1	0	0	0	0	1
--------	---	---	---	---	---	---	---	---	---	---	---

FCHS = Change Sign of ST(0)

Escape	0	0	1	1	1	1	0	0	0	0	0
--------	---	---	---	---	---	---	---	---	---	---	---

Transcendental

FPTAN = Partial Tangent of ST(0)

Escape	0	0	1	1	1	1	1	0	0	1	0
--------	---	---	---	---	---	---	---	---	---	---	---

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

Escape	0	0	1	1	1	1	1	0	0	1	1
--------	---	---	---	---	---	---	---	---	---	---	---

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

Escape	0	0	1	1	1	1	1	0	0	0	0
--------	---	---	---	---	---	---	---	---	---	---	---

FYL2X = ST(1) · LOG₂[ST(0)]

Escape	0	0	1	1	1	1	1	0	0	0	1
--------	---	---	---	---	---	---	---	---	---	---	---

FYL2XP1 = ST(1) · LOG₂[ST(0) + 1]

Escape	0	0	1	1	1	1	1	1	0	0	1
--------	---	---	---	---	---	---	---	---	---	---	---

Constants

FLDZ = Load + 0.0 into ST(0)

Escape	0	0	1	1	1	1	0	1	1	1	0
--------	---	---	---	---	---	---	---	---	---	---	---

FLD1 = Load + 1.0 into ST(0)

Escape	0	0	1	1	1	1	0	1	0	0	0
--------	---	---	---	---	---	---	---	---	---	---	---

FLDPI = Load π into ST(0)

Escape	0	0	1	1	1	1	0	1	0	1	1
--------	---	---	---	---	---	---	---	---	---	---	---

FLDL2T = Load log₂10 into ST(0)

Escape	0	0	1	1	1	1	0	1	0	0	1
--------	---	---	---	---	---	---	---	---	---	---	---

FLDL2E = Load log₂e into ST(0)

Escape	0	0	1	1	1	1	0	1	0	1	0
--------	---	---	---	---	---	---	---	---	---	---	---

FLDLG2 = Load log₁₀2 into ST(0)

Escape	0	0	1	1	1	1	0	1	1	0	0
--------	---	---	---	---	---	---	---	---	---	---	---

FLDLN2 = Load log_e2 into ST(0)

Escape	0	0	1	1	1	1	0	1	1	0	1
--------	---	---	---	---	---	---	---	---	---	---	---

Processor Control

FINIT = Initialize NDP

Escape	0	1	1	1	1	1	0	0	0	1	1
--------	---	---	---	---	---	---	---	---	---	---	---

FENI = Enable Interrupts

Escape	0	1	1	1	1	1	0	0	0	0	0
--------	---	---	---	---	---	---	---	---	---	---	---

FDISI = Disable Interrupts

Escape	0	1	1	1	1	1	0	0	0	0	1
--------	---	---	---	---	---	---	---	---	---	---	---

FLDCW = Load Control Word

Escape	0	0	1	mod	1	0	1	r/m	disp-low	disp-high
--------	---	---	---	-----	---	---	---	-----	----------	-----------

FSTCW = Store Control Word

Escape	0	0	1	mod	1	1	1	r/m	disp-low	disp-high
--------	---	---	---	-----	---	---	---	-----	----------	-----------

FSTSW = Store Status Word

Escape	1	0	1	mod	1	1	1	r/m	disp-low	disp-high
--------	---	---	---	-----	---	---	---	-----	----------	-----------

FCLEX = Clear Exceptions

Escape	0	1	1	1	1	1	0	0	0	1	0
--------	---	---	---	---	---	---	---	---	---	---	---

FSTENV = Store Environment

Escape	0	0	1	mod	1	1	0	r/m	disp-low	disp-high
--------	---	---	---	-----	---	---	---	-----	----------	-----------

Processor Control (Continued)

FLDENV = Load Environment

Escape	0	0	1	mod	1	0	0	r/m	disp-low	disp-high
--------	---	---	---	-----	---	---	---	-----	----------	-----------

FSAVE = Save State

Escape	1	0	1	mod	1	1	0	r/m	disp-low	disp-high
--------	---	---	---	-----	---	---	---	-----	----------	-----------

FRSTOR = Restore State

Escape	1	0	1	mod	1	0	0	r/m	disp-low	disp-high
--------	---	---	---	-----	---	---	---	-----	----------	-----------

FINCSTP = Increment Stack Pointer

Escape	0	0	1	1	1	1	1	0	1	1	1
--------	---	---	---	---	---	---	---	---	---	---	---

FDECSTP = Decrement Stack Pointer

Escape	0	0	1	1	1	1	1	0	1	1	0
--------	---	---	---	---	---	---	---	---	---	---	---

FFREE = Free ST(i)

Escape	0	0	1	1	1	0	0	0	ST(i)
--------	---	---	---	---	---	---	---	---	-------

FNOP = No Operation

Escape	0	0	1	1	1	0	1	0	0	0	0
--------	---	---	---	---	---	---	---	---	---	---	---

FWAIT = CPU Wait for NDP

1	0	0	1	1	0	1	1
---	---	---	---	---	---	---	---

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

8088 Instruction Set Matrix

LO	0	1	2	3	4	5	6	7	
HI	0	ADD b,f,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	SEG = 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 = SS	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 D1
	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/ LOOPE	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 = indirect
 is = immed. byte, sign ext.
 l = long ie. intersegment

m = memory
 r/m = EA is second byte
 si = short intrasegment
 sr = segment register
 t = to CPU reg
 v = variable
 w = word operation
 z = zero

8088 Instruction Set Matrix

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

Instruction Set Index

Mnemonic	Page	Mnemonic	Page	Mnemonic	Page
AAA	6-8	FRNDINT	6-20	JP	6-13
AAD	6-10	FRSTOR	6-23	JPE	6-13
AAM	6-9	FSAVE	6-23	JPO	6-14
AAS	6-9	FSCALE	6-20	JS	6-13
ADC	6-8	FSQRT	6-20	JZ	6-13
ADD	6-8	FST	6-17	LAHF	6-7
AND	6-10	FSTCW	6-22	LDS	6-7
CALL	6-12	FSTENV	6-22	LEA	6-7
CBW	6-10	FSTP	6-18	LES	6-7
CLC	6-16	FSTSW	6-22	LOCK	6-16
CLD	6-16	FSUB	6-19	LODS	6-12
CLI	6-16	FTST	6-19	LOOP	6-14
CMC	6-16	FWAIT	6-23	LOOPE	6-14
CMP	6-9	FXAM	6-19	LOOPNE	6-14
CMPS	6-11	FXCH	6-18	LOOPNZ	6-14
CWD	6-10	FXTRACT	6-20	LOOPZ	6-14
DAA	6-8	FYL2X	6-21	MOV	6-6
DAS	6-9	FYL2XP1	6-21	MOVS	6-11
DEC	6-9	HLT	6-16	MUL	6-9
DIV	6-9	IDIV	6-10	NEG	6-9
ESC	6-16	IMUL	6-9	NOP	6-16
F2XM1	6-21	IN	6-7	NOT	6-10
FABS	6-20	INC	6-8	OR	6-11
FADD	6-19	INT	6-15	OUT	6-7
FCHS	6-20	INTO	6-15	POP	6-6
FCLEX	6-22	IRET	6-15	POPF	6-7
FCOM	6-19	JA	6-14	PUSH	6-6
FCOMP	6-19	JAE	6-14	PUSHF	6-7
FCOMPP	6-19	JB	6-13	RCL	6-10
FDECSTP	6-23	JBE	6-13	RCR	6-10
FDISI	6-22	JCXZ	6-14	REP	6-11
FDIV	6-20	JE	6-13	RET	6-13
FENI	6-22	JG	6-14	ROL	6-10
FFREE	6-23	JGE	6-13	ROR	6-10
FINCSTP	6-23	JL	6-13	SAHF	6-7
FINIT	6-22	JLE	6-13	SAL	6-10
FLD	6-17	JMP	6-12	SAR	6-10
FLD1	6-21	JNA	6-13	SBB	6-9
FLDCW	6-22	JNAE	6-13	SCAS	6-11
FLDENV	6-23	JNB	6-14	SHL	6-10
FLDL2E	6-21	JNBE	6-14	SHR	6-10
FLD2T	6-21	JNE	6-13	STC	6-16
FLDLG2	6-21	JNG	6-13	STD	6-16
FLDLN2	6-21	JNGE	6-13	STI	6-16
FLDPI	6-21	JNL	6-13	STOS	6-12
FLDZ	6-21	JNLE	6-14	SUB	6-8
FMUL	6-20	JNO	6-14	TEST	6-11
FNOP	6-23	JNP	6-14	WAIT	6-16
FPATAN	6-21	JNS	6-14	XCHG	6-7
FPREM	6-20	JNZ	6-13	XLAT	6-7
FPTAN	6-21	JO	6-13	XOR	6-11

Notes:



SECTION 7. CHARACTERS, KEYSTROKES, AND COLORS

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

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

7-2 Characters, Keystrokes, and Colors

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

Value		As Characters			As Text Attributes		
					Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground	
4B	75	K	K	Note 4	Red	Light Cyan	High Intensity
4C	76	L	L	Note 4	Red	Light Red	High Intensity
4D	77	M	M	Note 4	Red	Light Magenta	High Intensity
4E	78	N	N	Note 4	Red	Yellow	High Intensity
4F	79	O	O	Note 4	Red	White	High Intensity
50	80	P	P	Note 4	Magenta	Black	Normal
51	81	Q	Q	Note 4	Magenta	Blue	Underline
52	82	R	R	Note 4	Magenta	Green	Normal
53	83	S	S	Note 4	Magenta	Cyan	Normal
54	84	T	T	Note 4	Magenta	Red	Normal
55	85	U	U	Note 4	Magenta	Magenta	Normal
56	86	V	V	Note 4	Magenta	Brown	Normal
57	87	W	W	Note 4	Magenta	Light Grey	Normal
58	88	X	X	Note 4	Magenta	Dark Grey	High Intensity
59	89	Y	Y	Note 4	Magenta	Light Blue	High Intensity Underline
5A	90	Z	Z	Note 4	Magenta	Light Green	High Intensity
5B	91	[[Magenta	Light Cyan	High Intensity
5C	92	\	\		Magenta	Light Red	High Intensity
5D	93]]		Magenta	Light Magenta	High Intensity
5E	94	^	^	Shift	Magenta	Yellow	High Intensity
5F	95	—	—	Shift	Magenta	White	High Intensity
60	96	·	·		Yellow	Black	Normal
61	97	a	a	Note 5	Yellow	Blue	Underline
62	98	b	b	Note 5	Yellow	Green	Normal
63	99	c	c	Note 5	Yellow	Cyan	Normal
64	100	d	d	Note 5	Yellow	Red	Normal
65	101	e	e	Note 5	Yellow	Magenta	Normal
66	102	f	f	Note 5	Yellow	Brown	Normal

7-4 Characters, Keystrokes, and Colors

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

Value		As Characters			As Text Attributes		
					Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground	
*** 80 to FF Hex are Flashing in both Color & IBM Monochrome ***							
80	128	Ç	Alt 128	Note 6	Black	Black	Non-Display
81	129	ü	Alt 129	Note 6	Black	Blue	Underline
82	130	é	Alt 130	Note 6	Black	Green	Normal
83	131	â	Alt 131	Note 6	Black	Cyan	Normal
84	132	ä	Alt 132	Note 6	Black	Red	Normal
85	133	à	Alt 133	Note 6	Black	Magenta	Normal
86	134	å	Alt 134	Note 6	Black	Brown	Normal
87	135	ç	Alt 135	Note 6	Black	Light Grey	Normal
88	136	ê	Alt 136	Note 6	Black	Dark Grey	Non-Display
89	137	ë	Alt 137	Note 6	Black	Light Blue	High Intensity Underline
8A	138	è	Alt 138	Note 6	Black	Light Green	High Intensity
8B	139	ï	Alt 139	Note 6	Black	Light Cyan	High Intensity
8C	140	î	Alt 140	Note 6	Black	Light Red	High Intensity
8D	141	ì	Alt 141	Note 6	Black	Light Magenta	High Intensity
8E	142	Á	Alt 142	Note 6	Black	Yellow	High Intensity
8F	143	À	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	ó	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	õ	Alt 153	Note 6	Blue	Light Blue	High Intensity Underline
9A	154	ü	Alt 154	Note 6	Blue	Light Green	High Intensity





7-6 Characters, Keystrokes, and Colors

Value		As Characters			As Text Attributes		
					Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground	
9B	155	¢	Alt 155	Note 6	Blue	Light Cyan	High Intensity
9C	156	£	Alt 156	Note 6	Blue	Light Red	High Intensity
9D	157	¥	Alt 157	Note 6	Blue	Light Magenta	High Intensity
9E	158	Pt	Alt 158	Note 6	Blue	Yellow	High Intensity
9F	159	∫	Alt 159	Note 6	Blue	White	High Intensity
A0	160	á	Alt 160	Note 6	Green	Black	Normal
A1	161	í	Alt 161	Note 6	Green	Blue	Underline
A2	162	ó	Alt 162	Note 6	Green	Green	Normal
A3	163	ú	Alt 163	Note 6	Green	Cyan	Normal
A4	164	ñ	Alt 164	Note 6	Green	Red	Normal
A5	165	Ñ	Alt 165	Note 6	Green	Magenta	Normal
A6	166	<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	ı	Alt 173	Note 6	Green	Light Magenta	High Intensity
AE	174	<<	Alt 174	Note 6	Green	Yellow	High Intensity
AF	175	>>	Alt 175	Note 6	Green	White	High Intensity
B0	176	⋮	Alt 176	Note 6	Cyan	Black	Normal
B1	177	⋱	Alt 177	Note 6	Cyan	Blue	Underline
B2	178	⋴	Alt 178	Note 6	Cyan	Green	Normal
B3	179		Alt 179	Note 6	Cyan	Cyan	Normal
B4	180		Alt 180	Note 6	Cyan	Red	Normal
B5	181		Alt 181	Note 6	Cyan	Magenta	Normal
B6	182		Alt 182	Note 6	Cyan	Brown	Normal

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

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

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

- NOTE 1 Asterisk (*) can easily be keyed using two methods:
1) hit the  key or 2) in shift mode hit the  key.
- NOTE 2 Period (.) can easily be keyed using two methods:
1) hit the  key or 2) in shift or Num Lock mode hit the  key.
- NOTE 3 Numeric characters (0—9) can easily be keyed using two methods: 1) hit the numeric keys on the top row of the typewriter portion of the keyboard or 2) in shift or Num Lock mode hit the numeric keys in the 10—key pad portion of the keyboard.
- NOTE 4 Upper case alphabetic characters (A—Z) can easily be keyed in two modes: 1) in shift mode the appropriate alphabetic key or 2) in Caps Lock mode hit the appropriate alphabetic key.
- NOTE 5 Lower case alphabetic characters (a—z) can easily be keyed in two modes: 1) in “normal” mode hit the appropriate key or 2) in Caps Lock combined with shift mode hit the appropriate alphabetic key.
- NOTE 6 The 3 digits after the Alt key must be typed from the numeric key pad (keys 71—73, 75—77, 79—82). Character codes 000 through 255 can be entered in this fashion. (With Caps Lock activated, Character codes 97 through 122 will display upper case rather than lower case alphabetic characters.)

Character Set (00-7F) 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	I	k	{	
12	C	♀ ⊥	,	<	L	\	l	!	
13	D	🎵 ↔	—	=	M	I	m	}	
14	E	🎵 ▲	.	>	N	^	n	~	
15	F	☀ ▼	/	?	O	_	o	△	

Character Set (80-FF) Quick Reference

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

Notes:



SECTION 8. COMMUNICATIONS

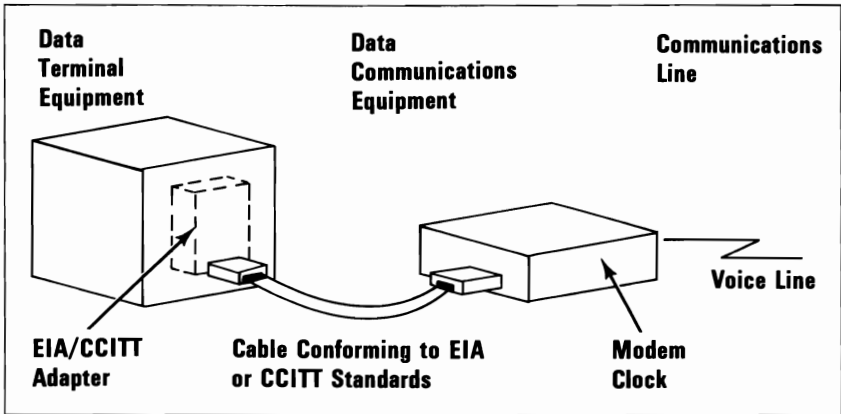
Contents

Communications	8-3
Establishing a Communications Link	8-5
Establishing Link on Nonswitched Point-to-Point Line	8-6
Establishing Link on Nonswitched Multipoint Line	8-8
Establishing Link on Switched Point-to-Point Line	8-10



Information processing equipment used for communications is called data terminal equipment (DTE). Equipment used to connect the DTE to the communications line is called data communications equipment (DCE).

An adapter is used to connect the data terminal equipment to the data communications line as shown in the following illustration:



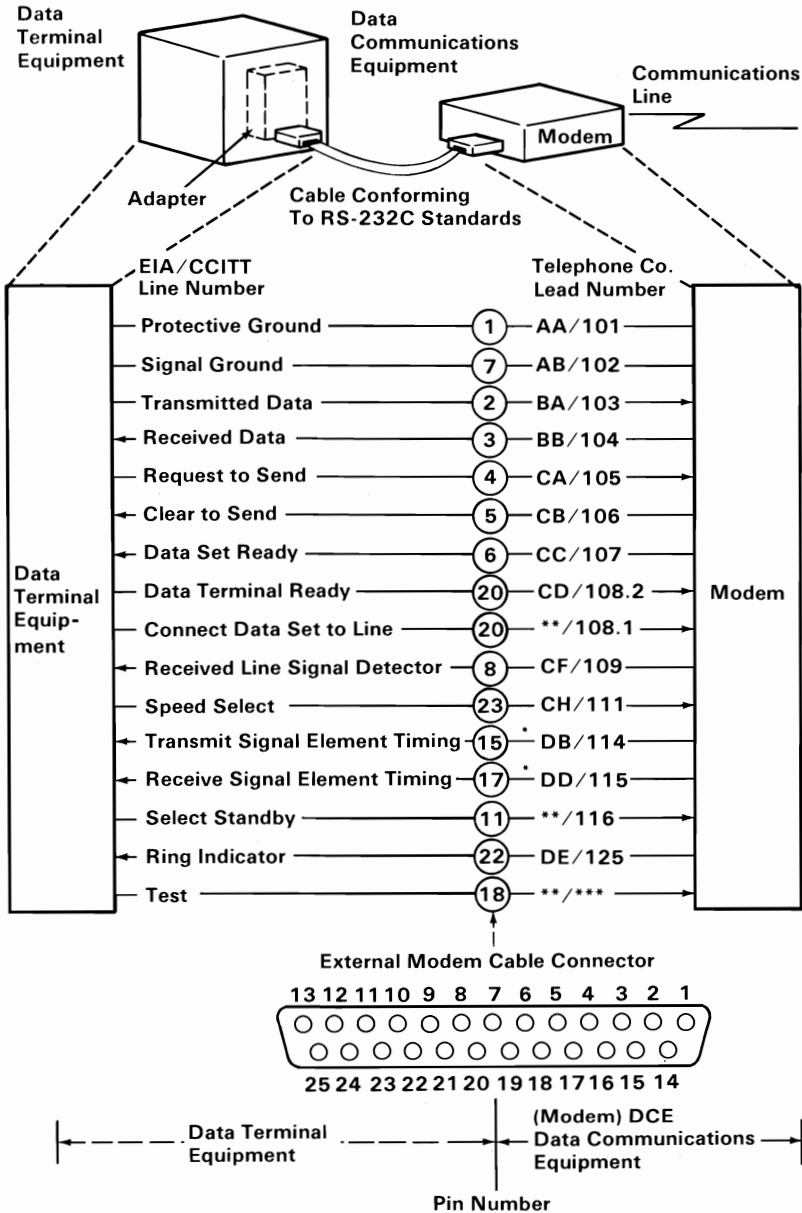
The EIA/ CCITT adapter allows data terminal equipment to be connected to data communications equipment using EIA or CCITT standardized connections. An external modem is shown in this example; however, other types of data communications equipment can also be connected to data terminal equipment using EIA or CCITT standardized connections.

EIA standards are labeled RS-x (Recommended Standards-x) and CCITT standards are labeled V.x or X.x, where x is the number of the standard.

The EIA RS-232 interface standard defines the connector type, pin numbers, line names, and signal levels used to connect data terminal equipment to data communications equipment for the purpose of transmitting and receiving data. Since the RS-232 standard was developed, it has been revised three times. The three revised standards are the RS-232A, the RS-232B, and the presently used RS-232C.

The CCITT V.24 interface standard is equivalent to the RS-232C standard; therefore, the descriptions of the EIA standards also apply to the CCITT standards.

The following is an illustration of data terminal equipment connected to an external modem using connections defined by the RS-232C interface standard:



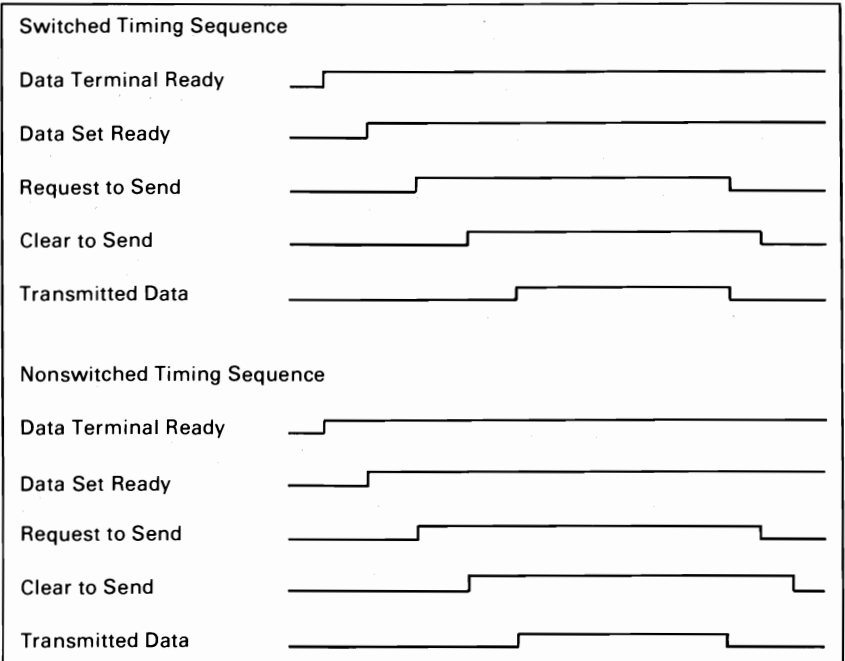
* Not used when business machine clocking is used.

** Not standardized by EIA (Electronic Industries Association).

*** Not standardized by CCITT

Establishing a Communications Link

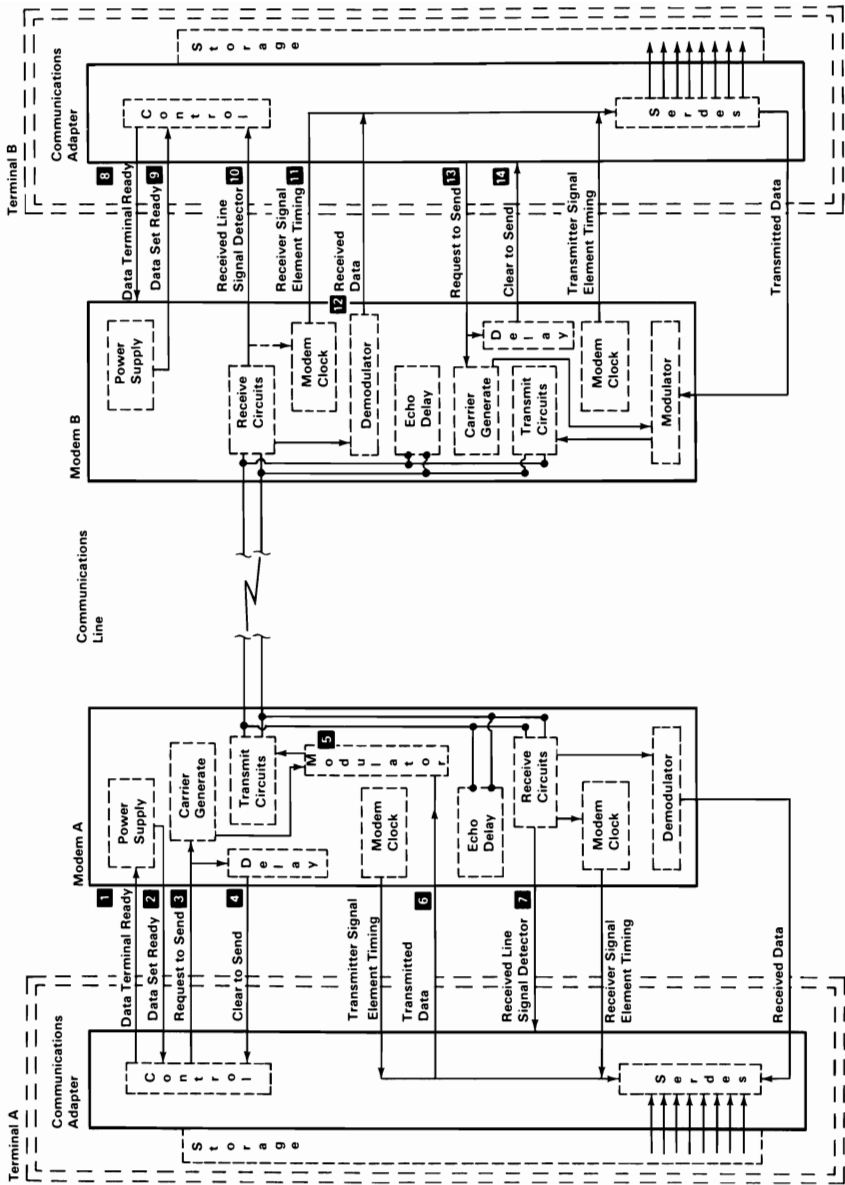
The following bar graphs represent normal timing sequences of operation during the establishment of communications for both switched (dial-up) and nonswitched (direct line) networks.



The following examples show how a link is established on a nonswitched point-to-point line, a nonswitched multipoint line, and a switched point-to-point line.

Establishing a Link on a Nonswitched Point-to-Point Line

- The terminals at both locations activate the 'data terminal ready' lines **1** and **8**.
- Normally the 'data set ready' lines **2** and **9** from the modems are active whenever the modems are powered on.
- Terminal A activates the 'request to send' line **1**, which causes the modem at terminal A to generate a carrier signal.
- Modem B detects the carrier, and activates the 'received line signal detector' line (sometimes called data carrier detect) **10**. Modem B also activates the 'receiver signal element timing' line (sometimes called receive clock) **11** to send receive clock signals to the terminal. Some modems activate the clock signals whenever the modem is powered on.
- After a specified delay, modem A activates the 'clear to send' line **4**, which indicates to terminal A that the modem is ready to transmit data.
- Terminal A serializes the data to be transmitted (through the serdes) and transmits the data one bit at a time (synchronized by the transmit clock) onto the 'transmitted data' line **6** to the modem.
- The modem modulates the carrier signal with the data and transmits it to the modem B **5**.
- Modem B demodulates the data from the carrier signal and sends it to terminal B on the 'received data' line **12**.
- Terminal B deserializes the data (through the serdes) using the receive clock signals (on the 'receiver signal element timing' line) **11** from the modem.
- After terminal A completes its transmission, it deactivates the 'request to send' line **3**, which causes the modem to turn off the carrier and deactivate the 'clear to send' line **4**.
- Terminal A and modem A now become receivers and wait for a response from terminal B, indicating that all data has reached terminal B. Modem A begins an echo delay (50 to 150 milliseconds) to ensure that all echoes on the line have diminished before it begins receiving. An echo is a reflection of the transmitted signal. If the transmitting modem changed to receive too soon, it could receive a reflection (echo) of the signal it just transmitted.
- Modem B deactivates the 'received line signal detector' line **10** and, if necessary, deactivates the receive clock signals on the 'receiver signal element timing' line **11**.
- Terminal B now becomes the transmitter to respond to the request from terminal A. To transmit data, terminal B activates the 'request to send' line **13**, which causes modem B to transmit a carrier to modem A.
- Modem B begins a delay that is longer than the echo delay at modem A before turning on the 'clear to send' line. The longer delay (called request-to-send delay) ensures that modem A is ready to receive when terminal B begins transmitting data. After the delay, modem B activates the 'clear to send' line **14** to indicate that terminal B can begin transmitting its response.
- After the echo delay at modem A, modem A senses the carrier from modem B (the carrier was activated in step 13 when terminal B activated the 'request to send' line) and activates the 'received line signal detector'; line **7** to terminal A.
- Modem A and terminal A are ready to receive the response from terminal B. Remember, the response was not transmitted until after the request-to-send to clear-to-send delay at modem B (step 14).

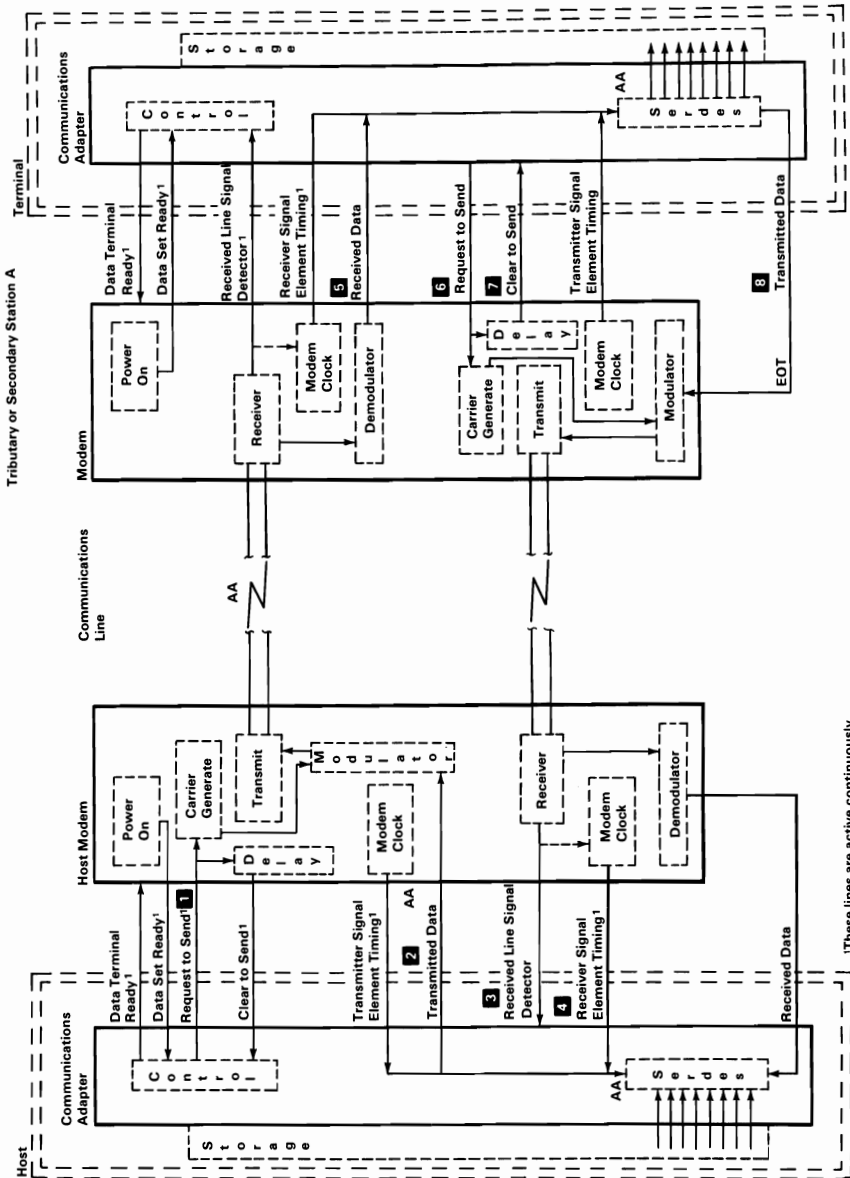


Section 8

Establishing a Link on a Nonswitched Multipoint Line

1. The control station serializes the address for the tributary or secondary station (AA) and sends its address to the modem on the 'transmitted data' line **2**.
 2. Since the 'request to send' line and, therefore, the modem carrier, is active continuously **1**, the modem immediately modulates the carrier with the address, and, thus, the address is transmitted to all modems on the line.
 3. All tributary modems, including the modem for station A, demodulate the address and send it to their terminals on the 'received data' line **5**.
 4. Only station A responds to the address; the other stations ignore the address and continue monitoring their 'received data' line. To respond to the poll, station A activates its 'request to send' line **6**, which causes the modem to begin transmitting a carrier signal.
 5. The control station's modem receives the carrier and activates the 'received line signal detector' line **3** and the 'receiver signal element timing' line **4** (to send clock signals to the control station). Some modems activate the clock signals as soon as they are powered on.
6. After a short delay to allow the control station modem to receive the carrier, the tributary modem activates the 'clear to send' line **7**.
 7. When station A detects the active 'clear to send' line, it transmits its response. (For this example, assume that station A has no data to send; therefore, it transmits an EOT **8**.)
 8. After transmitting the EOT, station A deactivates the 'request to send' line **6**. This causes the modem to deactivate the carrier and the 'clear to send' line **7**.
 9. When the modem at the control station (host) detects the absence of the carrier, it deactivates the 'received line signal detector' line **3**.
 10. Tributary station A is now in receive mode waiting for the next poll or select transmission from the control station.



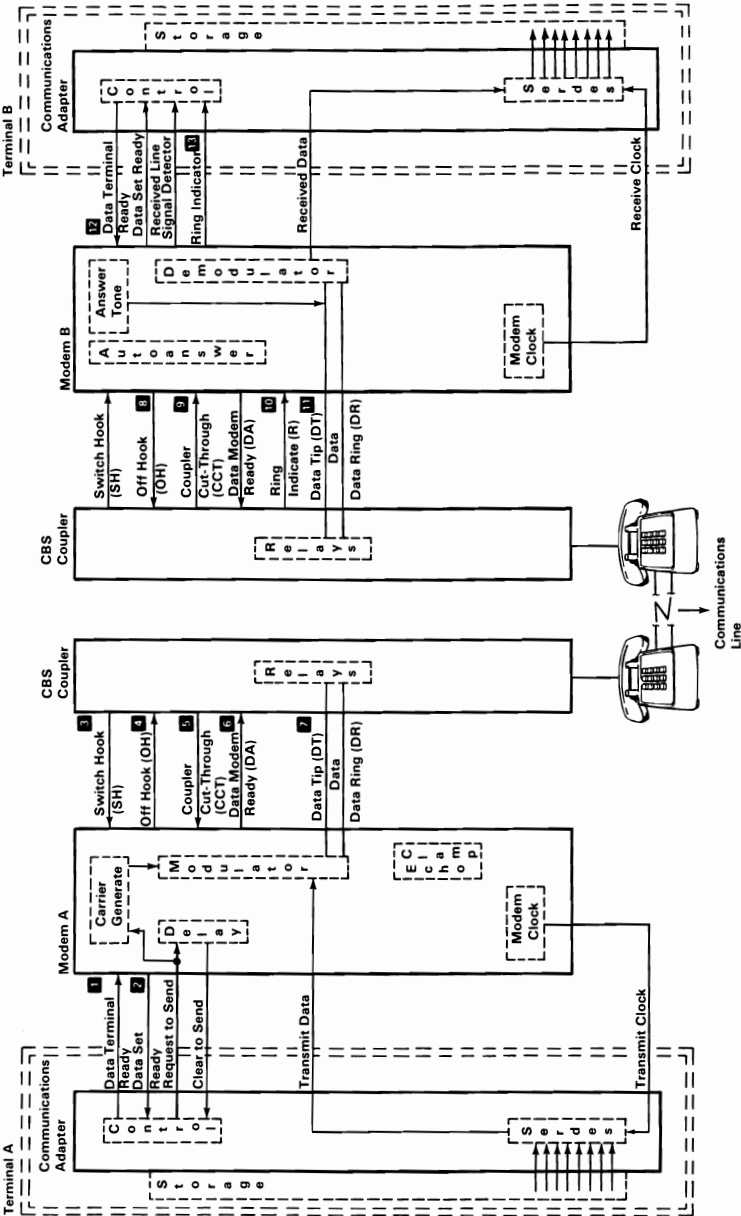


¹These lines are active continuously.

Establishing a Link on a Switched Point-To-Point Line

1. Terminal A is in communications mode; therefore, the 'data terminal ready' line **1** is active. Terminal B is in communication mode waiting for a call from terminal A.
2. When the terminal A operator lifts the telephone handset, the 'switch hook' line from the coupler is activated **3**.
3. Modem A detects the 'switch hook' line and activates the 'off hook' line **4**, which causes the coupler to connect the telephone set to the line and activate the 'coupler cut-through' line **5** to the modem.
4. Modem A activates the 'data modem ready' line **6** to the coupler (the 'data modem ready' line is on continuously in some modems).
5. The terminal A operator sets the exclusion key or talk/data switch to the talk position to connect the handset to the communications line. The operator then dials the terminal B number.
6. When the telephone at terminal B rings, the coupler activates the 'ring indicate' line to modem B **10**. Modem B indicates that the 'ring indicate' line was activated by activating the 'ring indicator' line **13** to terminal B.
7. Terminal B activates the 'data terminal ready' line to modem B **12** which activates the autoanswer circuits in modem B. (The 'data terminal ready' line might already be active in some terminals.)
8. The autoanswer circuits in modem B activate the 'off hook' line to the coupler **8**.
9. The coupler connects modem B to the communications line through the 'data tip' and 'data ring' lines **11** and activates the 'coupler cut-through' line **9** to the modem. Modem B then transmits an answer tone to terminal A.
10. The terminal A operator hears the tone and sets the exclusion key or talk/data switch to the data position (or performs an equivalent operation) to connect modem A to the communications line through the 'data tip' and 'data ring' lines **7**.
11. The coupler at terminal A deactivates the 'switch hook' line **3**. This causes modem A to activate the 'data set ready' line **2** indicating to terminal A that the modem is connected to the communications line.
The sequence of the remaining steps to establish the data link is the same as the sequence required on a nonswitched point-to-point line. When the terminals have completed their transmission, they both deactivate the 'data terminal ready' line to disconnect the modems from the line.





Notes:



Glossary

μ. Prefix micro; 0.000 001.

μs. Microsecond; 0.000 001 second.

A. Ampere.

ac. Alternating current.

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

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

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

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

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

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

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

alphameric. Synonym for alphanumeric.

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

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

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

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

A/N. Alphanumeric

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

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

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

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

APA. All points addressable.

ASCII. American National Standard Code for Information Exchange.

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

assembler. A computer program used to assemble.

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

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

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

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

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

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

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

BCC. Block-check character.

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

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

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

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

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

BIOS. Basic input/output system.

bit. Synonym for binary digit

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

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

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

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

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

bps. Bits per second.

BSC. Binary synchronous communications.

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

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

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

C. Celsius.

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

CAS. Column address strobe.

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

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

CCITT. International Telegraph and Telephone Consultative Committee.

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

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

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

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

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

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

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

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

CMOS. Complementary metal oxide semiconductor.

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

coding scheme. Synonym for code.

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

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

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

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

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

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

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

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

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

conjunction. Synonym for AND operation.

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

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

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

control storage. A portion of storage that contains microcode.

cps. Characters per second.

CPU. Central processing unit.

CRC. Cyclic redundancy check.

CRT. Cathode ray tube.

CRT display. Cathode ray tube display.

CTS. Clear to send. Associated with modem control.

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

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

cylinder. (1) The set of all tracks with the same nominal distance from the axis about which the disk rotates. (2) The tracks of a disk storage device that can be accessed without repositioning the access mechanism.

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

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

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

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

data transmission. Synonym for transmission.

dB. Decibel.

dBa. Adjusted decibels.

dc. Direct current.

debounce. An electronic means of overcoming the make/break bounce of switches to obtain one smooth change of signal level.

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

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

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

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

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

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

DIN. Deutsche Industrie Norm.

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

DIP. Dual in-line package.

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

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

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

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

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

disk. Loosely, a magnetic disk.

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.

DMA. Direct memory access.

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

dot printer. Synonym for matrix printer.

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

DSR. Data set ready. Associated with modem control.

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

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

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

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

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

EBCDIC. Extended binary-coded decimal interchange code.

ECC. Error checking and correction.

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

EIA. Electronic Industries Association.

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

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

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

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

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

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

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

EOB. End of block.

EOF. End of file.

EOT. End-of-transmission.

EPROM. Erasable programmable read-only memory.

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

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

ESC. The escape character.

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

ETB. End-of-transmission-block.

ETX. End-of-text.

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

F. Fahrenheit.

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

falling edge. Synonym for negative-going edge.

FCC. Federal Communications Commission.

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

FF. The form feed character.

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

fixed disk 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.

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

g. Gram.

G. (1) Prefix giga; 1 000 000 000. (2) When referring to computer storage capacity, 1 073 741 824. (1 073 741 824 = 2 to the 30th power.)

gate. (1) A combinational logic circuit having one output channel and one or more input channels, such that the output channel state is completely determined by the input channel states. (2) A signal that enables the passage of other signals through a circuit.

Gb. 1 073 741 824 bytes.

general-purpose register. A register, usually explicitly addressable within a set of registers, that can be used for different purposes; for example, as an accumulator, as an index register, or as a special handler of data.

giga (G). Prefix 1 000 000 000.

gram (g). A unit of weight (equivalent to 0.035 ounces).

graphic. A symbol produced by a process such as handwriting, drawing, or printing.

graphic character. A character, other than a control character, that is normally represented by a graphic.

half-duplex. (1) In data communication, pertaining to an alternate, one way at a time, independent transmission. (2) Contrast with duplex.

hardware. (1) Physical equipment used in data processing, as opposed to programs, procedures, rules, and associated documentation. (2) Contrast with software.

head. A device that reads, writes, or erases data on a storage medium; for example, a small electromagnet used to read, write, or erase data on a magnetic disk.

hertz (Hz). A unit of frequency equal to one cycle per second.

hex. Common abbreviation for hexadecimal.

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

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

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

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

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

Hz. Hertz

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

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

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

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

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

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

input/output (I/O). (1) Pertaining to a device or to a channel that may be involved in an input process, and, at a different time, in an output process. In the English language, "input/output" may be used in place of such terms 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.

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

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

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

I/O. Input/output.

I/O area. Synonym for buffer.

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

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

k. Prefix kilo; 1000.

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

Kb. 1024 bytes.

kg. Kilogram; 1000 grams.

kHz. Kilohertz; 1000 hertz.

kilo (k). Prefix 1000

kilogram (kg). 1000 grams.

kilohertz (kHz). 1000 hertz

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

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

LED. Light-emitting diode.

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

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

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

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

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

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

mA. Milliampere; 0.001 ampere.

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

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

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

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

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

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

masked. Synonym for disabled.

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

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

Mb. 1 048 576 bytes.

mega (M). Prefix 1 000 000.

megahertz (MHz). 1 000 000 hertz.

memory. Term for main storage.

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

MFM. Modified frequency modulation.

MHz. Megahertz; 1 000 000 hertz.

micro (μ). Prefix 0.000 001.

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

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

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

microsecond (μ s). 0.000 001 second.

milli (m). Prefix 0.001.

milliamperere (mA). 0.001 ampere.

millisecond (ms). 0.001 second.

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

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

modem (modulator-demodulator). A device that converts serial (bit by bit) digital signals from a business machine (or data

communication equipment) to analog signals that are suitable for transmission in a telephone network. The inverse function is also performed by the modem on reception of analog signals.

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

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

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

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

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

monitor. (1) A device that observes and verifies the operation of a data processing system and indicates any significant departure from the norm. (2) Software or hardware that observes, supervises, controls, or verifies the operations of a system.

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

ms. Millisecond; 0.001 second.

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

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

n. Prefix nano; 0.000 000 001.

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

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

nano (n). Prefix 0.000 000 001.

nanosecond (ns). 0.000 000 001 second.

negative true. Synonym for active low.

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

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

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

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

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

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

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

ns. Nanosecond; 0.000 000 001 second.

NUL. The null character.

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

odd-even check. Synonym for parity check.

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

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

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

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

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

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

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

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

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

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

overcurrent. A current of higher than specified strength.

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

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

overvoltage. A voltage of higher than specified value.

parallel. (1) Pertaining to the concurrent or simultaneous operation of two or more devices, or to the concurrent performance of two or more activities. (2) Pertaining to the concurrent or simultaneous occurrence of two or more related activities in multiple devices or channels. (3) Pertaining to the

simultaneity of two or more processes. (4) Pertaining to the simultaneous processing of the individual parts of a whole, such as the bits of a character and the characters of a word, using separate facilities for the various parts. (5) Contrast with serial.

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

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

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

PEL. Picture element.

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

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

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

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

port. An access point for data entry or exit.

positive true. Synonym for active high.

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

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

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

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

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

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

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

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

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

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

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

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

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

PROM. Programmable read-only memory.

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

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

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

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

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

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

RAM. Random access memory. Read/write memory.

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

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

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

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

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

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

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

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

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

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

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

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

RF. Radio frequency.

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

RGBI. Red-green-blue-intensity.

rising edge. Synonym for positive-going edge.

ROM. Read-only memory.

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

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

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

RTS. Request to send. Associated with modem control.

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

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

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

SDLC. Synchronous Data Link Control

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

SERDES. Serializer/deserializer.

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

serializer/deserializer (SERDES). A device that serializes output from, and deserializes input to, a business machine.

setup. (1) In a computer that consists of an assembly of individual computing units, the arrangement of interconnections between the units, and the adjustments needed for the computer to operate. (2) The preparation of a computing system to perform a job or job step. Setup is usually performed by an operator and often involves performing routine functions, such as mounting tape reels. (3) The preparation of the system for normal operation.

short circuit. A low-resistance path through which current flows, rather than through a component or circuit.

signal. A variation of a physical quantity, used to convey data.

sink. A device or circuit into which current drains.

software. (1) Computer programs, procedures, and rules concerned with the operation of a data processing system. (2) Contrast with hardware.

source. The origin of a signal or electrical energy.

square wave. An alternating or pulsating current or voltage whose waveshape is square.

square wave generator. A signal generator delivering an output signal having a square waveform.

SS. Start-stop.

start bit. (1) A signal to a receiving mechanism to get ready to receive data or perform a function. (2) In a start-stop system, a signal preceding a character or block that prepares the receiving device for the reception of the code elements.

start-of-text (STX). A transmission control character that precedes a text and may be used to terminate the message heading.

start-stop system. A data transmission system in which each character is preceded by a start bit and is followed by a stop bit.

start-stop (SS) transmission. (1) Asynchronous transmission such that a group of signals representing a character is preceded by a start bit and followed by a stop bit. (2) Asynchronous transmission in which a group of bits is preceded by a start bit that prepares the receiving mechanism for the reception and registration of a character and is followed by at least one stop bit that enables the receiving mechanism to come to an idle condition pending the reception of the next character.

static memory. RAM 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.

V. Volt.

video. Computer data or graphics displayed on a cathode ray tube, monitor, or display.

volt. The basic practical unit of electric pressure. The potential that causes electrons to flow through a circuit.

W. Watt.

watt. The practical unit of electric power.

word. (1) A character string or a bit string considered as an entity. (2) See computer word.

write. To make a permanent or transient recording of data in a storage device or on a data medium.

write precompensation. The varying of the timing of the head current from the outer tracks to the inner tracks of the diskette to keep a constant 'write' signal.



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.



Index

Special Characters

- MEMR (memory read command) 1-22
- MEMW (memory write command) 1-22

A

adapter card with ROM 5-13

address

- bits 0 to 19 (A0–A19), I/O channel 1-20

- enable (AEN), I/O channel 1-20

- latch enable (ALE), I/O channel 1-20

- map, I/O 1-24

AEN (address enable), I/O channel 1-20

ALE (address latch enable), I/O channel 1-20

B

BASIC reserved interrupts 5-8, 5-9

BASIC,

- DEF SEG 5-11

- reserved interrupt 5-8, 5-9

binary integers (coprocessor) 2-3, 2-4

BIOS,

- parameter passing 5-4

- quick reference 5-29

- software interrupt 5-6

- system ROM 5-29

use of 5-3
bit map, I/O 8255A 1-31
block diagram (coprocessor) 2-6
break (keyboard extended code) 5-21

C

CCITT 8-3
standards 8-3
CH CK, negative (-channel check), I/O channel 1-21
channel check, negative (-CH CK), I/O channel 1-21
character codes (keyboard) 5-15, 5-17
character set
quick reference 7-12
clock (CLK), I/O channel 1-20
communications 8-3
component diagram, system board 1-16

D

data
bits 0 to 7 (D0–D7) 1-21
flow, system board diagram 1-5
decimal integers (coprocessor) 2-3, 2-4
description I/O channel 1-20
diagram system board 1-16
diagram, I/O channel 1-18
DMA request 1 to 3 (DRQ1–DRQ3) 1-21
DOS,
keyboard function 5-8, 5-14, 5-22
keyboard functions 5-24

E

- EIA 8-3
 - standards 8-3
- establishing a communications link 8-5
 - establish a link 8-5

I

I/O channel

- address map 1-24
 - ALE (address latch enable) 1-20
 - bit map 8255A 1-31
 - CH CK (-I/O channel check) 1-21
 - CH RDY (I/O Channel Ready), I/O channel 1-21
 - check (-CH CK) 1-21
 - CLK 1-20
 - description 1-20
 - I/O channel diagram 1-18
 - oscillator (OSC) 1-23
 - read command (-IOR) 1-22
 - reset drive (RESET DRV) 1-23
 - terminal count (T/C) 1-23
 - write command (-IOW) 1-22
- Intel 8048 4-3
- Intel 8088 microprocessor, 6-17, instruction set extensions
- 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
 - 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

K

keyboard 4-3

- block diagram 4-4
- connector 4-12
- interface 4-4
- keyboard scan 4-3
- power-on self-test 4-3

keyboard extended codes,

- alt 5-19
- break 5-21
- caps lock 5-20
- ctrl 5-19
- encoding 5-14
- pause 5-21
- print screen 5-21
- scroll lock 5-20
- shift 5-19
- system reset 5-20

L

logic diagrams, system-board 1-36

M

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 5-9

memory map, BIOS 5-13

memory read command (-MEMR) 1-22

memory write command (-MEMW) 1-22

N

NMI (coprocessor) 2-5

O

OSC (oscillator), I/O channel 1-23

oscillator (OSC), I/O channel 1-23

P

- parameter passing (ROM BIOS) 5-4
 - software interrupt listing 5-6
- pause (keyboard extended code) 5-21

Q

- QS0 (coprocessor) 2-4
- QS1 (coprocessor) 2-4

R

- read command I/O channel 1-22
- read memory command (-MEMR) 1-22
- ready (RDY), I/O channel 1-21
- real numbers (coprocessor) 2-3, 2-4
- request interrupt 2 to 7 (IRQ2–IRQ7) 1-22
- reserved interrupts,
 - BASIC and DOS 5-8, 5-9
- RESET DRV, I/O channel 1-23

S

- screen editor keyboard function 5-24
- scroll lock (keyboard extended code) 5-20
- shift (key priorities (keyboard code) 5-20
- shift (keyboard extended code) 5-17, 5-19
- shift states (keyboard code) 5-19
- signals (I/O),
 - DACK0–DACK3 1-21

-I/O CH CK 1-21
-IOR 1-22
-IOW 1-22
-MEMR 1-22
-MEMW 1-22
AEN 1-20
ALE 1-20
A0–A19 1-20
CLK 1-20
DRQ1–DRQ3 1-21
D0–D7 1-21
I/O CH RDY 1-21
IRQ2–IRQ7 1-22
OSC 1-23
RESET DRV 1-23
T/C 1-23
software interrupt listing (8088) 5-6
speaker circuit 1-25
speaker drive system 1-25
system board
 data flow diagrams 1-5
 diagram 1-16
 logic diagrams 1-36
system clock (CLK), I/O channel 1-20
system reset 5-20
system ROM BIOS 5-29

T

terminal count (T/C), I/O channel 1-23
typematic 4-3

V

vectors with special meanings 5-6

W

write command (-IOW), I/O channel 1-22
write memory command (-MEMW) 1-22

Numerals

8088, (see Intel 8088 microprocessor) 1-3
8255A bit map 1-31
specifications I/O channel 1-34



Reader's Comment Form

Technical Reference
IBM Personal Computer

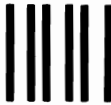
6361453

Your comments assist us in improving the usefulness of our publication; they are an important part of the input used for revisions.

IBM may use and distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Please do not use this form for technical questions regarding the IBM Personal Computer or programs for the IBM Personal Computer, or for requests for additional publications; this only delays the response. Instead, direct your inquiries or request to your authorized IBM Personal Computer dealer.

Comments:



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 321 BOCA RATON, FLORIDA 33432



POSTAGE WILL BE PAID BY ADDRESSEE

IBM PERSONAL COMPUTER
SALES & SERVICE
P.O. BOX 1328-C
BOCA RATON, FLORIDA 33432

.....
Fold here

Tape

Please do not staple

Tape