



*Personal Computer PCjr
Hardware Reference
Library*

Technical Reference





*Personal Computer PCjr
Hardware Reference
Library*

Technical Reference

First Edition Revised (November 1983)

Changes are periodically made to the information herein; these changes will be incorporated in new editions of this publication.

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.

A Reader's Comment Form is provided at the back of this publication. If this form has been removed, address comments to IBM Corporation, 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.

© Copyright International Business Machines Corporation 1983

FEDERAL COMMUNICATIONS COMMISSION RADIO FREQUENCY INTERFERENCE STATEMENT

Warning: This equipment has been certified to comply with the limits for a Class B computing device, pursuant to Subpart J of Part 15 of FCC rules. Only peripherals (computer input/output devices, terminals, printers, etc.) certified to comply with the Class B limits may be attached to this computer. Operation with non-certified peripherals is likely to result in interference to radio and TV reception.

INSTRUCTIONS TO USER

This equipment generates and uses radio frequency energy and if not installed and used properly, i.e., in strict accordance with the operating instructions, reference manuals, and the service manual, may cause interference to radio or television reception. It has been tested and found to comply with the limits for a Class B computing device pursuant to Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference when operated in a residential installation.

If this equipment does cause interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient the receiving antenna.
- Relocate the equipment with respect to the receiver.
- Move the equipment away from the receiver.
- Plug the equipment into a different outlet so that equipment and receiver are on different branch circuits.
- Ensure that side option mounting screws, attachment connector screws, and ground wires are tightly secured.
- If peripherals not offered by IBM are used with this equipment, it is suggested that you use shielded, grounded cables with in-line filters, if necessary.

If necessary, consult your dealer service representative for additional suggestions.

The manufacturer is not responsible for any radio or TV interference caused by unauthorized modifications to this equipment. It is the responsibility of the user to correct such interference.

CAUTION

This product is equipped with a UL listed and CSA-certified plug for the user's safety. It is to be used in conjunction with a properly grounded 115 Vac receptacle to avoid electrical shock.

Preface

The IBM PC_{jr} Technical Reference manual describes the hardware design and provides interface information for the IBM PC_{jr}. This publication also has information about the basic input/output system (BIOS) and programming support.

The information in this publication is both descriptive and reference oriented, and is intended for hardware and software designers, programmers, engineers, and interested persons who need to understand the design and operation of the IBM PC_{jr} computer.

You should be familiar with the use of the IBM PC_{jr}, and understand the concepts of computer architecture and programming.

This manual has five sections:

Section 1: "Introduction" is an overview of the basic system and available options.

Section 2: "Base System" describes each functional part of the base system. This section also has specifications for power, timing, and interfaces. Programming considerations are supported by coding tables, command codes, and registers.

Section 3: "System Options" describes each available option using the same format as Section 2: "Base System."

Section 4: “Compatibility with the IBM Personal Computer Family” describes programming concerns for maintaining compatibility between the IBM PCjr and the other IBM Personal Computers.

Section 5: “System BIOS and Usage” describes the basic input/output system (BIOS) and its use. This section also contains the software interrupt listing, a system memory map, descriptions of vectors with special meanings, and a set of low-storage maps. In addition, keyboard encoding and usage is discussed.

This publication has four appendixes:

Appendix A: “ROM BIOS Listing”

Appendix B: “Logic Diagrams”

Appendix C: “Characters, Keystrokes, and Color”

Appendix D: “Unit Specifications”

Prerequisite Publication:

Guide to Operations part number 1502291

Guide to Operations part number 1502292

Suggested Reading:

IBM PCjr Hands on BASIC part number 1504702

IBM PCjr BASIC Reference Manual part number 6182371

Disk Operating System (DOS) part number 6024061

Hardware Maintenance and Service Manual part number 1502294

Macro Assembler part number 6024002

Related publications are listed in “Bibliography.”

Contents

SECTION 1. INTRODUCTION	1-1
Introduction	1-3
SECTION 2. BASE SYSTEM	2-1
Introduction	2-5
Processor and Support	2-13
Performance	2-13
8259A Interrupt Controller	2-15
PCjr Hardware Interrupts	2-15
8259A Programming Considerations .	2-16
64K RAM	2-17
ROM Subsystem	2-19
Input/Output Channel	2-21
System Board I/O Channel Description	2-23
Input/Output	2-29
8255 Bit Assignments	2-30
Cassette Interface	2-39
Video Color Graphics Subsystem	2-43
Major Components Definitions	2-47
Palette	2-50
Alphanumeric Modes	2-54
Graphics Mode	2-55
Video Gate Array	2-63
Light Pen	2-74
CRT/Processor Page Register	2-79
Beeper	2-85
Sound Subsystem	2-87
Complex Sound Generator	2-88
Audio Tone Generator	2-89
Infra-Red Link	2-97
Infra-Red Receiver	2-97
IBM PCjr Cordless Keyboard	2-101
Transmitter	2-103

Program Cartridge and Interface	2-107
Program Cartridge Slots	2-107
Cartridge Storage Allocations	2-108
ROM Module	2-114
Games Interface	2-119
Interface Description	2-119
Input from Address Hex 201	2-120
Pushbuttons	2-122
Joystick Positions	2-122
Serial Port (RS232)	2-125
Modes of Operation	2-128
Interrupts	2-129
Interface Description	2-129
Voltage Interchange Information ...	2-130
System Power Supply	2-135
Operating Characteristics	2-136
Over-Voltage/Over-Current Protection	2-137

SECTION 3. SYSTEM OPTIONS 3-1

IBM PCjr 64KB Memory and Display Expansion	3-5
IBM PCjr Diskette Drive Adapter	3-13
Functional Description	3-15
System I/O Channel Interface	3-19
Drive Interface	3-22
Voltage and Current Requirements ..	3-24
IBM PCjr Diskette Drive	3-27
Functional Description	3-27
Diskette	3-31
IBM PCjr Internal Modem	3-33
Functional Description	3-34
Modem Design Parameters	3-37
Programming Considerations	3-40
Status Conditions	3-60
Dialing and Loss of Carrier	3-60
Default State	3-63
Programming Examples	3-63

Modes of Operation	3-68
Interrupts	3-70
Data Format	3-70
Interfaces	3-70
IBM PCjr Attachable Joystick	3-77
Hardware Description	3-77
Functional Description	3-77
IBM Color Display	3-81
Hardware Description	3-81
Operating Characteristics	3-82
IBM Connector for Television	3-85
IBM PCjr Keyboard Cord	3-87
IBM PCjr Adapter Cable for Serial Devices	3-89
IBM PCjr Adapter Cable for Cassette ...	3-91
IBM PCjr Adapter Cable for the IBM Color Display	3-93
IBM PCjr Parallel Printer Attachment ...	3-95
Description	3-96
System Interface	3-98
Programming Considerations	3-99
IBM Graphics Printer	3-107
Printer Specifications	3-107
Additional Printer Specifications ..	3-109
DIP Switch Settings	3-110
Parallel Interface Description	3-112
Printer Modes	3-115
Printer Control Codes	3-116
IBM PC Compact Printer	3-133
Printer Specifications	3-135
Serial Interface Description	3-139
Print Mode Combinations for the PC Compact Printer	3-140
Printer Control Codes and Functions	3-140

SECTION 4. COMPATIBILITY WITH THE IBM PERSONAL COMPUTER FAMILY	4-1
Compatibility Overview	4-3
Timing Dependencies	4-5
Unequal Configurations	4-7

Hardware Differences	4-9
User Ready/Write Memory	4-12
Diskette Capacity/Operation	4-13
IBM PCjr Cordless Keyboard	4-14
Color Graphics Capability	4-15
Black and White Monochrome Display	4-18
RS232 Serial Port and IBM PCjr	
Internal Modem	4-18
Summary	4-19
SECTION 5. SYSTEM BIOS USAGE	5-1
ROM BIOS	5-3
BIOS Usage	5-5
Vectors with Special Means	5-8
Other Read/Write Memory Usage	5-13
BIOS Programming Guidelines	5-18
Adapter Cards with System-Accessible	
ROM-Modules	5-18
Keyboard Encoding and Usage	5-21
Cordless keyboard Encoding	5-21
Special Handling	5-34
Non-Keyboard Scan-Code Architecture	5-42
BIOS Cassette Logic	5-47
Software Algorithms - Interrupt	
Hex 15	5-47
Cassette Write	5-48
Cassette Read	5-49
Data Record Architecture	5-50
Error Detection	5-51
Appendix A. ROM BIOS LISTING	A-1
Equates and Data Areas	A-3
Power-On Self-Test	A-7
Boot Strap Loader	A-26
Non-Keyboard Scan-Code Table	A-38
Time-of-Day	A-42
Graphics-Character Generator	
(Second 128 Characters)	A-54

I/O Support	A-97
System Configuration Analysis	A-97
Graphics-Character Generator (First 128 Characters)	A-103
Print Screen	A-108
Appendix B. LOGIC DIAGRAMS	B-1
System Board	B-3
Program Cartridge	B-20
Power Supply Board	B-23
64KB Memory and Display Expansion	B-25
Color Display	B-29
Diskette Drive Adapter	B-30
Internal Modem	B-36
Parallel Printer Attachment	B-37
Infra-Red Receiver Board	B-42
Graphics Printer	B-43
Compact Printer	B-47
Appendix C. CHARACTERS, KEYSTROKES, and COLOR	C-1
Appendix D. UNIT SPECIFICATIONS	D-1
System Unit	D-1
Cordless Keyboard	D-2
Diskette Drive	D-3
Color Display	D-5
Graphics Printer	D-6
Internal Modem	D-7
Compact Printer	D-8
Glossary	Glossary-1
Bibliography	Bibliography-1
Index	Index-1

Notes:

TAB INDEX

Section 1: Introduction

Section 2: Base System

Section 3: System Options

**Section 4: Compatibility With the IBM Personal
Computer Family**

Section 5: System BIOS Usage

Appendix A: ROM BIOS Listing

Introduction

Base System

System Options

Compatibility

BIOS Usage

Appendix A

Notes:

Appendix B: Logic Diagram

Appendix B

Appendix C: Characters, Keystrokes, and Color

Appendix C

Appendix D: Unit Specifications

Appendix D

Glossary

Glossary

Bibliography

Bibliography

Index

Index

Notes:

SECTION 1. INTRODUCTION

Contents

Introduction 1-3

Notes:

Introduction

The system unit, a desk top transformer, and a cordless keyboard make up the hardware for the *PCjr* base system.

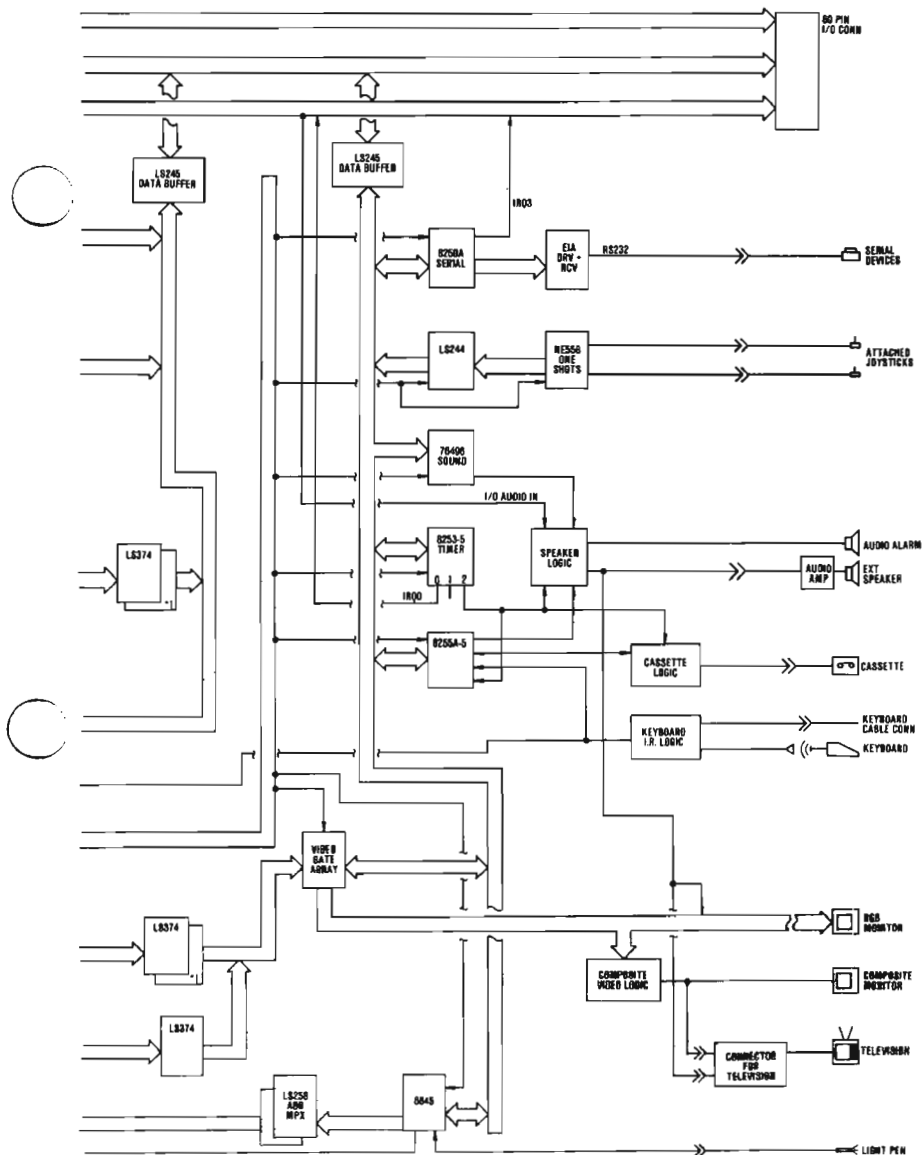
The following options are available for the base system:

- IBM *PCjr* 64KB Memory and Display Expansion
 - The 64KB Memory and Display Expansion enables the user to work with the higher density video modes while increasing the system's memory size by 64K Bytes to a total of 128K Bytes.
- IBM *PCjr* Diskette Drive Adapter
 - The IBM *PCjr* Diskette Drive Adapter permits the attachment of the IBM *PCjr* Diskette Drive to the IBM *PCjr* and resides in a dedicated connector on the IBM *PCjr* system board.
- IBM *PCjr* Diskette Drive
 - The IBM *PCjr* Diskette Drive is double-sided with 40 tracks for each side, is fully self-contained, and consists of a spindle drive system, a read positioning system, and a read/write/erase system.
- IBM *PCjr* Internal Modem
 - The IBM *PCjr* Internal Modem is an adapter that plugs into the *PCjr* system board modem connector and allows communications over standard telephone lines.

- **IBM PCjr Parallel Printer Attachment**
 - The IBM PCjr Parallel Printer Attachment is provided to attach various I/O devices that accept eight bits of parallel data at standard TTL logic levels. It attaches as a feature to the right side of the system unit.
- **IBM Personal Computer Graphics Printer**
 - IBM Graphics Printer is an 80 cps (characters-per-second), self-powered, stand-alone, tabletop unit.
- **IBM PCjr Joystick**
 - The IBM PCjr Joystick is an input device to provide the user with two-dimensional positioning-control. Two pushbutton switches on the joystick give the user additional input capability.
- **IBM Color Display**
 - The IBM Color Display is a Red/Green/Blue /Intensity (RGBI) Direct-Drive display, that is independently housed and powered.
- **IBM Connector for Television**
 - The IBM Connector for Television allows a TV to be connected to the IBM PCjr system.
- **IBM PCjr Keyboard Cord**
 - The IBM PCjr Keyboard Cord option is used to connect the IBM PCjr Cordless Keyboard to the system board.

- **IBM PC_{jr} Adapter Cable for Serial Devices**
 - This option is an adapter cable that allows connection of serial devices to the IBM PC_{jr} system board.
- **IBM PC_{jr} Adapter Cable for Cassette**
 - This option is an adapter cable that allows a cassette recorder to be connected to the IBM PC_{jr}.
- **IBM PC_{jr} Adapter Cable for Color Display**
 - This adapter cable allows the IBM Color Display to be connected to the IBM PC_{jr}.

The following is a block diagram of the IBM PC_{jr} system.



*1 These components are contained on the 84KE memory and display expansion card. They are included here for completeness.

System Block Diagram (Sheet 2 of 2)

Notes:

SECTION 2. BASE SYSTEM

Contents

Introduction	2-5
Processor and Support	2-13
Performance	2-13
8259A Interrupt Controller	2-15
PCjr Hardware Interrupts	2-15
8259A Programming Considerations	2-16
64K RAM	2-17
ROM Subsystem	2-19
Input/Output Channel	2-21
System Board I/O Channel Description	2-23
Input/Output	2-29
8255 Bit Assignments	2-30
8255 Bit Assignment Description	2-31
Port A0 Output Description	2-35
Port A0 Input Operation	2-36
Cassette Interface	2-39
Video Color/Graphics Subsystem	2-43
Major Components Definitions	2-47
Motorola 6845 CRT Controller	2-47
Storage Organization	2-47
Bandwidth	2-49
Character Generator	2-49
Video Gate Array	2-49
Palette	2-50

Alphanumeric Modes	2-54
Graphics Mode	2-55
Low-Resolution 16-Color Graphics	2-56
Medium-Resolution 4-Color Graphics ...	2-57
Medium-Resolution 16-Color Graphics ..	2-58
High-Resolution 2-Color Graphics	2-58
High-Resolution 4-Color Graphics	2-59
Graphics Storage Organization	2-60
Video Gate Array	2-63
Mode Control 1 Register	2-64
Palette Mask Register	2-65
Border Color Register	2-66
Mode Control 2 Register	2-66
Reset Register	2-69
Palette Registers	2-71
Status Register	2-73
Light Pen	2-74
Programming Considerations	2-75
CRT/Processor Page Register	2-79
Beeper	2-85
Sound Subsystem	2-87
Complex Sound Generator	2-88
Audio Tone Generator	2-89
Features	2-89
Infra-Red Link	2-97
Infra-Red Receiver	2-97
Functional Description	2-97
Application Notes	2-98
Programming Considerations	2-99
Detectable Error Conditions	2-99
Operational Parameters	2-100
IBM PCjr Cordless Keyboard	2-101
Transmitter	2-103

Program Cartridge and Interface	2-107
Program Cartridge Slots	2-107
Cartridge Storage Allocations	2-108
ROM Module	2-114
Games Interface	2-119
Interface Description	2-119
Input from Address hex 201	2-120
Pushbuttons	2-122
Joystick Positions	2-122
Serial Port (RS232)	2-125
Modes of Operation	2-128
Interrupts	2-129
Interface Description	2-129
Voltage Interchange Information	2-130
Output Signals	2-131
Accessible Registers	2-131
INS8250A Programmable Baud Rate Generator	2-132
System Power Supply	2-135
Operating Characteristics	2-136
Power Supply Input Requirements	2-136
DC Outputs	2-136
Over-Voltage/Over-Current Protection ...	2-137
Input (Transformer)	2-137
Output (Power Board)	2-137

Notes:

Introduction

The *PCjr* base-system hardware consists of the system unit, a 62-key cordless-keyboard, and a power transformer.

The *PCjr* system board is the center of the *PCjr* system unit. The system board fits horizontally in the base of the system unit and is approximately 255 mm by 350 mm (10 inches by 13.8 inches). It is double-sided, with an internal-power/ground plane. Low voltage ac power enters the power supply adapter, is converted to dc voltage, and enters the system board through the power supply adapter edge-connector. Other system board connectors provide interfaces for a variety of input/output (I/O) devices and are individually keyed to prevent improper installation. The following is a list of these connectors:

- 64KB Memory and Display Expansion Connector
- Diskette Drive Adapter Connector
- Internal Modem Connector
- Infra-Red (IR) Link Receiver Board Connector
- Program Cartridge Connectors (2)
- I/O Channel Expansion Connector
- Serial Port (RS232) Connector (with optional adapter cable)
- Direct Drive (RGBI) Video Connector
- Composite Video Connector
- IBM Connector for Television Connector (external RF modulator)
- Light Pen Connector
- External Audio Connector
- IBM *PCjr* Keyboard Cord Connector
- Cassette Connector (with optional adapter cable)
- IBM *PCjr* Attachable Joystick Connectors (2)

The system board consists of seven functional subsystems: the processor subsystem and its support elements, the read-only (ROM) subsystem, the read/write (R/W) subsystem, the audio subsystem, the video subsystem, the games subsystem, and the I/O channel. All are described in this section.

The nucleus of the system board is the Intel 8088 microprocessor. This processor is an 8-bit external bus version of Intel's 16-bit 8086 processor, and is software-compatible with the 8086. The 8088 supports 16-bit operations, including multiplication and division, and supports 20 bits of addressing (1 megabyte of storage). It operates in the minimum mode at 4.77 MHz. This frequency, which is derived from a 14.31818-MHz crystal, is divided by 3 for the processor clock, and by 4 to obtain the 3.58-MHz color-burst signal required for color televisions.

For additional information about the 8088, refer to the publications listed in "Bibliography".

The processor is supported by a set of high-function support-devices providing three 16-bit timer-counter channels, and nine prioritized-interrupt levels.

The three programmable timer/counters are provided by an Intel 8253-5 programmable interval-timer and are used by the system in the following manner: 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 deserialize the keyboard data and for time-of-day overflow during diskette operations. Channel 2 is used to support the tone generation for the audio speaker and to write data to the cassette.

Of the nine prioritized levels of interrupt, three are bused to the system's I/O channel for use by adapters. Five levels are used on the system board. Level 0, the

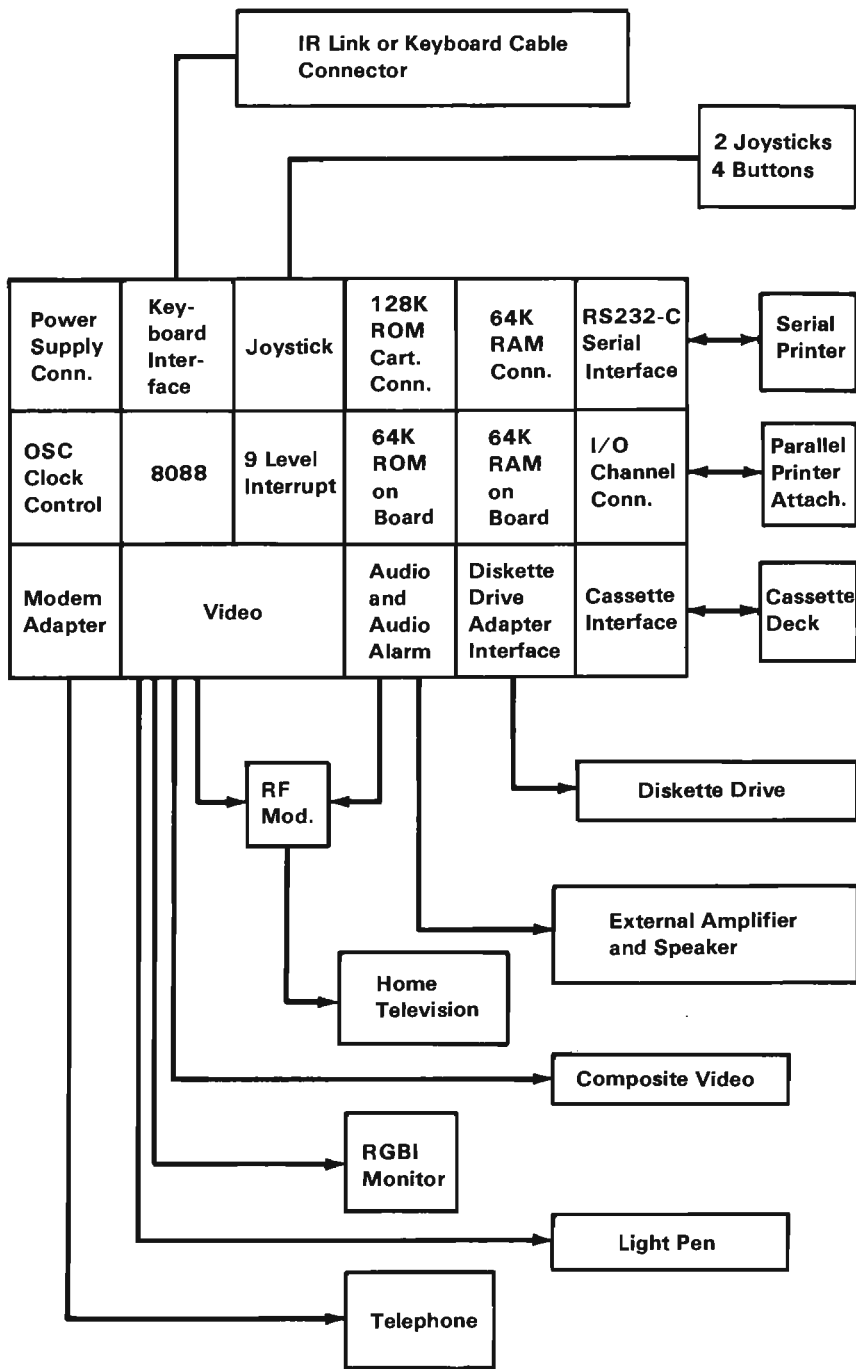
highest priority, is attached to Channel 0 of the timer/counter and provides a periodic interrupt for the time-of-day clock; level 3 is the serial-port-access interrupt; level 4 is the modem-access interrupt; level 5 is the vertical-retrace interrupt for the video; and level six is the diskette drive adapter-access interrupt. The non-maskable interrupt (NMI) of the 8088 is attached to the keyboard-interface circuits and receives an interrupt for each scan code sent by the keyboard.

The system board supports both read-only memory (ROM) and R/W memory (RAM). It has space for 64K bytes by 8 bits of ROM. There are two module sockets that accept a 32K byte by 8 bit ROM module. ROM is aligned at the top of the 8088's address space. This ROM contains the Power-On Self-Test, cassette-BASIC interpreter, cassette-operating system, I/O drivers, dot patterns for 256 characters in graphics mode, a diskette bootstrap-loader and user-selectable diagnostic-routines.

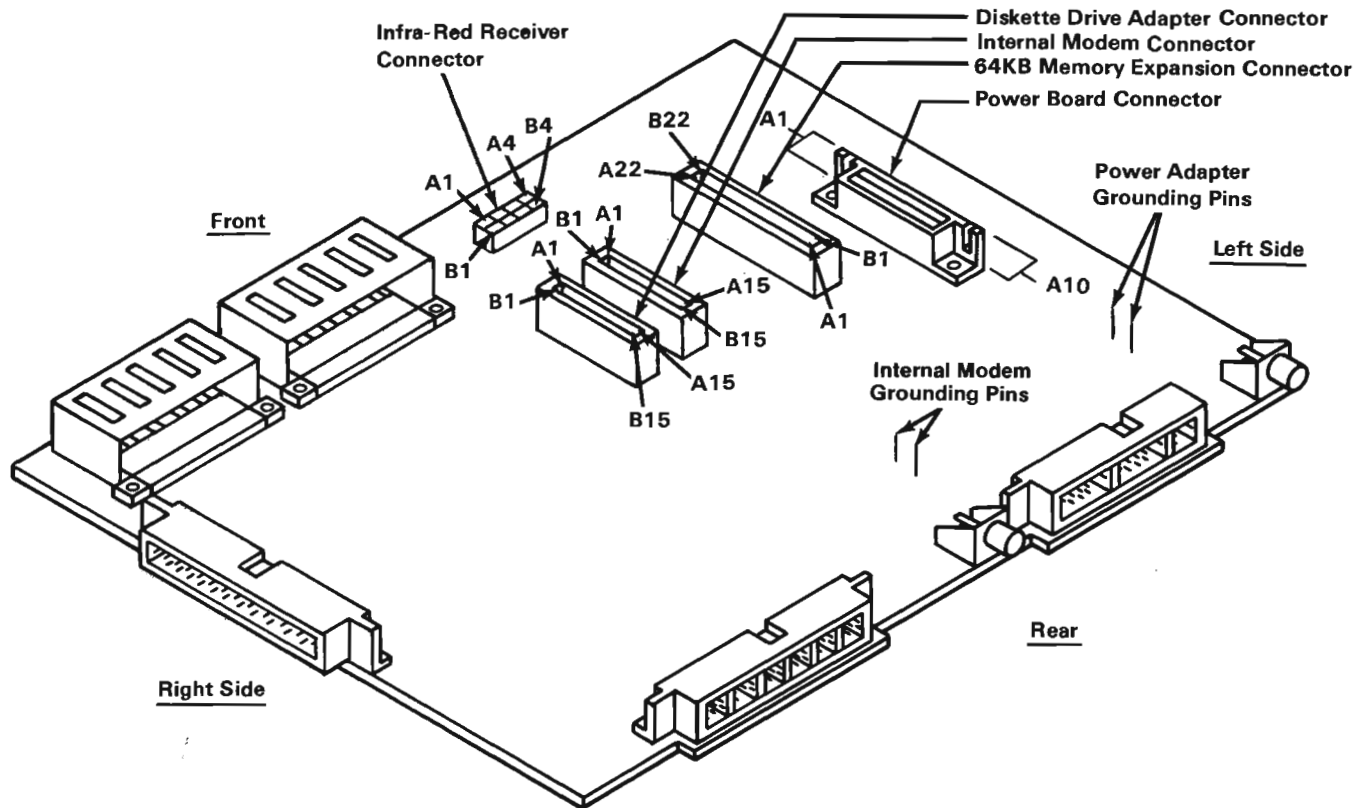
The system board contains the following major functional components:

- 8088 Microprocessor
- 64K ROM
- 128K ROM Cartridge Interface
- 64K Dynamic RAM
- 64KB Memory and Display Expansion Interface
- Serial Port (RS232)
- Audio Alarm (Beeper)
- Sound Subsystem
- Cassette Interface
- Joystick Interface
- Keyboard Interface
- Modem Interface
- Diskette Interface
- Video/Graphics Subsystem
- Light Pen Interface
- I/O Expansion Bus
- 9-Level Interrupt

The following is a block diagram of the System Board.

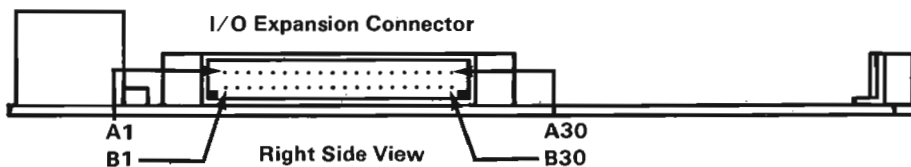
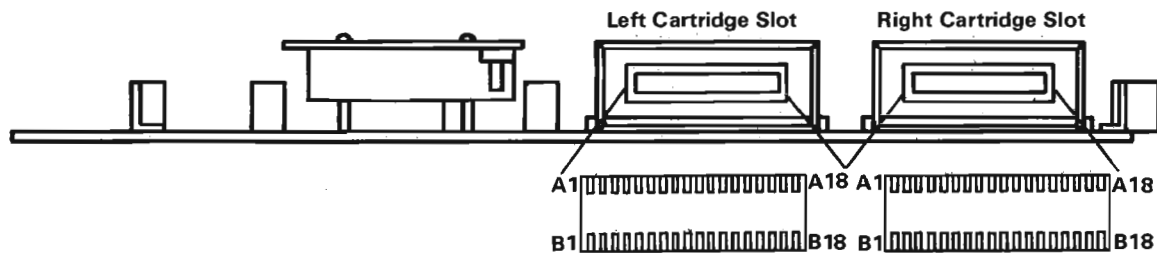


System Board Block Diagram



System Board Connector Specifications (Part 1 of 3)

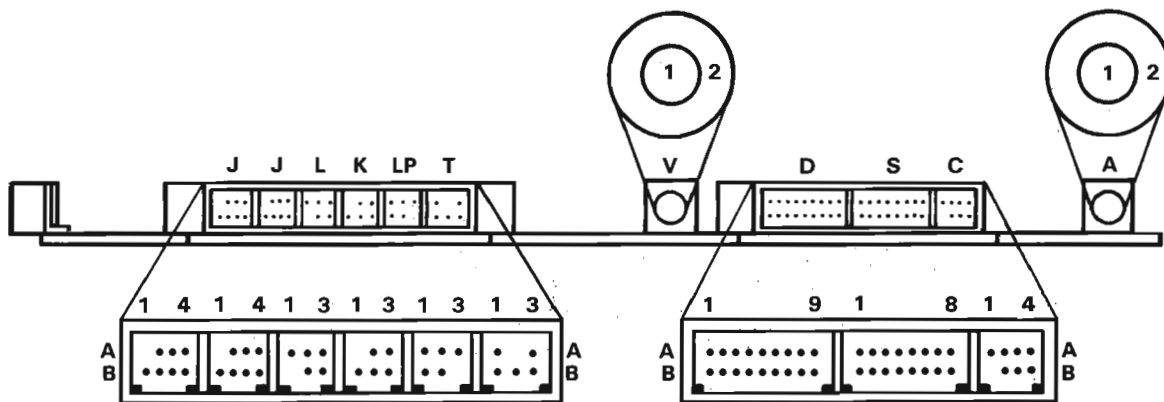
Front View



System Board Connector Specifications (Part 2 of 3)

Letter Designation	Connector Use
J	Left Joystick
J	Right Joystick
L	Spare
K	Keyboard
LP	Light Pen
T	Television

Letter Designation	Connector Use
V	Composite Video
D	Direct Drive Video
S	Serial Device
C	Cassette
A	Audio



System Board Connector Specifications (Part 3 of 3)

Processor and Support

The (R) Intel 8088 Microprocessor is used as the system's central processor. Some of its characteristics are:

- 4.77 MHz clock
- 20 bit address bus
- 8-bit memory interface
- 16-bit ALU (arithmetic/logic unit) and registers
- Extensive instruction set
- DMA and interrupt capabilities
- Hardware fixed-point multiply and divide

The system clock is provided by one Intel 8284A clock chip. The 8088 is operated in the minimum mode.

Performance

The 8088 is operated at 4.77 MHz which results in a clock cycle-time of 210 ns.

Normally four clock cycles are required for a bus cycle so that an 840 ns ROM memory cycle time is achieved. RAM write and read cycles will incur an average of two wait states because of sharing with video, leading to an average of six clock cycles. I/O reads and writes also take six clock cycles leading to a bus cycle time of 1.260 μ s.

Notes:

—

—

—

8259A Interrupt Controller

PCjr Hardware Interrupts

Nine hardware levels of interrupts are available for the PCjr system. The highest-priority interrupt is the NMI interrupt in the 8088. The NMI is followed by eight prioritized interrupt-levels (0-7) in the 8259A Programmable Interrupt Controller, with IRQ 0 as the highest and IRQ 7 as the lowest. The interrupt level assignments follow:

Level		Function
8088	NMI	Keyboard Interrupt
8259A	IRQ 0	Timer Clock Interrupt
8259A	IRQ 1	I/O Channel (Reserved)
8259A	IRQ 2	I/O Channel
8259A	IRQ 3	Asynchronous Port Interrupt (RS-232C)
8259A	IRQ 4	Modem Interrupt
8259A	IRQ 5	Vertical Retrace Interrupt (Display)
8259A	IRQ 6	Diskette Interrupt
8259A	IRQ 7	I/O Channel (Parallel Printer)

Hardware Interrupts

8259A Programming Considerations

The 8259A is set up with the following characteristics:

- Buffered Mode
- 8086 Mode
- Edge Triggered Mode
- Single Mode Master (No Cascading is Allowed)

The 8259A I/O is located at I/O address hex 20 and hex 21. The 8259A is set up to issue interrupt types hex 8 to hex F which use pointers to point to memory address hex 20 to hex 3F.

The following figure is an example setup.

0263	BO 13	MOV AL, 13H	; ICW1 - Reset edge sense circuit set single ; 8259 Chip and ICW4 read
0265	E6 20	OUT INTA00,AL	
0267	BO 08	MOV AL,8	; ICW2 - Set interrupt type 8 (8-F)
0269	E6 21	OUT INTA01,AL	
026B	BO 09	MOV AL,9	; ICW4 - Set buffered mode/master and 8086 mode
026D	E6 21	OUT INTA01,AL	

Example Set Up

64K RAM

The 64K bytes of R/W memory reside on the system board and require no user configuration.

Eight 64K byte by 1, 150 ns, dynamic memory modules are used to provide 64K byte of storage. The RAM has no parity. Sources of these memory modules include the Motorola MCM6665AL15 and the Texas Instruments TMS4164-15 or equivalent.

The system board 64K RAM is mapped at the bottom of the 1 MEG address space. The system board 64K RAM is mapped to the next 64K bytes of address space if the 64KB Memory and Display Expansion option is not installed. If read or written to, this higher block of address space will look just like the low-order 64K-byte block. This means the bottom 128K bytes of address space is always reserved for RAM. If the 64KB Memory and Display Expansion option is installed, it is mapped to the 'ODD' memory space within the 128K byte-reserved space while the system board memory is mapped to the 'EVEN' space. Memory refresh is provided by the 6845 CRT Controller and gate array. The gate array cycles the RAM and resolves contention between the CRT and processor cycles.

See "IBM PCjr 64KB Memory and Display Expansion" in Section 3 for a detailed description.

Notes:

ROM Subsystem

The ROM subsystem is made up of 64K bytes of ROM aligned at the top of the 1 MEG address space. The ROM is built using 32K byte by 8 ROM-modules. The ROM has no parity. The general memory specifications for the ROM are:

Access Time - 250 ns
Cycle Time - 375 ns

ROM modules Mk 38000 from Mostek, TMM23256P or equivalent are used. Address A14 is wired to both pin 1 and pin 27.

The following figure is a map of the sections of memory allocated for use by the system:

BIOS/Diagnostic/Cassette Basic Program Area	FFFFF	} Cartridge Chip Selects
Standard Application Cartridge	F0000	
Standard Application Cartridge	E8000	
Reserved For Future Cartridge	E0000	
Reserved For Future Cartridge	D8000	
Reserved for I/O ROM	D0000	
Video RAM	C0000	
Reserved Future Video	B8000	
Reserved Future User RAM	A0000	
Expansion RAM	20000	
Base RAM	10000	
	00000	

Memory Map

Input Output Channel

The Input/Out channel (I/O) is an extension of the 8088 microprocessor bus. It is however, demultiplexed, repowered, and enhanced by the addition of interrupts.

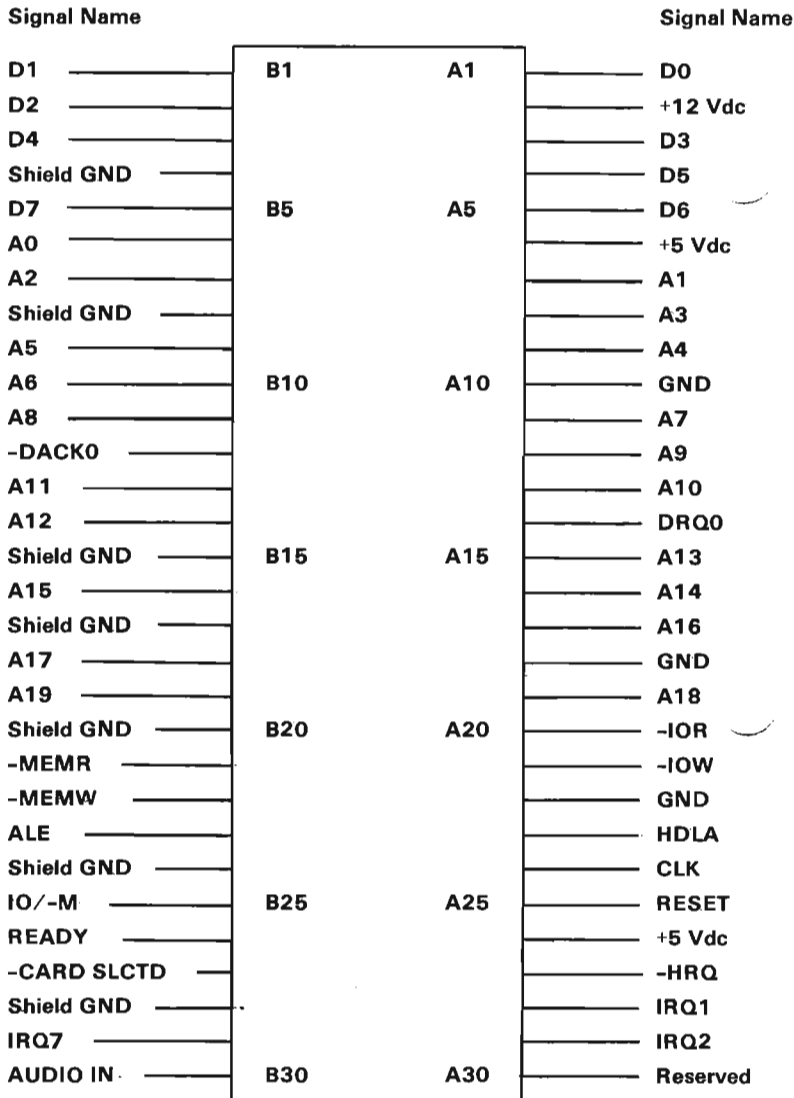
The I/O channel contains an 8-bit bidirectional bus, 20 address lines, 3 levels of interrupt, control lines for memory and I/O read or write, clock and timing lines, and power and ground for the adapters. Voltages of +5 dc and +12 dc are provided for external adapters. Any additional power needs will require a separate power-module.

All I/O Channel functions are bused to the right-hand side of the system unit and are provided by a right-angle, 60-pin connector. Each external adapter connects to the I/O bus and passes the bus along for the next attachment.

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 processor-generated memory-read and write cycles take four 210-ns clocks or 840-ns/byte. All processor-generated I/O-read or write cycles require six clocks for a cycle time of 1.26- μ s/byte.

The I/O Channel also contains the capability to add bus masters to the channel. These devices could be DMA devices or alternate processors.

The I/O Channel signals have sufficient drive to support five I/O Channel expansion-adapters and the internal modem and diskette drive adapter, assuming one standard TTL load per attachment. For information on power available for external adapters, see "System Power Supply", later in this Section.



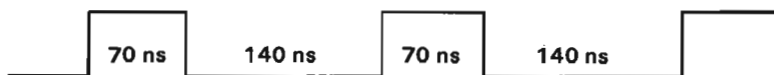
I/O Channel Expansion Connector Specifications

System Board I/O Channel Description

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

Signal	I/O	Description
--------	-----	-------------

CLK	O	System Clock: It is a divide-by-three of the 14.31818 MHz oscillator and has a period of 210 ns (4.77 MHz). The clock has a 33% duty cycle.
-----	---	---



Duty Cycle

RESET	O	This line is used to reset or initialize system logic upon power-up. This line is synchronized to the falling edge of the clock and is 'active high'. Its duration upon power up is 26.5 μ s.
-------	---	---

A0-A19	I/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 1 megabyte of memory. A0 is the least-significant-bit (LSB) while A19 is the most-significant-bit (MSB). These lines are normally driven by the 8088 microprocessor as
--------	-----	---

outputs, but can become inputs from an external bus-master by issuing an HRQ and receiving an HLDA.

D0-D7	I/O	Data Bits 0-7: These lines provide data-bus bits 0 to 7 for the processor, memory, and I/O devices. D0 is the least-significant-bit (LSB) and D7 is the most-significant-bit (MSB). These lines can be controlled by an external bus-master by issuing an HRQ and receiving an HLDA.
ALE	O	Address Latch Enable: This line is provided to allow the addition of wait states in memory and I/O cycles.
READY	I	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 requiring this line should drive it 'low' immediately upon detecting a valid address and IO/-M signal. Machine cycles (I/O and memory) are extended by an integral number of CLK cycles (210 ns). Any bus master on the I/O Channel should also honor this 'ready' line. It is pulled 'low' by the system board

on memory read and write cycles and outputting to the sound subsystem.

- IRQ1, IRQ2, IRQ7** I Interrupt Request 1, 2, and 7: These lines are used to signal the processor that an I/O device requires attention. They are prioritized with IRQ1 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 processor (interrupt-service routine).
- IOR** I/O I/O Read Command: This command line instructs an I/O device to drive its data onto the data bus. This signal may be driven by the 8088 microprocessor or by an external bus-master after it has gained control of the bus. This line is active 'low'.
- IOW** I/O I/O Write Command: This command line instructs an I/O device to read the data on the data bus. This signal may be driven by the 8088 microprocessor or by an external bus-master after it has gained control of the bus. This line is active 'low'.
- MEMR** I/O Memory Read Command: This command line instructs the

memory to drive its data onto the data bus. This signal may be driven by the 8088 microprocessor or by an external bus-master after it has gained control of the bus. This line is active 'low'.

-MEMW I/O **Memory Write Command:** This command line instructs the memory to store the data present on the data bus. This signal may be driven by the 8088 microprocessor or by an external bus-master after it has gained control of the bus. This line is active low.

IO/-M I/O **I/O or Memory Status:** This status line is used to distinguish a memory access from an I/O access. This line should be driven by a bus master after it has gained control of the bus. If this line is 'high' it indicates an I/O Address is on the Address Bus; if this line is 'low', it indicates a memory address is on the Address Bus.

-HRQ I **Hold Request:** This line indicates that another bus master is requesting the I/O Channel. To gain bus-master status, a device on the channel must assert -HRQ (active 'low'). The 8088 will respond to a -HRQ by asserting an HLDA. After receiving an HLDA, the new bus master may

control the bus, and must continue to assert the **-HRQ** until it is ready to relinquish the bus. A **-HRQ** is not an asynchronous signal and should be synchronized to the system clock. All channel devices with bus-master capabilities must latch data-bit D4 during any 'Out' instruction to A0-A7. The resulting signal should be used to qualify **-HRQ** as follows: Latched value = 1 --> **-HRQ** is inhibited. Latched value = 0 --> **-HRQ** is allowed. For more detail, see the explanation of the A0 port.

DRQ 0	0	This line comes from the floppy disk controller (FDC) and can be used by an external DMA to indicate that a byte should be transferred to the FDC.
-DACK 0	I	This line should come from an external DMA and should indicate that a byte is being transferred from memory to the FDC.
HLDA	O	Hold Acknowledge: This line indicates to a bus master on the channel that -HRQ has been honored and that the 8088 has floated its bus and control lines.

-CARD I This line should be pulled down
SLCTD by any adapter when it is selected
 with address and IO/-M. This
 line will be used for bus
 expansion. It is pulled up with a
 resistor and should be pulled
 down with an open collector
 device.

AUDIO IN I Channel devices may provide
 sound sources to the
 system-board sound-subsystem
 through this line. It is 1 volt
 peak-to-peak, dc biased at 2.5
 volts above ground.

Input/Output

Hex Range	9	8	7	6	5	4	3	2	1	0	Device
20-27	0	0	0	0	1	0	0	X	X	A0	PIC 8259
40-47	0	0	0	1	0	0	0	0	A1	A0	Timer 8253-5
60-67	0	0	0	1	1	0	0	X	A1	A0	PPI 8255-5
A0-A7	0	0	1	0	1	0	0	X	X	X	NMI Mask Reg.
C0-C7	0	0	1	1	0	0	0	X	X	X	Sound SN76496N
F0-FF	0	0	1	1	1	1	X	A2	A1	A0	Diskette
200-207	1	0	0	0	0	0	0	X	X	X	Joystick
2F8-2FF	1	0	1	1	1	1	1	A2	A1	A0	Serial Port
3D0-3DF	1	1	1	1	0	1	A3	A2	A1	A0	Video Subsystem
3F8-3FF	1	1	1	1	1	1	1	A2	A1	A0	Modem

I/O Map

X = Don't care (that is, not in decode.)

- Any I/O which is not decoded on the system board may be decoded on the I/O Channel.
- At Power-On time the NMI into the 8088 is masked 'off'. This mask bit can be set by system software as follows:

**Write to Port A0 D7=ENA NMI D6=IR TEST ENA
D5=SELCLK1 INPUT D4=+Disable HRQ**

8255 Bit Assignments

PA Output
PA0 Reserved for Keystroke Storage
PA1 Reserved for Keystroke Storage
PA2 Reserved for Keystroke Storage
PA3 Reserved for Keystroke Storage
PA4 Reserved for Keystroke Storage
PA5 Reserved for Keystroke Storage
PA6 Reserved for Keystroke Storage
PA7 Reserved for Keystroke Storage
PB Output
PB0 +Timer2 Gate (Speaker)
PB1 +Speaker Data
PB2 +Alpha (-Graphics)
PB3 +Cassette Motor Off
PB4 +Disable Internal Beeper and Cassette Motor
Relay
PB5 SPKR Switch 0
PB6 SPKR Switch 1
PB7 Reserved
PC Input
PC0 Keyboard Latched
PC1 -Internal MODEM Card Installed
PC2 -Diskette Drive Card Installed
PC3 -64KB Memory and Display Expansion Installed
PC4 Cassette Data In
PC5 Timer Channel 2 Output
PC6 +Keyboard Data
PC7 -Keyboard Cable Connected

8255 Bit Assignment Description

PA0 thru PA7 (Output Lines)

Port A is configured as an output. The output lines are not used by the hardware, but are used to store keystrokes. This is done to maintain compatibility with the Personal Computer, and Personal Computer XT.

PB0 (+Timer 2 Gate)

This line is routed to the gate input of timer 2 on the 8253-5. When this bit is 'low', the counter operation is halted. This bit and PB1 (+Speaker Data) controls the operation of the 8253-5 sound source.

PB1 (+Speaker Data)

This bit ANDS 'off' the output of the 8253-5 timer 2. It can be used to disable the 8253-5 sound source, or modify its output. When this bit is a 1, it enables the output, a 0 forces the output to zero.

PB2 (+Alpha -Graphics)

This bit is used to steer data from the memory into the Video Gate Array. This bit should be a 1 for all alpha modes, and a 0 for all graphics modes.

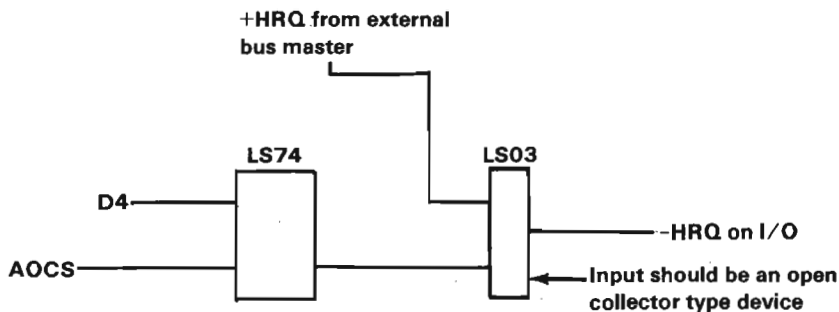
- PB3** (+Cassette Motor Off) When this bit is a 1, the cassette relay is 'open' and the cassette motor is 'off'. When this bit is a 0, and PB4 = 0, the cassette motor is 'on'.
- PB4** (+Disable internal beeper and cassette motor relay) When this bit is a 1, the internal beeper is 'disabled' and the 8253-5 timer 2 sound source can only be heard if it is steered to the audio output. This bit also disables the cassette motor when it is a 1. To 'enable' the cassette motor, this bit must be a 0. In this case, PB1 should be used to gate 'off' the internal beeper and 8253-5 sound source.
- PB5, PB6** (Speaker switch 0,1) These bits steer one of 4 sound sources. This is available to the RF modulator or the external audio jack. The sound sources selected are shown below.
- | PB6 | PB5 | Sound Source |
|-----|-----|----------------------|
| 0 | 0 | 8253-5 Timer 2 |
| 0 | 1 | Cassette Audio Input |
| 1 | 0 | I/O Channel Audio In |
| 1 | 1 | 76496 |
- PB7** (Open) Reserved for future use.

- PC0** (Keyboard latched) This input comes from a latch which is set to a 1 on the first rising edge of the Keyboard Data stream. The output of this latch also causes the NMI to occur. This latch is cleared by doing a dummy 'Read' operation to port A0. This input is provided so that a program can tell if a keystroke occurred during a time when the NMI was masked 'off' and a keystroke has been missed. The program will then be able to give an error indication of the missed keystroke.
- PC1** (-Modem card installed) When this bit is a 0, it indicates that the Internal Modem card is installed.
- PC2** (-Diskette card installed) When this bit is a zero, it indicates that the Diskette Drive Adapter is installed.
- PC3** (-64KB Memory and Display Expansion installed) When this bit is a 0, it indicates that the 64KB Memory and Display Expansion is installed.

PC4	(Cassette data in)	If the cassette-motor relay is 'closed', and the cassette motor is 'on', this pin will contain data which has been wave shaped from the cassette. If the cassette-motor relay is 'off', this pin will contain the same data as the 8253-5 timer 2 output.
PC5	(Timer channel 2 output)	This input is wired to the timer channel 2 output of the 8253-5.
PC6	(+Keyboard data)	This input contains keyboard data. The keyboard data comes from the cable if attached, or from the IR Receiver if the cable is not attached.
PC7	(-Keyboard cable connected)	If this bit is 'low', it indicates that the keyboard cable is connected.

Port A0 Output Description

- | | | |
|-----------|--------------------------|---|
| D7 | (Enable NMI) | When this bit is a 1, the NMI is 'enabled'. When it is a 0, it is 'disabled'. |
| D6 | (IR test ENA) | This bit enables the 8253-5 timer 2 output into an IR diode on the IR Receiver board. This information is then wrapped back to the keyboard input. If the cable is not connected, timer 2 should be set for 40 kHz which is the IR-modulation frequency. This feature is used only for a diagnostic test of the IR Receiver board. |
| D5 | (Selc Clk1 input) | This bit selects one of two input Clks to the 8253-5 timer 1. A 0 selects a 1.1925 MHz Clk input used to assist the program in de-serializing the keyboard data. A 1 selects the timer 0 output to be used as the Clk input to timer 1. This is used to catch timer 0 overflows during diskette drive operations when interrupts are masked 'off'. This is then used to update the time-of-day. |
| D4 | (+Disable HRQ) | This bit is not actually implemented on the system board, but is supported by the programming. This bit is used to disable -HRQs from external bus-masters (DMA, Alternate Processors, etc.) The logic for this bit must exist on each bus-master attachment. A 0 should 'enable' -HRQ, and a 1 should 'disable' -HRQ. |



Port A0 Output Description

Port A0 Input Operation

A 'read' to I/O port A0 will clear the keyboard NMI latch. This latch causes an NMI on the first rising edge of the keyboard data if the enable NMI bit (port A0 bit D7) is 'on'. This latch can also be read on the 8255 PC0. The program can determine if a keystroke occurred while the NMI was 'disabled' by reading the status of this latch. This latch must be cleared before another NMI can be received.

The System board provides for selection of keyboard data from either a cable or the IR-receiver board. The IR-receiver board is mounted on the system board and can receive data through an IR link. The source of the keyboard's data is determined by the -Cable Connected signal at the keyboard cable connector. Keyboard serial data is available to the 8088 at bit PC6 of the 8255 PPI.

The system board is responsible for the de-serialization of keyboard data. The start bit in the serial stream causes an NMI to be generated. The 8088 then reads the 8253 timer to determine when to interrogate the

serial stream. After de-serialization the NMI service-routine does a 'Read' from hex A0 to clear the NMI latch.

During certain time-critical operations, such as diskette I/O, the processor will mask 'off' the NMI interrupt. Keyboard inputs during this time cannot be serviced. A keyboard latch is provided so that at the end of such operations the processor will determine whether any keys were pressed and take appropriate actions. The keyboard latch is 'set' by any key being pressed and is 'reset' by 'Reading' the NMI port. (No data is presented to the microprocessor during this 'Read'.) Keyboard latch data is available to the processor at bit PC0 of the 8255 PPI.

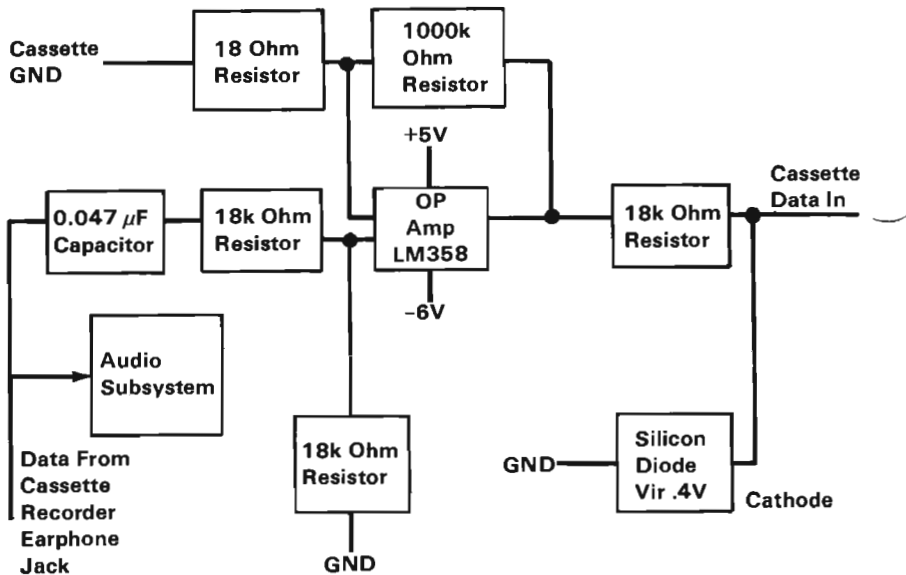
Notes:

Cassette Interface

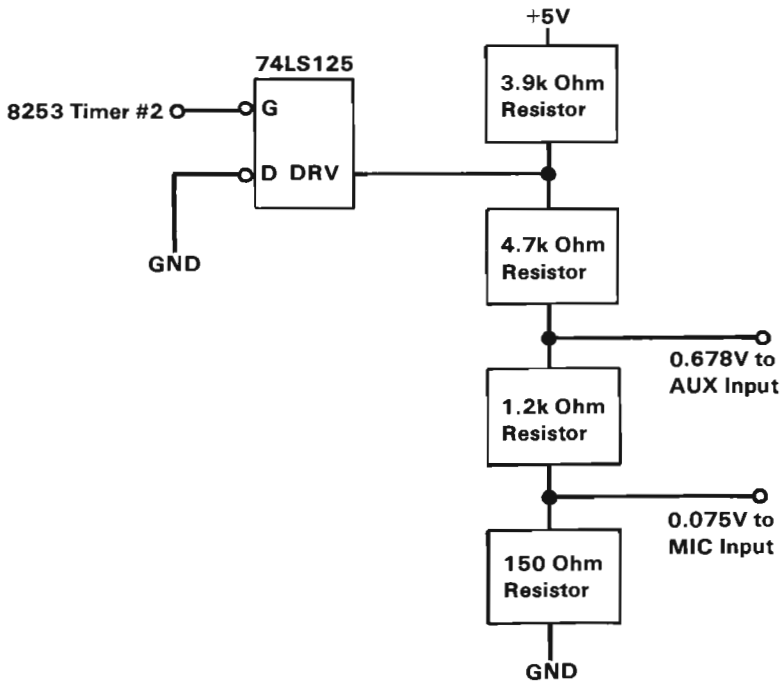
The cassette interface is controlled through software. An output from the 8253 timer controls the data to the cassette recorder through the cassette connector at the rear of the system board. The cassette-input data is read by an input-port bit of the 8255A-5 programmable-peripheral-interface (PPI) (8255A-5 PC4). Software algorithms are used to generate and read cassette-data. The cassette drive-motor is controlled by Bit PB3 of the 8255. Bit PB4, which 'enables' the 7547 relay driver, must be 'low' when the motor is to be turned on. The cassette interface has a wrap feature which connects the output to the input when the motor control is 'off'. See "BIOS Cassette Logic" in Section 5 for information on data storage and retrieval.

A mechanism is provided that will direct the cassette input to the audio subsystem. Please see "Sound Subsection" in Section 2.

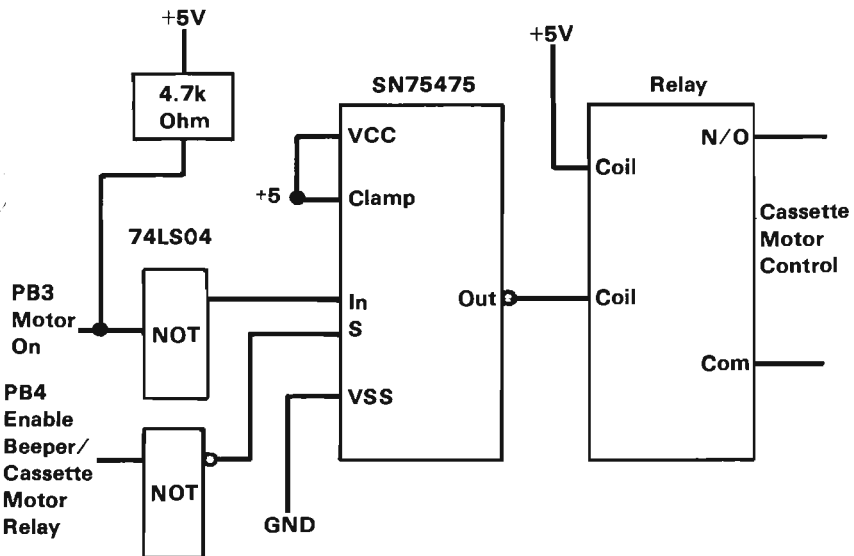
Circuit block diagrams for the cassette-interface read, write, and motor control are illustrated in the following figures.



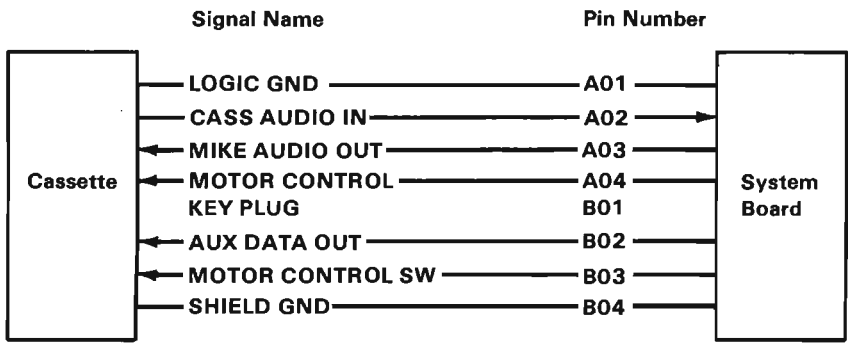
Cassette-Interface Read-Hardware Block Diagram



Cassette-Interface Write-Hardware Block Diagram



Cassette-Motor Control Block Diagram



Cassette Connector Specifications

Notes:

Video Color/Graphics Subsystem

The video subsystem is designed so that the IBM Color Display, composite monitors, and a home television set can be attached. It is capable of operating in black-and-white or color. It provides three video ports: a composite-video, a direct-drive, and a connector for an RF modulator to be used with home televisions. In addition, it contains a light pen interface.

Note: The IBM Personal Computer Monochrome Display cannot be used with the PCjr system.

Note: An IBM Connector for Television option must be obtained to attach a home TV.

The subsystem has two basic modes of operation: alphanumeric (A/N) and all points addressable graphics (APA). Additional modes are available within the A/N and APA modes.

In the A/N mode, the display can be operated in either a 40-column by 25-row mode for a low-resolution display home television, or an 80-column by 25-row mode for high-resolution monitors. In both modes, characters are defined in an 8-wide by 8-high character box and are 7-wide by 7-high, with one line of descender. Both A/N modes can operate in either color or black-and-white.

In the A/N black-and-white mode, the character attributes of reverse video, blinking, highlighting and gray shades are available.

In the A/N color mode, sixteen foreground-colors and sixteen background-colors are available for each character. In addition, blinking on a per-character basis

is available. When blinking is used, only eight background-colors are available. One of 16 colors, or gray shades can be selected for the screen's border in all A/N modes.

In both A/N modes, characters are formed from a ROM character-generator. The character generator contains dot patterns for 256 different characters. The character set contains the following major groupings of characters:

- 16 special characters for game support
- 15 characters for word-processing editing support
- 96 characters for the standard-ASCII-graphics set
- 48 characters for foreign-language support
- 48 characters for business block-graphics (allowing drawing of charts, boxes, and tables using single or double lines)
- 16 selected Greek symbols
- 15 selected scientific-notation characters

In the APA mode, there are three resolutions available: a low-resolution mode (160 PELs [Picture Elements] by 200 rows), a medium-resolution mode (320 PELs by 200 rows), and a high-resolution mode (640 PELs by 200 rows).

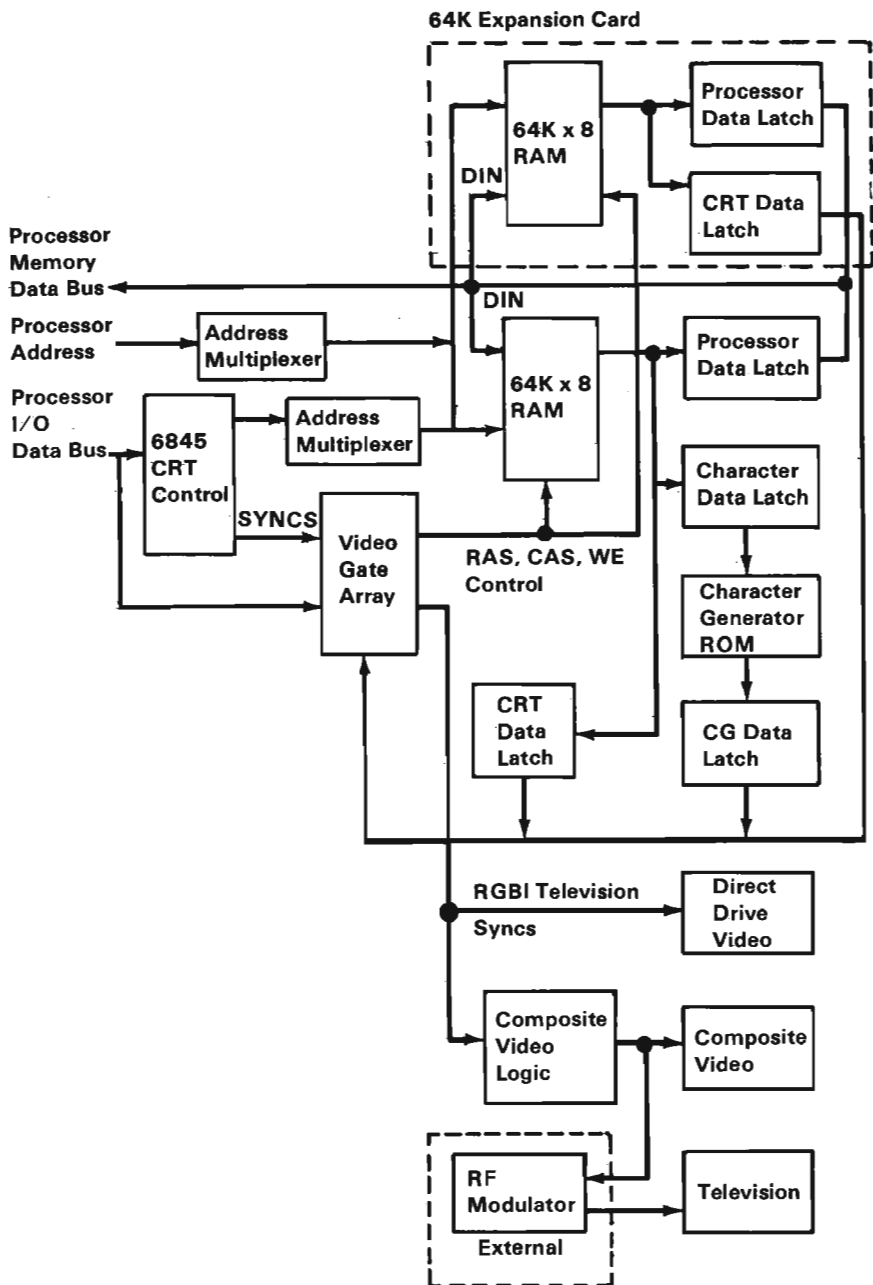
Different color modes exist within each of the APA resolutions. Two, four, or sixteen colors are available in APA color, and two, four, or sixteen gray shades are available in APA black-and-white.

One of sixteen colors, or grey shades can be selected for the screen's border in all APA modes.

The direct drive, composite video and RF Modulator connector are right-angle-mounted connectors extending through the rear of the system unit.

The video color/graphics subsystem is implemented using a Motorola 6845 CRT controller device and a Video Gate Array (VGA) (LSI5220). The video subsystem is highly programmable with respect to raster and character parameters. Thus many additional modes are possible with the proper programming.

The following figure shows a block diagram of the video color/graphics subsystem.



Video Color/Graphic Subsystem Block Diagram

Major Components Definitions

Motorola 6845 CRT Controller

This device provides the necessary interface to drive a raster-scan CRT. Additional information about this component is provided in publications listed in "Bibliography".

Storage Organization

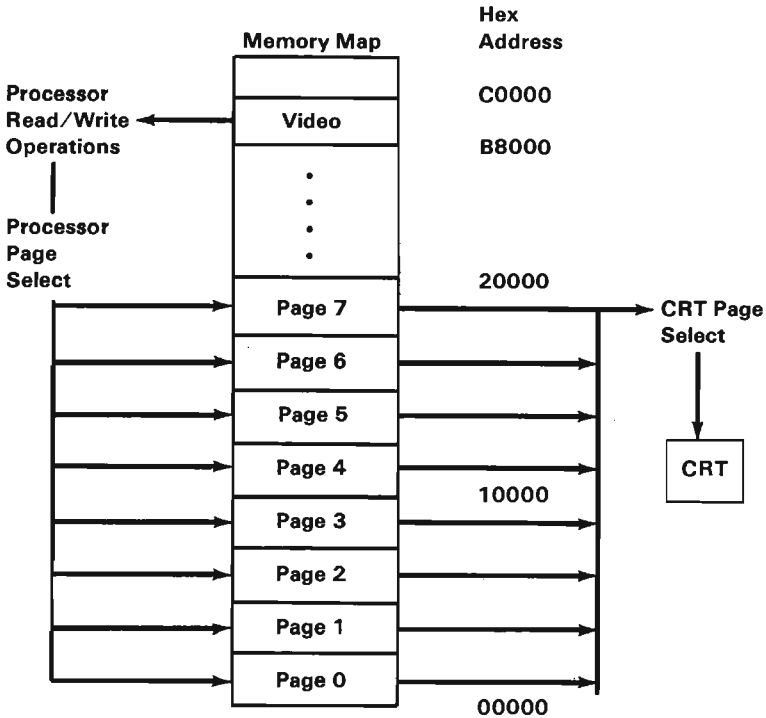
The base video-color/graphics-subsystem accesses 64K bytes of read/write memory (RAM). A 64KB Memory and Display Expansion can be added to increase the amount of system RAM to 128K bytes. This memory-storage area serves two functions; as the video-display buffer and as the system processor is (8088) main-RAM.

The RAM is located at address hex 0000 and is either 64K bytes or 128K bytes with the memory expansion option. The 8088 can access the memory by reading from and writing to address locations hex 00000 to 1FFFF or by reading from or writing to the 16K-byte region starting at address hex B8000. The page affected by a read or write operation is determined by the processor's page register. The processor can access the RAM at any time in all modes with no adverse effect to the video information. The page that the video information is taken from is determined by the CRT page register.

The processor and CRT page registers are write only registers and can be changed at any time. These registers allow the processor to work in one page while the display is displaying another page. The processor can switch pages at the vertical-retrace time. This will aid animation on the video color/graphics subsystem.

Also, since all 128K bytes of read/write memory are available for display purposes, the application can use as little or as much memory as needed for the display.

The following figure is a map of the video color/graphics subsystem.



Video Color/Graphics Subsystem Memory Map

Bandwidth

The video bandwidth is either 3.5, 7 or 14 MHz depending on the mode of operation. The processor bandwidth is the same for all modes. The processor is allowed one cycle every 1.1 microseconds. An average of two wait states will be inserted in a processor RAM read cycle, because the average latency time for the processor to get a cycle is 560 ns and the cycle time is 350 ns. There is no performance penalty for redirecting processor reads and writes through the B8000 - BFFFF address area.

Character Generator

The ROM character-generator consists of 2K bytes of storage which cannot be read from, or written to under software control. It is implemented with a MCM68A316E or equivalent. Its specifications are 350 ns access, 350 ns cycle static operation. The device is pin compatible with 2716 and 2732 EPROMS.

Video Gate Array

A CMOS gate array is used to generate storage-timing (RAS, CAS, WE), direct-drive, composite-color and status signals. See "Video Gate Array" later in this section.

Palette

The video color/graphics subsystem contains a 16-word by 4-bit palette in the Video Gate Array which takes PEL (Picture Element) information from the read/write memory and uses it to select the color to display. This palette is used in all A/N and APA modes. Any input to the palette can be individually masked 'off' if a mode does not support the full complement of 16 colors. This masking allows the user to select a unique palette of colors whenever any mode does not support all 16 colors.

In two-color modes, the palette is defined by using one bit (PA0), with the following logic:

Palette Address Bit	
PA0	Function
0	Palette Register 0
1	Palette Register 1

Palette Logic (1 of 3)

In four-color modes, the palette is defined by using two bits (PA1 and PA0), with the following logic:

Palette Address Bits		Function
PA1	PA0	
0	0	Palette Register 0
0	1	Palette Register 1
1	0	Palette Register 2
1	1	Palette Register 3

Palette Logic (2 of 3)

In sixteen-color modes, the palette is defined by using four bits (PA3, PA2, PA1, and PA0), with the following logic:

Palette Address Bits				Function
PA3	PA2	PA1	PA0	
0	0	0	0	Palette Register 0
0	0	0	1	Palette Register 1
0	0	1	0	Palette Register 2
0	0	1	1	Palette Register 3
0	1	0	0	Palette Register 4
0	1	0	1	Palette Register 5
0	1	1	0	Palette Register 6
0	1	1	1	Palette Register 7
1	0	0	0	Palette Register 8
1	0	0	1	Palette Register 9
1	0	1	0	Palette Register 10
1	0	1	1	Palette Register 11
1	1	0	0	Palette Register 12
1	1	0	1	Palette Register 13
1	1	1	0	Palette Register 14
1	1	1	1	Palette Register 15

Palette Logic (3 of 3)

The sixteen colors available to all A/N and APA modes are selected through combinations of the I (Intensity), R (Red), G (Green), and B (Blue) bits. These colors are listed in the following figure:

I	R	G	B	Color
0	0	0	0	Black
0	0	0	1	Blue
0	0	1	0	Green
0	0	1	1	Cyan
0	1	0	0	Red
0	1	0	1	Magenta
0	1	1	0	Brown
0	1	1	1	Light Gray
1	0	0	0	Dark Gray
1	0	0	1	Light Blue
1	0	1	0	Light Green
1	0	1	1	Light Cyan
1	1	0	0	Pink
1	1	0	1	Light Magenta
1	1	1	0	Yellow
1	1	1	1	White

Note: The "I" bit provides extra luminance (brightness) to each available shade. This results in the light colors listed above, except for monitors that do not recognize the "I" bit.

Summary of Available Colors

Alphanumeric Modes

Every display-character position in the alphanumeric mode is defined by two bytes in the system read/write memory, using the following format:

Display Character Code Byte	Attribute Byte
7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0

Display Format

The functions of the attribute byte are defined by the following figure:

Attribute Function	Attribute Byte Definition							
	7	6	5	4	3	2	1	0
Normal Reverse Video Nondisplay (Off) Nondisplay (On)	Fore- Ground Blink	PA2	PA1	PA0	PA3	PA2	PA1	PA0
		Background			Foreground			
	B	0	0	0	I	1	1	1
	B	1	1	1	I	0	0	0
B	0	0	0	I	0	0	0	
B	1	1	1	I	1	1	1	
I = Highlighted Foreground (Character) B = Blinking Foreground (Character)								

Attribute Functions

Graphics Mode

The Video Color/Graphics Subsystem can be programmed for a wide variety of modes within the graphics mode. Five graphics-modes are supported by the system's ROM BIOS. They are low-resolution 16-color graphics, medium-resolution 4-color graphics, medium-resolution 16-color graphics, high-resolution 2-color graphics, and high-resolution 4-color graphics. The table in the following figure summarizes the five modes:

Graphics Mode	Horiz. (PELs)	Vert. (Rows)	Number of Colors Available (Includes Background Color)
Low-Resolution 16-Color	160	200	16 (Includes b-and-w)
Medium-Resolution 4-Color	320	200	4 Colors of 16 Available
Medium-Resolution 16-Color	320	200	16 (Includes b-and-w)
High-Resolution 2-Color	640	200	2 Colors of 16 Available
High-Resolution 4-Color	640	200	4 Colors of 16 Available

Note: The screen's border color in all modes can be set to any 1 of the 16 possible colors. This border color is independent of the screen's work area colors. In Black and White each color maps to a distinct gray shade.

Graphics Modes

Low-Resolution 16-Color Graphics

The low-resolution mode supports home-television sets, low-resolution displays, and high-resolution displays. It has the following characteristics:

- Contains a maximum of 200 rows of 160 PELs
- Specifies 1 of 16 colors for each PEL by the I, R, G, and B bits
- Requires 16K bytes of read/write memory
- Formats 2 PELs per byte for each byte in the following manner:

7	6	5	4	3	2	1	0
PA3	PA2	PA1	PA0	PA3	PA2	PA1	PA0
First Display PEL				Second Display PEL			

Low-Resolution 16-Color Graphics

Medium-Resolution 4-Color Graphics

The medium-resolution mode supports home-television sets, low-resolution displays, and high-resolution displays. It has the following characteristics:

- Contains a maximum of 200 rows of 320 PELs
- Selects one of four colors for each PEL
- Requires 16K bytes of read/write memory
- Supports 4 of 16 possible colors
- Formats 4 PELs per byte for each byte in the following manner:

7	6	5	4	3	2	1	0
PA1	PA0	PA1	PA0	PA1	PA0	PA1	PA0
First Display PEL		Second Display PEL		Third Display PEL		Fourth Display PEL	

Medium-Resolution 4-Color Graphics

Medium-Resolution 16-Color Graphics

The medium-resolution 16-color graphics mode supports home television sets, low-resolution displays, and high-resolution displays. It has the following characteristics:

- Requires system configuration of 128K bytes of read/write memory
- Requires 32K bytes of read/write memory
- Contains a maximum of 200 rows of 320 PELs.
- Specifies 1 of 16 colors for each PEL
- Formats 2 PELs per byte for each byte in the following manner.

7	6	5	4	3	2	1	0
PA3	PA2	PA1	PA0	PA3	PA2	PA1	PA0
First Display PEL				Second Display PEL			

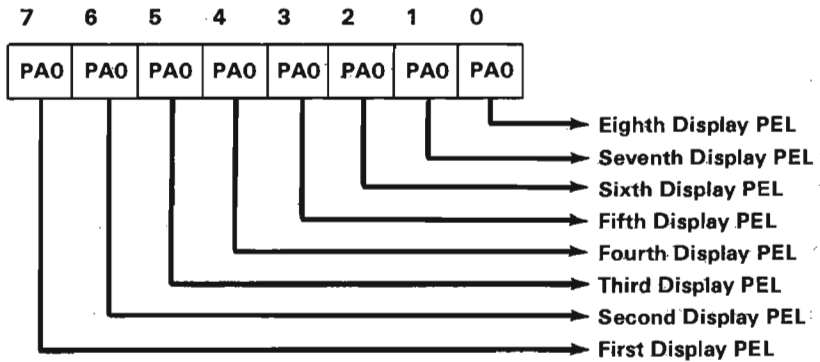
Medium-Resolution 16-Color Graphics

High-Resolution 2-Color Graphics

The high-resolution 2-color mode supports high-resolution monitors only. This mode has the following characteristics:

- Contains a maximum of 200 rows of 640 PELs
- Supports 2 of 16 possible colors.

- Requires 16K bytes of read/write memory.
- Formats 8 PELs per byte for each byte in the following manner:

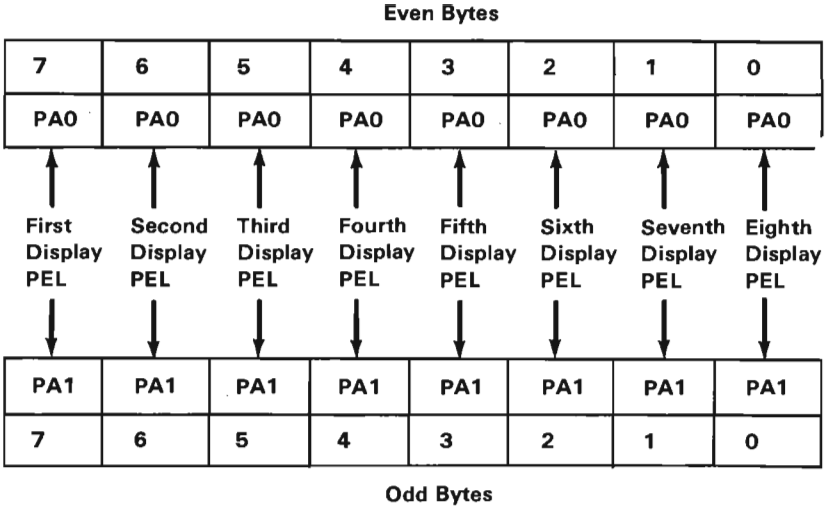


High-Resolution 2-Color Graphics

High-Resolution 4-Color Graphics

The high-resolution mode is used only with high-resolution monitors. This mode has the following characteristics:

- Requires system configuration of 128K Bytes read/write memory
- Requires 32K bytes of read/write memory
- Contains a maximum of 200 rows of 640 PELs
- Selects one of four colors for each PEL
- Supports 4 out of 16 colors
- Formats 8 PELs per two bytes (consisting of one even-byte and one odd-byte) in the following manner:

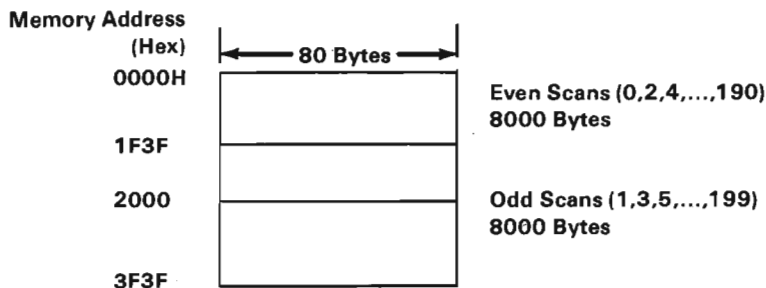


High-Resolution 4-Color Graphics

Graphics Storage Organization

For the low-resolution 16-color graphics, the medium-resolution 4-color graphics, and the high-resolution 2-color graphics, storage is organized into two banks of 8000 bytes each.

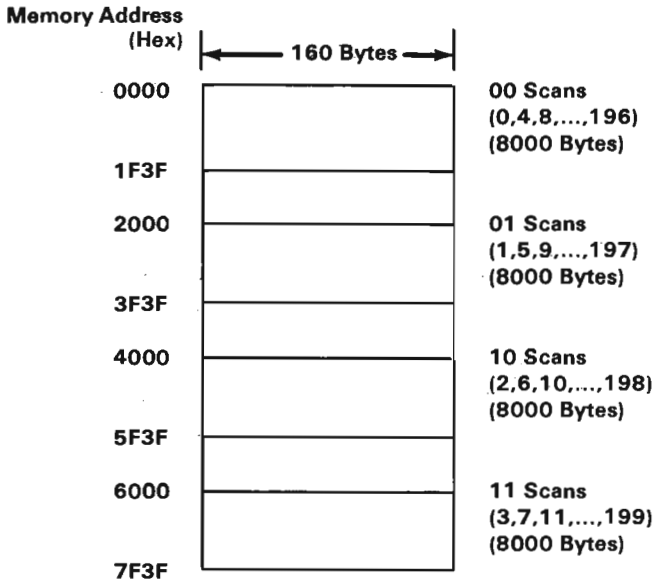
The following figure shows the organization of the graphics storage.



Graphics Storage Organization (Part 1 of 2)

Address 0000 contains PEL information for the upper-left corner of the display area.

For the medium-resolution 16-color graphics, and the high-resolution 4-color graphics modes, the graphics storage is organized into four banks of 8000 bytes each.



Graphics Storage Organization (Part 2 of 2)

Address 0000 contains PEL information for the upper-left corner of the display.

Video Gate Array

The Video Gate Array is located at I/O address hex 3DA, and is programmed by first writing a register address to port hex 3DA and then writing the data to port hex 3DA.

Any I/O 'write'-operations to hex address 3DA continuously toggle an internal address/data flip-flop. This internal flip-flop can be set to the address state by issuing an I/O 'read' instruction to port hex 3DA. An I/O 'read' instruction also 'reads' the status of the Video Gate Array. A description of each of the registers in the Video Gate Array follows.

Hex Address	Register
00	Mode Control 1
01	Palette Mask
02	Border Color
03	Mode Control 2
04	Reset
10-1F	Palette Registers

Video Gate Array Register Addresses

Mode Control 1 Register

This is a 5-bit 'write'-only register, it cannot be 'read'. Its address is 0 within the Video Gate Array. A description of this register's bit functions follows.

Bit 0	+HIBW/-LOBW
Bit 1	+Graphics/-Alpha
Bit 2	+B/W
Bit 3	+Video Enable
Bit 4	+16 Color Graphics

Mode Control 1 Register

- Bit 0** This bit is 'high' (1) for all high-bandwidth modes. These modes are all modes which require the 64KB Memory and Display Expansion for a system total of 128K bytes of read/write memory. The high bandwidth modes are the 80 by 25 alphanumeric mode, the 640 by 200 4-color graphics mode, and the 320 by 200 16-color graphics mode. This bit is 'low' (0) for all low-bandwidth modes.
- Bit 1** This bit is 'high' (1) for all graphics modes and is 'low' (0) for all alphanumeric modes.
- Bit 2** When this bit is 'high' (1), the composite-video color-burst and chrominance are disabled, leaving only the composite intensity-levels for gray shades. When this bit is 'low' (0), the composite-video color is 'enabled'. This

bit should be set 'high' for high-resolution black-and-white display applications.

Note: This bit has no effect on direct-drive colors.

Bit 3 When this bit is 'high' (1), the video signal is 'enabled'. The video signal should be 'disabled' when changing modes. When the video signal is 'disabled', the screen is forced to the border color.

Bit 4 This bit must be 'high' (1) for all 16-color graphics-modes. These modes are the 160 by 200 16-color graphics-mode and the 320 by 200 16-color graphics-mode.

Palette Mask Register

This is a 4-bit write-only register, it cannot be 'read'. Its address in the Video Gate Array is hex 01. A description of this register's bit functions follows.

Bit 0	-Palette Mask 0
Bit 1	-Palette Mask 1
Bit 2	-Palette Mask 2
Bit 3	-Palette Mask 3

Palette Mask Register

When bits 0-3 are 0, they force the appropriate palette address to be 0 regardless of the incoming color

information. This can be used to make some information in memory a 'don't care' condition until it is requested.

In the 2-color and 4-color modes, the palette addresses should be 'masked' because only 1 or 2 color-lines contain valid information. For 4-color modes, the palette mask register should contain a hex 03 and, for 2-color modes, it should contain a hex 01.

Border Color Register

This is a 4-bit 'write'-only register, it cannot be 'read'. Its address in the Video Gate Array is hex 02. The following is a description of the register's bit functions:

Bit Number	Function
0	+ B (Blue) Border Color Select
1	+ G (Green) Border Color Select
2	+ R (Red) Border Color Select
3	+ I (Intensity) Border Color Select

Border Color Register

A combination of bits 0-3 selects the screen-border color as one of 16 colors, as listed in the "Summary of Available Colors" table in this section.

Mode Control 2 Register

This is a 4-bit, 'write'-only register, it cannot be 'read'. Its address inside the Video Gate Array is hex

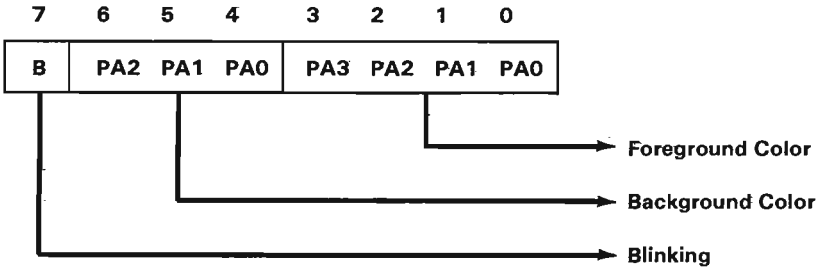
03. The following is a description of the register's bit functions:

Bit Number	Function
0	- Reserved = 0
1	+ Enable Blink
2	- Reserved = 0
3	+ 2-Color Graphics

Mode Control 2 Register

Bit 0 This bit is reserved, but should always be programmed as a 0.

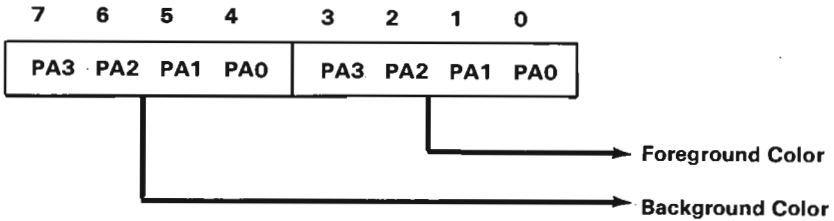
Bit 1 When this bit is 'high' (1) in the alphanumeric mode, the attribute byte has the following definition:



Where PA0 to PA3 are palette addresses.

Attribute Byte Definition (Part 1 of 2)

If the enable-blink bit is 'off' in the alphanumeric mode, the attribute byte takes on the following definition:



Attribute Byte Definition (Part 2 of 2)

If the enable-blink bit is on in a graphics mode, the high-order address of the palette (PA3) is replaced with the character-blink rate. This causes displayed colors to switch between two sets of colors.

If the colors in the lower half of the palette are the same as in the upper half of the palette, no color changes will occur. If the colors in the upper half of the palette are different from the lower half of the palette, the colors will alternately change between the 2 palette colors at the blink rate.

Only eight colors are available in the 16-color modes when using this feature. Bit 3 of the palette mask has no effect on this mode.

Bit 2 This bit is reserved, but should always be programmed as a 0.

Bit 3 This bit should be 'high' (1) when in the 640 by 200 2-color graphics-mode. It should be 'low' (0) for all other modes.

Reset Register

This is a 2-bit 'write'-only register, it cannot be 'read'. Its address inside the Video Gate Array is hex 04. The following is a description of the register's bit functions:

Bit 0	+Asynchronous Reset
Bit 1	+Synchronous Reset

Reset Register

Bit 0 When 'high' (1), this bit will issue an 'asynchronous reset' to the Video Gate Array. This will cause all memory cycles to stop and all output signals to be tri-stated. The 'asynchronous reset' should only be issued once at the system power-on time. This bit should be 'high' (1), the Video Gate Array and the 6845 programmed, and then it should be 'low' (0).

The system read/write memory (RAM) will not work until this power-on sequence is finished. After this power-on sequence, subsequent 'resets' should be 'synchronous resets'.

Note: Issuing an 'asynchronous reset' can cause the contents of RAM to be destroyed.

Bit 1 When 'high' (1), this bit will issue a 'synchronous reset' to the Video Gate Array. This will cause all memory cycles to stop and all output signals to stop. Bit 1 should be 'low' (0) before changing modes.

Before issuing a 'synchronous reset', the program should read 256 locations in RAM as every other location in 512 locations. The program should then issue the 'synchronous reset' and change the mode. This changes the Video Gate Array mode-control registers and the 6845 registers.

Next, the 'synchronous reset' should be removed and the 256 RAM locations should be 'read' again as above. This procedure will ensure system RAM data-integrity during mode changes. 'Synchronous resets' need only be issued when changing between high-bandwidth, and low- bandwidth modes. (Bit 0 in mode control 1 register)

Note: No accesses to RAM can be made while the video gate array is in a 'reset' state. 'Resets' must be done from code in ROM or EPROM's.

Palette Registers

There are sixteen 4-bit-wide palette-registers. These registers are 'write'-only, they cannot be 'read'. Their addresses in the Video Gate Array are from hex 10 to 1F.

Palette address hex 10 is accessed whenever the color code from memory is a hex 0, address hex 11 is accessed whenever the color code from memory is a hex 1, and so forth. A description of the color codes is in "Summary of Available Colors" in this section.

Note: The palette address can be 'masked' by using the palette mask register.

The following is a description of the register's bit functions:

Bit Number	Function
0	+ Blue
1	+ Green
2	+ Red
3	+ Intensity

Palette Register Format

When loading the palette, the video is 'disabled' and the color viewed on the screen is the data contained in the register being addressed by the processor.

When the program has completed loading the palette, it must change the hex address to some address less than hex 10 for video to be 'enabled' again.

If a programmer does not wish a user to see the adverse effects of loading the palette, the palette should be loaded during the vertical-retrace time. The program must modify the palette and change the video gate array address to less than hex 10 within the vertical-retrace time. A vertical-retrace interrupt and a status bit are provided to facilitate this procedure.

Status Register

This is a 5-bit 'read'-only register, it cannot be 'written'. The internal address of the video gate array is a 'don't care' condition for the status-register read-operation. A description of the register's bit functions follows:

Bit 0	+Display Enable
Bit 1	+Light Pen Trigger Set
Bit 2	-Light Pen Switch Made
Bit 3	+Vertical Retrace
Bit 4	+Video Dots

Status Register

- Bit 0** When 'high' (1), this bit indicates video is being displayed.
- Bit 1** When 'high' (1), this bit indicates that a positive- going edge from the light pen input has set the light pen trigger. This trigger is 'low' (0) upon a system power-on, and may also be cleared by performing an I/O 'Out' command to address hex 3DB. No specific data is required, this action is address-activated.
- Bit 2** This bit indicates the status of the light pen switch. The switch is not latched or debounced. When this bit is 'low' (0), the light pen switch is 'on'.
- Bit 3** When 'high' (1), this bit indicates the vertical retrace is 'active'.

Bit 4 When 'high' (1), this bit indicates that video-dot information is available. The two low-order bits of the address register determine the video-dot information presented through the following logic:

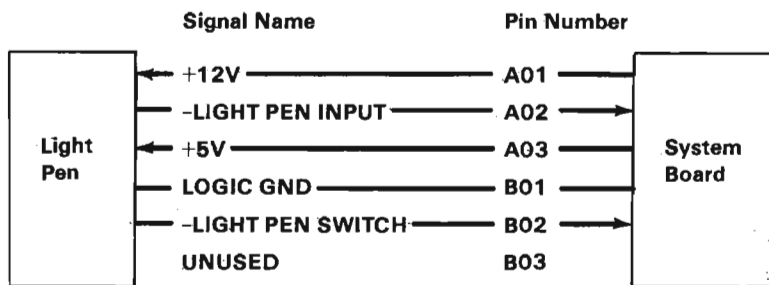
Address Register Bit 1	Address Register Bit 0	Video Dot Information Selected
0	0	Blue
0	1	Green
1	0	Red
1	1	Intensity

Address Register

This bit is provided for testing purposes. It verifies that video is occurring properly, and that the palette registers and all other 'write'-only registers are operating correctly.

Light Pen

A light pen can be used on the PCjr by connecting it to the six-pin connector for light pens on the back of the system board.



Connector Specifications

Note: The light pen interface is set for RGBI (Red, Green, Blue, Intensity). Due to timing differences between different displays (Different phosphors take longer to turn on, and different circuits take longer to accomplish their task.) the row, column value returned from the CRT can vary. This difference must be compensated for through software.

Programming Considerations

Programming the 6845 CRT Controller

The 6845 has 19 accessible, internal registers, which are used to define and control a raster-scanned CRT display. One of these registers, the Index Register, is actually used as a pointer to the other 18 registers. It is a 'write'-only register, which is loaded from the processor by executing an 'Out' instruction to I/O address hex 3D4. The five least-significant-bits of the I/O bus are loaded into the Index Register.

In order to load any of the other 18 registers, the Index Register is first loaded with the necessary pointer; then the Data Register is loaded with the information to be

placed in the selected register. The Data Register is loaded from the processor by executing an 'Out' instruction to I/O address hex 3D5.

The following table defines the values that must be loaded into the 6845-CRT-Controller registers to control the different modes of operation supported by the attachment:

Hex Addr.	Register		Units	I/O	Alphanumeric		Low/High Band Width Graphics
	#	Type			40x25	80x25	
0	R0	Horizontal Total	Char.	Write Only	38	71	38/71
1	R1	Horizontal Display	Char.	Write Only	28	50	28/50
2	R2	Horizontal Sync Position	Char.	Write Only	2C	5A	2B/56
3	R3	Horizontal Sync Width	Char.	Write Only	06	0C	06/0C
4	R4	Vertical Total	Char. Row	Write Only	1F	1F	7F/3F
5	R5	Vertical Total Adjustment	Scan Line	Write Only	06	06	06/06

Note: All register values are given in hexadecimal.

6845 Register Table (Part 1 of 3)

Hex Addr.	Register		Units	I/O	Alphanumeric		Low/High Band Width Graphics
	#	Type			40x25	80x25	
6	R6	Vertical Displayed	Char. Row	Write Only	19	19	64/32
7	R7	Vertical Sync Position	Char. Row	Write Only	1C	1C	70/38
8	R8	Interlace Mode	—	Write Only	02	02	02/02
9	R9	Maximum Scan Line Address	Scan Line	Write Only	07	07	01/03
A	R10	Cursor Start	Scan Line	Write Only	06	06	26/26
B	R11	Cursor End	Scan Line	Write Only	07	07	07/07

Note: All register values are given in hexadecimal.

6845 Register Table (Part 2 of 3)

Hex Addr.	Register		Units	I/O	Alphanumeric		Low/High Band Width Graphics
	#	Type			40x25	80x25	
C	R12	Start Addr. (H)	—	Write Only	00	00	00/00
D	R13	Start Addr. (L)	—	Write Only	00	00	00/00
E	R14	Cursor Addr. (H)	—	Read/ Write	00	00	00/00
F	R15	Cursor Addr. (L)	—	Read/ Write	00	00	00/00
10	R16	Light Pen (H)	—	Read Only	NA	NA	NA/NA
11	R17	Light Pen (L)	—	Read Only	NA	NA	NA/NA

Note: All register values are given in hexadecimal.

6845 Register Table (Part 3 of 3)

CRT/Processor Page Register

This register is an 8-bit 'write'-only register, that cannot be read. Its address is hex 3DF. The following is a description of the Register functions.

Bit Number	Description
0	CRT Page 0
1	CRT Page 1
2	CRT Page 2
3	Processor Page 1
4	Processor Page 2
5	Processor Page 3
6	Video Address Mode 0
7	Video Address Mode 1

CRT/Processor Page Register (Part 1 of 2)

CRT Page 0-2

These bits select which 16K byte memory-page between 00000 to hex 1FFFF is being displayed. If there is no expansion RAM in the system, the high-order bit is a 'don't care', and only 4 pages are supported. For graphics modes which require 32K bytes the low-order bit is a 'don't care'.

Processor Page 0-2

These bits select the 16K byte memory-page region where memory cycles to B8000 are redirected. If there is no expansion RAM installed in the system, the high-order bit is a 'don't care' and only 4 pages are supported.

Video Adr Mode 0-1

These bits control whether the row scan addresses are used as part of the memory address. These should be programmed as follows:

Video Address Mode		Resulting Modes
1 (Bit 7)	0 (Bit 6)	
0	0	All Alpha Modes
0	1	Low-Resolution-Graphics Modes
1	1	High-Resolution-Graphics Modes
1	0	Unused, Reserved

CRT/Processor Page Register (Part 2 of 2)

The following I/O devices are defined on the video color/graphics subsystem:

Hex Address	A9 A8 A7 A6 A5 A4 A3 A2 A1 A0	Function of Register
3DA	1 1 1 1 0 1 1 0 1 0	Gate Array Address and Status Register
3DB	1 1 1 1 0 1 1 0 1 1	Clear Light Pen Latch
3DC	1 1 1 1 0 1 1 1 0 0	Preset Light Pen Latch
3D0,3D4	1 1 1 1 0 1 0 x x 0	6845 Index Register
3D1,3D5	1 1 1 1 0 1 0 x x 1	6845 Data Register
3DF	1 1 1 1 0 1 1 1 1 1	CRT, Processor Page Register

x = "don't care" condition

Video I/O Devices

Mode Selection Summary

Four registers of the Video Gate Array allow the user to access all the alphanumeric and graphics modes supported by the system ROM BIOS. The following table summarizes the modes and their register settings:

Mode	Video Gate Array Reg.			
	00	01	02	03
40 by 25 Alphanumeric Black-and-White	0C	0F	00	02
40 by 25 Alphanumeric Color	08	0F	00	02
80 by 25 Alphanumeric Black-and-White	0D	0F	00	02
80 by 25 Alphanumeric Color	09	0F	00	02
160 by 200 16-Color Graphics	1A	0F	00	00
320 by 200 4-Color Graphics	0A	03	00	00
320 by 200 4-Shade Black-and-White	0E	03	00	00
320 by 200 16-Color Graphics	1B	0F	00	00
640 by 200 2-Color Graphics	0E	01	00	08
640 by 200 4-Color Graphics	0B	03	00	00

Note: All values are given in hexadecimal.

Mode Summary

Sequence of Events for Changing Modes

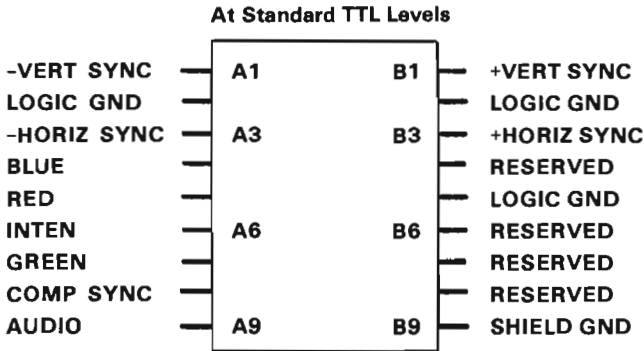
1. Determine the mode of operation.
2. Reset the 'video enable' bit in the Video Gate Array to disable video.
3. Program the 6845 CRT Controller to select the mode.
Read 256 bytes of memory
Reset gate array
4. Program the Video Gate Array registers.

- Remove gate-array reset
- Read 256 bytes of memory
- 5. Re-enable video.

Note: The gate array needs to be reset only when changing the high-bandwidth/low-bandwidth register.

Interrupt Information

The Video Gate Array uses interrupt level 5 of the Intel 8259 to provide the vertical retrace interrupt to the system.



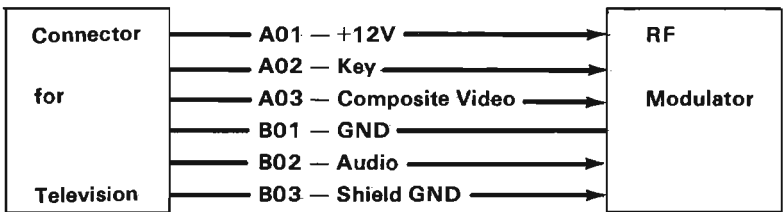
Connector Specifications

The direct-drive signals are standard TTL levels except the audio output which is a 1V peak-to-peak signal biased at 0V which can drive a 10K ohm or greater input-impedence.



Connector Specifications

The composite-video signal is 1V peak to peak biased at .7V with a 75 ohm load.



Television Connector Specifications

The Connector for Television connector has the composite-video signal at 1V peak to peak biased at .7V with a 75 ohm load. The connector also has the audio output which is 1V peak-to-peak signal biased at 0V which can drive a 10K ohm or greater input impedance.

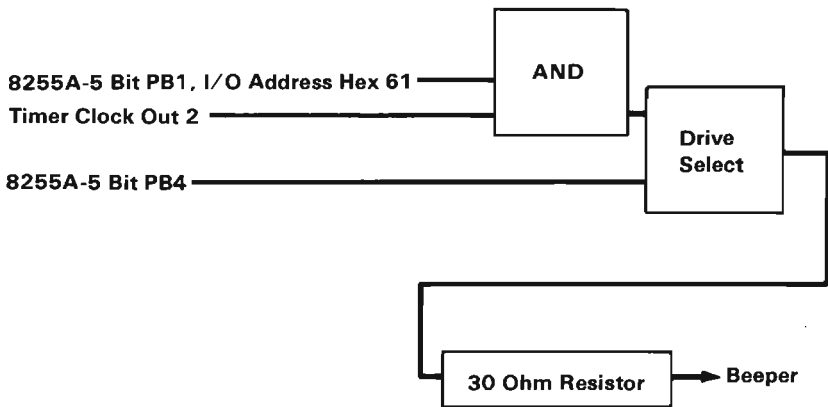
Notes:

Beeper

The system beeper is a small, piezoelectric- speaker, which can be driven from one or both of two sources. The two sources are:

- The 8255A-5 PPI output-bit PB1
- A timer clock out of an 8253-5 timer which has a 1.19 MHz-clock input. The timer gate is also controlled by an 8255-5 output bit PB0.

Note: The TI76496 Sound Generator cannot be directed through the beeper.



Beeper Block Diagram

Notes:

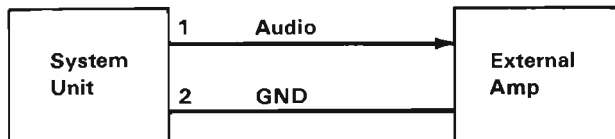
—

—

—

Sound Subsystem

The nucleus of the sound subsystem is an analog multiplexer (mpx) which allows 1 of 4 different sound sources to be selected, amplified, and sent to the audio outputs. The mpx and amplifier are configured so the amplifier's gain is unique to and consistent with each sound source. This provides a consistent level of output with any of the sound sources. The output of the amplifier is supplied to the IBM Connector for Television interface and external-amplifier interface. If an external speaker is used, an external amplifier must be used to drive it. The amplifier is configured as a single-pole low pass filter with a 3 dB cut-off frequency of 4.8 kHz. This filter is used to "round" off the corners of the square-wave signals. BIOS Power-on will initialize the sound subsystem to use the 8253 programmable-timer mode.



Connector Specifications

The audio output is a 1V peak-to-peak signal biased at 0V. It can drive a 10k ohm or greater input-impedence.

Source	Port	Bits
	PB6	PB5
Complex Sound Generator (TI 76496)	1	1
Programmable Timer (8253)	0	0
Cassette Audio	0	1
I/O Channel Audio	1	0

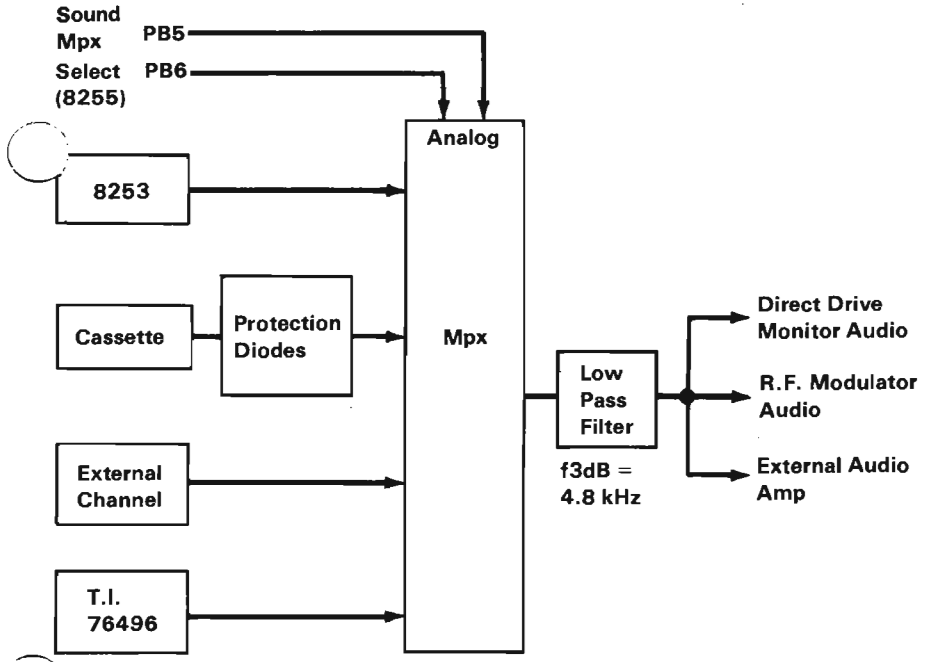
Port bits PB5 and PB6, of the 8255, control which source is selected.

Sound Sources

Complex Sound Generator

The Complex Sound Generator chip (SN76496N) has 3 programmable frequencies which may be mixed to form chords and a white noise generator which may also be mixed for special effects. Each of the 3 channels as well as the white noise generator can be independently attenuated. The processor controls the sound chip by writing to port hex C0.

The Sound Generator is described in greater detail later in this section. More information can be obtained by referring to Texas Instruments' data sheets and application notes.



Sound Block Diagram

Audio Tone Generator

Features

- 3 Programmable Tone-Generators
- Programmable White Noise
- Programmable Attenuation
- Simultaneous Sounds
- TTL Compatible
- 3.579 MHz Clock Input
- Audio Mixer

Processor to Sound-Generator Interface

The system microprocessor communicates with the SN76496N through the 8 data lines and 3 control lines

(WE, CE and READY). Each tone generator requires 10 bits of information to select the frequency and 4 bits of information to select the attenuation. A frequency update requires a double-byte transfer, while an attenuator update requires a single-byte transfer.

If no other control registers on the chip are accessed, a tone generator may be rapidly updated by initially sending both types of frequency and register data, followed by just the second byte of data for succeeding values. The register address is latched on the chip, so the data will continue going into the same register. This allows the 6 most-significant bits to be quickly modified for frequency sweeps.

Control Registers

The sound generator has 8 internal registers which are used to control the 3 tone generators and the noise source. During all data transfers to the sound generator, the first byte contains a 3-bit field which determines the destination control register. The register address codes are as follows:

Register Address Field			Destination Control Register
MSB R0	R1	LSB R2	
0	0	0	Tone 1 Frequency
0	0	1	Tone 1 Attenuation
0	1	0	Tone 2 Frequency
0	1	1	Tone 2 Attenuation
1	0	0	Tone 3 Frequency
1	0	1	Tone 3 Attenuation
1	1	0	Noise Control
1	1	1	Noise Attenuation

Register Address Field

1	Reg. Addr.			Low Data				0	High Data							
	R0	R1	R2	F6	F7	F8	F9		X	F0	F1	F2	F3	F4	F5	
Bit	First Byte						Bit	Bit	Second Byte							Bit
0							7	0								7
MSB							LSB	MSB								LSB

Frequency (Double or Single Byte Transfer)

Frequency Generation

Each tone generator consists of a frequency-synthesis section and an attenuation section. The frequency-synthesis section requires 10 bits of information (hex F0-F9) to define half the period of the desired frequency (n). Hex F0 is the most-significant bit and hex F9 is the least-significant bit. This information is

loaded into a 10-stage tone-counter, which is decremented at an $N/16$ rate where N is the input-clock frequency. When the tone counter decrements to 0, a borrow signal is produced. This borrow signal toggles the frequency flip-flop and also reloads the tone counter. Thus, the period of the desired frequency is twice the value of the period register.

The frequency can be calculated by the following:

$$f = \frac{N}{32n}$$

where N = ref clock in Hz (3.579 MHz)

n = 10-bit binary-number

Attenuator

1	Reg. Addr.			Data			
	R0	R1	R2	A0	A1	A2	A3
Bit 0	Second			Bit 7			
MSB	Byte			LSB			

Update Attenuation (Single Byte Transfer)

The output of the frequency flip-flop feeds into a four-stage attenuator. The attenuator values, along with their bit position in the data word, are shown in the following figure. Multiple-attenuation control-bits may be 'true' simultaneously. Thus, the maximum theoretical attenuation is 28 dB typically.

Bit Position				
MSB A0	A1	A2	LSB A3	Weight
0	0	0	1	2dB
0	0	1	0	4dB
0	1	0	0	8dB
1	0	0	0	16db
1	1	1	1	OFF

Attenuator Values

Noise Generator

1	Reg. Addr.			X	FB	SHIFT	
	R0	R1	R2			NF0	NF1
	1	1	0				
MSB				LSB			

Update Noise Source (Single Byte Transfer)

The noise generator consists of a noise source and an attenuator. The noise source is a shift register with an exclusive-OR feedback-network. The feedback network has provisions to protect the shift register from being locked in the zero state.

FB	Configuration
0	Periodic Noise
1	White Noise

Noise Feedback Control

Whenever the noise-control register is changed, the shift register is cleared. The shift register will shift at one of four rates as determined by the two NF bits. The fixed shift-rates are derived from the input clock.

Bits		Shift Rate
NF0	NF1	
0	0	$N/512$
0	1	$N/1024$
1	0	$N/2048$
1	1	Tone Generator #3 Output

Noise Generator Frequency Control

The output of the noise source is connected to a programmable attenuator.

Audio Mixer/Output Buffer

The mixer is a conventional operational-amplifier summing-circuit. It will sum the three tone-generator

outputs, and the noise-generator output. The output buffer will generate up to 10 mA.

Data Transfer

The sound generator requires approximately 32 clock cycles to load the data into the register. The open collector READY output is used to synchronize the microprocessor to this transfer and is pulled to the false state (low voltage) immediately following the leading edge of CE. It is released to go to the true state (external pull-up) when the data transfer is completed.

This will insert approximately 42 wait states (8.9 μ s) for each data transfer.

Warning: Do not attempt to issue an I/O read operation to the TI76496 port (COH). Such an operation will cause the system to hang indefinitely.

Note: If DMA is added to the system on the I/O channel, I/O WRITES to the 76496 will increase the latency time.

Notes:

—

—

—

Infra-Red Link

The infra-red link provides cordless communications between the keyboard and the system unit. Two infra-red-emitting diodes, mounted in the keyboard, transmit coded information to the system unit. The keyboard transmitter is fully discussed in “Cordless Keyboard” in this section. The infra-red receiver, which is located in the system unit, has an infra-red-sensitive device that demodulates the signal transmitted from the keyboard and sends it to the system.

Infra-Red Receiver

The receiver card measures 57.15 mm wide by 63 mm (2.25 in. by 2.50 in.) long. The infra-red receiver is mounted on the system board, component-side down, with two snap-in-type standoffs. Signal output and power input is through an 8-pin connector, located at the rear of the infra-red receiver. The infra-red-sensitive device is located on the front of the board and receives its input through an opening in the front of the system unit’s cover. There is also an infra-red transmitter mounted on the receiver board for diagnostic purposes.

Functional Description

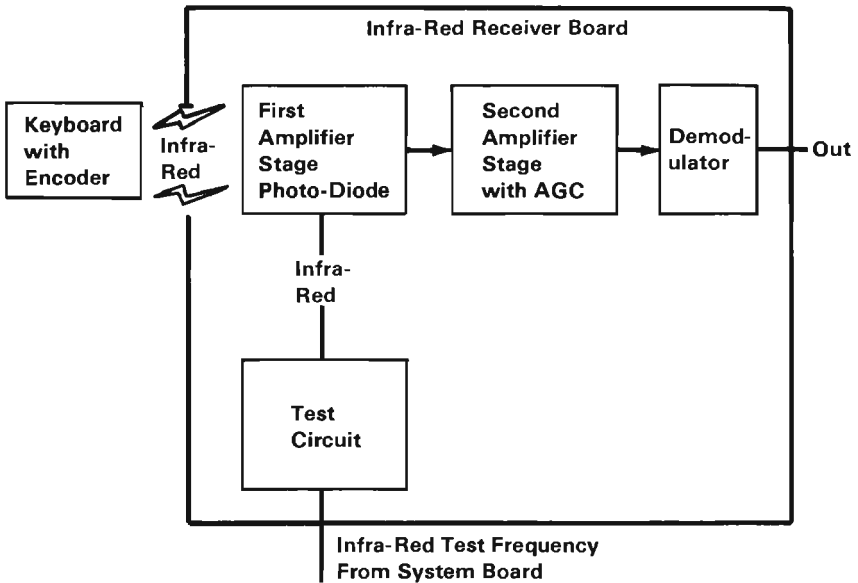
The following figure is the Infra-Red Receiver Block Diagram. During keyboard operation, the emitted light is modulated, transmitted, and received in the following sequence:

1. A key is pushed.

2. The data stream is sent using the infra-red-emitting diodes.
3. The receiver amplifies and processes the signal.
4. The demodulated signal is sent to the system board.

The signal received consists of an infra-red-light transmission modulated at 40 kHz.

An input is available (I/R Test Frequency) to the system for receiver-circuit-operational verification.



Infra-Red Receiver Block Diagram

Application Notes

The Infra-Red Receiver Board can serve as a general-purpose infra-red-receiver, however, the

demodulator timings are tailored to the needs of the system.

Programming Considerations

The serially-encoded word is software de-serialized by the 8088 processor on the system unit. The leading edge of the start bit will generate a non-maskable interrupt (NMI). Once the processor enters the NMI routine to handle the deserialization, the keyboard-data line is sampled and the processor waits to sample the trailing edge of the start bit. When the trailing edge of the start bit is sampled, the processor will wait for 310 μs and sample the first half of the first data bit. This delay causes the processor to sample in the nominal center of the first half of the first data bit. The processor then samples the keyboard data every half-bit cell-time. The sampling interval is 220 μs . The processor samples each half-bit-sample 5 times and will determine the logical level of the sample by majority rule. This enables the processor to discriminate against transient glitches and to filter out noise. The 8088 processor utilizes one 8255 PPI bit (PORT C BIT 6) and shares one 8253 timer channel (CHANNEL 1) to do the software de-serialization of the keyboard data. See the "Cordless Keyboard" in this section for more information on the data-transmission protocol.

Detectable Error Conditions

Errors	Cause
Phase Errors	The 1st half of the bit-cell sample is not equal to the inverse of the 2nd half of the bit-cell sample.
Parity Errors	The received encoded word did not maintain odd parity.

Note: Errors will be signaled by the processor with a short tone from the audio alarm or external speaker.

Operational Parameters

The operational distance from infra-red devices to the system should not exceed 6.1 meters (20 feet) (line-of-sight). Operational efficiency can be impaired by outside sources. These sources are, excessively-bright lights, and high-voltage lines, which include some TV sets. High-energy sources will generally cause an audible alarm within the system unit. These sources may downgrade the operational distance from the keyboard to the system. A keyboard cable is recommended if the above interference conditions are not controllable.

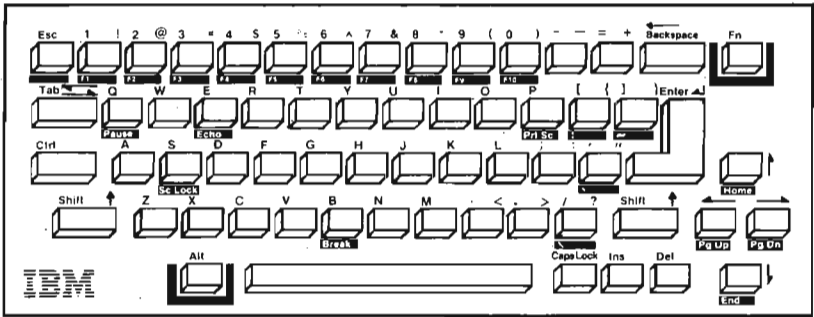
Pin	Signal	Input/Output
A01	+12 Volts	Input
A02	Ground	Input
A03	Ground-Shield	Input
A04	I.R. TEST FREQ.	Input
B01	GROUND	Input
B02	+5 Volts	Input
B03	-I.R. KBD DATA	Output
B04	GROUND	Input

Infra-Red Connector Specifications

IBM PCjr Cordless Keyboard

The keyboard is a low-profile, 62-key, detached keyboard with full-travel keys. The keys are arranged in a standard typewriter layout with the addition of a function key and cursor-control keys. The keybuttons are unmarked; however, an overlay is used to provide the keys' functional descriptions.

The following figure shows the layout of the cordless keyboard.



The keyboard is battery powered and communicates to the system unit with an infra-red (IR) link. The infra-red link makes the remote keyboard a truly portable hand-held device. An optional-cord connection to the system unit is available. Power is sent to the keyboard and serially-encoded data received by the system unit through the optional cord. When connected, the cord's keyboard-connector removes the battery power and the -CABLE CONNECT signal disables the infra-red-receiver circuit. The disabling of the circuit also allows other infrared devices to be used

without interfering with the system. The data which is received through the IR link or by the cord, have the same format.

The keyboard interface is designed to maximize system-software flexibility in defining keyboard operations such as shift states of keys, and typematic operation. This is accomplished by having the keyboard return scan codes rather than American National Standard Code for Information Interchange (ASCII) codes. The scan codes are compatible with Personal Computer and Personal Computer XT scan codes at the BIOS interface level. All of the keys are typematic 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 the make codes. The keyboard I/O driver can define keyboard keys as shift keys or typematic, as required by the application.

The microprocessor in the keyboard performs keyboard scanning, phantom-key detection, key debounce, buffering of up to 16 key-scan-codes, and transfer of serially-encoded data to the system unit. The keyboard microprocessor is normally in a standby power-down mode until a key is pressed. This causes the microprocessor to scan the keyboard. The microprocessor then transmits the scan code, and re-enters the power-down mode if its buffer is empty and no keys are pressed.

The keyboard electronics is designed with low-power CMOS integrated-circuitry for battery power operation. Four AA-size batteries are required. Because the keyboard is normally in the standby power-down mode, which uses very little power, no on/off switch is needed.

Unlike other keyboards in the IBM Personal Computer family, the IBM PCjr Cordless Keyboard has phantom-key detection. Phantom-key detection occurs when invalid combinations of three or more keys are pressed simultaneously, causing a hex 55 scan-code to be sent to the keyboard's processor. The phantom-key scan-code instructs the keyboard's processor to ignore all of the keys that were pressed at that time. BIOS ignores the resulting scan-code that is sent to it.

The keyboard-cord connector provides a battery-disconnect function and also disables the infra-red-transmission circuitry when the mating plug for the modular jack is connected.

Note: See "Keyboard Encoding and Usage" in Section 5, for scan codes and further information.

Transmitter

Serially encoded words are transmitted to the system unit using the Infra-Red Link or the cable link. Encoded words are sent to the system unit with odd parity. Both the Infra-Red Link and the cable link use biphasic serial-encoding and each is a simplex link.

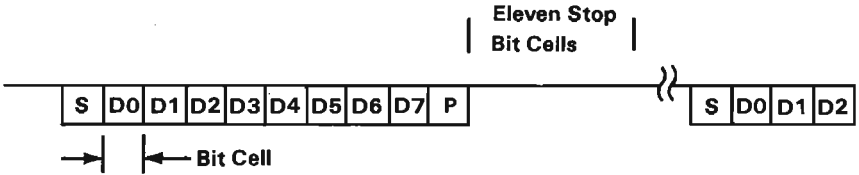
The 80C48 microprocessor does the biphasic serial encoding with a bit cell of 440 μ s. A biphasic logically-encoded 1 is transmitted as logical 1 for the first half of the bit cell time and as a logical 0 for the second half of the bit cell. A biphasic logically-encoded 0 is transmitted as a logical 0 for the first half of the bit cell time and as a logical 1 for the second half of the bit cell.

Each logical 1 transmission for the Infra-Red Link consists of a 40 kHz carrier burst at a 50% duty cycle.

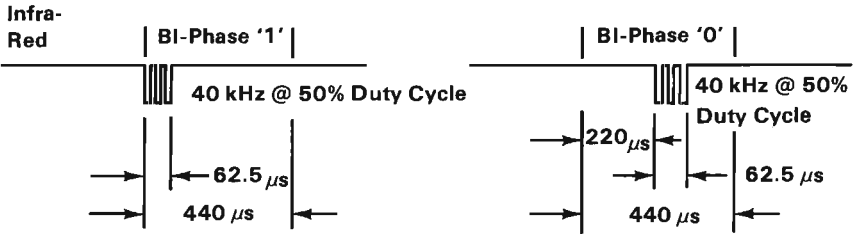
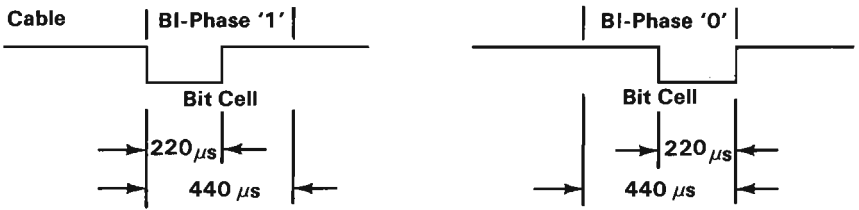
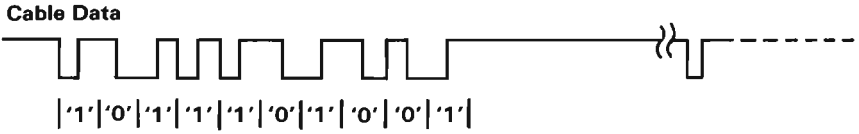
First Bit	Start Bit
Second Bit	Data Bit 0 (Least Significant Bit)
Third Bit	Data Bit 1
Fourth Bit	Data Bit 2
Fifth Bit	Data Bit 3
Sixth Bit	Data Bit 4
Seventh Bit	Data Bit 5
Eight Bit	Data Bit 6
Ninth Bit	Data Bit 7 (Most Significant Bit)
Tenth Bit	Parity Bit
Eleventh Bit	Stop Bit

Data Stream Sequence

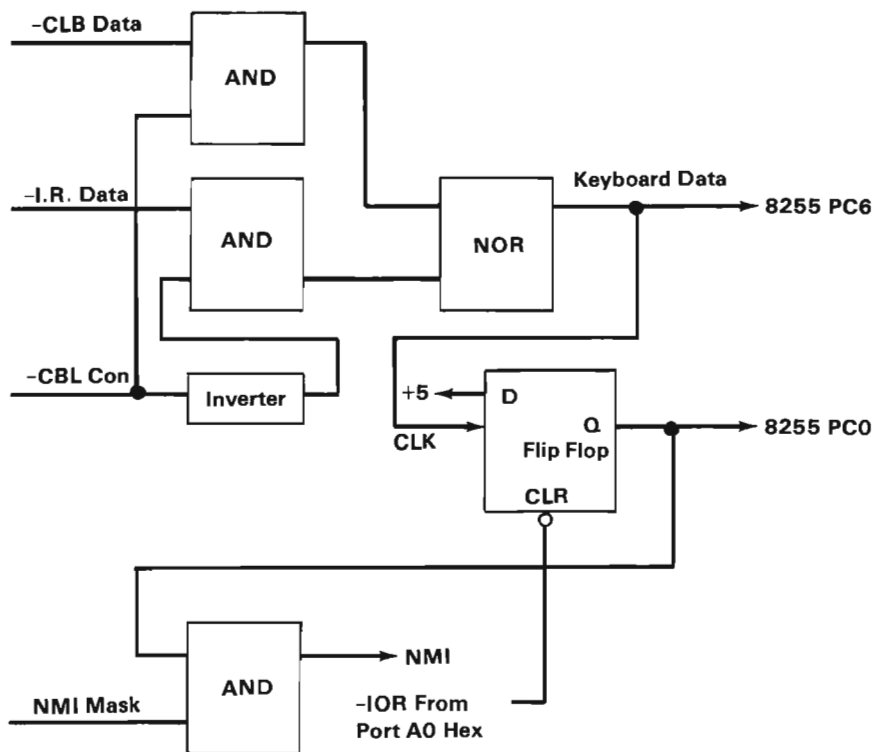
Eleven stop bits are inserted after every scan-code transmission. This is to allow some processor bandwidth between keystrokes to honor other types of interrupts, such as serial and time-of-day.



Example: DATA = "2EH" PARITY = '1'



Keyboard Transmission Timing



Keyboard Interface Logic

Program Cartridge and Interface

The Program Cartridge allows the addition of ROM to the system without removing the cover by plugging it into either of two slots in the front of the machine.

The 48 by 72 mm (2 by 3 inch) cartridge can hold one or two 32K byte by 8 ROMS (64K bytes total) of program storage. Smaller ROMS such as the 8K byte by 8 modules can be used in the cartridge. When a smaller module is used, the higher address lines are not used. To allow two smaller modules to be mapped to adjacent memory segments, each module's contents is addressed to multiple adjacent-memory segments, within the addressable range of the module's socket (32k).

Program Cartridge Slots

The Program Cartridge is designed to plug into either of two identical slots in the front of the machine. Each slot has 15 address signals, 8 data signals, 6 chip selects, 2 control signals, and power. Cartridge selection is accomplished by the chip selects, each of which addresses one of the high 32K memory-blocks. Each cartridge uses up to two of the six chip selects. Selection is determined on the basis of the intended use of the cartridge. This is done at the factory.

Two of the chip selects are used by the internal system-ROM. These two signals can be used to allow the internal ROM to be replaced by a Program Cartridge. This allows the machine to assume a different personality from the standard machine. To use this option of mapping the internal-ROM space to a cartridge, the Base-ROM-in-Cartridge function must be inserted. This function is a factory-installed

signal-jumper manufactured into particular program-cartridges that are intended to replace the system ROM.

Note: When the cartridge is inserted or removed with the system turned on, the system will 'reset' and go through a warm power-up. Any data in the system RAM will be lost.

Cartridge Storage Allocations

A. The following conventions will be followed for "Initial Program Loadable" program cartridges:

Location	Contents
0	055H
1	0AAH
2	Length
3,4,5	Jump to Initialize Code
6	0
Last 2 Addresses	CRC Bytes

Storage Conventions

- Locations 0 and 1 contain the word hex 55AA. This is used as a test for the presence of the cartridge during the configuration- determination portion of the power-on routines.
- Location 2 contains a length indicator representing the entire address space taken by the ROM on the cartridge. The algorithm for determining the

contents of this byte is (length/512). The contents of this byte is used by the CRC (cyclic-redundancy-check) routine to determine how much ROM to check.

- Location 3 contains the beginning of an initialization routine that is reached by a 'Long' call during the power-on sequence. For cartridges that are 'IPL-able' (BASIC or assembler program) this routine should set the INT hex 18 vector to point to their entry points. Other types of cartridges (BASIC or whatever) should merely 'return' to the caller. Setting the INT hex 18 vector will enable transfer of control to the cartridge program by the IPL routine.
- This location 6 should be 00.
- CRC bytes: The last two locations of the address space used by the cartridge must be blank. CRC characters will be placed in these bytes when the cartridge is built. See the routine at label "CRC Check", in the BIOS listing for the CRC algorithm.

B. The following conventions will be followed for cartridges that wish to be recognized by DOS 2.1 as containing code associated with DOS command words:

Location	Contents
0	055H
1	0AAH
2	Length
3-5	Jump to Initialize
6	Command Name Length (Offset Y-Offset Z)
Z	First Character in Command Name
Y	Last Character in Command Name
W	Word Pointing to Routine that is Jumped to if "Name" is Typed
X	Next Command Name Length or "00" if No More Command Names
Last 2 Addresses	CRC Bytes

DOS Conventions

- Locations 0 and 1 contain the word hex 55AA. This is used as a test for the presence of the cartridge during the configuration- determination portion of the power-on routines.
- Location 2 contains a length indicator representing the entire address space taken by the ROM on the cartridge. The algorithm for determining the contents of this byte is (length/512). The contents of this byte is used by the CRC routine to determine how much ROM to check.
- Location 3 contains a 'jump' to the initialization code for this ROM. (May just be a 'Far Return')
- Starting at location 6 may be a sequence of command name pointers consisting of 1: Count of length name, 2: Name in ASCII, and 3: Word

containing offset within this segment to the code that is entered when this name is called. There can be as many names as desired, providing that a hex 00 is placed in the count field following the last name pointer. If a cartridge has a routine called 'TEST' at location hex 0FB5 (offset from start of segment that the cartridge is in) that needs to be executed when 'test' is entered as a DOS command the entry at location 6 would be hex 04,54,45,53,54,B5,0F.

- CRC bytes: The last two locations of the address space used by the cartridge must be blank. CRC characters will be placed in these bytes when the cartridge is built. See the routine at label "CRC Check", in the BIOS listing for the CRC algorithm.

C. The following conventions will be followed for cartridges that wish to be recognized by "Cartridge BASIC" as containing interpretable-BASIC Code:

- The cartridge-chip selects must address hex D0000 since the BASIC cartridge addresses hex E0000. When "Cartridge BASIC" is activated, it will check for a second cartridge program at hex D0000. If the second cartridge is present and formatted properly, then the BASIC code is loaded into RAM and run.
- The format for this interpretable-BASIC code must be as follows:

Location	Contents
0	055H
1	0AAH
2	Length
3	0CBH
4	0AAH
5	055H
6	0
7	0FFH if unprotected Basic program or 0FEH if protected Basic program
8	Start of interpretable Basic code
n	0FFH Padding to next 2048 byte boundary
Last 2 Addresses	CRC Bytes

Cartridge Format

1. Locations 0 and 1 contain the word hex 55AA. This is used as a test for the presence of the cartridge during the configuration-determination portion of the power-on routines.
2. Location 2 contains a length indicator representing the entire address space taken by the ROM on the cartridge. The algorithm for determining the contents of this byte is $(\text{length}/512)$. The contents of this byte is used by the CRC routine to determine how much ROM to check.
3. Location 3 must be hex 0CB for a 'far return' instruction.

4. Locations 4 and 5 contain the word hex AA55. This is used as a test for the presence of the second cartridge by "Cartridge Basic".
5. Location 6 must be a 0 to follow the DOS conventions.
6. Location 7 can be either hex FF to indicate an unprotected BASIC program, or hex FE to indicate a protected program.
7. Location 8 must be the start of the BASIC program. It must be interpretable Basic and not compiled. Also, at the end of the program PAD to the next 2048 byte boundary with hex OFF.
8. CRC bytes: The last two locations of the address space used by the cartridge must be blank. CRC characters will be placed in these bytes when the cartridge is built. See the routine at label "CRC Check", in the BIOS listing for the CRC algorithm.

ROM Module

The ROM modules used are 250 ns devices. Typical modules are the Mostek MK37000 and MK38000, the TMM 23256, the SY23128, and other compatible devices.

ROM Chip Select	Hex Address Space	Typical Use
CS0	X	Not Used
CS1	X	Not Used
CS2	D0000-D7FFF	Optional Cartridge ROM #2
CS3	D8000-DFFFF	Optional Cartridge ROM #1
CS4	E0000-E7FFF	Standard Cartridge ROM #2
CS5	E8000-EFFFF	Standard Cartridge ROM #1
CS6	F0000-F7FFF	System Board ROM #2
CS7	F8000-FFFFF	System Board ROM #1

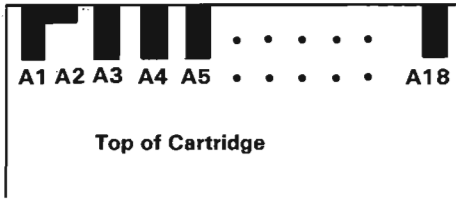
ROM Chip Select Table

Signal	I/O	Description
A0 - A14	0	Processor Address lines A0 - A14
D0 - D7	I	Processor Data lines

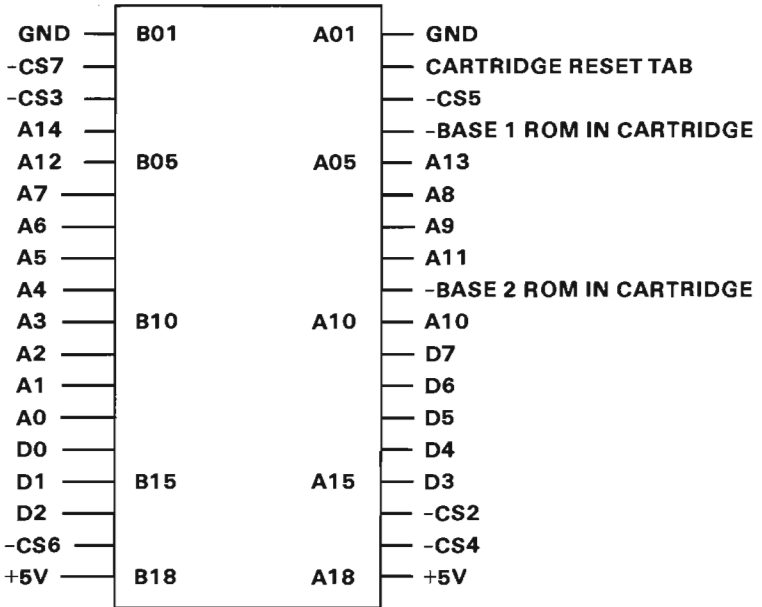
- CS2
THRU
-CS7** **0** These chip-select lines are used to select ROM modules at different addresses. The addresses for each chip-select are shown in the ROM-chip select-table. -CS6 and -CS7 are used on the system board for BIOS, Power-On-Self-Test (POST) and cassette-basic ROMs. In order to use these chip selects on a cartridge, **-BASE 1 ROM IN CARTRIDGE** or **-BASE 2 ROM IN CARTRIDGE** must be pulled 'low'
- BASE 1
ROM IN
CARTRIDGE** **I** This line when pulled 'low' instructs the system board to de-gate the ROM module from hex F8000 - FFFFF on the system board. This ROM module can then be replaced by a ROM module on the cartridge by using **-CS7**.
- BASE 2 ROM
IN
CARTRIDGE** **I** This line when pulled 'low' instructs the system board to de-gate the ROM module from hex F0000 - F7FFF on the system board. This ROM module can then be replaced by a ROM module on the cartridge by using **-CS6**.

Cartridge Reset I
Tab

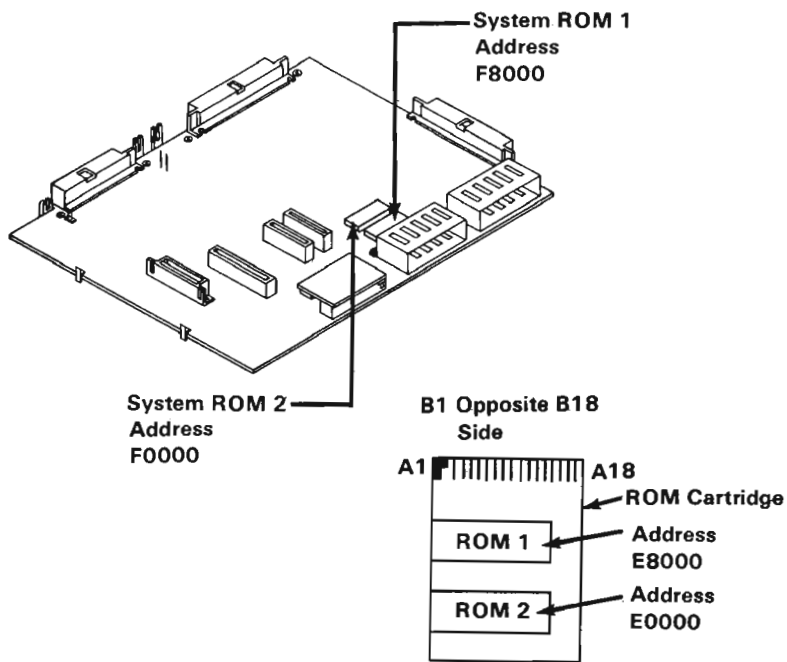
This input when 'low' causes a 'reset' to the system. The system will remain 'reset' until this line is brought back 'high'. This tab is usually wired with an L shaped land pattern to the GND at A02 which provides a momentary 'reset' when a cartridge is inserted or removed.



Momentary Reset Land



Connector Specification



Cartridge ROM Locations

Games Interface

Interface Description

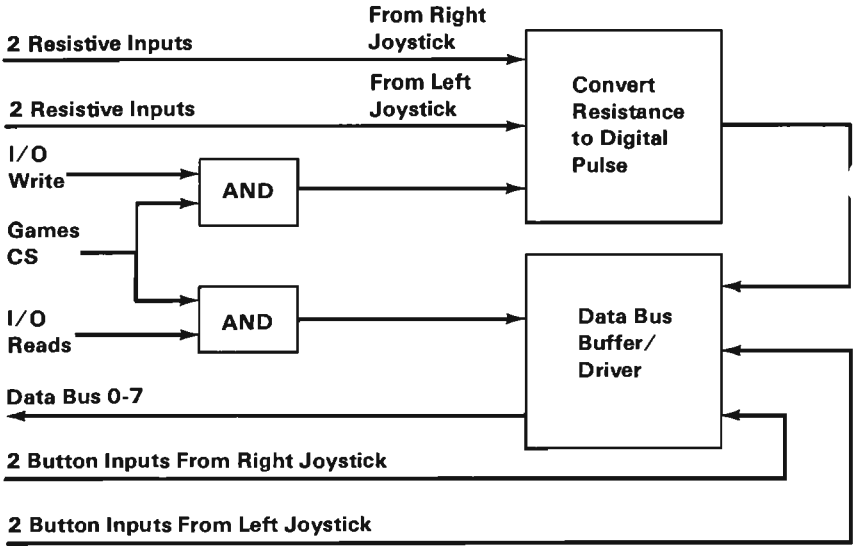
The Game Interface has two connectors located at the rear of the System unit for four paddles (two per connector) or two joysticks. Each connector has four input lines: two digital inputs and two resistive inputs. All the inputs are 'read' with one 'IN' from address hex 201. The interface, plus system software, converts the present resistive value to a relative paddle or joystick-position. On receipt of an output signal, four timing circuits are started. By determining the time required for the circuit to time out (a function of the resistance), the paddle or joystick position can be determined.

The four digital inputs each have a 1K ohm resistor to pull the voltage up to +5V. With no drive on these inputs, a 1 is read. For a 0 reading, the inputs must be pulled to ground.

The four resistive inputs are converted to a digital pulse with a duration proportional to the resistive load, according to the following equation:

$$\text{Time} = 24.2 \mu\text{s} + 0.011 (r) \mu\text{s}$$

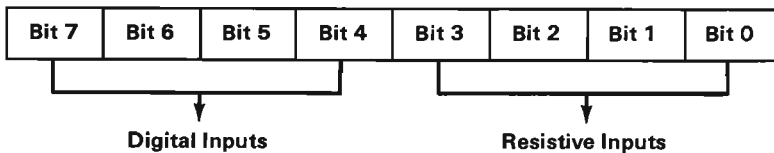
Where r is the resistance in ohms



Games Interface Block Diagram

Any program application must first begin the conversion by an 'OUT' to address hex 201. An 'IN' from address hex 201 will show the digital pulse go 'high' and remain 'high' for the duration according to the resistance value. All four bits (Bit 3 through Bit 0) function in the same manner. Each bits digital pulse goes high simultaneously and resets independently according to the input resistance value.

Input from Address Hex 201



Input From Address Hex 201

Joysticks typically have one or two buttons and two variable resistances each. The variable resistances are mechanically linked to have a range from 0 to 100k ohms. One variable resistance indicates the X coordinate and the other variable resistance indicates the Y coordinate. The joysticks are attached to give the following input data:

Joystick B		Joystick A		Joystick B		Joystick A	
Button #2	Button #1	Button #2	Button #1	Coord. Y	Coord. X	Coord. Y	Coord. X
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

Joystick Input Data

The game paddles have one button each and one variable resistance each. The variable resistance is mechanically linked to have a range from 0 to 100k ohms. The paddles are attached to give the following input data.

Buttons				Coordinates			
Paddle D	Paddle C	Paddle B	Paddle A	Paddle D	Paddle C	Paddle B	Paddle A
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

Paddle Input Data

Pushbuttons

The pushbutton inputs are 'read' by an 'IN' from address hex 201. These values are seen on data bits 7 through 4. These buttons default to an 'open' state and are 'read' as 1. When a button is pressed, it is 'read' as 0.

Note: Software should be aware that these buttons are not debounced in hardware.

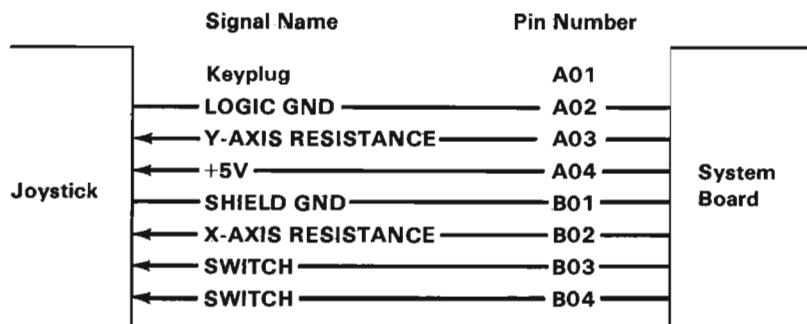
Joystick Positions

The joystick position is indicated by a potentiometer for each coordinate. Each potentiometer has a range from 0 to 100k ohms that varies the time constant for each of the four one-shots. As this time constant is set at different values, the output of the one-shot will be of varying durations.

All four one-shots are fired simultaneously by an 'OUT' to address hex 201. All four one-shot outputs

will go 'true' after the fire pulse and will remain 'high' for varying times depending on where each potentiometer is set.

These four one-shot outputs are 'read' by an 'IN' from address hex 201 and are seen on data bits 3 through 0.



Connector Specification

Notes:

Serial Port (RS232)

The PCjr serial port is fully programmable and supports asynchronous communications only. It will add and remove start bits, stop bits, and parity bits. A programmable baud-rate generator allows operation from 50 baud to 4800 baud. Five, six, seven or eight bit characters with 1, 1-1/2, or 2 stop bits are supported. A fully-prioritized interrupt-system controls transmit, receive, line status and data-set interrupts. Diagnostic capabilities provide loopback functions of transmit/receive and input/output signals.

The nucleus of the adapter is a 8250A LSI chip or functional equivalent. Features in addition to those previously listed are:

- Full double-buffering eliminates the need for precise synchronization
- Independent receiver clock input
- Modem control functions: clear to send (CTS), request to send (RTS), data set ready (DSR), data terminal ready (DTR)
- Even, odd, or no-parity-bit generation and detection
- False start bit detection
- Complete status reporting capabilities
- Line-break generation and detection
- Break, parity, overrun, and framing error simulation
- Full prioritized interrupt system controls

All communications protocol is a function of the system ROM and must be loaded before the adapter is operational. All pacing of the interface and control-signal status must be handled by the system software. It should be noted that Asynchronous (Async) receive operations cannot overlap diskette operation since all but the Diskette Interrupt are masked 'off' during diskette operations. If Async receive

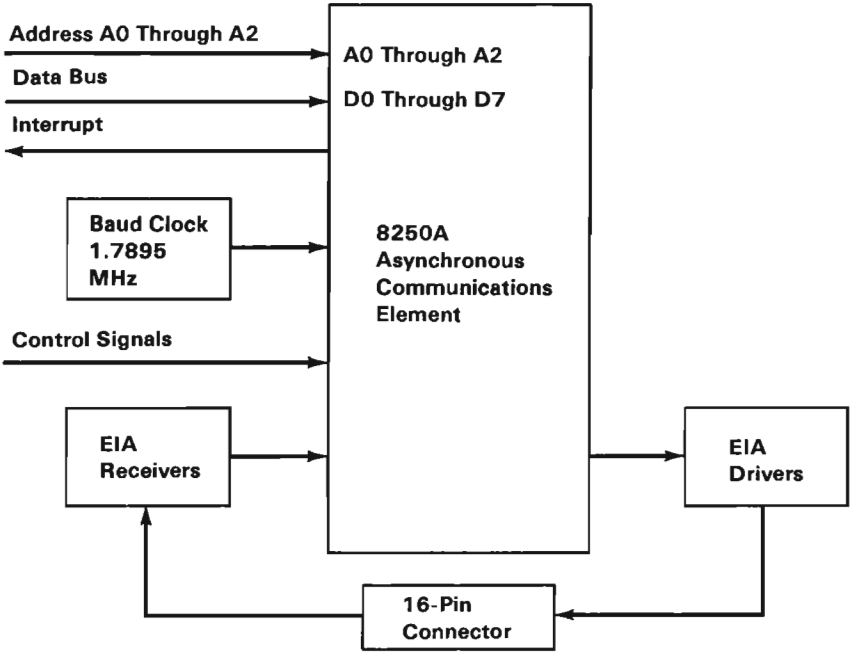
operations are going to be overlapped with keyboard receive operations, the Async Receiver rate cannot exceed 1200 baud. This is due to the processor deserialization of the keyboard. See IBM PCjr Cordless Keyboard in this section for more information.

Programming Note: Due to the read/write cycle-time of the 8250A, it is recommended that back-to-back I/O operations to the 8250A be avoided. A good Programming Technique would be to insert a short 'jump' between every consecutive 8250 I/O instruction. This action will flush the queue and provide 15 clock periods between I/O operations.

Note: This note only applies to programmers using the 8250A directly. It is **STRONGLY** suggested that the user not communicate directly with the physical hardware, but use the system BIOS instead.

Note: It is important to note that when the IBM PCjr has the Internal Modem installed it is logically COM1 and the RS232 serial port is logically COM2 in BIOS, DOS, and BASIC. Without the Internal Modem installed the RS232 serial port is logically addressed as COM1 in BIOS, DOS, and BASIC even though its address is still hex 2F8 using Interrupt level 3.

The following figure is a Serial Port Block Diagram:



Serial Port Block Diagram

Modes of Operation

The different modes of operation are selected by programming the 8250A asynchronous communications element. This is done by selecting the I/O address (hex 2F8 to 2FF) and 'writing' data out to the card. Address bits A0, A1, and A2 select the different registers that define the modes of operation. Also, the divisor-latch access-bit (bit 7) of the line-control register is used to select certain registers.

I/O Decode (in Hex)	Register Selected	DLAB State
2F8	TX Buffer	DLAB=0 (Write)
2F8	RX Buffer	DLAB=0 (Read)
2F8	Divisor Latch LSB	DLAB=1
2F9	Divisor Latch MSB	DLAB=1
2F9	Interrupt Enable Register	DLAB=0
2FA	Interrupt Identification Registers	(Don't Care)
2FB	Line Control Register	(Don't Care)
2FC	Modem Control Register	(Don't Care)
2FD	Line Status Register	(Don't Care)
2FE	Modem Status Register	(Don't Care)
2FF	Scratch Register	(Don't Care)

I/O Decodes

Address Range hex 2F8 - 2FF

Note: The state of the divisor-latch access-bit (DLAB), which is the most-significant bit of the line-control register, affects the selection of certain 8250A registers. The DLAB must be set 'high' by the system software to access the baud-rate-generator divisor latches.

Interrupts

One interrupt line is provided to the system. This interrupt is IRQ3 and is 'positive active'. To allow the serial port to send interrupts to the system, bit 3 of the modem control register must be set to 1 'high'. At this point, any of the following interrupt types 'enabled' by bits in the interrupt-enable register will cause an interrupt: Receiver-line status, Received Data available, Transmitter-Holding-Register empty, or Modem Status.

Interface Description

The communications adapter provides an EIA RS-232C electrically-compatible interface. One 2 by 8-pin Berg connector is provided to attach to various peripheral devices.

The voltage interface is a serial interface. It supports data and control signals as follows:

Pin A04	Transmit Data
Pin A08	Receive Data
Pin A03	Request to Send
Pin A07	Clear to Send
Pin A06	Data Set Ready
Pin B02-B08	Signal Ground
Pin A05	Carrier Detect
Pin A02	Data Terminal Ready
Pin B01	Shield Ground

The adapter converts these signals to/from TTL levels to EIA voltage levels. These signals are sampled or generated by the communications-control chip. These

signals can then be sensed by the system software to determine the state of the interface or peripheral device.

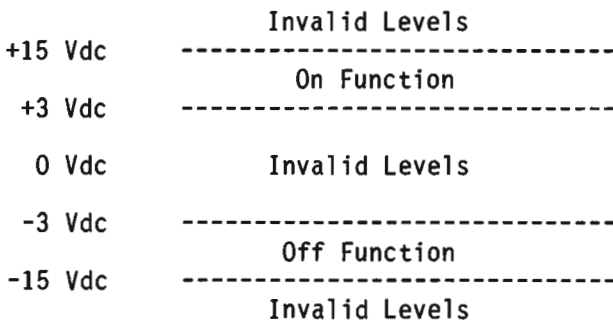
Note: The above nomenclature describes the communications adapter as a DTE (Data Terminal Equipment) device. Suitable adapters must be used to attach other devices such as serial printers.

Note: Ring Indicate is not supported on the PCjr.

Voltage Interchange Information

Interchange Voltage	Binary State	Signal Condition	Interface Control Function
Positive Voltage = Negative Voltage =	Binary (0) Binary (1)	= Spacing = Marking	= On = Off

Voltage Interchange Information



The signal will be considered in the 'marking' condition when the voltage on the interchange circuit, measured at the interface point, is more negative than

-3 Vdc with respect to signal ground. The signal will be considered in the 'spacing' condition when the voltage is more positive than +3 Vdc with respect to signal ground. The region between +3 Vdc and -3 Vdc is defined as the transition region, and considered an invalid level. The voltage which is more negative than -15 Vdc or more positive than +15 Vdc will also be considered an invalid level.

During the transmission of data, the 'marking' condition will be used to denote the binary state 1, and the 'spacing' condition will be used to denote the binary state 0.

For interface control circuits, the function is 'on' when the voltage is more positive than +3 Vdc with respect to signal ground and is 'off' when the voltage is more negative than -3 Vdc with respect to signal ground.

For detailed information regarding the INS8250A Communications Controller, refer to "Bibliography".

Output Signals

Output 1 (OUT 1), Pin 34: Output 1 of the 8250A is not supported in PCjr hardware.

Output 2 (OUT 2), Pin 31: Output 2 of the 8250A is not supported in PCjr hardware.

Accessible Registers

The INS8250A has a number of accessible registers. The system programmer may access or control any of

the INS8250A registers through the processor. These registers are used to control INS8250A operations and to transmit and receive data. For further information regarding accessible registers, refer to "Bibliography".

INS8250A Programmable Baud Rate Generator

The INS8250A contains a programmable baud rate generator that is capable of taking the clock input (1.7895 MHz) and dividing it by any divisor from 1 to (65535). The output frequency of the Baud Rate Generator is 16 x the baud rate [divisor number = (frequency input) / (baud rate x 16)]. Two 8-bit latches store the divisor in a 16-bit binary-format. These divisor latches must be loaded during initialization in order to ensure desired operation of the baud rate generator. Upon loading either of the divisor latches, a 16-bit baud-counter is immediately loaded. This prevents long counts on initial load.

The following figure illustrates the use of the baud rate generator with a frequency of 1.7895 MHz. For baud rates of 4800 and below, the error obtained is minimal.

Note: The maximum operating frequency of the baud generator is 3.1 MHz. In no case should the data rate be greater than 4800 baud.

Desired Baud Rate	Divisor Used to Generate 16x Clock		Percent Error Per Bit Difference Between Desired and Actual
	(Decimal)	(Hex)	
50	2237	8BD	.006
75	1491	5D3	.017
110	1017	1A1	.023
134.5	832	167	.054
150	746	12C	.050
300	373	175	.050
600	186	BA	.218
1200	93	5D	.218
1800	62	3E	.218
2000	56	38	.140
2400	47	2F	.855
3600	31	1F	.218
4800	23	17	1.291

Baud Rate at 1.7895 MHz

Note: These divisions are different than that used in the IBM Personal Computer. For portability, all initialization should be done through the system BIOS.

Note: Receive rates should not exceed 1200 baud if the receive operation is overlapped with keyboard keystrokes.

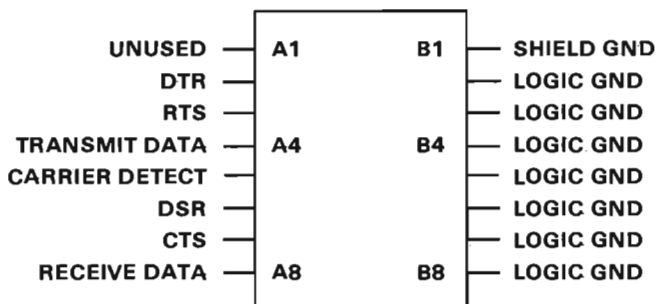
The following Assembly language sample program initializes the 8250. The baud rate is set to 1200 baud. It's data word is defined: 8 bits long with 1 stop bit odd parity.

```

BEGIN      PROC      NEAR
MOV        AL,80H    ; SET DLAB = 1
MOV        DX,2FBH  ; To Line Control Register
OUT        DX,AL
JMP        $+2      ; I/O DELAY
MOV        DX,2F8H  ; Point to LSB of Divisor Latch
MOV        AL,5DH   ; This is LSB of Divisor
OUT        DX,AL
JMP        $+2      ; I/O DELAY
MOV        DX,2F9H  ; Point to MSB of Divisor Latch
MOV        AL,0     ; This is MSB of Divisor
OUT        DX,AL
JMP        $+2      ; I/O DELAY
MOV        DX,2FBH  ; Line Control Register
MOV        AL,0BH   ; 8 Bits/Word, 1 Stop Bit,
                   ; Odd Parity, DLAB = 0
OUT        DX,AL
JMP        $+2      ; I/O DELAY
MOV        DX,2F8H
IN         AL,DX    ; In Case Writing to Port LCR Caused
                   ; Data Ready to go high
ENDP
BEGIN

```

Assembly Language Sample Program



Connector Specifications

System Power Supply

The system power supply is a 33 Watt, three voltage-level, two-stage supply. The first stage is an external power transformer that provides a single-fuse protected, extra low, ac-voltage output. The power cord is 3.08 meters (10.16 feet) long. The second stage is an internal, printed-circuit board, which is vertically mounted into the system board. The second stage converts the transformer's ac-output into three dc-output levels.

The amount of power available on the I/O connector for a machine that is fully configured with internal features is 400 mA of +5 Vdc, 0 mA of +12 Vdc and 0 mA of -6 Vdc.

Power is supplied to the system board through a printed-circuit-board edge-connector. The diskette drive is powered through a separate four-pin connector mounted on the front edge of the Power Board. The power for the diskette drive fan is provided by a three-pin Berg-type connector mounted directly below the diskette-drive connector. Power is removed from the system board and diskette drive by a switch mounted on the rear of the Power Board. Both the switch and the transformer connector are accessible from the rear of the system.

Operating Characteristics

Power Supply Input Requirements

Voltage (Vac)			Frequency	Current (Amps)
Nominal	Minimum	Maximum	± 5 Hz	Maximum
120	104	127	60 Hz	.65 at 104 Vac

Voltage ac

D.C Outputs

Vdc Voltage	Current (Amps)		Regulation Tolerance
Nominal	Minimum	Maximum	$\pm\%$
+5	*1.5	3.6	5
+12	.04	1.2	5
-6	0.0	.025	16

Voltage dc

* There must be a minimum of a 1.5 Amp load on the +5 Vdc output for the -6 Vdc to be present.

Over-Voltage/Over-Current Protection

Input (Transformer)

The following table describes the transformer input protection:

Voltage (Nominal)	Type Protection	Rating (Amps)
120 Vac	Non-resettable Fuse Thermal/Over-Current	5A Slo Blow

Input Protection

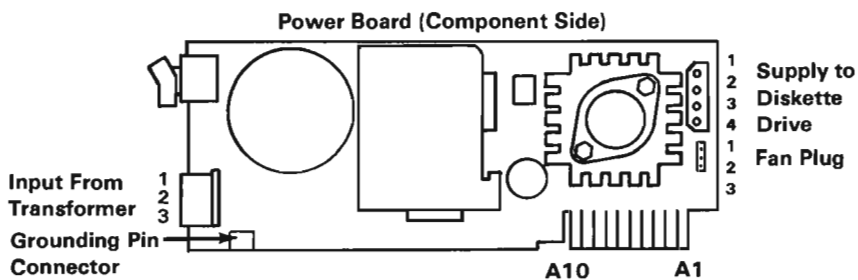
Output (Power Board)

The following table describes the Power Board's output protection:

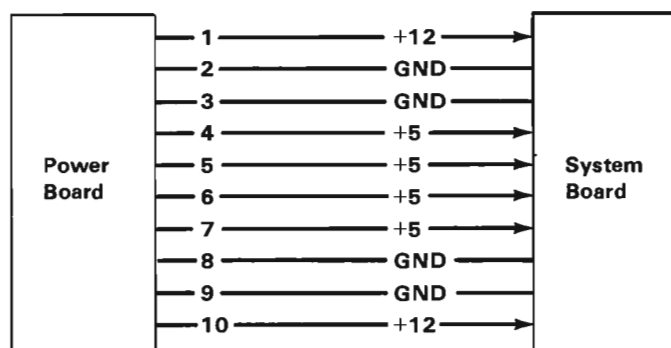
Output Voltages	Protection Condition	
	Over-Voltage	Over-Current
+5 Vdc	* $6.3 \pm .7$ Vdc	** $3.9 \pm .25$ Amps
12 Vdc	* 14.4 ± 1.4 Vdc	$2.2 \pm .9$ Amps

* Over-Voltage protection is provided by fuse F1.
**Resettable by removing the fault condition and removing power for at least 5 seconds and then applying power.

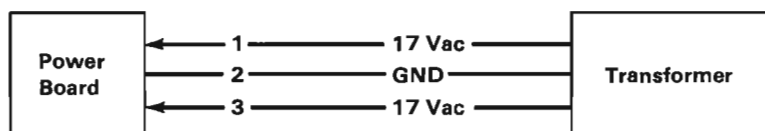
Output Protection



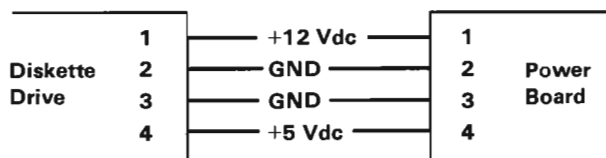
Connector Specifications



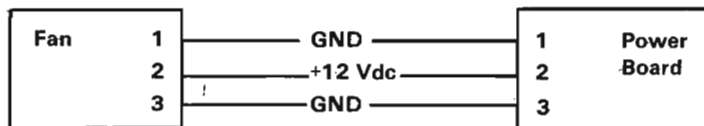
Connector Specifications



Connector Specifications



Connector Specifications



Fan Connector Specifications

Notes:

SECTION 3. SYSTEM OPTIONS

Contents

IBM PCjr 64KB Memory and Display Expansion	3-5
IBM PCjr Diskette Drive Adapter	3-13
Functional Description	3-15
Digital Output Register	3-15
WatchDog Timer	3-16
Floppy Disk Controller (FDC)	3-16
Programming Summary	3-17
Comments	3-16
System I/O Channel Interface	3-19
Drive Interface	3-22
Adapter Outputs	3-22
Adapter Inputs	3-24
Voltage and Current Requirements	3-24
IBM PCjr Diskette Drive	3-27
Functional Description	3-27
Diskette	3-31
IBM PCjr Internal Modem	3-33
Functional Description	3-34
Modem Design Parameters	3-37
Programming Considerations	3-40
Command Format	3-40
Commands	3-42
Responses	3-59
Editing/Changing Command Lines	3-59
Status Conditions	3-60
Dialing and Loss of Carrier	3-60

Default State	3-63
Programming Examples	3-63
Modes of Operation	3-68
Interrupts	3-70
Data Format	3-70
Interfaces	3-70
8250A to Modem Interface	3-70
Telephone Company Interface	3-74
System I/O Channel	3-74
IBM PCjr Attachable Joystick	3-77
Hardware Description	3-77
Functional Description	3-77
IBM Color Display	3-81
Hardware Description	3-81
Operating Characteristics	3-82
IBM Connector for Television	3-85
IBM PCjr Keyboard Cord	3-87
IBM PCjr Adapter Cable for Serial Devices ...	3-89
IBM PCjr Adapter Cable for Cassette	3-91
IBM PCjr Adapter Cable for the IBM Color Display	3-93
IBM PCjr Parallel Printer Attachment	3-95
Description	3-96
System Interface	3-98
Programming Considerations	3-99
Command Definition	3-99
IBM Graphics Printer	3-107
Printer Specifications	3-107

DIP Switch Settings	3-101
Parallel Interface Description	3-103
Data Transfer Sequence	3-103
Interface Signals	3-104
Printer Modes	3-106
Printer Control Codes	3-107
IBM PC Compact Printer	3-133
Printer Specifications	3-135
Serial Interface Description	3-139
Print Mode Combinations for the PC	
Compact Printer	3-140
Printer Control Codes and Functions	3-140

Notes:

IBM PCjr 64KB Memory and Display Expansion

The 64KB Memory and Display Expansion option enables the user to work with the higher density video modes while increasing the system's memory size by 64K bytes to a total of 128K bytes. The memory expansion option plugs into the 44-pin memory expansion connector on the system board. Only one memory expansion is supported.

The Memory Expansion Option does not require the user to reconfigure the system to recognize the additional memory.

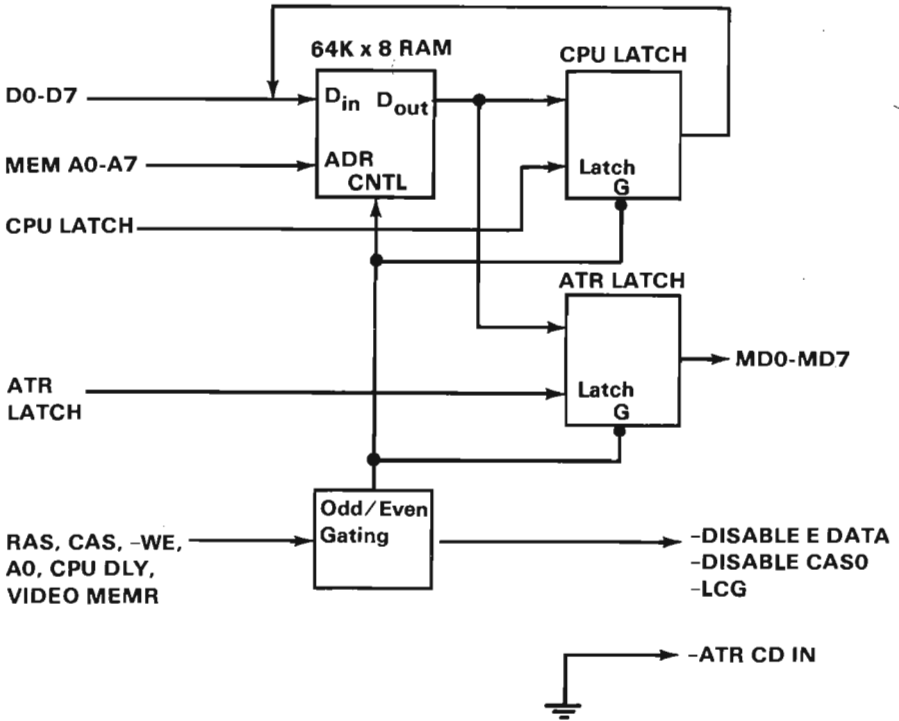
Eight 64K-by-1, 150 ns, dynamic memory modules provide 64K bytes of storage. The memory modules are Motorola's MCM6665AL15, and Texas Instrument's TMS4164-15, or equivalent.

When inserted, the memory expansion option uses the ODD memory space, while the system memory is decoded as the EVEN memory. Thus, when used as video memory, the memory expansion option has the video attributes while the on-board system memory has the video characters. This arrangement provides a higher bandwidth of video characters.

In addition to the eight memory modules, the expansion card has logic to do the EVEN/ODD address decoding, video data multiplexing, and a CARD PRESENT wrap.

Dynamic-refresh timing and address generation are done on the system board and used by the memory expansion option.

The following is a block diagram of the IBM PCjr 64KB Memory and Display Expansion.



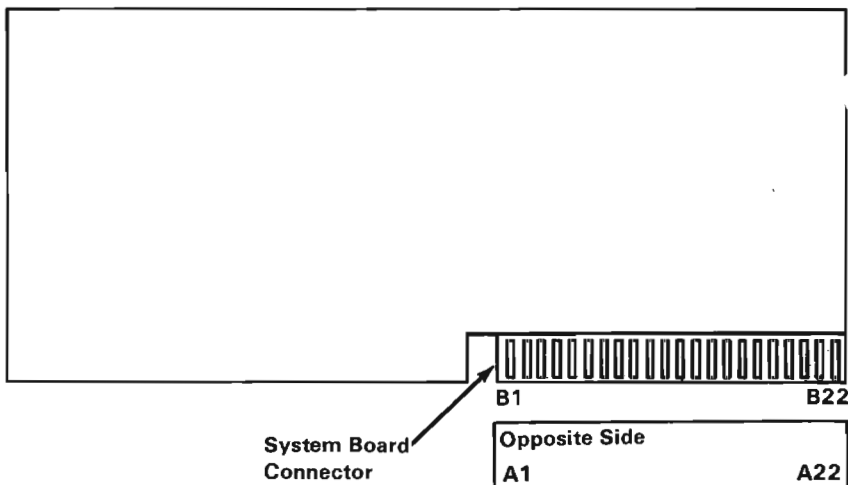
Memory Expansion Block Diagram

Signal	I/O	Description
+RAS	I	+Row Address Strobe. This line is inverted and then becomes the -RAS for the RAM modules.
+A0	I	Microprocessor Address 0. This is used to determine whether the microprocessor access is from the system board RAM (Low) or from the expansion RAM (High).
-DISABLE EDATA	O	When the expansion RAM card is in and the microprocessor is reading an ODD byte of data the expansion card tri-states the latch for EVEN data on the system board using this line.
ATR LATCH	I	This signal indicates that the expansion RAM card should 'latch' up data from the expansion RAM into the attribute latch.
MD0 thru MD7	O	These data lines contain CRT information from the attribute latch and go to the Video Gate Array.
D0 thru D7	I/O	These data lines are from the microprocessor and are bidirectional.
MEM A0 thru A7	I	These are the multiplexed address lines for the dynamic-RAM modules. These lines are multiplexed between row address and column

VIDEO MEMR	I	address, and also between microprocessor and CRT addresses. When this signal is 'high' it indicates a MEMR is accessing the system board or expansion RAM is being accessed. This line along with A0 determines if the expansion RAM microprocessor latch should 'gate' its data onto the D0 thru D7 Bus.
CPU DLY	I	This line when 'high' indicates that a microprocessor RAM cycle is occurring. It is used to gate 'off' the expansion RAM CAS or used with A0 to generate the -DISABLE CAS 0 signal.
-DISABLE CAS 0	O	This line is used to disable the system board CAS0 when a system microprocessor 'write' is occurring to the expansion RAM. This line keeps the 'write' from occurring to the system board RAM.
+CAS	I	Column Address Strobe. This line instructs the expansion RAM to 'latch' up the address on the MEM A0 thru A7 address lines.

-LCG	O	This line is used to instruct the system board that attributes or ODD graphics data should be 'read' from the expansion RAM card for use by the Video Gate Array.
GATE	I	This line is 'wrapped' and becomes the -LCG output.
-WE	I	This line instructs the memory that the cycle is a microprocessor 'write' cycle.
CPU LATCH	I	This line instructs the expansion RAM card to 'latch' the data from the expansion RAM into the microprocessor latch.
-ATR CD IN	O	This line is a wrap of the ground line on the expansion RAM card. It pulls 'down' an 8255 input so that the microprocessor can tell if this card is installed or not.

The following is the connector specifications for the IBM PCjr 64KB Memory and Display Expansion.



64KB Memory and Display Expansion

Connector Pin	Signal Name	Signal Name	Connector Pin
A01	+RAS	VIDEO MEMR	B01
A02	A0	CPU DLY	B02
A03	-DISABLE	-DISABLE	B03
A04	EDATA	CAS 0	B04
A05	ATR LATCH	+CAS	B05
A06	MD4	-LCG	B06
A07	MD5	GATE	B07
A08	MD6	Ground	B08
A09	MD7	Ground	B09
A10	MD0	Ground	B10
A11	MD1	-WE	B11
A12	MD2	CPU LATCH	B12
A13	MD3	-ATR CD IN	B13
A14	GND	GND	B14
A15	VCC	VCC	B15
A16	D7	D6	B16
A17	D5	D4	B17
A18	D3	D2	B18
A19	D1	D0	B19
A20	MEM A6	MEM A7	B20
A21	MEM A4	MEM A5	B21
A22	MEM A2	MEM A3	B22
	MEM A0	MEM A1	

Connector Specifications

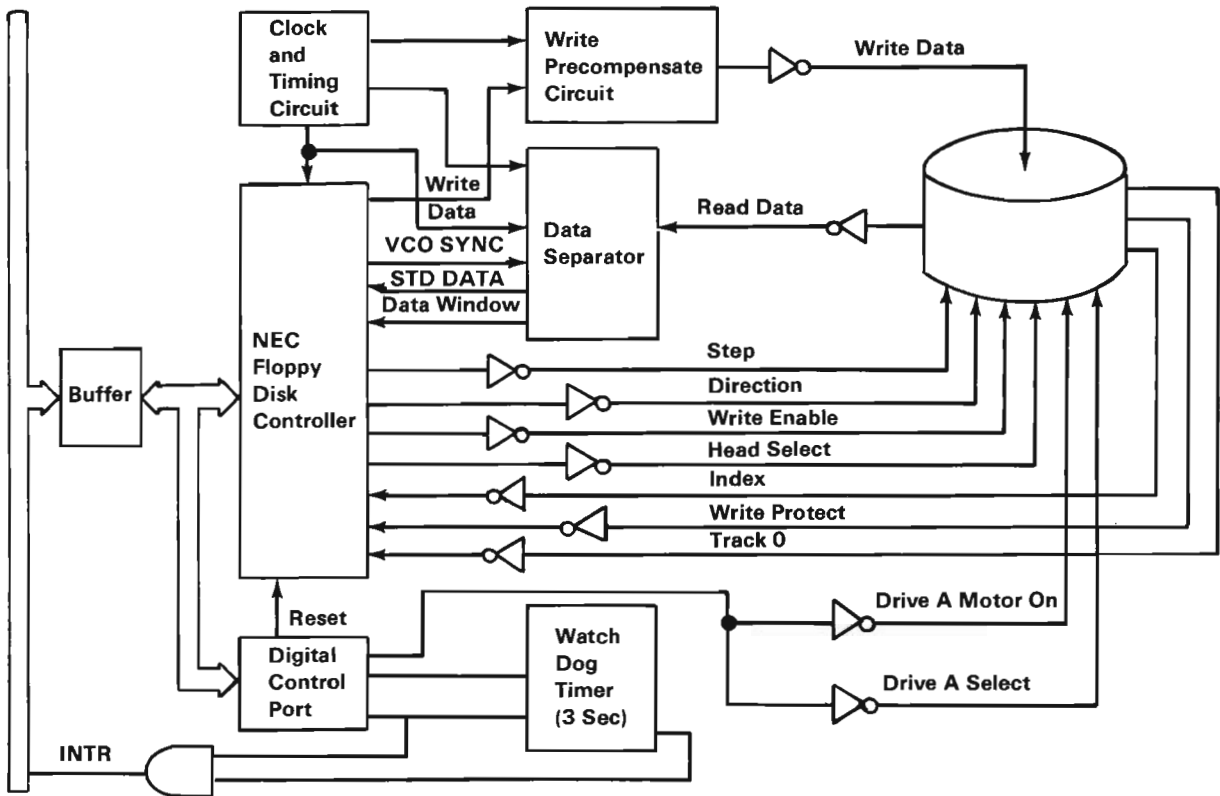
Notes:

IBM PCjr Diskette Drive Adapter

The diskette drive adapter resides in a dedicated connector on the IBM PCjr system board. It is attached to the single diskette drive through a flat, internal, 60-conductor, signal cable.

The general purpose adapter is designed for a double-density, Modified Frequency Modulation (MFM)-coded, diskette drive and uses write precompensation with an analog phase-lock loop for clock and data recovery. The adapter uses the NEC μ PD765 or compatible controller, so the μ PD765 characteristics of the diskette drive can be programmed. In addition, the attachment supports the diskette drive's write-protect feature. The adapter is buffered on the I/O bus and uses the system ROM BIOS for transferring record data. An interrupt level is also used to indicate an error status condition that requires processor attention.

A block diagram of the diskette drive adapter follows.



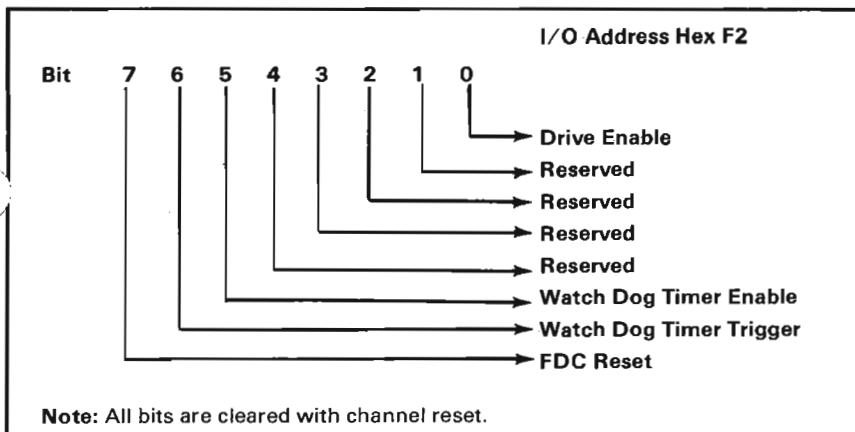
Diskette Drive Adapter Block Diagram

Functional Description

From a programming point of view, the diskette drive adapter consists of a 4-bit digital output register (DOR) in parallel with a NEC μ PD765 or equivalent floppy disk controller (FDC).

Digital Output Register

The digital output register (DOR) is an output-only register used to control the drive motor and selection. All bits are cleared by the I/O interface reset line. The bits have the following functions:



Digital Output Register

Bit 0 This bit controls the motor and enable lines to the drive. When 'high' (1), this bit will turn 'on' the drive motor and 'enable' the drive. When 'low' (0), this bit will turn 'off' the drive motor and 'disable' the drive.

Bits 1-4 These bits are reserved.

- Bit 5** When 'high' (1), this bit 'enables' the WatchDog Timer function and interrupt. When 'low' (0), this bit 'disables' the WatchDog Timer and interrupt.
- Bit 6** This bit controls the start of a watchdog timer cycle. Two output commands are required to operate the trigger. A 1 and then a 0 must be written in succession to 'strobe' the trigger.
- Bit 7** This bit is the hardware 'reset' for the floppy diskette controller chip. When 'low' (0), this bit holds the FDC in its 'reset' state. When 'high' (1), this bit releases the 'reset' state on the FDC.

WatchDog Timer

The WatchDog Timer (WDT) is a one to three-second timer connected to interrupt request line 6 (IRQ6) of the 8259. This timer breaks the program out of data transfer loops in the event of a hardware malfunction. The WatchDog Timer starts its cycle when 'triggered.'

Floppy Disk Controller (FDC)

The floppy disk controller (FDC) contains two registers that can be accessed by the system microprocessor: a status register and a data register. The 8-bit main-status register contains the status information of the FDC and can be accessed at any time. The 8-bit data register consists of several registers in a stack with only one register presented to the data bus at a time. The data register stores data, commands, parameters, and provides floppy disk drive (FDD) status information. Data bytes are read from or written to the data register in order to program or obtain results after

a particular command. The main status register can only be read and is used to facilitate the transfer of data between the system microprocessor and FDC.

FDC Register	I/O Address
Data Register	hex F5
Main Status Register	hex F4

Programming Summary

The FDC is set up with the following Parameters during system power up:

Parameter	Power-up Condition
Sector Size	hex 02 for 512 Byte Sectors
Sector Count	9
Head Unload	hex 0F - Has no effect on system operation.
Head Step Rate	hex D - This gives a step rate of 6 milliseconds.
Head Load Time	hex 1 Minimum head load time.
Format Gap	hex 50
Write Gap	hex 2A
Non-DMA Mode	hex 1
Fill byte for Format	hex F6

FDC Power-up Parameters Settings

The IBM PCjr Diskette Drive Adapter and BIOS use and support the following FDC commands:

- Specify
- Recalibrate
- Seek
- Sense interrupt status
- Sense Drive status
- Read data
- Write data
- Format a track

Note: Please refer to the Diskette section of the BIOS listing for details of how these commands are used.

The following FDC hardware functions are not implemented or supported by the IBM PCjr Diskette Drive Adapter.

- DMA data transfer
- FDC interrupt
- Drive polling and overlapped seek
- FM data incoding
- Unit select status bits

2 Heads (1 per side) 40 Cylinders (Tracks)/Side 9 Sectors/Track 512 Bytes/Sector Modified Frequency Modulation (MFM)
--

Diskette Format

Constant	Value
Head Load	Not Applicable
Head Settle	21 Milliseconds
Motor Start	500 Milliseconds

Drive Constants

Comments

1. Head loads when diskette is clamped.
2. Following access, wait Head Settle time before RD/WR.
3. Drive motor should be 'off' when not in use. Wait Motor Start time before RD/WR.
4. All system interrupts except IRQ6 must be 'disabled' during diskette data transfer in order to prevent data under-run or over-run conditions from occurring.

System I/O Channel Interface

All signals are TTL-compatible:

Most-Positive Up-Level	+ 5.5 Vdc
Least-Positive Up-Level	+ 2.7 Vdc
Most-Positive Down-Level	+ 0.5 Vdc
Least-Positive Down-Level	- 0.5 Vdc

The following lines are used by this adapter:

+D0 thru 7 (Bidirectional, Load: 1 74LS,
Driver: 74LS 3-state)

+A0 thru 3	<p>These eight lines form a bus through which all commands, status, and data are transferred. Bit 0 is the low-order bit.</p> <p>(Adapter Input, Load: 1 74LS)</p>
-IOW	<p>These four lines form an address bus by which a register is selected to receive or supply the byte transferred through lines D0-7. Bit 0 is the low-order bit.</p> <p>(Adapter Input, Load: 1 74LS)</p>
-IOR	<p>The content of lines D0-7 is stored in the register addressed by lines A0-3 at the trailing edge of this signal.</p> <p>(Adapter Input, Load: 1 74LS)</p>
-RESET	<p>The content of the register addressed by lines A0-3 is 'gated' onto lines D0-7 when this line is 'active.'</p> <p>(Adapter Input, Load: 1 74LS)</p>
+IRQ6	<p>A down level 'aborts' any operation in process and 'clears' the digital output register (DOR).</p> <p>(Adapter Output, Driver: 74LS 3-state)</p>
-DISKETTE CARD INSTALLED	<p>This line is made 'active' when the WatchDog timer times out.</p> <p>(Adapter Output, Driver: Gnd.)</p>
	<p>This line is pulled 'up' on the System Board and is wired to input port bit PC2 on port hex 62 of the</p>

-Diskette CS 8255. This line is used by the program to determine if the diskette drive adapter is installed.
(Adapter Input, Load: 1 74LS)

A9 This line is shared with the modem CS line and is 'low' whenever the microprocessor is doing IOR or IOW to either the diskette adapter or the modem. This line should be conditioned with A9 being 'low' to generate a DISKETTE CS.
(Adapter Input, Load: 1 74LS)

DRQ 0 This line is the microprocessor address line 9. When this line is 'low' and -DISKETTE CS is 'low', IOR and IOW are used by the diskette adapter.
(adapter Output, Driver: NEC μ pd 765)

DAK 0 This output would indicate to a DMA device on the external I/O Channel that the diskette controller wants to 'receive' or 'transmit' a byte of data to or from memory.
(Adapter input, Load: NEC μ pd 765)

This line should come from an external DMA and should indicate that a byte is being transferred from/to the Floppy Disk Controller to/from memory.

Drive Interface

All signals are TTL-compatible:

Most Positive Up Level	+ 5.5 Vdc
Least Positive Up Level	+ 2.4 Vdc
Most Positive Down Level	+ 0.4 Vdc
Least Positive Down Level	- 0.5 Vdc

All adapter outputs are driven by active collector gates. The drive should not provide termination networks to Vcc (except Drive Select which has a 2,000 ohm resistor to Vcc).

Each attachment input is terminated with a 2,000 ohm resistor to Vcc.

Adapter Outputs

-Drive Select (Driver: MC3487)

This line is used to 'degate' all drivers to the adapter and receivers from the adapter (except Motor Enable) when the line is not 'active.'

-Motor Enable (Driver: 74LS04)

The drive must control its spindle motor to 'start' when the line becomes 'active' and 'stop' when the line becomes 'inactive.'

-Step (Driver: MC3487)

The selected drive must move the read/write head one cylinder in or

out as instructed by the Direction line for each pulse present on this line.

-Direction

(Driver: MC3487)

For each recognized pulse of the step line the read/write head should move one cylinder toward the spindle if this line is active, and away from the spindle if not-active.

-Write Data

(Driver: 74LS04)

For each 'inactive' to 'active' transition of this line while Write Enable is 'active', the selected drive must cause a flux change to be stored on the diskette.

-Write Enable

(Driver: MC3487)

The drive must 'disable' write current in the head unless this line is 'active.'

**-HEAD
SELECT 1**

(Driver: MC3487)

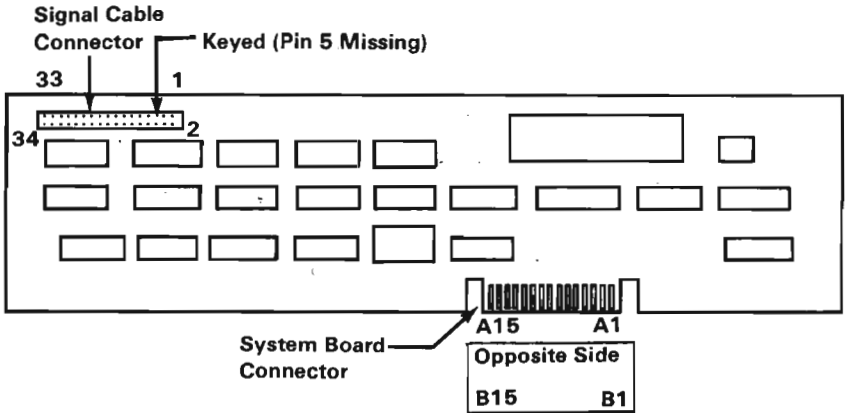
This interface signal defines which side of a two-sided diskette is used for data recording or retrieval. A 'high' level on this line selects the R/W head on the side 1 surface of the diskette. When switching from side 0 to side 1 and conversely, a 100 μ s delay is required before any 'read' or 'write' operation can be initiated.

Adapter Inputs

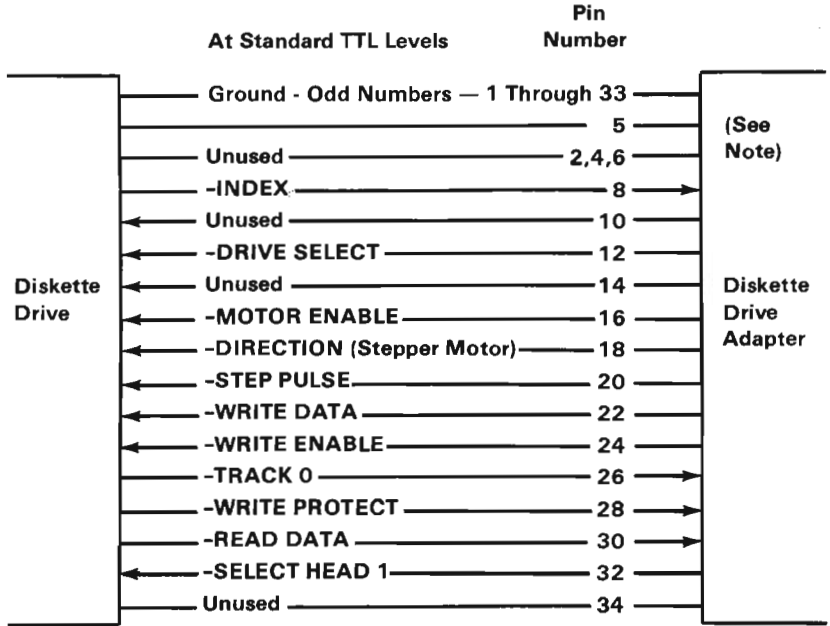
- Index** The selected drive must supply one pulse per diskette revolution on this line.
- Write Protect** The selected drive must make this line 'active' if a write-protected diskette is mounted in the drive.
- Track 0** The selected drive must make this line 'active' if the read/write head is over track 0.
- Read Data** The selected drive must supply a pulse on this line for each flux change encountered on the diskette.

Voltage and Current Requirements

The diskette drive adapter requires a voltage supply of +5 Vdc +/- 5% and draws a nominal current of 525 mA and a maximum current of 700 mA.



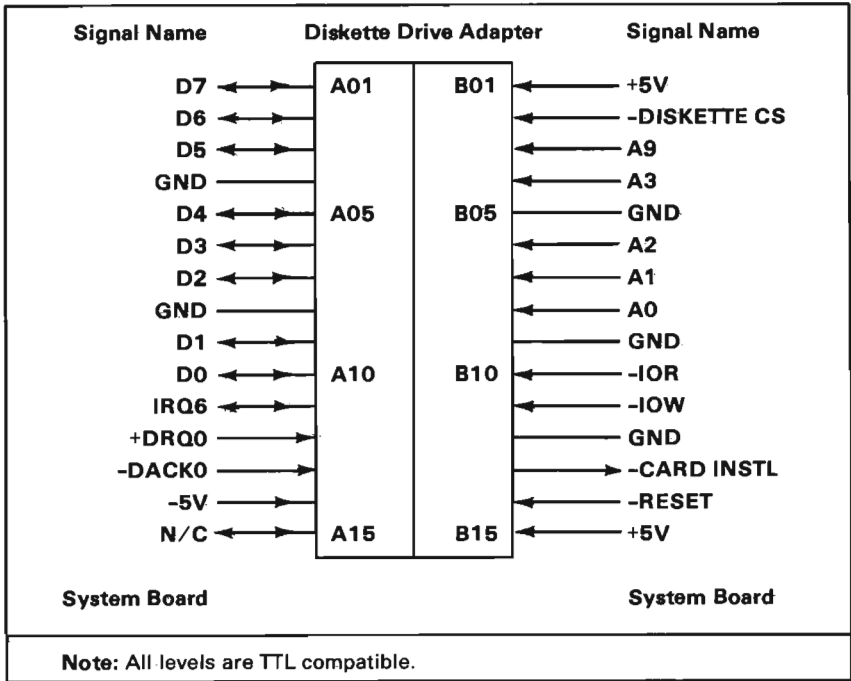
Diskette Drive Adapter



Note: Pin 5 is missing to match the key plug on the signal cable.

Connector Specifications (Part 1 of 2)

System Options



Connector Specifications (Part 2 of 2)

IBM PCjr Diskette Drive

The system unit has space and power for one diskette drive. The drive is double-sided with 40 tracks for each side, is fully self-contained, and consists of a spindle-drive system, a read-positioning system, and a read/write/erase system.

Functional Description

The diskette drive uses modified frequency modulation (MFM) to read and write digital-data, with a track-to-track access time of 6 milliseconds.

To load a diskette, the operator rotates the load lever at the front of the diskette drive clockwise and inserts the diskette into the slot. Plastic guides in the slot ensure the diskette is in the correct position. Closing the load lever centers the diskette and clamps it to the drive hub. This same action also loads the Read/Write heads against the surfaces of the diskette. The load lever is mechanically interlocked to prevent closing of the lever if a diskette is not installed.

The head-positioning system moves the magnetic head to come in contact with the desired track of the diskette. Operator intervention is not required during normal operation. If the diskette is write-protected, a write-protect sensor 'disables' the drive's circuitry, and an appropriate signal is sent to the interface.

Data is read from the diskette by the data-recovery circuitry, which consists of a low-level read-amplifier, differentiator, zero-crossing detector, and digitizing circuits. All data decoding is done by the adapter card.

The IBM PCjr Diskette Drive is equipped with a media cooling fan, which gets its power from the power supply board.

The diskette drive also has the following sensor systems:

- The track 00 sensor, senses when the head/carriage assembly is at track 00.
- The index sensor, which consists of an LED light source and phototransistor. This sensor is positioned so that when an index hole is detected, a digital signal is generated.
- The write-protect sensor 'disables' the diskette drive's electronics whenever it senses a write-protect tab on the diskette.

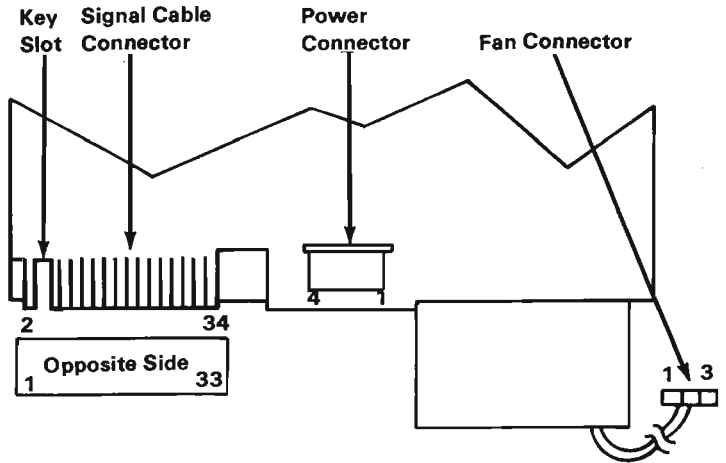
The drive requires power within the following specifications:

Specification	+5 Vdc Input	+12 Vdc Input
Nominal Supply	+5 Vdc	+12 Vdc
Ripple (0 to 50 kHz)	100 mV	100 mV
Tolerance (Including Ripple)	±5%	±5%
Standby Current (Nominal)	600 mA	400 mA
Standby Current (Worst Case)	700 mA	500 mA
Operating Current (Nominal)	600 mA	900 mA
Operating Current (Worst Case)	700 mA	2400 mA

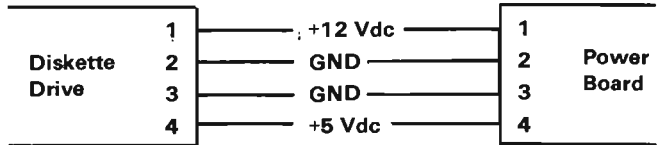
Diskette Drive Power Specifications

For interface information refer to "Diskette Drive Adapter" in this section.

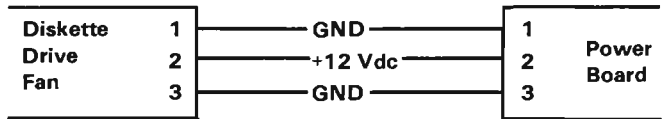
For mechanical and electrical specifications see Appendix D.



Diskette Drive Connectors



Connector Specifications (Part 1 of 2)



Connector Specifications (Part 2 of 2)

Notes:

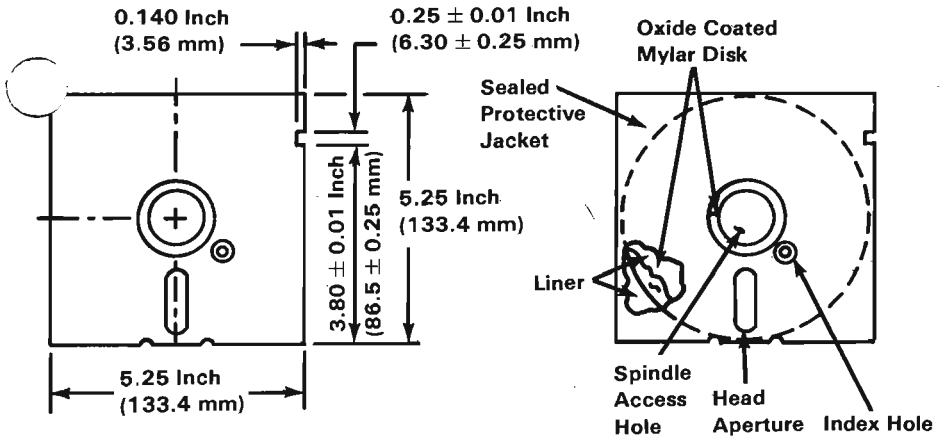
1

2

3

Diskette

The IBM PCjr Diskette Drive uses a standard 133.4 mm (5.25 in.) diskette. For programming considerations, single-sided, double-density, soft-sectored diskettes are used for single-sided drives. Double-sided drives use double-sided, double-density, soft-sectored diskettes. The figure below is a simplified drawing of the diskette used with the diskette drive. This recording medium is a flexible magnetic disk enclosed in a protective jacket. The protected disk, free to rotate within the jacket, is continuously cleaned by the soft fabric lining of the jacket during normal operation. Read/write/erase head access is through an opening in the jacket. Openings for the drive hub and diskette index hole are also provided.



Recording Medium

Notes:

IBM PCjr Internal Modem

The IBM PCjr Internal Modem is a 65 mm (2.5 inch) by 190 mm (7.5 inch) adapter that plugs into the PCjr system board modem connector. The modem connector is an extension of the system I/O bus. All system control signals and voltage requirements are provided through a 2 by 15 position card-edge tab with 0.254 cm (0.100-inch) spacing on the modem adapter.

Functional Description

The Internal Modem consists of two major parts: (1) the INS8250A Asynchronous Communication Element, and (2) the Smart 103 Modem. Therefore, the programming must be considered in two parts. The INS8250A communications protocol is a function of the system ROM BIOS, and is discussed later in this section. All 'pacing' of the interface and control-signal status must be handled by the system software. After the INS8250A is initialized, the modem is controlled by ASCII characters transmitted by the INS8250A.

Key features of the INS8250A used in the modem adapter are:

- Adds or deletes start bits, stop bits, and parity bits to or from the serial data stream
- Full double-buffering eliminates the need for precise synchronization
- Independently-controlled transmit, receive, line status, and data-set interrupts
- Programmable baud-rate-generator allows division of the baud clock by 373 (hex 175) for a 300-bps transmission-speed or 1017 (hex 3F9) for a 110-bps transmission-speed to generate the internal 16 x clock.

- Modem-control functions: Clear to Send (CTS), Data Set Ready (DSR), Data Terminal Ready (DTR), Ring Indicator (RI), and Data Carrier Detect (DCD)
- Fully-programmable serial-interface

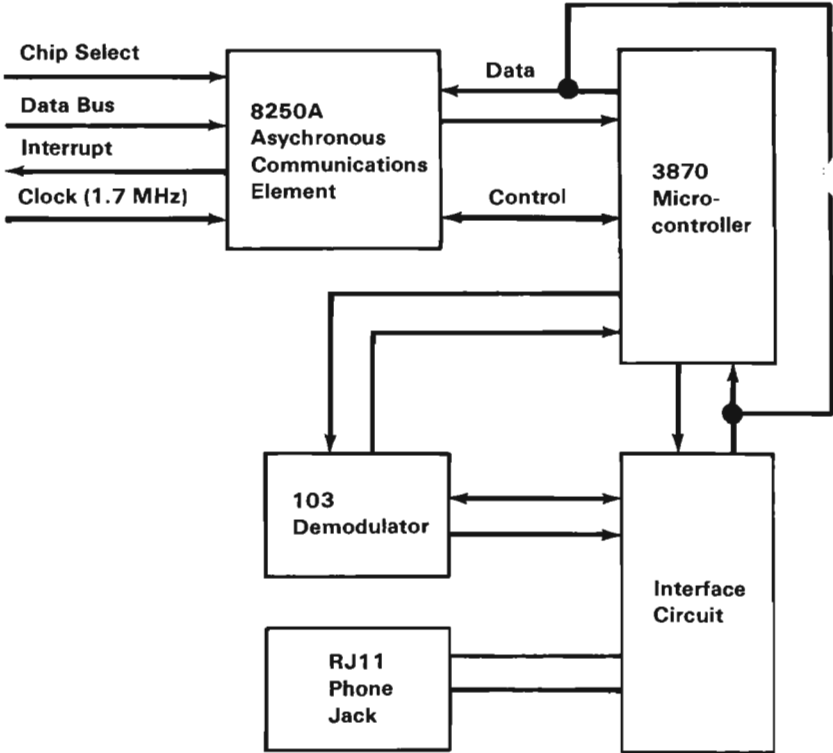
characteristics:

- 7, or 8-bit characters
- Even, odd, or no-parity bit generation and detection
- 1 stop-bit generation
- Baud-rate generation
- False-start bit detection
- Complete status reporting capabilities
- Line-break generation and detection
- Internal-diagnostic capabilities
 - Loopback controls for communications-link fault-isolation
 - Break, parity, overrun, framing-error simulation
- Fully prioritized-interrupt system-controls

Key features of the Smart 103 Modem used on the IBM PCjr Internal Modem are:

- Direct connection to a telephone company line through an FCC Part-68-approved permissive connection
- Compatible to Bell Series 100 originate/answer for modulation and handshaking
- All functions controlled by ASCII characters and INS8250A modem-control lines
- Uses modular phone-jack (USOC RJ11)
- Data rate is either 300 or 110 bits-per-second
- Auto/manual originate
- Auto/manual answer
- Communication mode is full duplex on two-wire, switched-network channels

- Auto dialer; either DTMF ([dual-tone modulated-frequency] touch-tone) or pulse-dialing (rotary dial) by software command
- Tandem dialing
- Call-progress reporting
- Dial-tone, ring-back tone, and busy-tone detection



IBM PCjr Internal Modem Block Diagram

Modem Design Parameters

The following tables describe the design parameters of the Smart 103 Modem.

Dialer Type:	Two modes 1. Forced Touch-Tone (DTMF) dialing 2. Forced pulse dialing
Tandem Dialing:	The ASCII character P (hex 50 or 70) in the dial string causes a delay of up to 10 seconds while the modem is searching for another dial tone. A time out will cause the modem to hang up and post status. The ASCII character W (hex 57 or 77) in the dial string causes a 5-second dead wait before continuing to dial. Multiple ASCII W's will cause multiple waits.
Pulse Dialing:	Rate: 10 + 1, -0 pulses per second Duty Cycle: 60% make, 40% break Interdigit Delay: 800 ms \pm 50 ms
DTMF Dialing:	Tone Duration: 85 ms \pm 10 ms Intertone Duration: 80 ms \pm 10 ms

Dialer Parameters (Part 1 of 2)

Tone Pair Frequencies:		
ASCII Digit Code	Frequency (Hz)	
0	941	1336
1	697	1209
2	697	1336
3	697	1477
4	770	1209
5	770	1336
6	770	1477
7	852	1209
8	852	1336
9	852	1477
*	941	1209
#	941	1477

Dialer Parameters (Part 2 of 2)

Time Out Duration: A data call will time out if an answer tone is not detected within 45 seconds of the last digit dialed.

Failed Call Time Out Parameter

Modulation: Conforms to Bell 103/113 specification using binary phase-coherent frequency shift keying (FSK).

Modulation Parameter

Mode	Originating End	Answering End
Transmit	1070 Space 1270 Mark	2025 Space 2225 Mark
Receive	2025 Hz Space 2225 Hz Mark	1070 Hz Space 1270 Hz Mark

Transmitter/Receiver Frequency Parameters

Receive Sensitivity	More negative or equal to -42 dBm.
---------------------	------------------------------------

Receive Sensitivity Parameters

Transmitter Level	Fixed at -10 dBm as per FCC Part 68 Permissive connection.
-------------------	--

Transmitter Level Parameter

Programming Considerations

The modem and the IBM PCjr system can communicate commands or data between each other. Any commands sent to the modem from the IBM PCjr are stripped from the data stream and executed but are not transmitted to the receiving station. The data is transparent to the modem. The modem is capable of causing hardware interrupts as the result of certain conditions, and in response to queries for its status.

Commands to the modem are a sequence of characters preceded by a single command character. The command character tells the modem that the following character sequence, until a carriage return, is a command. The carriage return completes the command sequence and causes the modem to execute the commands. The command character (represented by [cc] in the following text) is programmable (with the NEW command) to any ASCII character (hex 00 thru 7F). The default for the command character is Ctrl N (ASCII hex 0E).

Commands can occur anywhere in the data stream if properly formatted but are not to be executed by the modem until a carriage return is received.

Multiple commands are allowed if separated by commas and preceded by a single command character.

Command Format

The following is the command format that all commands must follow.

```
[cc][command word][delimiter][arguments] [,more][CR]
```

where:

[cc]	is the single ASCII command character.
[command word]	is the command word or the first letter of the command word.
[delimiter]	is always a space when separating an argument and command word. Any spaces thereafter are ignored until the modem sees a comma, an argument or a carriage return.
[arguments]	is a variable that is replaced by any character allowed by the command definition.
[,more]	is any additional commands preceded by a comma.
[CR]	is a carriage return that completes the command sequence and causes the modem to execute the commands.

The following are two examples of command format.

```
[cc] COUNT 5 [CR]
sample test [cc] VOICE, D (408)
555-1234,QUERY [CR]
```

Format Guidelines

1. Commands can occur anywhere in the data stream if properly formatted but are not be executed by the modem until a carriage return is received.
2. Multiple commands are allowed if separated by commas and preceded by a single command-character.
3. Only the first character of the command word is significant. All remaining characters are ignored up to the first space following the command word. In other words, the **DIAL** command and **DUMMY** are treated identically.

4. The modem does not discriminate between upper-case and lower-case characters.
5. There are three ways to send the current command-character as data to a receiving station:
 - a. Consecutively sending it twice:
 [cc][cc]
 This would send the character a single time.
 - b. Change the command character (with the **NEW** command) to another ASCII character and then transmit the previous command-character.
 - c. Place the modem in the Transparent mode and then transmit the character.

Commands

The commands that are used with the integrated modem are listed on the following pages in alphabetical order.

Each of the commands has its syntax described according to the following conventions:

1. Words in capital letters are keywords. Only the first letter of the keyword is required, the others are optional.
2. You must supply any arguments which are in lower-case letters. Valid characters for arguments are defined as:
 - m - ASCII decimal digits 0 to 9, *, #, I, P, and W
 - n - ASCII hexadecimal digits 0 to F
 - o - ASCII hexadecimal digits 0 to 9
 - p - any ASCII character

3. All arguments are examined for validity. If extra characters are used in an argument, the extra characters are ignored. If the argument is invalid, the command is ignored.
4. An ellipsis (...) indicates an item may be repeated as many times as you wish.
5. All command lines must begin with a command character. The default command-character is (CONTROL N).
6. Multiple commands separated by commas can follow a single command-character.

An example of the **DIAL** command is given below:

Command format - **DIAL m...m**

Command line - **DIAL 1 800 555 1234**

If an invalid argument or no argument is given, the command is not executed. Also, a question mark (?) is given as the error response and the command line is aborted.

The commands are as follows:

Format: ANSWER

A

Purpose: To logically take the phone off the hook and force ANSWER mode. This is logically like a manual answer.

Format: Break n

Purpose: To send a space or break character for a duration equal to a multiple of 100 ms ($n \times 100$ ms).

Format: COUNT n

C n

Where n is the number of complete rings in the range of hex 0 to hex F.

When answering an incoming call, the modem answers the phone after n complete incoming rings, where n is any value from hex 0 to F.

A value of zero specifies that the modem not answer an incoming call, but still carry out any instructions from the host.

When dialing, the modem waits n + 3 complete ringbacks before cancelling the call.

If n exceeds 4, the 45-second abort timer cancels an outgoing call with an "UNSUCCESSFUL" response, as more than seven ringbacks exceeds 45 seconds.

Purpose: Sets the ring count when the modem is answering an incoming call or dialing a call.

Default: 0

Format: DIAL m...m

D m...m

Where m...m is a dial string of ASCII decimal digits 0 through 9, *, #, I, P, and W. A maximum of 33 characters are allowed in the dial string. The first character of the string defaults to P (a 10-second delay while searching for the dial tone). W causes the modem to delay five seconds, then continue dialing.

W or P must start a string, can also occur anywhere within a string, and causes the digits to be tone dialed.

The characters * and # represent the two extra buttons on a push-button phone, but may be used for other things.

I causes the next digits to be pulse dialed. The I stays in effect until a (P,), (W,), or end of command. The modem then searches for line busy, ringing, or incoming carriers while posting the status.

Purpose: To cause the modem to dial.

Default: P (10-second timeout). (If this command is used without an argument, the last number dialed is redialed once.)

Format: **FORMAT n**

F n

Where **n** is one of the following:

n	Parity	Data Length	Stop Bit
0	Mark	7	1
1	Space	7	1
2	Odd	7	1
3	Even	7	1
4	None	8	1
5-7	Reserved		

The 8250A line control register (LCR) must specify the same format as defined in the **FORMAT n** command to 'enable' data/command communication.

Do not combine this command with any other commands except the **SPEED** command on a single command line.

Note: If programming in BASIC, this command must be used in addition to specifying the same parity and data length in the BASIC 'open' statement.

Purpose: To change the parity and number of stop-bits being transmitted at either end, to a new format.

Default: 3

Format: HANGUP

H

Purpose: To perform a clean disconnect and go on-hook.
Logically the same as manually hanging up.

Format: INITIALIZE

I

This command is executed in 10 seconds and is the same as a cold start. An "OK" response is not returned after execution and the integrity test code in the QUERY command is set.

Purpose: Places the modem in the power-up default-state.

Format: LONG RESPONSE o

L o

Where o is one of the following:

o	Mode	Responses
0	Verbose	"BUSY" "CONNECTED" "NO ANSWER" "NO DIAL TONE" "OK" "RING" "UNSUCCESSFUL" "?" (Question Mark)
1	Terse (Hex code)	30 31 32 33 34 35 36 37

Note: The dial string is not echoed in the terse mode.

Purpose: Modifies message feedback. Information is posted in the status area.

Default: 0 (Verbose mode)

Format: **MODEM**

M

Purpose: Forces the modem into the data state where the carrier is placed on the telephone line and proper connection-protocols are followed.

This command is equivalent to **ANSWER** if the data state started as autoanswer.

Format: **NEW p**

N p

where **p** is any ASCII character.(hex 0E)

Purpose: Changes the command character to an ASCII character.

Default: Ctrl N (ASCII hex 0E)

Format: **ORIGINATE**

O

Purpose: Logically takes the phone off-hook and forces the **ORIGINATE** mode. Logically equivalent to manual originate.

Format: PICKUP

P

Purpose: Logically takes the phone off-hook and puts the modem in the voice state.

Format: **QUERY**

Q

Purpose: To query the modem for its status information.

Possible characters returned by the modem are as follows:

Responses	Meaning
H0 or H1	Hook status: H0 = on-hook, H1 = off-hook.
S0 to SF	Current ringcount setting in hex.
B	Line busy.
D	Line dead: no dial-tone found or no ring/no busy timeout after dialing.
L	Successful dial and handshake.
N	Dial not recorded: dial tone present after dialing.
X	No answer: ringcount plus 3 exceeded.
T0	Integrity test passed.
T1	Integrity test failed.

The first group of characters is always returned for a **QUERY** command. The second group of characters is returned only after a dialing sequence has been started or a change has occurred in the dialing status. The third group of characters is returned when a **TEST** command has occurred. All characters except the first group are erased by being read and do not appear in response to the next **QUERY** unless the

condition has recurred in the interim. The **QUERY** response overrides any incoming data from the telephone line.

Format: **RETRY**

R

Purpose: When placed after a **DIAL** command, it causes the modem to execute up to 10 redials at a rate of one per 40 seconds. The redials are triggered by a busy detection after dialing.

Format: **SPEED o**

S o

Where **o** is one of the following:

- o** **bps**
- 0 -** **110**
- 1 -** **300**
- 2 -** **Reserved**

Note: Do not combine this command with other commands except the **FORMAT** command on a single command line.

The **SPEED** command must be issued before the 8250A baud rate is changed.

Note: If programming in **BASIC**, this command must be used in addition to specifying the same bps rate in the **BASIC** 'open' statement.

Purpose: Sets the baud rate.

Default: 1 (300 bps)

Format: **TRANSPARENT n...n**

T n...n

Where **n...n** is the number of bytes to transmit in the range of hex 0 to hex FFFF.

Purpose: Places the modem in the transparent mode for the next **n...n** bytes.

The modem does not look for command sequences but instead transmits every character it receives.

The argument can be up to four ASCII-coded hex digits long. This provides a range of 65,536 bytes.

If an argument is not included with the **TRANSPARENT** command, the command is ignored because it has no default.

The transparent mode is terminated when:

1. **n...n** characters have been transmitted.
2. Loss of carrier timeout.
3. INS8250A OUT 1 pin goes 'active.' (The INS8250A -OUT 1 signal should remain 'active' until the transparent mode is requested again.)

The modem exits the transparent mode before processing the next complete character from the host.

To re-enter the transparent mode, the sequence is:

1. The INS8250A -OUT 1 pin changes to, or remains in the 'inactive' state.
2. The command string containing the **TRANSPARENT** command is issued.

An argument of 0 causes a permanent transparent mode which can be exited by the INS8250A -OUT 1 pin going 'active.'

Format: VOICE

V

Purpose: Forces the modem to the voice state where no tones or carriers are placed or searched for on the telephone line.

This state is used for voice communication, when the modem is an autodialer or answering device only. It is also necessary to be in the voice state to transmit DTMF tone-pairs.

This command 'disables' the autoanswer function.

The status responses are:

1. If a busy signal is detected "BUSY OK".
2. Any other condition "OK...(16 dots)...CONNECTED".

Format: WAIT

W

Purpose: Causes the modem to take no action, including autoanswer, until the next command is received from the host. All commands following the WAIT command in a single command-line are ignored.

Format: XMIT m...m

X m...m

Purpose: Instructs the modem to transmit the DTMF tone-pairs found in the argument string m...m. This is only valid in the voice state. Delays between digits can be caused by inserting W's in the string.

Each W causes a five-second delay.

Format: ZTEST o

Z o

Where o is one of the following:

o Test

0 - Hardware Integrity Test

1 - Analog Loop Back Test

Purpose: Places the modem in the test mode specified by the argument.

For modes other than the integrity test, the modem stays in the test mode until any other command is received.

For the integrity test, the test is performed, status posted, and then the modem returns to service immediately. The integrity test takes eight to 10 seconds to execute and its completion is signaled by an "OK" message.

All commands following the ZTEST command in a single command-line are ignored.

Responses

Autoanswer

If -DTR is 'active', the modem goes off-hook and proper connection protocols including the two-second billing delay are followed. If connection is made, the modem sends "CONNECTED" to the host and posts the status in the status area.

Editing/Changing Command Lines

Corrections to the command line can be performed by aborting current-command lines and typing a new line or by entering the correct command later on in the current-command line.

The last command entered on a single command-line supersedes any previously entered command that performs an opposite function.

A Control X or backspace received by the modem immediately aborts the entire command line.

Opposite Commands

The command line is scanned after its completion (after [CR] is entered). Commands which cause an action during the scan (for example, **DIAL**) are not candidates for opposite treatment. Only commands which 'preset' a static condition can be opposites.

They include:

Count (n)	two entries, latest are used
Format (n)	two entries, latest are used
New (p)	two entries, latest are used
Speed (n)	two entries, latest are used
Transparent n..n	two entries, latest are used
Modem - Voice	these are opposites only when on-hook

Note: Answer and originate are not opposites; each of these causes an action when scanned.

Status Conditions

The modem sends the host messages as defined in the **LONG RESPONSE** command for dialing success or failure. Hardware interrupts for carrier loss and detecting incoming rings are provided on the 8250A.

Dialing and Loss of Carrier

The dialing process begins with the modem searching for a dial tone if it is not in the blind dialing mode. If a dial tone is not detected, the modem hangs up, the appropriate status characters are posted, and the "NO DIAL TONE" message is returned to the host.

If a dial tone is found, the modem continues to dial. When a P is encountered in the dial string, the modem

delays for up to 10 seconds to search for another dial tone and returns the "NO DIAL TONE" message to the host if a dial tone is not detected. When a W is encountered in the dial string, the modem delays for five seconds before continuing to dial. Consecutive W's are allowed in a dial string.

Anytime a P or W is not followed with an I in a dial string, the next digits are tone-dialed. When an I follows a P or W, all following digits are pulse-dialed until a P, W, or end of command ([CR]) is detected.

The modem ignores any character except 0 through 9, *, #, I, P, or W while dialing. This allows the user to place parentheses and dashes in the dial string for greater legibility.

The modem checks the telephone line again after it has dialed the digits in the dial string. If a dial tone is found immediately, the dialed digits are not recorded and the modem posts this to the status characters, hangs up, and sends the "UNSUCCESSFUL" message to the host. If the line is busy, this is also posted to the status characters and the modem hangs up and returns the "BUSY" message to the host. If the line is ringing, the modem begins counting the number of rings. If this count exceeds the value of COUNT + 3, the modem hangs up and takes the same actions as above. If no answer tone is detected within 45 seconds after completion of dialing, the modem hangs up and takes the same actions as above.

Finally, if the call is answered, the modem either looks for a carrier and begins the handshake sequence (if it is in the data or modem state) or remains silent (if it is in the voice state). In the voice state, the modem looks for busy, and transmits a response (1) when the line is

found not busy, or (2) if it is found busy, in which case it also hangs up and possibly dials again. In voice state, ringback count and abort time out are not used.

If, during the process of establishing the data link after dialing, the modem receives any character from the host or - DTR goes 'inactive', the modem aborts the call with a clean disconnect, clears the balance of the command line, and sends an "OK" message. Also, the modem does not carry out the instruction sent from the host, even if the character is a command character.

In the data state, the modem transmits a message after successful completion of the handshake, or after it has determined that the handshake failed. An unsuccessful handshake is evidenced by absence of carrier at the proper time.

If a carrier drops out for more than two seconds in the data state, the modem begins a timeout lasting approximately 17 seconds. At the end of the timeout, the modem hangs up. Any command received during the 17 seconds resets the timer.

The modem does not automatically reestablish the connection if the carrier returns after this dropout interval. This allows the user or software to intercede by commanding the modem to go into the voice state, to hang up immediately, or to take some other action. The data connection may also be terminated by a **HANGUP** command while carriers are still present. A voice connection is always terminated by a **HANGUP** command.

Default State

Upon power up or after an **INITIALIZE** command is given, the modem returns to the default state as follows:

- A verification of hardware integrity is performed and the result posted to the status characters.
- The remaining status characters cleared.
- The modem is placed in the data state awaiting a dialing request or incoming ring.
- The Transparent mode is cleared.
- All loopback modes are cleared.
- The wait mode is cleared.
- The command character is set to Control-N.
- The data format is set to 7 data bits, even parity, and one stop bit.
- Ringcount is set to 0 (auto answer 'disabled')
- The modem is set to on-hook.
- The message mode is set to verbose.

Programming Examples

Call progress reporting is done in two modes, verbose messages or terse messages as defined in **LONG RESPONSE** command to the Serial In (SIN) pin of the 8250A. The power-up default is the verbose messages mode, and these messages from the modem are in capital letters. Also, in call progress reporting, the status area is updated.

The following examples are representative of real-time call-progress reporting. The italicized entries are user entries.

Example 1:

OK [cc]Dial 555-1234 [CR]
NO DIAL TONE
OK

In this example, no dial tone is detected within the time out period.

Example 2:

OK
[cc]Dial 555-1234 [CR]

5551234.....
RINGCONNECTED OK

In this example, a modem answer tone is detected.

Example 3:

OK
[cc]Dial 1(301)555-1234 [CR]
13015551234..... BUSY
OK

In this example, busy is detected.

Example 4:

OK
[cc]Dial 555-1234 [CR]
5551234.....
RING.....
RING.....
RING.....NO ANSWER
OK

In this example, ring count is exceeded before ringing stops.

Example 5:

OK
[cc]Dial 555-1234 [CR]
5551234.....
RING.....
.....
.....UNSUCCESSFUL
OK

In this example, a failed-call time-out occurred because an answer tone was not detected within the allotted time.

Example 6:

OK
[cc]Dial 99P555-1234 [CR]
99.....
.....NO DIAL TONE
OK

In this example, the second dial-tone is not detected within the time out period.

Example 7:

OK
[cc]Dial 99P421-7229 [CR]
99.....BUSY
OK

In this example, busy is detected within the time-out period.

Example 8:

OK
 [cc]Dial 99WW555-1234 [CR]
 99.....

 4217229....
 RING.....CONNECTED OK

In this example, the access code is dialed and two dead waits are performed. Then, the second number is dialed and a modem answers.

Example 9:

OK
 [cc]Dial 555-1234, Retry [CR]
 5551234.....BUSY
 5551234.....BUSY
 5551234.....CONNECTED OK

In this example, the modem dials a number with auto redial. The first two times, the number is busy. The third time, a modem answers.

Modes of Operation

The different modes of operation are selected by programming the 8250A Asynchronous Communication Element. This is done by selecting the I/O address (hex 3F8 to 3FF) and writing data out to the card.

The 8250A is externally programmed to provide asynchronous, ASCII, 10 bit character length including start, stop, and parity on the serial-output pin (SOUT, pin 11). The data rate is 110 or 300 bits-per-second. The commands can be either upper-case or lower-case characters. See the command, **Format [n]**, earlier in this section for additional information.

For further information refer to "Bibliography."

Hex Address	Register Selected	Input/Output	Mode		Notes
			1	2	
3F8	Transmit Buffer	Write	XX	XX	*
3F8	Receive Buffer	Read	XX	XX	*
3F8	Divisor Latch LSB	Write	75	F9	**
3F9	Divisor Latch MSB	Write	01	03	**
3F9	Interrupt Enable	Write	0F	0F	*
3FA	Interrupt Identification	Read	XX	XX	
3FB	Line Control	Write	1A	03	
3FC	Modem Control	Write	01	01	
3FD	Line Status	Read	XX	XX	
3FE	Modem Status	Read	XX	XX	
3FF	Scratch Pad	Write	XX	XX	

*DLAB = 0 (Bit 7 in line control Register).
**DLAB = 1 (Bit 7 in line control Register).
Mode 1 - 300 BPS - 7 Data Bits, 1 Stop Bit, Even Parity.
Mode 2 - 110 BPS - 8 Data Bits, 1 Stop Bit, No Parity.

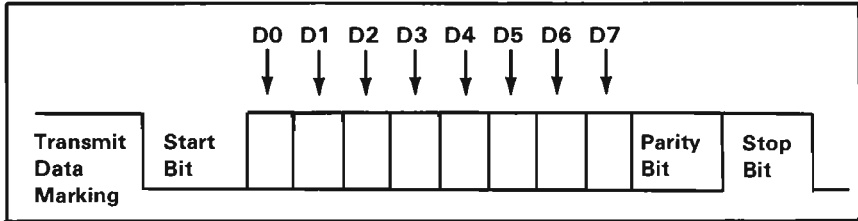
8250A Register Description

Interrupts

One interrupt line is provided to the system. This interrupt is IRQ4 and is 'positive active.' The interrupt enable register must be properly programmed to allow interrupts.

Data Format

The data format is as follows:



Transmitter Output and Receiver Input Data Format

Data bit 0 is the first bit to be transmitted or received. The attachment automatically inserts the start bit, the correct parity-bit if programmed to do so, and the stop bit.

Interfaces

8250A to Modem Interface

The following describes the 8250A to 103 modem interface:

Signal	Description
--------	-------------

INS8250A -OUT 1	The 'inactive' state enables entry into the transparent mode using the UNLISTEN command. The 'active' state 'disables' the transparent mode.
-OUT 2	No connection.
SOUT	Serial output from the 8250A.
-RTS	-Request To Send No connection.
-DTR	-Data Terminal Ready <ol style="list-style-type: none">1. To accept a command, -DTR must be 'active.'2. If -DTR goes 'inactive', the modem does a clean disconnect sequence.3. In auto-answer mode, the modem does not go off-hook, but RI on the 8250A will be toggled if the ringing signal is present.
SIN	Serial input to the 8250A.
-RI	The ring indicator pulses with an incoming ring voltage.
-CTS	-Clear To Send

	This line is wired 'active' on the modem adapter.
-DSR	-Data Set Ready This line is wired 'active' on the modem adapter.
-RLSD	-Received Line Signal Detect When 'low', this line indicates the data carrier has been detected. If the carrier drops out for longer than two seconds, this line goes 'inactive' and starts the timeout timer.
-RESET, +XRESET	These lines are used to reset or initialize the modem logic upon power-up. These lines are synchronized to the falling edge of the clock. Its duration upon power up is 26.5 ms. -RESET is 'active low'. +XRESET is 'active high.'
A0,A1,A2,A9	Address bits 0 to 3 and bit 9. These bits are used with -MODEM CS to select a register on the modem card.
-MODEM CS DISKETTE CS	This line is 'active' for addresses hex 0F0 thru 0FF and 3F8 thru 3FF. It is gated with A9 in the 8250A to exclusively decode hex 3F8 thru 3FF.

D0 thru D7

Data bits 0 thru 7:

These eight lines form a bus through which all data is transferred. Bit 0 is the least significant bit (LSB).

-IOR

The content of the register addresses by line A0 thru A2 is gated onto lines D0 thru D7 when this line is 'active', -MODEM CS is 'active', and A9 is 'high.'

-IOW

The content of lines D0 thru S7 is stored in the register addressed by A0 thru A2 at the leading edge of this signal when -MODEM CS is 'active', and A9 is 'high.'

BAUDCLK

This is a 1.7895 MHz clock signal used to drive the Baud Rate Generator.

+MODEM INTR

This line is connected to the +IQRP4 on the 8259A Interrupt Controller.

-CARD INSTALL

This line indicates to the system BIOS that an IBM PC^{jr} Internal Modem is installed in the feature location.

Telephone Company Interface

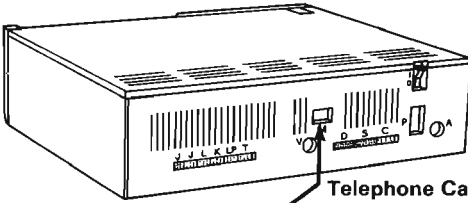
The telephone company interface is a 600 Ohm, balanced, two-wire telephone-interface design that meets the FCC Part 68 rules. A 2.13 meter (7 foot) modular telephone cord is included with the modem adapter.

Line-status detection of dial tone, ringback tone, busy, and incoming ring is provided along with automated routines which react to detected conditions.

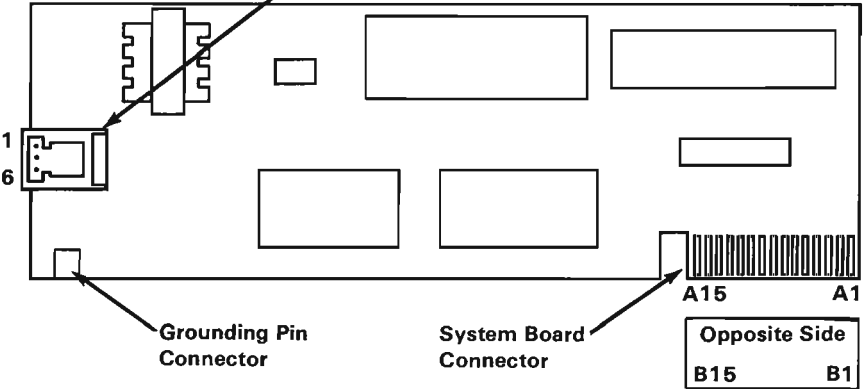
The modem card has one USOC RJ11 jack.

System I/O Channel

The following shows pin assignments for the system board modem connector. Pins A1 to A15 are on the component side.



Telephone Cable Connector

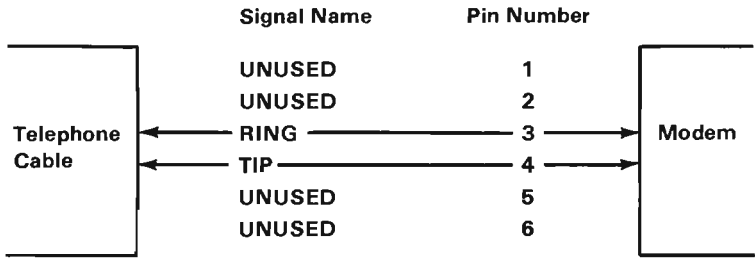


Grounding Pin Connector

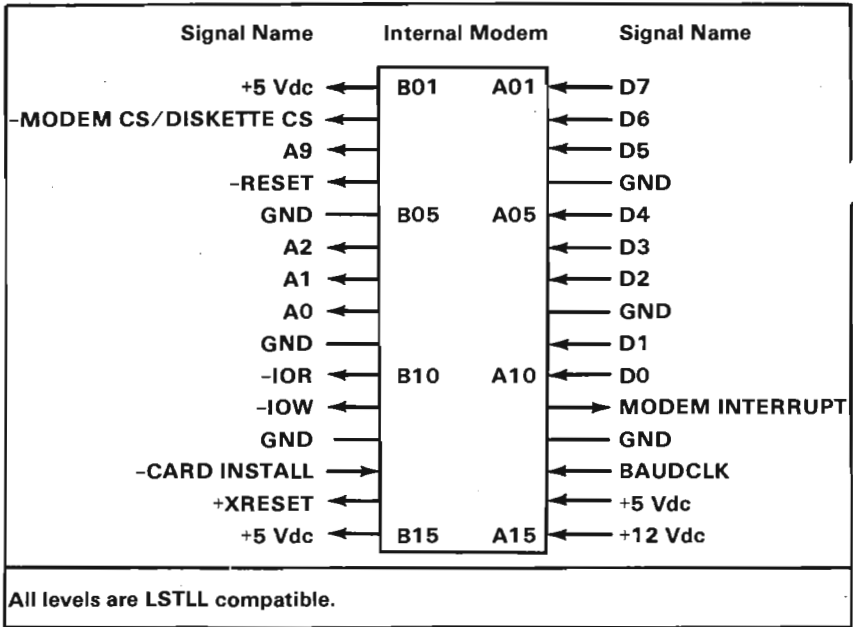
System Board Connector

Opposite Side
B15 B1

Internal Modem Connectors



Connector Specifications (Part 1 of 2)



Connector Specifications (Part 2 of 2)

IBM PCjr Attachable Joystick

The Attachable Joystick is an input device intended to provide the user with two-dimensional positioning-control. Two pushbutton switches on the joystick give the user additional input capability.

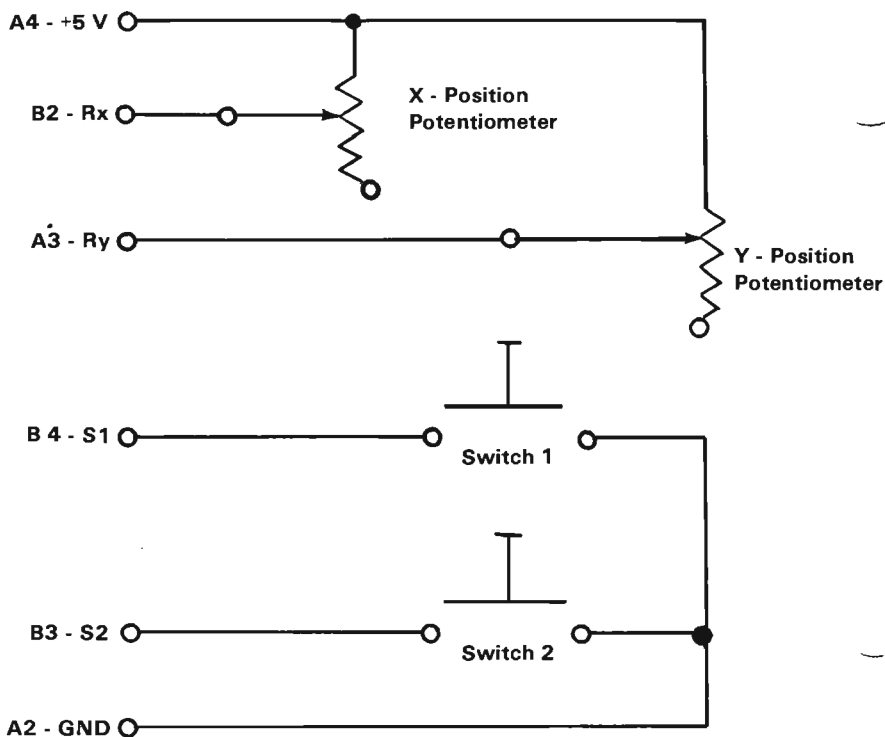
Hardware Description

Two modes of operation of the joystick are available. In the "Spring Return" mode the control stick returns to the center position when released. The "Free Floating" mode allows smooth, force free operation with the control stick remaining in position when released. Selection of these modes can be made for each axis independently. Two controls are provided for individual adjustment to the electrical center of each axis.

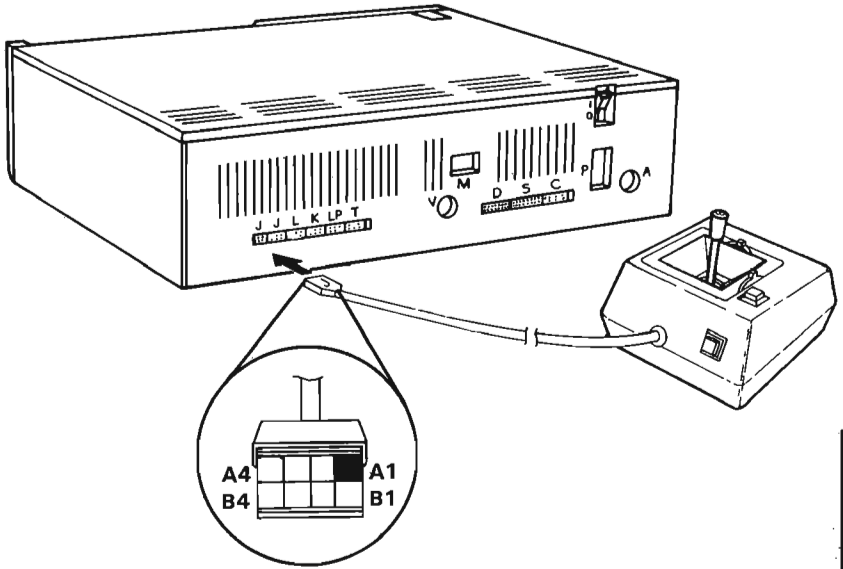
Functional Description

Positional information is derived from two potentiometers Rx and Ry. The resistance of these potentiometers will vary from 0 to 100K ohms nominally as the position of the control stick moves from left to right (X-axis) and from top to bottom (Y-axis). A linear taper is used on the potentiometers so that a linear relationship exists between angular displacement of the stick and the resulting resistance. Electrical centering for each axis is accomplished with the controls by mechanically rotating the body of the potentiometer. Adjustment in this manner has the effect of varying the minimum and maximum resistance relative to the extremes of the angular displacement. The two pushbuttons provided on the joystick are single-pole, single-throw, normally-open pushbuttons.

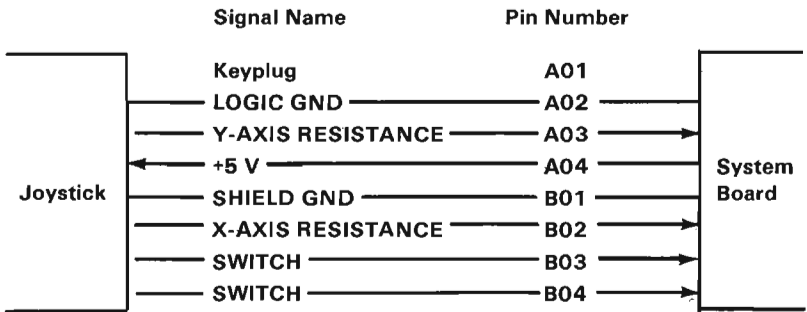
The following are the logic diagram and specifications for the two Attachable Joystick connectors.



Attachable Joystick Logic Diagram



Attachable Joystick Connector



Connector Specifications

Notes:

IBM Color Display

The IBM Color Display is a Red/Green/Blue/Intensity (RGBI)-Direct-Drive display, that is independently housed and powered.

Hardware Description

The IBM Color Display's signal cable is approximately 1.5 meters (5 feet) in length. This signal cable must be attached to the IBM PCjr with the IBM PCjr Adapter Cable for the IBM Color Display which provides a direct-drive connection from the IBM PCjr

A second cable provides ac power to the display from a standard wall outlet. The display has its own power control and indicator. The display will accept either 120-volt 60-Hz power or 220-volt 50-Hz power. The power supply in the display automatically switches to match the applied power.

The display has a 340 mm (13 in.) CRT. The CRT and analog circuits are packaged in an enclosure so the display may be placed separately from the system unit. Front panel controls and indicators include: Power-On control, Power-On indicator, Brightness and Contrast controls. Two additional rear-panel controls are the Vertical Hold and Vertical-Size controls.

Operating Characteristics

Screen

- High contrast (black) screen.
- Displays up to 16 colors.
- Characters defined in an 8-high by 8-wide matrix.

Video Signal

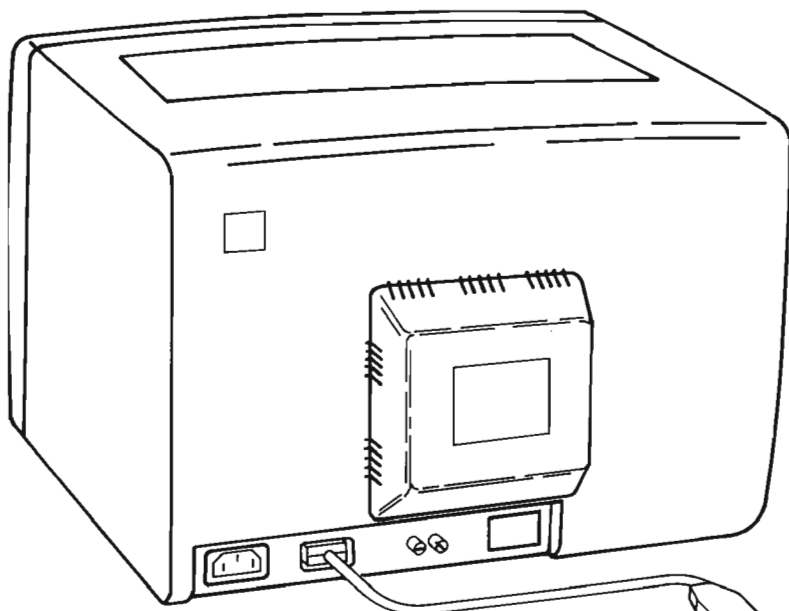
- Maximum video bandwidth of 14 MHz.
- Red, green, and blue video-signals, vertical sync, horizontal sync, and intensity are all independent. All input signals are TTL compatible.

Vertical Drive

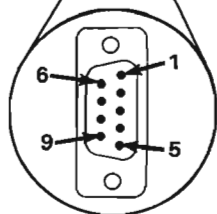
- Screen refreshed at 60 Hz with 200 vertical lines of resolution.

Horizontal Drive

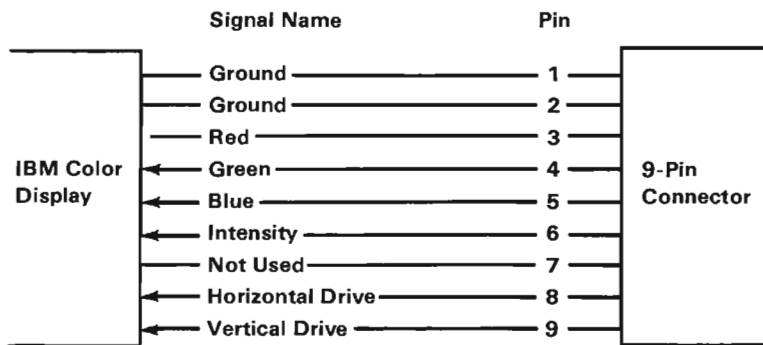
- The horizontal drive frequency is 15.75 kHz.



Color Direct-Drive 9-Pin D-Shell Connector



Color-Display Connector



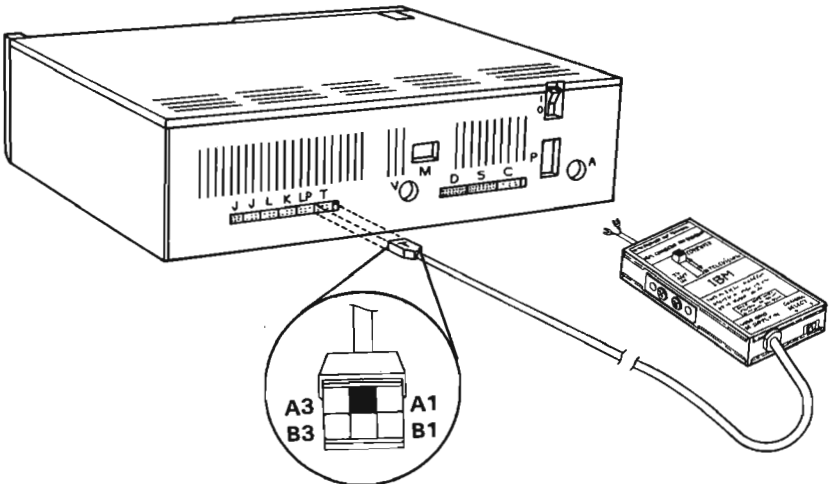
Connector Specifications

Notes:

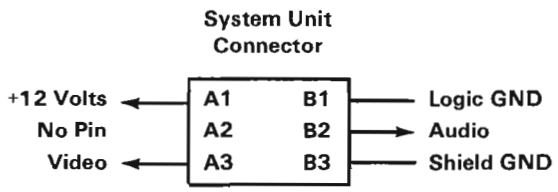
IBM Connector for Television

The Connector for Television is a sealed Radio Frequency (RF) Modulator that imposes the composite video and audio signals onto the RF carrier-wave supplied by the modulator. The connector unit has two two-position switches. One switch selects between the computer's signal or the standard-TV signal from an antenna as the input to the TV. The other switch selects either channel 3's or channel 4's carrier-wave frequency for input to the TV. This allows users to select the weaker TV channel for their area reducing the amount of interference with the computer's input signal. Signal input from the computer is provided by a five-conductor cable with a six-pin IBM PCjr-dedicated connector. Two spade-lug terminals provide for TV-antenna-cable connection. One twin-lead flat-type TV-cable provides input to the TV.

The following is the connector specifications for the IBM Connector for Television.



Connector for TV Connector



Connector Specifications

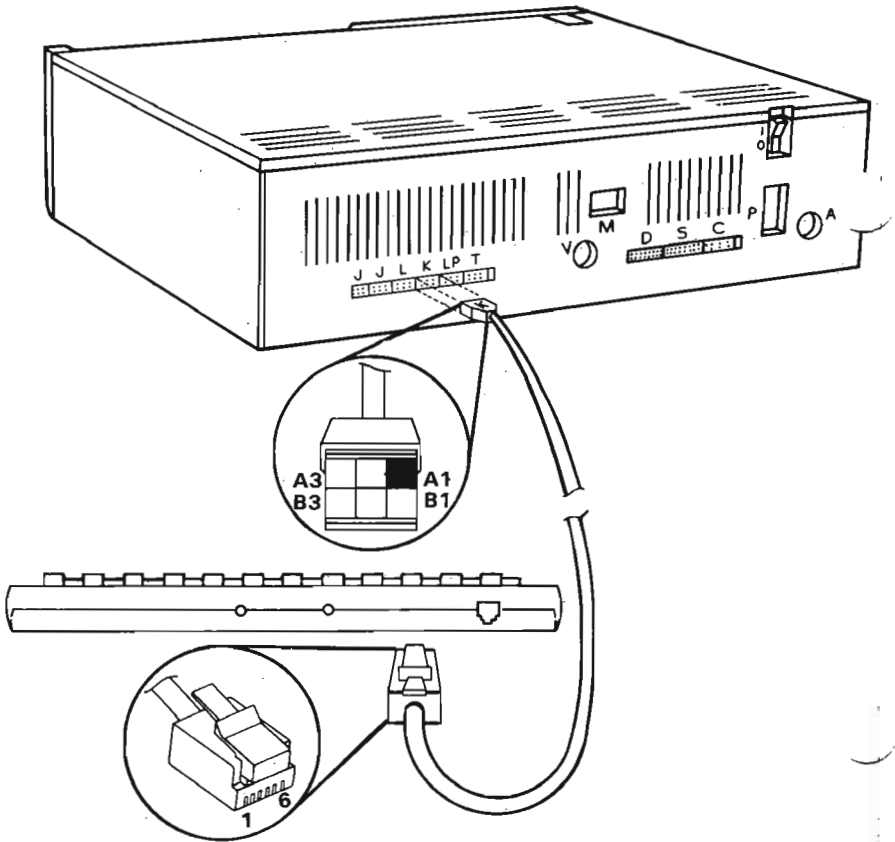
IBM PCjr Keyboard Cord

The IBM PCjr Cordless Keyboard can be attached to the PCjr using the optional Keyboard Cord. The Keyboard Cord is a 1.8 meter (6 foot), two twisted-pair cable, with a six-position RJ11-type connector for the keyboard and a six-position Berg-type connector for the system unit.

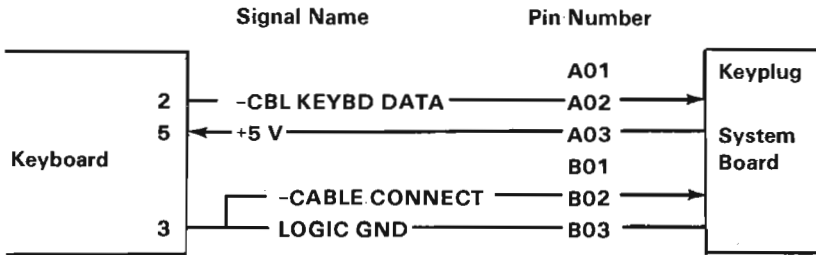
The Keyboard Cord option should be used in an environment that is unfavorable for use of the infra-red link. For instance, brightly lit high-intensity light areas, or multiple IBM PCjr areas where keyboards can conflict with one another.

Insertion of the cord's keyboard connector into the keyboard actuates switches internal to the keyboard. The switches 'deactivate' the IR transmitter by removing the power supplied by the keyboard's batteries. The system unit's infra-red (IR) receiver circuit is 'disabled' by the -CABLE CONNECT signal, supplied when the system-unit end of the cord is connected.

The following figures show the connector specifications for the Keyboard Cord.



Keyboard Cord Connectors

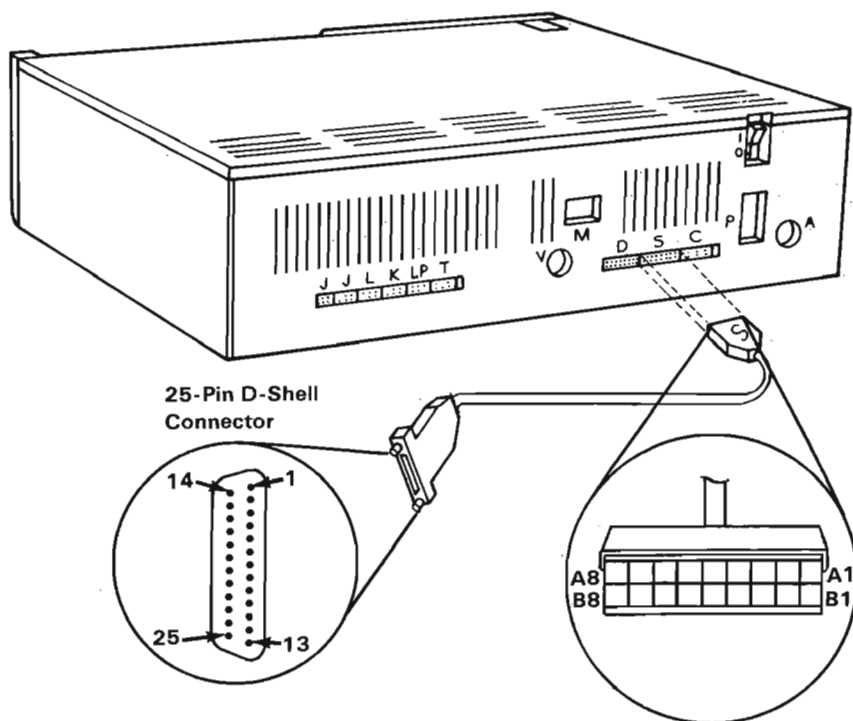


Connector Specifications

IBM PCjr Adapter Cable for Serial Devices

The Adapter Cable for Serial Devices is a 72 mm (3-inch) long, nine-conductor cable terminated with a 16-position Berg-type connector and a 25-pin "D"-shell connector. This cable allows serial devices that terminate with a standard EIA-RS232C 25-pin "D"-shell connector to be connected to the IBM PCjr.

The following figures show the connector specifications for the Adapter Cable for Serial Devices.

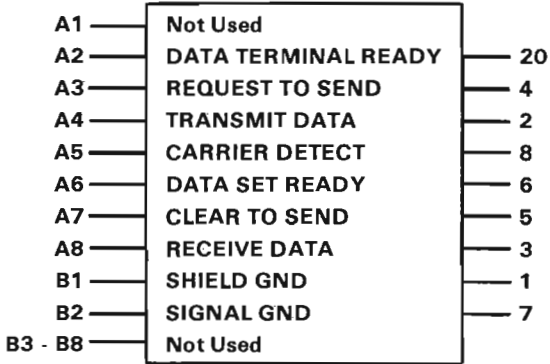


Adapter Cable for Serial Devices

**System
Connector**

Cable

**25-Pin D-Shell
Connector**



Connector Specifications

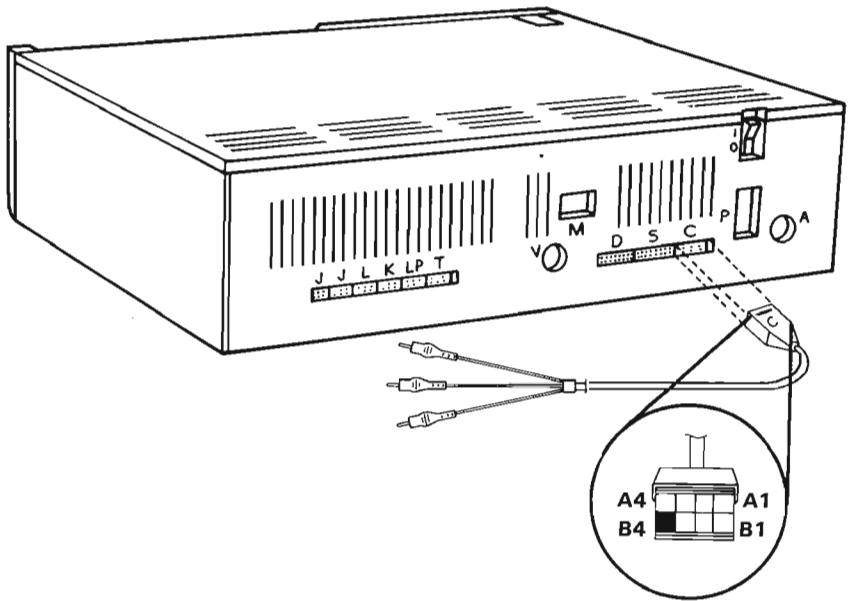
IBM PCjr Adapter Cable for Cassette

This option is an adapter cable that allows connection of a cassette recorder to the IBM PCjr cassette connector.

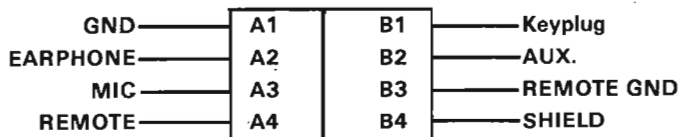
The cassette recorder to be connected must use the following type connectors:

- Belden Style-51 miniature phone-plug (Auxiliary)
- Belden Style-51 miniature phone-plug (Earphone)
- Belden Style-56 subminiature phone-plug (Remote)

The following figures show the connector specifications for the Adapter Cable for Cassette.



Adapter Cable for Cassette Connectors



**Connector Specifications (System End)
(Part 1 of 2)**

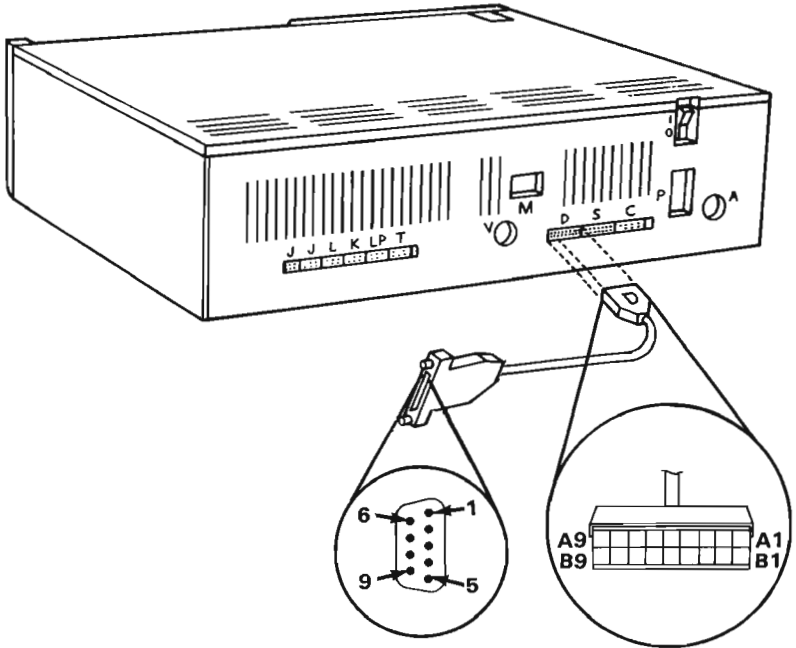
Cassette Connector		System Connector Pin
Aux. (Red)	Signal	B2
	Gnd	A1
Ear (Black)	Signal	A2
	Gnd	A1
Remote (Gray)	Signal	A4
	Gnd	B3

**Connector Specifications (Recorder End)
(Part 2 of 2)**

IBM PCjr Adapter Cable for the IBM Color Display

This adapter cable allows the IBM Color Display to be connected to the IBM PCjr.

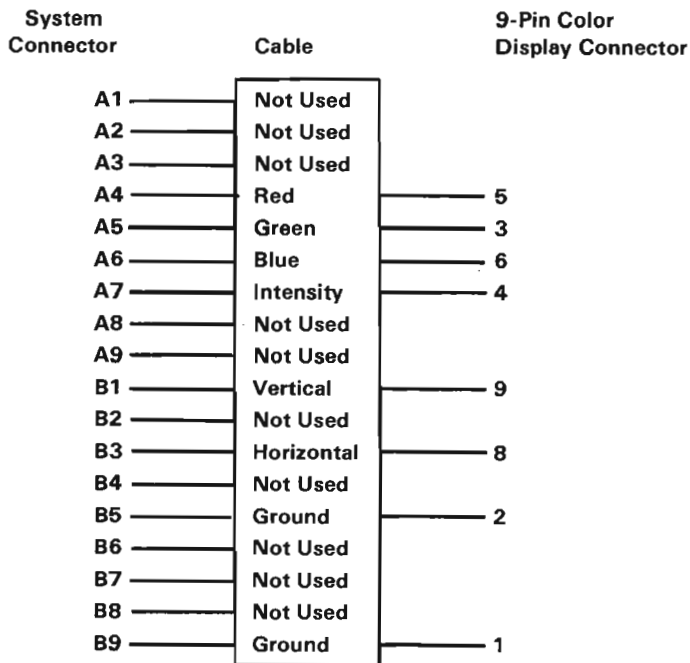
The following figures show the connector specifications for the adapter cable for the IBM Color Display.



System Options

Color Direct-Drive 9-Pin D-Shell Connector

Adapter Cable for IBM Color Display Connectors



Connector Specifications

IBM PCjr Parallel Printer Attachment

The Parallel Printer Attachment is provided to attach various I/O devices that accept eight bits of parallel data at standard TTL-logic levels. The card measures 76mm (3 inches) high by 244mm (9.6 inches) long.

The Parallel Printer Attachment attaches as a feature to the right-hand side of the system unit. It connects to the 60-pin Input/Output (I/O) connector where power and system-input signals are received. A parallel printer attaches to the Parallel Printer Attachment through a 25-pin female "D"-shell connector located on the rear edge of the attachment, where a cable and shield can be attached. The logic design is compatible with the IBM Personal Computer printer adapter.

The attachment card has 12 TTL buffer-output points which are latched and can be 'written' and 'read' under program control using the processor 'IN' or 'Out' instructions. The attachment card also has five steady-state input-points that may be 'read' using the processors' 'IN' instructions.

In addition, one input can also be used to create a processor interrupt. This interrupt can be 'enabled' and 'disabled' under program control. 'Reset' from the power-on circuit is also **ORed** with a program-output point allowing a device to receive a power-on 'reset' when the processor is 'reset.'

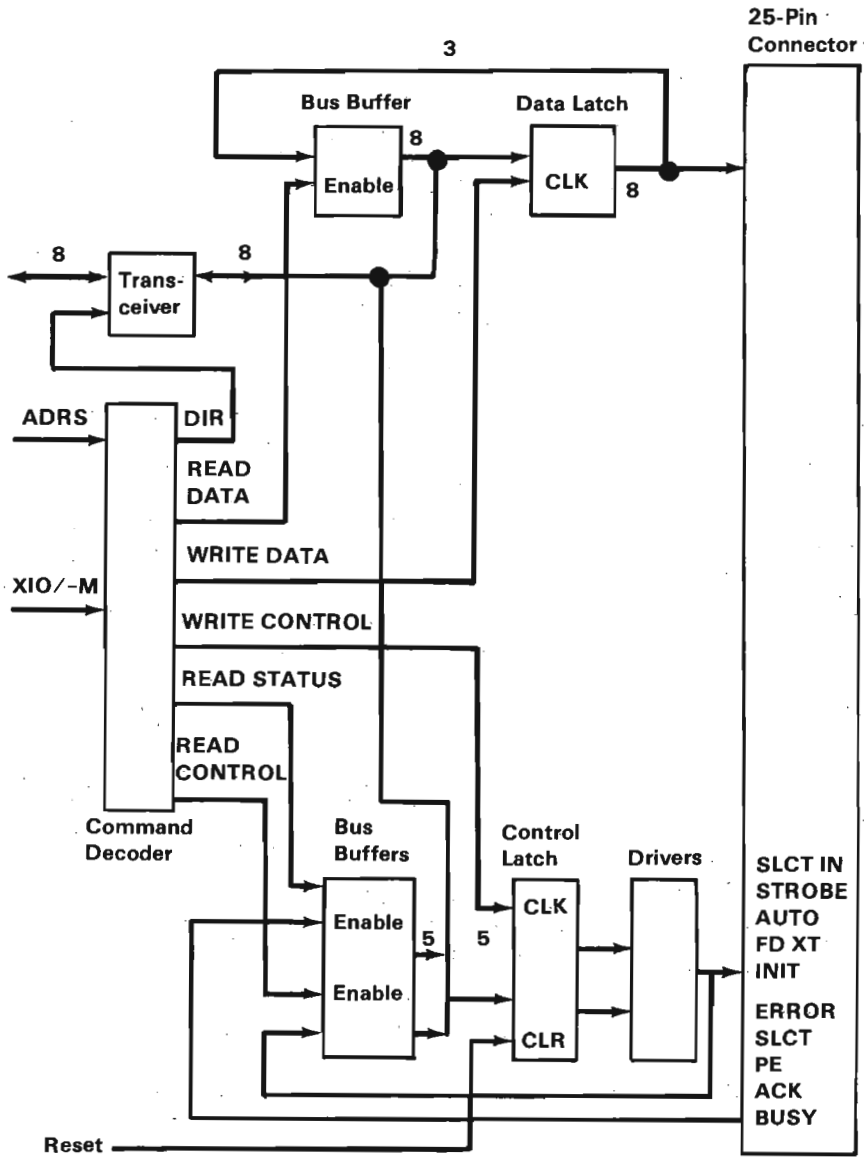
When the Parallel Printer Attachment is used to attach a printer, data or printer commands are loaded into an 8-bit latched output-port, then the strobe line is 'activated' to 'write' data to the printer. The program can then 'read' the input ports for printer

status indicating when the next character can be written. or it may use the interrupt line to indicate **not busy** to the software.

The output ports can also be 'read' at the card's interface for diagnostic-loop functions. This allows fault-isolation determination between the printer attachment and the attached printer.

Description

During a system I/O 'read' or 'write', with the proper address selection, data may be 'written' to or 'read' from the Parallel Printer Attachment. The data and Control Registers must be manipulated by the system software to be consistent with the attaching hardware. The following is a block diagram of the Parallel Printer Attachment card.



System Options

Parallel Printer Interface Block Diagram

System Interface

The Parallel Printer Attachment reserves addresses hex 378, through hex 37F. **IO/-M** must also be 'active high' when addressing the Parallel Printer Attachment.

A card selected signal (**-CARD SLCTD**) is provided to the system I/O when the above addresses are used, and the **IO/-M** bit is 'active high.'

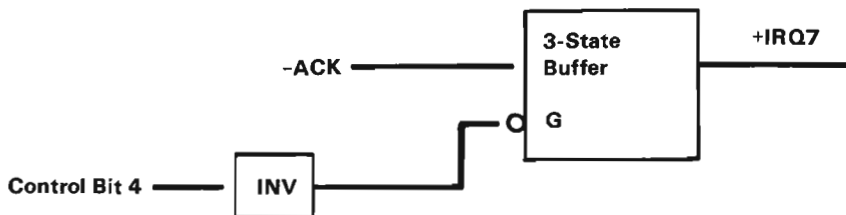
Specific commands are decoded from **A0**, **A1**, **RD**, and **WR** per the following table. Input **A2** is not used.

Addresses (hex)	Operation	Comments
378	'Read'	Read Data Latch
379	'Read'	Read Status
37A	'Read'	Read Control Latch
37B	'Read'	Unused
37B	'Write'	Write Data Latch
379	'Write'	Unused
37A	'Write'	Write Control Latch
37B	'Write'	Unused

All data transfers take place over the 8-bit I/O data-bus with timing provided by the 8088 microprocessor. (**IOR**, **IOW**, **IO/-M**)

An interrupt is provided to the system through the I/O connector of the Parallel Printer Attachment. This

interrupt is 'positive active', Interrupt Level 7 (+IRQ7). Bit 4 of the control latch must be 'written high' to allow interrupts. When the -ACKnowledge signal ('low active' signal goes 'high') the I/O device causes a level 7 interrupt. See the following figure.



+IRQ7/-ACK Logic Diagram

Programming Considerations

The Parallel Printer Attachment can serve as a general purpose peripheral driver. This section describes a configuration which supports attachment to the IBM Graphics Printer.

Command Definition

For the parallel-printer application, the following bit definitions apply.

Data Latch - Address hex 378

A 'write' to this address causes data to be latched onto the printer data bits. A 'read' from this address presents the contents of the data latch to the processor.

MSB	7	6	5	4	3	2	1	0	LSB
	Data	Data	Data	Data	Data	Data	Data	Data	
	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	
	7	6	5	4	3	2	1	0	

Data Latch Format

Printer Status - Address hex 379, hex 7D, Input Only

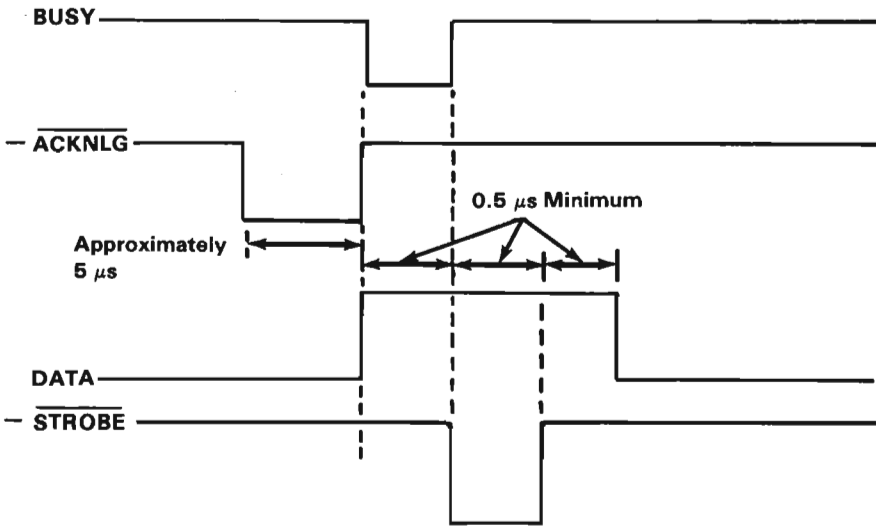
This port provides real-time feedback and status to the system from the printer.

Bit	Signal Name	Description
MSB 7	-BUSY	When this signal is at a low level, the printer is busy and cannot accept data. It can become low during data entry, off-line printing, head translation, or error state.
6	-ACK	When port B is read, this bit will represent the current state of the printer ACK signal. A low level means that a character has been received and the printer is ready to accept another. Normally, this signal will be low for approximately 5 microseconds before BUSY goes away.
5	-PE	A low level indicates that the printer has detected an end of form.
4	+SLCT	A high level indicates that the printer is selected.
3	-ERROR	A low level indicates that the printer has encountered an error condition.
2 Through 0 LSB		Unused.

Printer Status

Printer Control - Address hex 37A

This port contains printer control signals. A 'write' latches control bits to the printer; a 'read' presents the contents of the latches to the processor. See the following timing diagram:



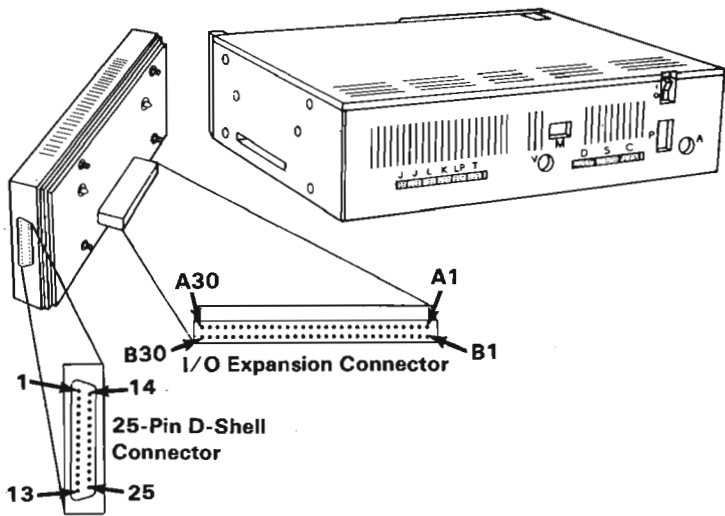
Parallel Interface Timing Diagram

The following figure describes the printer control signals.

Bit	Signal Name	Description
MSB 7 Through 5		Unused.
4	+INTERRUPT ENABLE	A high level in this bit position will allow an interrupt to occur when -ACK goes high.
3	SLCT IN	A low level in this bit position selects the printer.
2	INIT	A low level will initialize the printer (50 microseconds minimum).
1	AUTO FD XT	A low level will cause the printer to line feed anytime a line is printed.
LSB 0	STROBE	A 5 microsecond (minimum) low active pulse clocks data into the printer. Valid data must be present for 5 microseconds (minimum) before and after the STROBE pulse.

Printer Control Signal

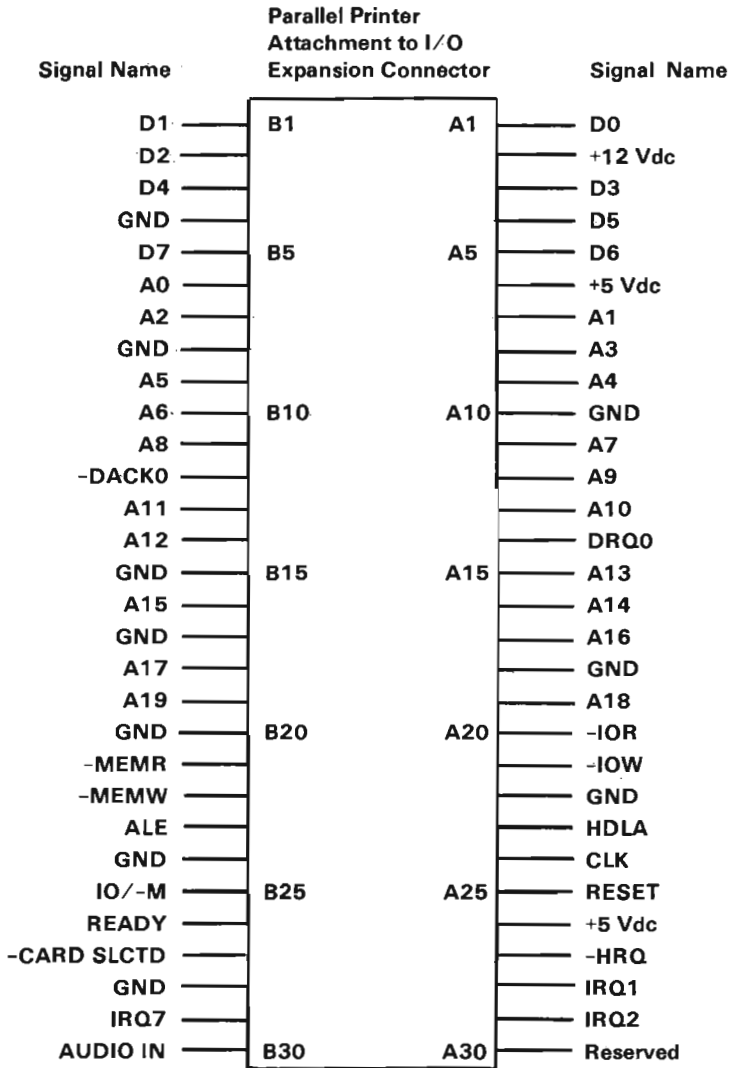
The following are the connector specifications for the IBM PCjr Parallel Printer Attachment.



Parallel Printer Attachment Connectors

25-Pin "D"-Shell Connector				
Pin	Signal	I _{OL} Max	I _{OH} Max	Source
1	-STROBE	14 ma	-.6 ma	Attachment Card
2 Through 9	DATA BIT 0 Through DATA BIT 7	24 ma	-2.6 ma	Attachment Card
10	-ACK	74LS Input	74LS Input	Printer
11	BUSY	74LS Input	74LS Input	Printer
12	PE	74LS Input	74LS Input	Printer
13	SLCT	74LS Input	74LS Input	Printer
14	-AUTO FD XT	14 ma	.6 ma	Attachment Card
15	-ERROR	74LS Input	74LS Input	Printer
16	-INIT PRINTER	14 ma	.6 ma	Printer
17	-SELECT INPUT	14 ma	.6 ma	Attachment Card
18 Through 25	GND	N/A	N/A	

Connector Specifications (Part 1 of 2)



System Options

Connector Specifications (Part 2 of 2)

Notes:

IBM Graphics Printer

The IBM Graphics Printer is a self-powered, stand-alone, tabletop unit which attaches to the system unit through a 6-foot parallel-signal cable, and obtains 120 Vac power from a standard wall outlet through a separate cable. It is an 80 CPS (characters per second), bidirectional, wire-matrix device that can print in a compressed mode of 132 characters per line, in a standard mode of 80 characters per line, in a double width-compressed mode of 66 characters per line, and in a double width mode of 40 characters per line. It can also print double-size and double-strike characters. It prints the standard ASCII, 96-character, uppercase and lowercase character sets and also has a set of 64 special block characters. It has an extended character set for international languages, subscript, superscript, an underline mode, and programmable graphics. The Graphics printer accepts commands that set the line-feed control desired for the application.

It attaches to the system unit through the IBM PCjr Parallel Printer Attachment. The cable is a 25-conductor, shielded cable with a 25-pin "D"-shell connector at the system unit end, and a 36-pin connector at the printer end.

Printer Specifications

Print Method: Serial-impact dot matrix

Print Speed: 80 CPS

Print Direction: Bidirectional with logic seeking

Number of Pins in Head: 9

Line Spacing: 1/16 inch (4.23 mm) or programmable

Matrix Characteristics: 9 by 9

Character Set: Full 96-character ASCII with descenders plus 9 international characters/symbols

Graphic Characters: See "Additional Printer Specifications"

Printing Sizes:

Normal	10 characters-per-inch with a maximum of 80 characters-per-line
Double Width	5 characters-per-inch with a maximum of 40 characters per line
Compressed	16.5 characters-per-inch with a maximum of 132 characters per line
Double Width-Compressed	8.25 characters-per-inch with a maximum of 66 characters per line
Subscript	10 characters-per-inch with a maximum of 80 characters per line
Superscript	10 characters-per-inch with a maximum of 80 characters per line

Media Handling: Adjustable sprocket-pin-feed with 4-inch (101.6 mm) to 10-inch (254 mm) width paper, one original plus two carbon copies (total thickness not to exceed 0.012 inch (0.3 mm)), minimum paper thickness of 0.0025 inch (0.064 mm)

Interface: Parallel 8-bit data and control lines

Inked Ribbon: Black, cartridge type with a life expectancy of 3 million characters

Environmental Conditions: Operating temperature is 5 to 35 degrees centigrade (41 to 95 degrees Fahrenheit), operating humidity is 10 to 80% non-condensing

Power Requirements: 120 Vac, 60 Hz, 1 A maximum with a power consumption of 100 VA maximum

Physical Characteristics:

Height 107 mm (4.2 inches)
Width 374 mm (14.7 inches)
Depth 305 mm (12 inches)
Weight 5.5 kg (12 pounds)

Additional Printer Specifications

Printing Characteristics

Extra Character Set

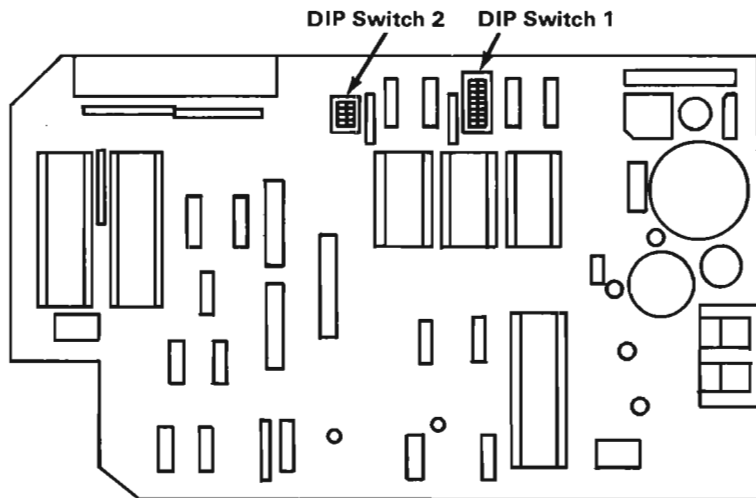
Set 1 Additional ASCII numbers 160 to 175 contain European characters. Numbers 176 to 223 contain graphic characters. Numbers 224 to 239 contain selected Greek-characters. Numbers 240 to 255 contain math and extra symbols.

Set 2 The differences in Set 2 are ASCII numbers 3,4,5,6, and 21. ASCII numbers 128 to 175 contain European characters.

Graphics There are 20 block characters and programmable graphics.

DIP Switch Settings

There are two Dual-Inline-Package (DIP) switches on the control circuit-board. In order to satisfy the user's specific requirements, desired control modes are selected by the DIP switches. The functions of these switches and their preset conditions at the time of shipment are shown in the following figures.



Location of DIP Switches

Switch Number	Function	On	Off	Factory Position
1-1	Not Applicable	—	—	On
1-2	CR	Print Only	Print and Line Feed	On
1-3	Buffer Full	Print Only	Print and Line Feed	Off
1-4	Cancel Code	Invalid	Valid	Off
1-5	Not Applicable	—	—	On
1-6	Error Buzzer	Sound	No Sound	On
1-7	Character Generator	Set 2	Set 1	Off
1-8	SLCT IN Signal	Fixed Internally	Not Fixed Internally	On

Functions and Conditions of DIP Switch 1

Switch Number	Function	On	Off	Factory Position
2-1	Form Length	12 Inches	11 Inches	Off
2-2	Line Spacing	1/8 Inch	1/6 Inch	Off
2-3	Auto Feed XT Signal	Fixed Internally	Not Fixed Internally	Off
2-4	1 Inch Skip Over Perforation	Valid	Invalid	Off

Functions and Conditions of DIP Switch 2

Parallel Interface Description

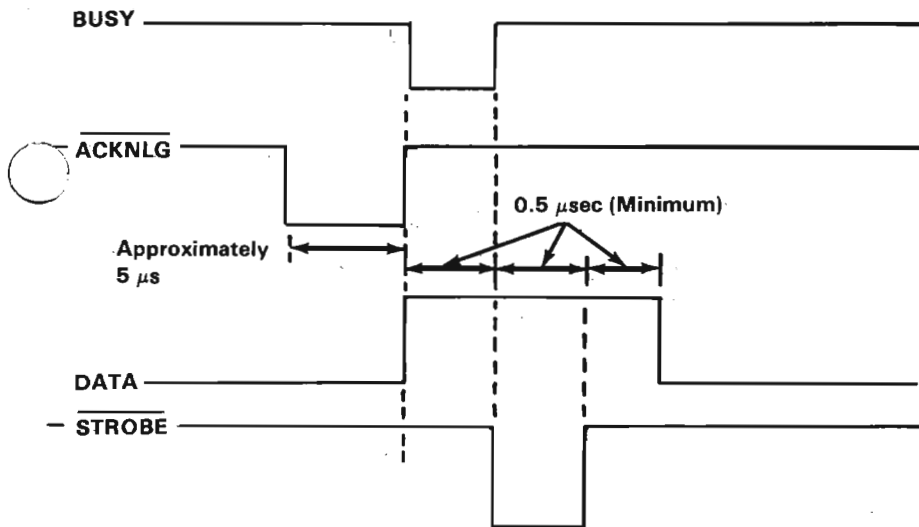
Specifications

Data Transfer Rate	1000 cycles-per-second (cps)-(maximum)
Synchronization	By externally-supplied STROBE pulses
Signal Exchange Logic level	-ACKNLG or BUSY signals Input data and all interface-control signals are compatible with the Transistor-Transistor Logic (TTL) level.
Connector	Plug 57-30360 (Amphenol)

Connector-pin assignments and descriptions of respective interface-signals are provided in the following figures.

Data Transfer Sequence

The following figure shows the Parallel Interface Timing.



Parallel Interface Timing Diagram

Interface Signals

- Strobe** STROBE pulse to read data in. Pulse width must be more than $0.5 \mu\text{s}$ at the receiving terminal. The signal is normally 'high'; however read-in of data is performed at the 'Low' level of this signal.
- Data 1-8** These signals are the first to eight bits of parallel data. Each signal is at a 'high' level when data is a logical 1 and 'low' when data is a logical 0.
- ACKNLG** Approximately $0.5 \mu\text{s}$ pulse (low) indicates that data has been received and the printer is ready to accept data.
- BUSY** A 'high' signal indicates that the printer cannot receive data. The signal is 'high' in the following cases:
 - During data entry

	<ul style="list-style-type: none"> • During printing operation • In the “off-line” state • During printer-error status
PE	A 'high' signal indicates that the printer is out of paper.
SLCT	This signal indicates that the printer is in the selected state.
Auto Feed XT	When this signal is 'low' paper is fed one line after printing. This signal level can be fixed 'low' by DIP switch pin 2-3.
INT	When this signal is 'low' the printer controller is reset to its initial state and the print buffer is cleared. This signal is normally 'high' and its pulse width must be more than 50 μ s at the receiving terminal.
Error	This signal is 'low' when the printer is in the “Paper End,” “Off Line,” and “Error” state.
-SLCTIN	Data entry to the printer is possible only when this signal is 'low'. This signal can be fixed 'low' by DIP switch 1-8.

Notes:

1. All interface conditions are based on TTL level. Both the rise and fall times of each signal must be less than 0.2 μ s.
2. Data transfer must not be carried out by ignoring the -ACKNLG or BUSY signal. Data transfer can only occur after confirming the -ACKNLG signal or when the BUSY signal is 'low'.

The following figure shows the pin assignment and direction of each signal.

Signal	Signal Pin #	Return Pin #	Direction
-STROBE	1	19	In
DATA 1	2	20	In
DATA 2	3	21	In
DATA 3	4	22	In
DATA 4	5	23	In
DATA 5	6	24	In
DATA 6	7	25	In
DATA 7	8	26	In
DATA 8	9	27	In
-ACKNLG	10	28	Out
BUSY	11	29	Out
PE	12	30	Out
SLCT	13	—	Out
AUTO FEED XT	14	—	In
NC	15	—	—
OV	16	—	—
CHASSIS GND	17	—	—
NC	18	—	—
GND	19-30	—	—
INT	31	—	In
ERROR	32	—	Out
GND	33	—	—
NC	34	—	—
	35	—	—
-SLCT IN	36	—	In

Pin Assignments

Printer Modes

The IBM Graphics Printer can use any of the combinations listed in the following table and the print mode can be changed at any place within the line.

Modes can be selected and combined if they are in the same vertical column.

Printer Modes									
Normal	X	X	X						
Compressed				X	X	X			
Emphasized							X	X	X
Double Strike	X			X			X		
Subscript		X			X			X	
Superscript			X			X			X
Double Width	X	X	X	X	X	X	X	X	X
Underline	X	X	X	X	X	X	X	X	X

Printer Modes

Printer Control Codes

On the following pages are complete codes for printer characters, controls, and graphics. You may want to keep them handy for future reference. The printer codes are listed in ASCII-decimal numeric-order (from NUL which is 0 to DEL, which is 127). The examples given in the Printer-Function descriptions are written in the BASIC language. The "input" description is given when more information is needed for programming considerations.

ASCII decimal values for the printer control codes can be found under "Printer Character Sets."

The Descriptions that follow assume that the printer DIP switches have not been changed from their factory settings.

Printer code
NUL

Printer Function
Null:

Used with ESC B and ESC D as a list terminator. NUL is also used with other printer.

control codes to select options (for example, ESC S).

Example:

```
LPRINT CHR$(0);
```

BEL

Bell:

Sounds the printer buzzer for 1 second.

Example:

```
LPRINT CHR$(7);
```

HT

Horizontal Tab:

Tabs to the next horizontal tab stop. Tab stops are set with ESC D. Tab stops are set every 8 columns when the printer is powered on.

Example:

```
LPRINT CHR$(9);
```

LF

Line Feed:

Spaces the paper up one line. Line spacing is 1/16-inch unless reset by ESC A, ESC 0, ESC 1, ESC 2, or ESC 3.

Example:

```
LPRINT CHR$(10);
```

Form Feed:

Advances the paper to the top of the next page.

Note: The location of the paper, when the printer is powered on, determines the top of the page. The next top of page is 11 inches from that position. ESC C can be used to change the page length.

Example:

```
LPRINT CHR$(12);
```

CR

Carriage Return:

Ends the line that the printer is on and prints the data remaining in the printer buffer. (No Line Feed operation takes place.)

Note: IBM Personal Computer BASIC adds a Line Feed unless 128 is added [for example CHR\$(141)].

Example:

LPRINT CHR\$(13);

SO Shift Out (Double Width):

Changes the printer to the Double-Width print-mode.

Note: A Carriage Return, Line Feed or DC4 cancels Double-Width print-mode.

Example:

LPRINT CHR\$(14);

SI Shift In (Compressed):

Changes the printer to the Compressed-Character print-mode. Example:

LPRINT CHR\$(15);

DC2 Device Control 1 (Compressed Off):

Stops printing in the Compressed print-mode.

Example:

LPRINT CHR\$(18);

DC4 Device Control 4 (Double Width Off):

Stops printing in the Double-Width print-mode.

Example:

LPRINT CHR\$(20);

CAN Cancel:

Clears the printer buffer. Control codes, except SO, remain in effect.

Example:

LPRINT CHR\$(24);

ESC Escape:

Lets the printer know that the next data sent is a printer command.

Example:

LPRINT CHR\$(27);

ESC - Escape Minus (Underline)

Format: ESC -;n;

ESC - followed by a 1, prints all of the following data with an underline.

ESC - followed by a 0 (zero), cancels the Underline print-mode.

Example:

LPRINT CHR\$(27);CHR\$(45);CHR\$(1);

Escape Zero (1/8-Inch Line Feeding)

Changes paper feeding to 1/8-inch.

Example:

LPRINT CHR\$(27);CHR\$(48);

Escape One (7/72-Inch Line Feeding)

Changes paper feeding to 7/72-inch.

Example:

LPRINT CHR\$(27);CHR\$(49);

Escape Two (Starts Variable Line-Feeding)

ESC 2 is an execution command for ESC A. If no ESC A command has been given, line feeding returns to 1/6-inch.

Example:

LPRINT CHR\$(27);CHR\$(50);

Escape Three (Variable Line-Feeding)

Format: ESC 3;n;

Changes the paper feeding to n/216-inch. The example that follows sets the paper feeding to 54/216 (1/4)-inch. The value of n must be between 1 and 255.

Example:

LPRINT CHR\$(27);CHR\$(51);CHR\$(54);

Escape Six (Select Character Set 2)

Selects Character Set 2. (See "Printer Character set 2")

Example:

LPRINT CHR\$(27);CHR\$(54);

Escape Seven (Select Character Set 1)

ESC 0

ESC 1

ESC 2

ESC 3

ESC 6

ESC 7

Selects character set 1. (See "Printer Character Set 1")

Character set 1 is selected when the printer is powered on or reset.

Example:

ESC 8 LPRINT CHR\$(27);CHR\$(55);

Escape Eight (Ignore Paper End)

Allows the printer to print to the end of the paper. The printer ignores the Paper End switch.

Example:

ESC 9 LPRINT CHR\$(27);CHR\$(56);

Escape Nine (Cancel Ignore Paper End)

Cancels the Ignore Paper End command. ESC 9 is selected when the printer is powered on or reset.

Example:

ESC < LPRINT CHR\$(27);CHR\$(57);

Escape Less Than (Home Head)

The printer head returns to the left margin to print the line following ESC <. This occurs for one line only.

Example:

ESC A LPRINT CHR\$(27);CHR\$(60);

Escape A (Sets Variable Line Feeding)

Format: ESC A;n;

Escape A sets the line-feed to n/72-inch.

The example that follows tells the printer to set line feeding to 24/72-inch. ESC 2 must be sent to the printer before the line

feeding changes. For example, ESC A;24 (text) ESC 2 (text). The text following ESC A;24 spaces at the previously set

line-feed increments. The text following ESC 2 prints with new line-feed

increments of 24/72-inch. Any increment between 1/72 and 85/72-inch may be used.

Example:

LPRINT

**CHR\$(27);CHR\$(65);CHR\$(24);
CHR\$(27);CHR\$(50);**

ESC C

Escape C (Set Lines-per-Page)

Format: ESC C;n;

Sets the page length. The ESC C command must have a value following it to specify the length of page desired. (Maximum form length for the printer is 127 lines.) The example below sets the page length to 55 lines. The printer defaults to 66 lines-per-page when powered on or reset.

Example:

LPRINT CHR\$(27);CHR\$(67);CHR\$(55);

Escape C (Set Inches-per-Page)

Format: ESC C;n;m;

Escape C sets the length of the page in inches. This command requires a value of 0 (zero) for n, and a value between 1 and 22 for m.

Example:

LPRINT CHR\$(27);CHR\$(67);CHR\$(0);CHR\$(12);

ESC D

Escape D (Sets Horizontal Tab Stops)

Format: ESC D;n1;n2;...nk;NUL;

Sets the horizontal-tab stop-positions. The example that follows shows the horizontal-tab stop-positions set at printer column positions of 10, 20, and 40. They are followed by CHR\$(0), the NUL code. They must also be in ascending numeric order as shown. Tab stops can be set between 1 and 80. When in the Compressed-print mode, tab stops can be set up to 132.

The Graphics Printer can have a maximum of 28 tab stops. The HT (CHR\$(9)) is used to execute a tab operation.

Example:

LPRINT
CHR\$(27);CHR\$(68);CHR\$(10)
;CHR\$(20);CHR\$(40);
CHR\$(0);

ESC E

Escape E (Emphasized)
Changes the printer to the Emphasized-print mode. The speed of the printer is reduced to half speed during the Emphasized-print mode.
Example:

ESC F

LPRINT CHR\$(27);CHR\$(69);
Escape F (Emphasized Off)
Stops printing in the Emphasized-print mode.
Example:

ESC G

LPRINT CHR\$(27);CHR\$(70);
Escape G (Double Strike)
Changes the printer to the Double-Strike print-mode. The paper is spaced 1/216 of an inch before the second pass of the print head.
Example:

ESC H

LPRINT CHR\$(27);CHR\$(71);
Escape H (Double Strike Off)
Stops printing in the Double-Strike mode.
Example:

ESC J

LPRINT CHR\$(27);CHR\$(72);
Escape J (Sets Variable Line Feeding)
Format: ESC J;n;
When ESC J is sent to the printer, the paper feeds in increments of n/216 of an inch. The value of n must be between 1 and 255. The example that follows gives a line feed of 50/216-inch. ESC J is canceled after the line feed takes place.
Example:

ESC K

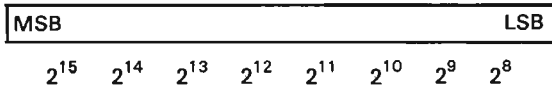
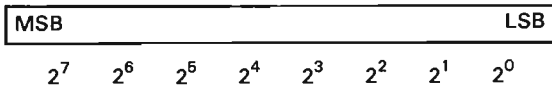
LPRINT CHR\$(27);CHR\$(74);CHR\$(50);
Escape K (480 Bit-Image Graphics Mode)
Format ESC K;n1;n2;v1;v2;...vk;
Changes from the Text mode to the Bit-Image

Graphics mode. n1 and n2 are one byte, which specify the number of bit-image data bytes to be transferred. v1 through vk are the bytes of the bit-image data. The number of bit-image data bytes (k) is equal to $n1 + 256n2$ and cannot exceed 480 bytes. At every horizontal position, each byte can print up to 8 vertical dots. Bit-image data may be mixed with text data on the same line.

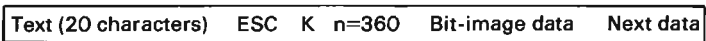
Note: Assign values to n1 and n2 as follows:
 n1 represents values from 0 - 255.
 n2 represents values from 0 - 1 x 256.

MSB is most-significant bit and LSB is least-significant bit.

The following figures show the format.

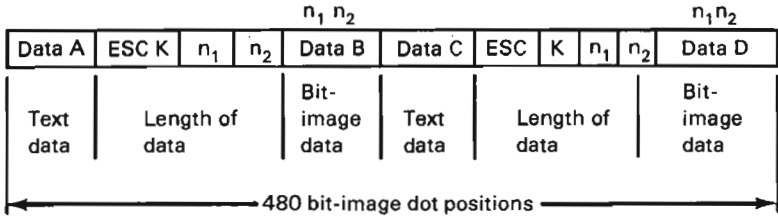


Data sent to the printer.



In text mode, 20 characters in text mode correspond to 120 bit-image positions ($20 \times 6 = 120$). The printable portion left in Bit-Image mode is 360 dot positions ($480 - 120 = 360$).

Data sent to the printer.



```

Example: 1 'OPEN PRINTER IN RANDOM MODE
WITH LENGTH OF 255
2 OPEN "LPT1:"AS #1
3 WIDTH "LPT1:",255
4 PRINT #1,CHR$(13)+CHR$(10);
5 SLASH$=CHR$(1)+CHR$(02)
+CHR$(04)+CHR$(08)
6 SLASH$=SLASH$+CHR$(16)+CHR$(32)
+CHR$(64)+CHR$(128)+CHR$(0)
7 GAP$=CHR$(0)+CHR$(0)+CHR$(0)
8 NDOTS=480
9 'ESC K N1 N2
10 PRINT #1,CHR$(27);"K";CHR$(NDOTS
MOD 256);CHR$(FIX(NDOTS/256));
11 'SEND NDOTS NUMBER OF BIT
IMAGE BYTES
12 FOR I=1 TO NDOTS/12 'NUMBER
OF SLASHES TO
PRINT USING GRAPHICS
13 PRINT #1,SLASH$;GAP$;
  
```

14 NEXT I

15 CLOSE

16 END

ESC L

This example gives you a row of slashes printed in the Bit-Image mode.

Escape L (960-Bit-Image Graphics-Mode)

Format: ESC L;n1;n2;v1;v2;...vk;

Changes from the Text mode to the Bit-Image Graphics mode. The input is similar to ESC K. The 960 Bit-Image mode prints at half the speed of the 480 Bit-Image Graphics mode, but can produce a denser graphic image. The number of bytes of bit-image Data (k) is $n1 + 256n2$ but cannot exceed 960. $n1$ is in the range of 0 to 255.

ESC N

Escape N (Set Skip Perforation)

Format ESC N;n;

Sets the Skip Perforation function. The number following ESC N sets the value for the number of lines of Skip Perforation. The example shows a 12-line skip perforation. This prints 54 lines and feeds the paper 12 lines. The value of n must be between 1 and 127. ESC N must be reset anytime the page length (ESC C) is changed.

Example:

```
LPRINT CHR$(27);CHR$(78);CHR$(12);
```

ESC O

Escape O (Cancel Skip Perforation)

Cancels the Skip Perforation function.

Example:

```
LPRINT CHR$(27);CHR$(79);
```

ESC S

Escape S (Subscript/Superscript)

Format: ESC S;n;

Changes the printer to the Subscript print mode when ESC S is followed by a 1, as in the example that follows. When ESC S is followed by a 0 (zero), the printer prints in the

Superscript print mode.

Example:

```
LPRINT CHR$(27);CHR$(83);CHR$(1);
```

ESC T

Escape T (Subscript/Superscript Off)

The printer stops printing in the Subscript or Superscript print mode.

Example:

```
LPRINT CHR$(27);CHR$(84);
```

ESC U

Escape U (Unidirectional Printing)

Format: ESC U;n;

The printer prints from left to right following the input of ESC U;1. When ESC U is followed by a 0 (zero), the left to right printing operation is canceled. The Unidirectional print-mode (ESC U) ensures a more accurate print-start position for better print quality.

Example:

```
LPRINT CHR$(27);CHR$(85);CHR$(1);
```

ESC W

Escape W (Double Width)

Format: ESC W;n;

Changes the printer to the Double-Width print mode when ESC W is followed by a 1. This mode is not canceled by a line-feed operation and must be canceled with ESC W followed by a 0 (zero).

Example:

```
LPRINT CHR$(27);CHR$(87);CHR$(1);
```

ESC Y

Escape Y (960 Bit-Image Graphics

Mode Normal Speed)

Format: ESC Y n1;n2;v1;v2;...vk;

Changes from the Text mode to the 960 Bit-Image Graphics mode. The printer prints at normal speed during this operation and cannot print dots on consecutive dot position. The input of data is similar to ESC L.

ESC Z

Escape Z (1920 Bit-Image Graphics Mode)

Format: ESC Z;n1;n2;v1;v2;...vk;
Changes from the Text mode to the 1920
Bit-Image Graphics mode. The input is
similar to the other Bit-Image Graphics
modes. ESC Z can print only every third dot
position.

0	1	2	3	4	5	6	7	8	9
NUL							BEL		HT
10	11	12	13	14	15	16	17	18	19
LF		FF	CR	SO	SI			DC2	
20	21	22	23	24	25	26	27	28	29
DC4				CAN			ESC		
30	31	32	33	34	35	36	37	38	39
		SP	!	''	#	\$	%	&	'
40	41	42	43	44	45	46	47	48	49
()	*	+	,	-	.	/	0	1
50	51	52	53	54	55	56	57	58	59
2	3	4	5	6	7	8	9	:	;
60	61	62	63	64	65	66	67	68	69
<	=	>	?	Ⓞ	A	B	C	D	E
70	71	72	73	74	75	76	77	78	79
F	G	H	I	J	K	L	M	N	O
80	81	82	83	84	85	86	87	88	89
P	Q	R	S	T	U	V	W	X	Y
90	91	92	93	94	95	96	97	98	99
Z	[\]	^	_	`	a	b	c
100	101	102	103	104	105	106	107	108	109
d	e	f	g	h	i	j	k	l	m
110	111	112	113	114	115	116	117	118	119
n	o	p	q	r	s	t	u	v	w
120	121	122	123	124	125	126	127	128	129
x	y	z	{		}	~		NUL	

Printer Character Set 1 (Part 1 of 2)

130	131	132	133	134	135	136	137	138	139
					BEL		HT	LF	
140	141	142	143	144	145	146	147	148	149
FF	CR	SO	SI			DC2		DC4	
150	151	152	153	154	155	156	157	158	159
		CAN			ESC				
160	161	162	163	164	165	166	167	168	169
á	í	ó	ú	ñ	Ñ	<u>a</u>	<u>o</u>	¿	␣
170	171	172	173	174	175	176	177	178	179
␣	½	¼		<<	>>	■	■	■	■
180	181	182	183	184	185	186	187	188	189
⌈	⌈	⌈	⌈	⌈	⌈	⌈	⌈	⌈	⌈
190	191	192	193	194	195	196	197	198	199
⌋	⌋	⌋	⌋	⌋	⌋	⌋	⌋	⌋	⌋
200	201	202	203	204	205	206	207	208	209
⌋	⌋	⌋	⌋	⌋	⌋	⌋	⌋	⌋	⌋
210	211	212	213	214	215	216	217	218	219
⌈	⌈	⌈	⌈	⌈	⌈	⌈	⌈	⌈	■
220	221	222	223	224	225	226	227	228	229
■	■	■	■	α	β	Γ	Π	Σ	σ
230	231	232	233	234	235	236	237	238	239
μ	τ	ϕ	θ	Ω	δ	∞	∅	ε	∩
240	241	242	243	244	245	246	247	248	249
≡	±	≥	≤	∫	J	÷	≈	◦	■
250	251	252	253	254	255				
-	√	∩	2	■	SP				

Printer Character Set 1 (Part 2 of 2)

0	1	2	3	4	5	6	7	8	9
NUL			♥	♦	♣	♠	BEL		HT
10	11	12	13	14	15	16	17	18	19
LF		FF	CR	SO	SI			DC2	
20	21	22	23	24	25	26	27	28	29
DC4	§			CAN			ESC		
30	31	32	33	34	35	36	37	38	39
		SP	!	”	#	\$	%	&	'
40	41	42	43	44	45	46	47	48	49
()	*	+	,	-	.	/	0	1
50	51	52	53	54	55	56	57	58	59
2	3	4	5	6	7	8	9	:	;
60	61	62	63	64	65	66	67	68	69
<	=	>	?	⌚	A	B	C	D	E
70	71	72	73	74	75	76	77	78	79
F	G	H	I	J	K	L	M	N	O
80	81	82	83	84	85	86	87	88	89
P	Q	R	S	T	U	V	W	X	Y
90	91	92	93	94	95	96	97	98	99
Z	[\]	^	_	`	a	b	c
100	101	102	103	104	105	106	107	108	109
d	e	f	g	h	i	j	k	l	m
110	111	112	113	114	115	116	117	118	119
n	o	p	q	r	s	t	u	v	w
120	121	122	123	124	125	126	127	128	129
x	y	z	{		}	~		Ç	ü

Printer Character Set 2 (Part 1 of 2)

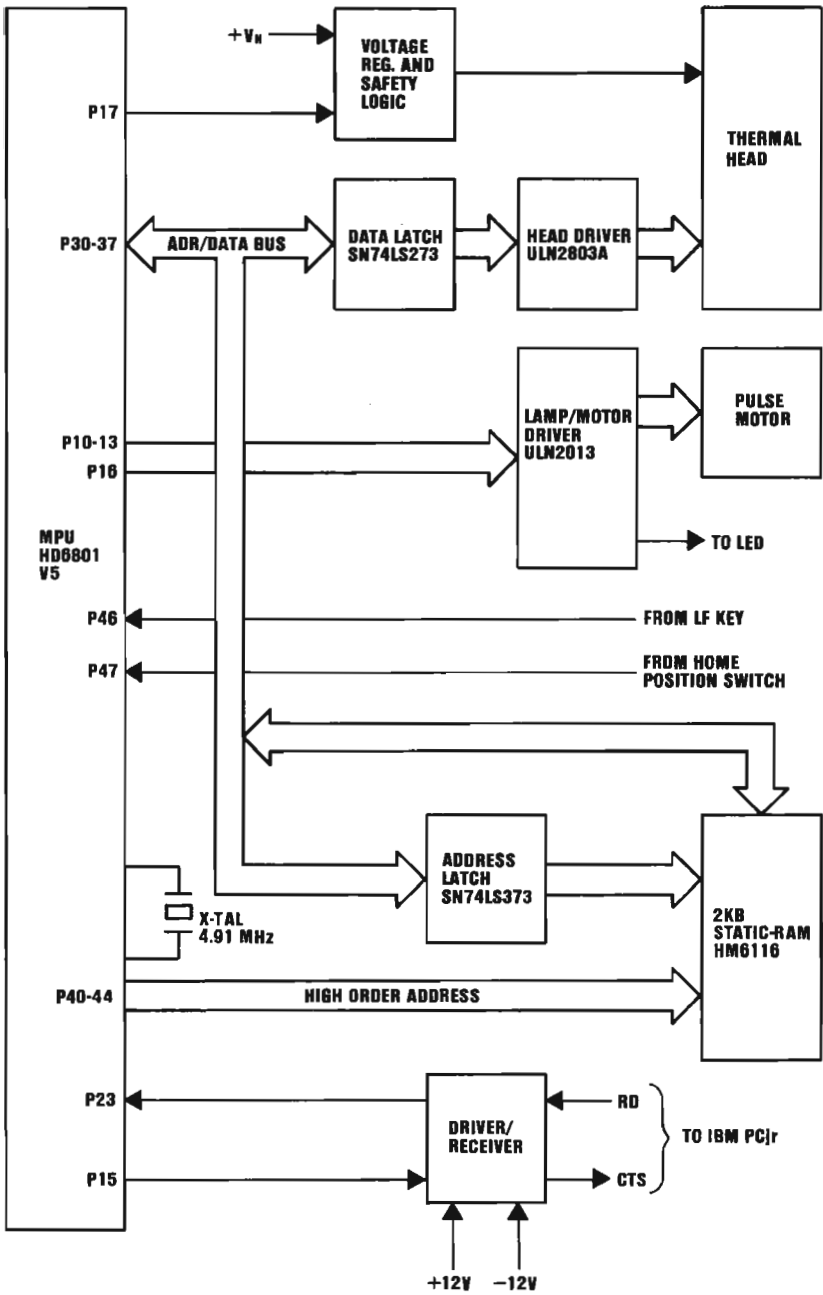
130	131	132	133	134	135	136	137	138	139
é	â	ä	à	å	ç	ê	ë	è	ï
140	141	142	143	144	145	146	147	148	149
î	ì	Ä	Â	É	æ	Æ	ô	ö	ò
150	151	152	153	154	155	156	157	158	159
û	ù	ÿ	ö	ü	ç	£	¥	₪	₯
160	161	162	163	164	165	166	167	168	169
á	í	ó	ú	ñ	Ñ	ₐ	ₒ	¿	₯
170	171	172	173	174	175	176	177	178	179
₯	½	¼		<<	>>	▒	▒	▒	▒
180	181	182	183	184	185	186	187	188	189
†	†	†	†	†	†	†	†	†	†
190	191	192	193	194	195	196	197	198	199
†	†	†	†	†	†	†	†	†	†
200	201	202	203	204	205	206	207	208	209
†	†	†	†	†	†	†	†	†	†
210	211	212	213	214	215	216	217	218	219
†	†	†	†	†	†	†	†	†	▒
220	221	222	223	224	225	226	227	228	229
▒	▒	▒	▒	α	β	Γ	Π	Σ	σ
230	231	232	233	234	235	236	237	238	239
μ	τ	ϕ	θ	Ω	δ	∞	∅	€	∩
240	241	242	243	244	245	246	247	248	249
≡	±	≥	≤	∫	J	÷	≈	◦	■
250	251	252	253	254	255				
-	√	∩	2	■	SP				

Notes:

IBM PC Compact Printer

The PC Compact Printer is a stand-alone, tabletop unit that plugs into a standard wall outlet. Using an eight-wire print head, the printer can print characters from the standard ASCII, 96-character, uppercase and lowercase character sets, and prints the characters in a 5-by-7 dot matrix at 56 characters-per-second (cps). It prints in one direction (left-to-right) and has four print modes. In the standard mode, the printer prints 80 characters-per-line; in the compressed mode, 136 characters; in the double-width mode, 40 characters, and in the compressed double-width mode, 68 characters-per-line. The PC Compact Printer can also underline characters, has an extended character-set for international languages, and can accept special characters programmed by the user.

The printer has a 1.89 meter (6-foot), 16-lead, printer cable that connects, through an Amphenol connector, to the serial port (RS-232-C) at the rear of the system unit.



Printer Specifications

Print Method:	Thermal, non-impact, Dot-matrix
Print Speed:	56 cps
Print Direction:	Left to right only
Number of Pins in Print Head:	8
Line Spacing:	4.23 mm (1/6 in)
Matrix Pattern:	5 by 7 Dots
Character Set:	Full 96-character ASCII with descenders, plus international characters/symbols
Graphics:	None

System Options

Print Modes:	Characters per Inch	Maximum Characters per Line
Standard	10	80
Double Width	5	40
Compressed	17.5	136
Compressed/ Double Width	8.75	68
Paper Feed:	Friction Feed	
Paper Width:	216 mm (8.5 in)	
Copies:	Single sheet only	
Paper Path:	Top	
System Interface:	Serial Data and Control Lines	
Print Color:	Black only	

Environmental Conditions

Temperature:	5°C (+41°F) to 40°C (104°F)
Humidity:	10 to 80% non-condensing
Power Requirement	
Voltage:	110 Vac 60 Hz
Current:	245 mA
Power Consumption:	36 watts
Heat Output:	57.6 kJ (54.6 BTU)/hr (maximum)

Physical Characteristics

Height:	88.9 mm (3.5 in)
Width:	312.4 mm (12.3 in)
Depth:	221 mm (8.7 in)
Weight:	2.99 kg (6.6 lb)
Power Cable Length:	1.98 m (6.5 ft)
Size:	28 AWG
Printer Cable Length:	1.83 m (6 ft)
Size:	3 by 18 AWG

Character Set:

ASCII numbers 0 to 31 contain control codes and special characters. ASCII numbers 32 to 127 contain the standard printable characters. ASCII numbers 128 to 175 contain European characters. ASCII numbers 224 to 255 contain math and extra symbols.

Serial Interface Description

Specifications:

Data Transfer Rate: 1200 bps (maximum)

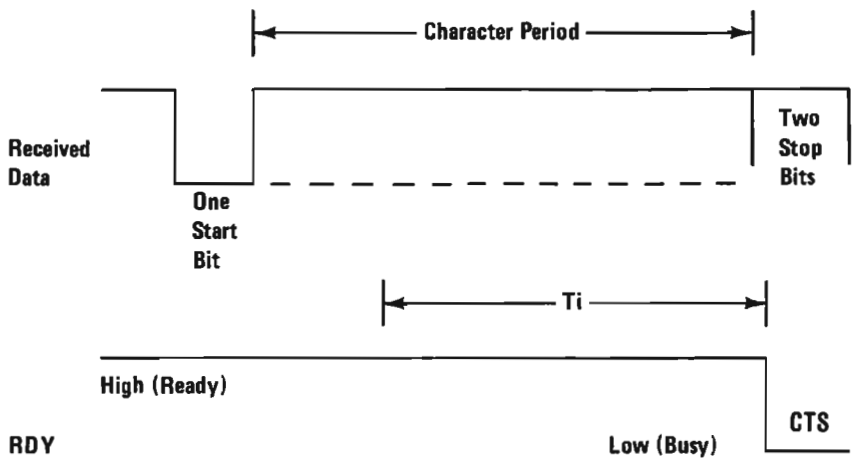
Synchronization: internal clocking

Handshaking: CTS (Clear to Send) Pacing

Logic Level: Input data and all interface control- signals are EIA Levels

Connector Plug: 9804 (Amphenol)

The following figure shows the timing of the Serial Interface.



Serial Interface Timing Diagram

Print Mode Combinations for the PC Compact Printer

The following figure shows the print-mode combinations possible with the PC Compact Printer. Modes shown in the same column can be combined. A print mode can be changed at any time within a line; however, the double-width mode effects the entire line.

Modes					
Standard	XXX				
Compressed		XXX		XXX	XXX
Double-Width			XXX	XXX	XXX
Underline	XXX	XXX	XXX		XXX

Printer Control Codes and Functions

On the following pages you will find a detailed list of the printer control codes and functions. This list also includes descriptions of the functions and examples of the printer control codes.

The examples (LPRINT statements) given in the detailed descriptions of the printer control codes and functions list, are written in BASIC. Some knowledge of BASIC programming is needed to understand these codes. Some of the printer control codes also show a "Format" description when more information is needed for programming considerations.

CODE	PRINTER FUNCTION
CAN	<p>Cancel Clears the printer buffer. Control codes, except SO, remain in effect. Reinitializes the printer to the power on defaults. LPRINT CHR\$(24);</p>
CR	<p>Carriage Return Ends the line the printer is on and prints any data remaining in the printer buffer. The logical character position is moved to the left margin. (No Line Feed operation takes place.) Note: IBM Personal Computer BASIC adds a Line Feed unless 128 is added. LPRINT CHR\$(13);</p>
DC2	<p>Device Control 2 (Compressed Off) Stops printing in the Compressed mode. LPRINT CHR\$(18);</p>
DC4	<p>Device Control 4 (Double Width Off) Stops printing in the Double Width mode. LPRINT CHR\$(20);</p>
ESC	<p>Escape Informs the printer that the following data is a printer command. (See the following ESC commands.) LPRINT CHR\$(27);</p>

ESC B**Escape B (Set Vertical Tabs)**

Sets vertical tab stop positions. Up to 64 vertical tab stop positions are recognized by the printer. Tab stop positions must be received in ascending numeric order. The tab stop numbers do not become valid until you type the NUL code. Once vertical tab stops are established, they are valid until new tab stops are specified. (If the printer is reset or switched Off, set tab stops are cleared.) If no tab stop is set, the Vertical Tab command acts as a Line Feed command. ESC B followed only by NUL cancels tab stops. The form length must be set by the ESC C command prior to setting tabs.

LPRINT

```
CHR$(27);CHR$(66);CHR$(10);CHR$(20);  
CHR$(40);CHR$(0);
```

ESC C**Escape C (Set lines per page)**

Format: ESC C;n; Sets the page length. The ESC C command must be followed by a value to specify the length of page desired. (Maximum form length for the printer is 127 lines.) The following example sets the page length to 55 lines. The printer default is 66 lines per page when switched On or reset.

```
LPRINT CHR$(27);CHR$(67);CHR$(55);
```

ESC D **Escape D (Set Horizontal Tab Stops)**
Sets the horizontal tab stop positions. The following example shows the horizontal tab stop positions set at printer column positions of 10, 20 and 40. The horizontal tab stops are followed by CHR\$(0), the NUL code. They must also be in ascending numeric order as shown. You can set tab stops between 1 and 80. When in the Compressed print mode, you can set tabs up to column 136. The maximum number of tabs that can be set is 112. HT (CHR\$(9)) is used to execute a tab operation.

LPRINT

```
CHR$(27);CHR$(68);CHR$(10)CHR$(20)  
CHR$(40);CHR$(0);
```

ESC K **Escape K (480 Bit-Image Graphics Mode)**
Format: ESC K;n1;n2; v1; v2;.....vk;
Changes the printer to the Bit-Image Graphics mode. Dot density is 82.5 by 82.5 dots per inch. If the graphics data exceeds the space remaining on the line, the printer ignores the excess data. Only the excess data is lost.

The numbers n1 and n2 specify, in binary form, the number of bit image data bytes to be transferred. Assign values to n1 to represent values from zero to 255 and assign values to n2 to represent values from 0-1 x 256. The total number of bit image data bytes cannot exceed 480. (n1 + (n2 X 256)).

The bit-image data bytes are v1 through vk.

All eight of the print head wires are used to print Bit-image graphics. Each bit of a bit-image data byte represents a dot position within a vertical line. The least significant bit (LSB) represents the bottom dot position, and the most significant bit (MSB) represents the top dot position. For example, if vX is hex 80, the top dot will print only in that vertical position; if vX is hex 01, the bottom dot will print; and if vX is hex FF, all eight dots will print.

	Dot	Bit Number
Top	O	--- 8
	O	--- 7
	O	--- 6
	O	--- 5
	O	--- 4
	O	--- 3
	O	--- 2
Bottom	O	--- 1

LPRINT CHR\$(27);CHR\$(75);n1;n2

ESC N

Escape N (Set Skip Perforation)

Format: ESC N;n; Sets the Skip Perforation function. The number following ESC N sets the number of lines to be skipped. The example shows a 12-line skip perforation. This command will print 54 lines and feed the paper 12 lines. The value of n must be between 1 and 127. ESC N must be reset anytime the page length (ESC C) is changed. The default for skip perforation is 25.4 mm (1 inch).

LPRINT CHR\$(27);CHR\$(78);CHR\$(12);

- ESC O** **Escape O (Cancel Skip Perforation)**
Cancels the Skip Perforation function.
LPRINT CHR\$(27);CHR\$(79);
- ESC R** **Escape R (Clear Tabs)**
Resets all tab stops, both horizontal and vertical to the powered-on defaults.
LPRINT CHR\$(27);CHR\$(82);
- ESC W** **Escape W (Double Width)**
Format: **ESC W;n**; Changes the printer to the Double Width mode when **ESC W** is followed by 1. This mode is not canceled by a line feed operation. It is canceled when **ESC W** is followed by 0 (zero).
LPRINT CHR\$(27);CHR\$(87);CHR\$(1);
- ESC 0** **Escape Zero (1/9-Inch Line Feed)**
Changes the line feed to 2.82 mm (1/9 inch).
LPRINT CHR\$(27);CHR\$(48);
- ESC 1** **Escape One (1/9-inch Line Feed)**
Changes the line feed to 2.82 mm (1/9 inch). **ESC 1** functions the same as **ESC 0**.
LPRINT CHR\$(27);CHR\$(49);
- ESC 2** **Escape Two (Start Variable Line Feeding)**
Resets line spacing to 4.23 mm (1/6 inch). This is the powered-on default for vertical line spacing.
LPRINT CHR\$(27);CHR\$(50);
- ESC 5** **Escape Five (Sets Automatic Line Feed)**
With automatic line feed on, when a CR code is received, a line feed automatically follows after the carriage return. **ESC 5 (1)** sets auto line feed; **ESC 5 (0)** resets it.
LPRINT CHR\$(27);CHR\$(53);

- ESC -** **Escape Minus (Underline)**
Format: ESC -;n; ESC - followed by 1, prints all of the following data with an underline. ESC - followed by 0 (zero), cancels the Underline print mode.
LPRINT CHR\$(27);CHR(45);CHR\$(1); [or CHR\$(0);]
- ESC <** **Escape Less Than (Home Head)**
The print head returns to the left margin to print the line following ESC <. This occurs for one line only.
LPRINT CHR\$(27);CHR\$(60);
- FF** **Form Feed**
Advances the paper to the top of the next page. Note: The location of the paper, when the printer power switch is set to the On position, determines the top of the page. The next top-of-page is 279 mm (11 inches) from that position. ESC C can be used to change the page length. Always separate multiple Form Feed commands with spaces.
LPRINT CHR\$(12);
- HT** **Horizontal Tab**
Tabs to the next horizontal tab stop. Tab stops are set with ESC D. (Tab stops are automatically set at every 8 columns when the printer power switch is set to the On position.)
LPRINT CHR\$(9);
- LF** **Line Feed**
Advances the paper one line. Line spacing is 4.23 mm (1/6 inch) unless reset by ESC 0, ESC 1, ESC 2.
LPRINT CHR\$(10);

- NUL** **Null**
Used with ESC B and ESC D as terminator for the tab set and clear commands.
LPRINT CHR\$(0);
- SI** **Shift In (Compressed On)**
Changes the printer to the Compressed Character mode. This command is canceled by a DC2 code (Compressed Off).
LPRINT CHR\$(15);
- SO** **Shift Out (Double Width)**
Changes the printer to the Double Width mode. Note: A Carriage Return, Line Feed or DC4 code cancels Double Width mode.
LPRINT CHR\$(14);
- VT** **Vertical Tab**
Spaces the paper to the next vertical tab position. VT are set by the ESC B sequence. The VT command is the same as the LF command, if no tabs are set. The paper is advanced one line after printing or advanced to the next vertical tab stop.
LPRINT CHR\$(11);

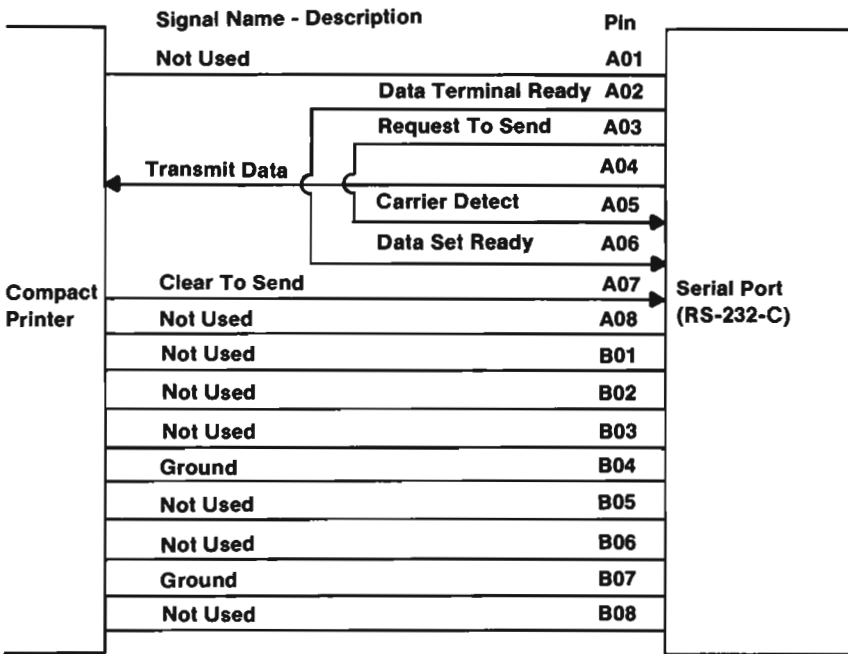
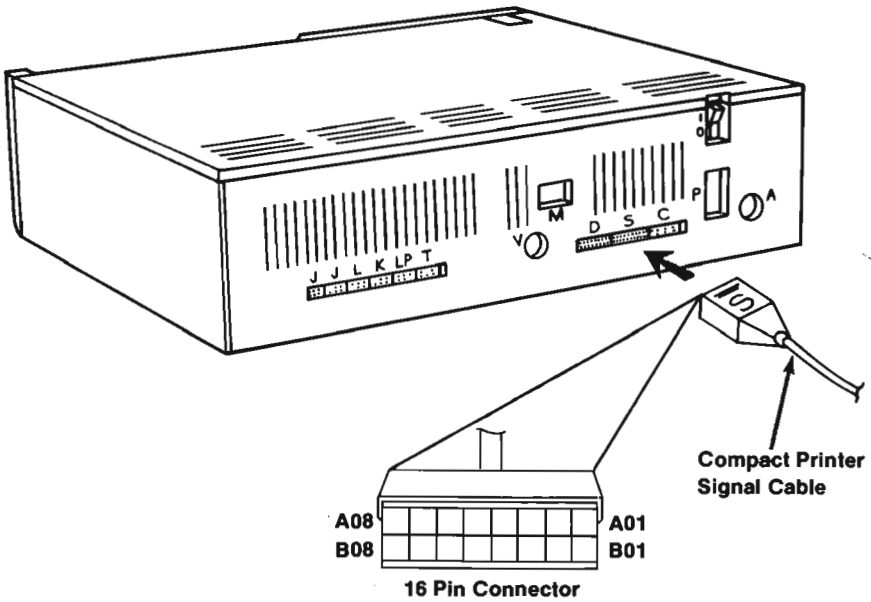
The following charts list the printer control codes and characters in ASCII decimal numeric order, (for example, NUL is 0 and ESC W is 87).

0	1	2	3	4	5	6	7	8	9
NUL			♥	♦	♣	♠	●	●	HT
10	11	12	13	14	15	16	17	18	19
LF	VT	FF	CR	SO	SI	▶	◀	DC2	!!
20	21	22	23	24	25	26	27	28	29
DC4	§	■	↕	CAN	↓	→	ESC	L	↔
30	31	32	33	34	35	36	37	38	39
◀	▶	SP	!	"	#	\$	%	&	'
40	41	42	43	44	45	46	47	48	49
()	*	+	,	-	.	/	0	1
50	51	52	53	54	55	56	57	58	59
2	3	4	5	6	7	8	9	:	;
60	61	62	63	64	65	66	67	68	69
<	=	>	?	⌚	A	B	C	D	E
70	71	72	73	74	75	76	77	78	79
F	G	H	I	J	K	L	M	N	O
80	81	82	83	84	85	86	87	88	89
P	Q	R	S	T	U	V	W	X	Y
90	91	92	93	94	95	96	97	98	99
Z	[\]	^	_	`	a	b	c
100	101	102	103	104	105	106	107	108	109
d	e	f	g	h	i	j	k	l	m
110	111	112	113	114	115	116	117	118	119
n	o	p	q	r	s	t	u	v	w
120	121	122	123	124	125	126	127	128	129
x	y	z	{		}	~	DEL	Ç	ü

Character Set (Part 1 of 2)

130	131	132	133	134	135	136	137	138	139
é	â	ä	à	å	ç	ê	ë	è	ï
140	141	142	143	144	145	146	147	148	149
î	ì	Ä	Å	É	æ	Æ	ô	ö	ò
150	151	152	153	154	155	156	157	158	159
û	ù	ÿ	ö	ü	ç	£	¥	₪	₯
160	161	162	163	164	165	166	167	168	169
á	í	ó	ú	ñ	Ñ	à	õ	ı	ƒ
170	171	172	173	174	175	176	177	178	179
↳	½	¼	¡	«	»				
180	181	182	183	184	185	186	187	188	189
190	191	192	193	194	195	196	197	198	199
200	201	202	203	204	205	206	207	208	209
210	211	212	213	214	215	216	217	218	219
220	221	222	223	224	225	226	227	228	229
				α	β	Γ	Π	Σ	σ
230	231	232	233	234	235	236	237	238	239
μ	τ	ϕ	θ	Ω	δ	∞	∅	ε	∩
240	241	242	243	244	245	246	247	248	249
≡	±	≥	≤	∫	∫	÷	≈	°	■
250	251	252	253	254	255				
-	√	∩	2	■	SP				

Character Set (Part 2 of 2)



**Data Terminal Ready Looped in Cable to Data Set Ready
Request to Send Looped in Cable to Carrier Detect**

Connector Specifications

SECTION 4. COMPATIBILITY WITH THE IBM PERSONAL COMPUTER FAMILY

Contents

Compatibility Overview	4-3
Timing Dependencies	4-5
Unequal Configurations	4-7
Hardware Differences	4-9
User Read/Write Memory	4-12
Diskette Capacity/Operation	4-13
IBM PCjr Cordless Keyboard	4-14
Color Graphics Capability	4-15
Black and White Monochrome Display	4-18
RS232 Serial Port and IBM PCjr Internal Modem	4-18
Summary	4-19

Notes:

Compatibility Overview

The IBM PCjr is a different Computer than the IBM Personal Computer and IBM Personal Computer XT. Even though it is different, the IBM PCjr has a high level of programming compatibility with the IBM Personal Computers. It is possible to create PCjr software applications that can run without modification on other IBM Personal Computers. In order to create such programs or to assess if a current program is compatible, you must understand the differences between the Personal Computers in the IBM family and know the proper way to communicate with them.

Normally, it would be impossible for a program written for one computer to run on a different computer since the microprocessors would be different; and the language of the application could not be executed by different processors. In this case, the application would have to be re-written entirely in the language of the other processor. Since the IBM PCjr and the other IBM Personal Computers use exactly the same microprocessors (Intel 8088), most assembler language programs need not be modified.

This alone is not enough, since applications normally take advantage of a computers device services (BIOS) and operating system (IBM DOS 2.1). In order to allow for maximum program compatibility, the IBM PCjr has maintained all BIOS system interrupts and utilizes the same IBM DOS. This means that applications which use the BIOS and the IBM DOS interrupts on the IBM Personal Computers operate the same on the IBM PCjr.

Note: The BIOS micro-code of the IBM PCjr is not identical to that of the IBM Personal Computers. If an application bypasses the BIOS interrupt calls and

directly accesses routines and/or storage locations in one system, it may not run in the other system. Some routines may be similar and some BIOS storage locations may be the same. It is strongly recommended that applications use only the BIOS and DOS interrupt interfaces in order to achieve compatibility in the IBM Personal Computer family.

Using the same language and the BIOS and DOS interfaces go a long way in achieving application compatibility. However, there are still several factors which need to be taken into consideration:

- **Timing Dependencies**
- **Unequal Configurations**
- **Hardware Differences**

Timing Dependencies

Programs running in user read/write memory normally run slower on the PCjr than on the IBM Personal Computers. Programs running in read-only memory (ROM) normally run a little faster on the PCjr than on the IBM Personal Computers. This may or may not cause a difference depending upon the application. Most applications are very I/O dependent in which case the execution time is not the critical factor and may not be noticeable. In other cases, the application runs the same but merely take a different amount of time.

If an application has very critical timing dependencies, any timing differences (faster or slower) may adversely affect its usability. Using an application's program execution speed to achieve a desired timing can effect the application. In these cases, the application may need to be modified.

Note: It is strongly recommended not to depend on instruction execution speed to achieve specific application timing. The system timer can provide short interval timing for assembly language programs. Similar timing functions are available in BASIC.

Performance of specific I/O devices (such as diskette or printer) may also differ between the PCjr and the other IBM Personal Computers. You should also avoid using timing of any I/O device as a dependency for the application.

Notes:

Unequal Configurations

In designing an application to run on both the IBM PCjr and the IBM Personal Computers, you need to make sure that the required hardware configuration is available on all machines. This means the application's minimum requirements are met by all IBM Personal Computers.

Notes:

Hardware Differences

To be able to run on either computer without change, an application utilizing a specific I/O device must have access to identical devices (or devices with identical operating characteristics and interfaces). The IBM PCjr and the IBM Personal Computers have very compatible I/O device capabilities.

The following table lists the hardware features and I/O devices supported by the IBM PCjr and the IBM Personal Computers and summarizes the differences:

Device	PC	PCXT	PCjr	PCjr Comments
Maximum User Memory	640KB	640KB	128KB	Shares user RAM with Video Buffer
Cordless Keyboard	No	No	Yes	Scan codes compatible and full 83 key capability
83 Key Keyboard	Yes	Yes	No	Compatible, but Hardware interface differences
Diskette Drive	Yes	Yes	Yes	Compatible, but different address and no DMA support
Hard Disk File	No	Yes	No	
Parallel Printer	Yes	Yes	Yes	Compatible
RS 232 Serial Port	Yes	Yes	Yes	Compatible, hex 2F8 address, Interrupt Level 3, Baud-Rate-Frequency divisor difference
Game Control	Yes	Yes	Yes	Compatible interface with potential timing differences
Cassette	Yes	No	Yes	Compatible
Internal Modem	No	No	Yes	Compatible to PC Serial Port hex 3F8 address, Interrupt Level 4, frequency divisor difference
IBM Monochrome Display	Yes	Yes	No	
Color Graphics and Display	Yes	Yes	Yes	Compatible, with some register differences and enhancements
Light Pen	Yes	Yes	Yes	Compatible

PCjr and Personal Computers Comparison (Part 1 of 2)

4-10 Hardware Differences

Device	PC	PCXT	PCjr	PCjr Comments
Attachable Joystick	Yes	Yes	Yes	Compatible
8253 Timer (time of day)	Yes	Yes	Yes	Compatible
8259 Interrupt	Yes	Yes	Yes	Some difference in interrupt levels
Internal Sound	Yes	Yes	Yes	Compatible but less frequency response
TI 76496 Sound	No	No	Yes	
ROM Cartridge Interface	No	No	Yes	
Future I/O ROM Architecture	Yes	Yes	Yes	Compatible

PCjr and Personal Computers Comparison (Part 2 of 2)

The hardware differences between the IBM PCjr and the IBM Personal Computers may lead to incompatibilities depending upon the specific application. Once again; if your application maintains an interface to the Personal Computer Family at the BIOS and DOS interrupt levels, then all hardware differences are handled transparently to your application. If your application goes below the BIOS level and directly addresses the hardware, then there could be an incompatibility.

User Read/Write Memory

Memory difference can be a problem even with programs written for the same computer, if the available memory is not the same from one machine to the next. Thus, the deciding factor is to state what the minimum memory requirement is for the application, and require that amount on the computer in question.

It is important to understand the memory aspects of the IBM PC*jr* in relationship to that of the IBM Personal Computers. The IBM PC*jr* can be configured for 64K bytes or 128K bytes (with memory expansion).

However, this user memory is not all available to the application. The IBM PC*jr* video architecture utilizes a minimum of 16K bytes (in graphic mode) and 2K bytes (in alpha numeric mode) for the screen buffer.

Therefore (in graphics mode), the IBM PC*jr* really has 48K bytes or 112K bytes (with memory expansion) available for system software. This is not the case with the IBM Personal Computers, since the color graphics adapter contains a separate 16K byte screen buffer. Thus, a 64K bytes Personal Computer with color graphics (extra 16K bytes) is an 80K byte system compared to a 64K byte IBM PC*jr*. The IBM PC*jr* also has graphic enhancements which allow more than the 16K bytes to be utilized for video screen buffers. If these enhanced features are used in an application, then even less is available for user memory.

Another aspect of available memory is the amount taken away by operating systems and language interpreters. In the case of the IBM DOS, both the IBM PC*jr* and the IBM Personal Computers support the same DOS. If your application requires the BASIC interpreter, then there may be a difference. The IBM Personal Computer Cassette BASIC resides entirely in the system ROM; taking no user memory. However, Disk BASIC or Advanced BASIC utilizes

approximately 10K bytes and 14K bytes respectively from user memory. In the IBM PC_{jr}, Advanced BASIC capabilities (cartridge BASIC) reside in ROM, taking no user memory.

As you can see, many items factor into user available memory requirements. The most frequent comparison is for the assembler language or compiled application using a 16K-byte screen buffer operating under DOS 2.1. In this case, an application requiring 64K bytes of user memory on an IBM Personal Computer cannot run on the IBM PC_{jr} without its expansion memory (128K byte capability). This is because of the IBM PC_{jr} video usage of 16K bytes. Also, any application requiring more than 112K bytes of user memory with DOS 2.1 on the IBM Personal Computers cannot run on an IBM PC_{jr}.

Diskette Capacity/Operation

Since the IBM PC_{jr} maximum stand-alone configuration is one diskette drive with a maximum capacity of 360K bytes diskette storage, an IBM PC_{jr} application is either limited by this diskette capacity or is impacted by the user having to change diskettes more frequently. The IBM Personal Computers can have multiple diskette drives with a capacity of 360K bytes diskette storage each or even possess hard files with a much larger disk storage capacity. This capacity difference may or may not be a concern depending upon the specific application.

In terms of diskette interfacing, the IBM PC_{jr} and the IBM Personal Computers both utilize the NEC μ PD765 floppy diskette controller, but with different hardware addresses, and the IBM PC_{jr} does not operate through direct memory access (DMA). Since the IBM PC_{jr} does not have DMA capability, application programs

cannot overlap diskette I/O operations. When diskette I/O takes place, the entire system is masked (operator keystrokes and asynchronous communications cannot take place). Therefore, the application must insure that asynchronous operations do not take place while diskette I/O is active.

IBM PCjr Cordless Keyboard

The Cordless Keyboard is unique to the IBM PCjr. Even though it does not possess all 83 keys of the IBM Personal Computers' keyboards, it does have the capability to generate all of the scan codes of the 83-key keyboard.

The following shows the additional functions available on the PCjr.

PCjr Special Functions	Required Key Combinations
Shift screen to the left	Alt + Ctrl + cursor left
Shift screen to the right	Alt + Ctrl + cursor right
Audio Feedback (System clicks when a key is pressed.)	Alt + Ctrl + Caps Lock
Customer Diagnostics	Alt + Ctrl + Ins

PCjr Special Functions

For more detail see "Keyboard Encoding and Usage" in Section 5.

Since all scan codes can be generated, any special application requirements can be met on the Cordless Keyboard.

The highest level of compatibility to interface to keyboards is through BIOS Interrupt hex 16 (read keystroke). Below that level is risky since there are hardware differences between the *PCjr* keyboard and the IBM Personal Computers' keyboards. The *PCjr* system utilizes the non-maskable (NMI) Interrupt to deserialize the scan codes and pass it to Interrupt hex 48 for compatible mapping to 83-key format. Interrupt level 9 remains a compatible interface for 83-key scan-code handling. It is not recommended to replace Interrupt level 9 even though a high degree of compatibility is maintained. If necessary, analyze this architecture carefully.

Color Graphics Capability

The IBM *PCjr* color graphic architecture is quite different from that of the IBM Personal Computers. The main difference (as previously discussed) is that the video buffer is taken from main user memory rather than having separate memory for video (as in the IBM Personal Computers). Normally, this would be an incompatibility since applications directly address the color graphics buffer at hex B8000. However, the IBM *PCjr* has special hardware to redirect hex B8000 addressing to any specific 16K-byte block of its user memory. The IBM *PCjr* defaults the video buffer to the high end 16K-byte block of user memory and applications can continue to address the video buffer at hex B8000. In addition all IBM Personal Computers' color graphics adapter modes are BIOS compatible and memory structure (bit map) compatible. These modes are:

Modes	Requirements
Alphanumeric: 40x25 BW 40x25 Color 80x25 Color 80x25 BW	None None Note None
Graphics: 320x200 4 Color 320x200 BW 640x200 BW	None None None
Note: PCjr requires the 64KB Memory and Display Expansion.	

Modes Available on the IBM Personal Computers and PCjr

In addition the IBM PCjr provides some new enhanced graphic modes which are not available to the IBM Personal Computers.

Modes	Requirements
Graphics: 320x200 16 Color 640x200 4 Color 160x200 16 Color	Note Note None
Note: PCjr requires the 64KB Memory and Display Expansion.	

Modes Available Only on PCjr

The IBM PCjr and IBM Personal Computers utilize the 6845 controller, but the hardware interface is not completely the same. Hardware addresses hex 3D8 and

hex 3D9 are not supported by the IBM PCjr video interface. Requests using these two addresses are not honored.

Also there are differences in the actual video used by the hardware. BIOS maintains compatibility by using the appropriate PCjr video parameters (addressed through Interrupt hex 1D) and maintains all video calls (through Interrupt hex 10). Application can still specify video parameter overrides by modifying Interrupt hex 1D to address their own parameters; however, since there are hardware differences the recommended approach is as follows:

1. Copy the original parameters from the BIOS of the system.
2. Change only those parameters desired.
3. Consider the specific video differences between systems.

Other differences to be aware of are:

- The IBM PCjr defaults the colorburst mode to be off, whereas the IBM Personal Computers default colorburst to on. Thus applications should not assume either default but set colorburst mode (through BIOS call) to the desired setting.
- The IBM PCjr video supports a full gray scale capability which the IBM Personal Computers do not.
- There can be some color differences between the IBM Personal Computers and the IBM PCjr; especially when color mixing techniques are used.

Black and White Monochrome Display

The IBM PC*jr* does not support the IBM Personal Computers black and white monochrome display. Programs which directly address the IBM Personal Computers monochrome display are not compatible. For example, any direct addressing of the B&W video buffer at hex B8000 is not redirected by the IBM PC*jr*. Applications should support Personal Computer video capabilities through BIOS, and the video buffer address is either transparent to the application or the address is provided indirectly in the BIOS data area.

RS232 Serial Port and IBM PC*jr* Internal Modem

The IBM PC*jr* serial port address is hex 2F8 and is associated with hardware Interrupt level 3. This is compatible with a second Asynchronous Communications Adapter on the IBM Personal Computers. The Internal Modem address is hex 3F8 and is associated with Interrupt level 4. This is compatible with the first Asynchronous Communications Adapter on the IBM Personal Computers. It is important to note that when the IBM PC*jr* has the Internal Modem installed it is logically COM1 and the RS232 serial port is logically COM2 in BIOS, DOS, and BASIC. Without the Internal Modem installed the RS232 serial port is logically addressed as COM1 in BIOS, DOS, and BASIC even though its address is still hex 2F8 using Interrupt level 3. Other hardware differences on the PC*jr* serial devices are:

- A different frequency divisor is needed to generate baud rate. This is transparent to applications using BIOS to initialize the devices (Interrupt Hex 14).
- No ring indicate capability on the RS232 serial port.

- Asynchronous communications input cannot be overlapped with IBM PCjr diskette I/O. Since diskette I/O operates in a non-DMA mode any asynchronous data received during diskette activity may be overrun (and lost). Thus, applications must insure that no diskette activity is active while receiving asynchronous communication data. This can be done by pacing the asynchronous device (tell it to hold from sending). The ASCII characters XOFF and XON are frequently used by some host computers for this purpose.

Summary

In summary, the IBM PCjr is a member of the IBM Personal Computer family by way of its strong architecture compatibility. The highest degree of application compatibility can be achieved by using a common high level language, and/ or accessing the system only through BIOS and DOS interrupts. It's not recommended to go below the BIOS level even though there are other hardware compatibilities. When it is necessary to design for particular computer differences, the application should determine at execution time which particular computer it is running on. This can be done by inspecting the ROM memory location at segment address hex F000 and offset hex FFFE for the following values

hex FF	= the IBM Personal Computer
hex FE	= the IBM Personal Computer XT
hex FD	= the IBM PCjr

Once determined, dual paths would handle any differences.

Notes:

SECTION 5. SYSTEM BIOS USAGE

Contents

ROM BIOS	5-3
BIOS Usage	5-5
Vectors with Special Meanings	5-8
Interrupt Hex 1B - Keyboard Break Address	5-8
Interrupt Hex 1C - Timer Tick	5-8
Interrupt Hex 1D - Video Parameters	5-9
Interrupt Hex 1E - Diskette Parameters ..	5-9
Interrupt Hex 1F and hex 44 - Graphics	
Character Pointers	5-9
Interrupt Hex 48 - Cordless Keyboard	
Translation	5-10
Interrupt Hex 49 - Non-Keyboard	
Scan-Code Translation-Table Address ...	5-10
Other Read Write Memory Usage	5-13
BIOS Programming Guidelines	5-18
Adapter Cards with System-Accessible	
ROM-Modules	5-18
Keyboard Encoding and Usage	5-21
Cordless Keyboard Encoding	5-21
Character Codes	5-26
Extended Codes	5-30
Shift States	5-31
Special Handling	5-34
System Reset	5-34
Break	5-34
Pause	5-34
Print Screen	5-34
Scroll Lock	5-35

Functions 1 thru 10	5-35
Function Lock	5-35
Screen Adjustment	5-35
Enable/Disable Keyboard Click	5-36
Run Diagnostics	5-36
Phantom-Key Scan-Code (Hex 55)	5-36
Other Characteristics	5-36
Non-Keyboard Scan-code Architecture	5-42
BIOS Cassette Logic	5-47
Software Algorithms - Interrupt Hex 15	5-47
Cassette Write	5-48
Cassette Read	5-49
Data Record Architecture	5-50
Error Detection	5-51

ROM BIOS

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 assembly-language programmer to perform block (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, 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.

Notes:

BIOS Usage

Access to BIOS is through the software interrupts. Each BIOS entry-point is available through its own interrupt, which can be found in "Personal Computer BIOS Interrupt Vectors", later in this section.

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 hex 12

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

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
MOV DX,LOW_COUNT
INT 1AH           ;set the time.
```

To read 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 can be seen 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	Keyboard NMI	KBDNMI
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	Vertical retrace	D_EOI
38-3B	E	Diskette Error Handler	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	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_PARMS
78-7B	1E	Diskette Parameters	DISK_BASE
7C-7F	1F	Video Graphics Chars	CRT_CHARH

Personal Computer BIOS Interrupt Vectors

Vectors with Special Meanings

The following are vectors with special meanings.

Interrupt Hex 1B - Keyboard Break Address

This vector points to the code to be executed when **Break** is 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 occurs when **Break** is pressed unless the application program sets a different value.

Control may be retained by this routine, with the following problem. The 'Break' may have occurred during interrupt processing, so that one or more 'End of Interrupt' commands must be issued 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 occurs unless the application modifies the pointer. It is the responsibility of the application to save and restore all registers that are modified.

Interrupt Hex 1D - Video Parameters

This vector points to a data region containing the parameters required for the initialization of the 6845 CRT Controller. 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. It is recommended that if a programmer wishes to use a different parameter table, that the table contained in ROM be copied to RAM and just modify the values needed for the application.

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 machine. Changing this parameter block may be necessary to reflect the specifications of the other drives attached. It is recommended that if a programmer wishes to use a different parameter table, that the table contained in ROM be copied to RAM and just modify the values needed for the application. The motor start-up-time parameter (parameter 10) is overridden by BIOS to force a 500-ms delay (value 04) if the parameter value is less than 04.

Interrupt Hex 1F and hex 44 - Graphics Character Pointers

When operating in the graphics modes, the

read/write-character interface forms the character from the ASCII code-point, using a table of dot patterns where each code point is comprised of 8 bytes of graphics information. The table of dot patterns for the first 128 code-points contained in ROM is pointed to by Interrupt Hex 44 and the second table of 128 code-points contained in ROM is pointed to by Interrupt Hex 1F. The user can change this vector to point to his own table of dot patterns. It is the responsibility of the user to restore these vectors to point to the default code-point-tables at the termination of the program.

Interrupt Hex 48 - Cordless Keyboard Translation

This vector points to the code responsible for translating keyboard scan-codes that are specific to the Cordless Keyboard. The translated scan-codes are then passed to the code pointed to by Interrupt Hex 9 which then handles the 83-key Keyboard scan codes.

Interrupt Hex 49 - Non-Keyboard Scan-Code Translation-Table Address

This interrupt contains the address of a table used to translate non-keyboard scan-codes (scan codes greater than 85 excluding 255.) If Interrupt hex 48 detects a scan code greater than 85 (excluding 255) it translates it using the table pointed to by Interrupt Hex 49. The address that Interrupt Hex 49 points to can be changed by users to point to their own table if different translations are required.

Note: It is recommended that a programmer save default pointers and restore them to their original values when the program has terminated.

Notes:

Other Read Write Memory Usage

The IBM BIOS routines use 256 bytes of memory starting at absolute hex 400 to hex 4FF. Locations hex 400 to 407 contain the base addresses of any RS-232C attachments to the system. This includes the optional IBM PCjr Internal Modem and the standard RS232 serial-port. Locations hex 408 to 40F contain the base addresses of any parallel printer attachments.

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.

The following is a list of the interrupts reserved for BIOS, DOS, and BASIC.

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
100-115	40-43	Reserved for BIOS
116-119	44	First 128 Graphics Characters
120-131	45-47	Reserves for BIOS
132-135	48	Cordless-Keybaord Translation
136-139	49	Non-keyboard Scan-code Translation Table
140-17F	50-5F	Reserved for BIOS
100-17F	40-5F	Reserved for BIOS
180-19F	60-67	Reserved for User Software Interrupts
1A0-1FF	68-7F	Reserved
200-217	80-85	Reserved for Basic
218-3C3	86-F0	Used by Basic Interpreter while BASIC is running
3C4-3FF	F1-FF	Reserved

BIOS, BASIC, and DOS Reserved Interrupts

The following is a list of reserved memory locations.

Address (Hex)	Mode	Function
400-48F 490-4EF 500-5FF	ROM BIOS	See BIOS Listing Reserved for System Usage Communication Area for any application
500	DOS	Reserved for DOS and BASIC, Print Screen Status Flag Store, O-Print Screen Not Active or Successful Print Screen Operation, I-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

The following is a list of the BASIC workspace variables.

If you do DEF SEG (Default workspace segment):	Offset (Hex)	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>) L H</p> <p>(</p> <p>100 <table border="1" data-bbox="199 794 456 850"> <tr> <td data-bbox="199 794 332 850">hex 64</td> <td data-bbox="332 794 456 850">hex 00</td> </tr> </table></p>			hex 64	hex 00
hex 64	hex 00			

BASIC Workspace Variables

The following shows the mapping of the BIOS memory

Starting Address in Hex

00000

**BIOS
Interrupt
Vectors**

00400

**BIOS
Data
Area**

00500

**User
Read/Write
Memory**

A0000

**Reserved
for Future
Video**

B8000

**Reserved
for Video**

C0000

**Reserved
for Future
I/O ROM**

D0000

**Reserved
for
Cartridges**

E0000

**Reserved
for
Cartridges**

F0000

**BIOS/
Diagnostics/
Cassette and
BASIC
Program
Area**

BIOS System Map

BIOS Programming Guidelines

The BIOS code is invoked through software interrupts. The programmer should not 'hard code' BIOS addresses into applications. **The internal workings and absolute addresses within BIOS are subject to change without notice.**

If an error is reported by the diskette code, you should 'reset' the drive adapter and retry the operation. A specified number of retries should be required on diskette 'reads' to insure the problem is not due to motor start-up.

When altering I/O-port bit-values, the programmer should change only those bits which 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 systems.

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 Power-On Self-Test (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 C0000 through hex D0000 are scanned in 2K-byte 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:** length (multiple of 2K bytes) - 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'.

Notes:

Keyboard Encoding and Usage

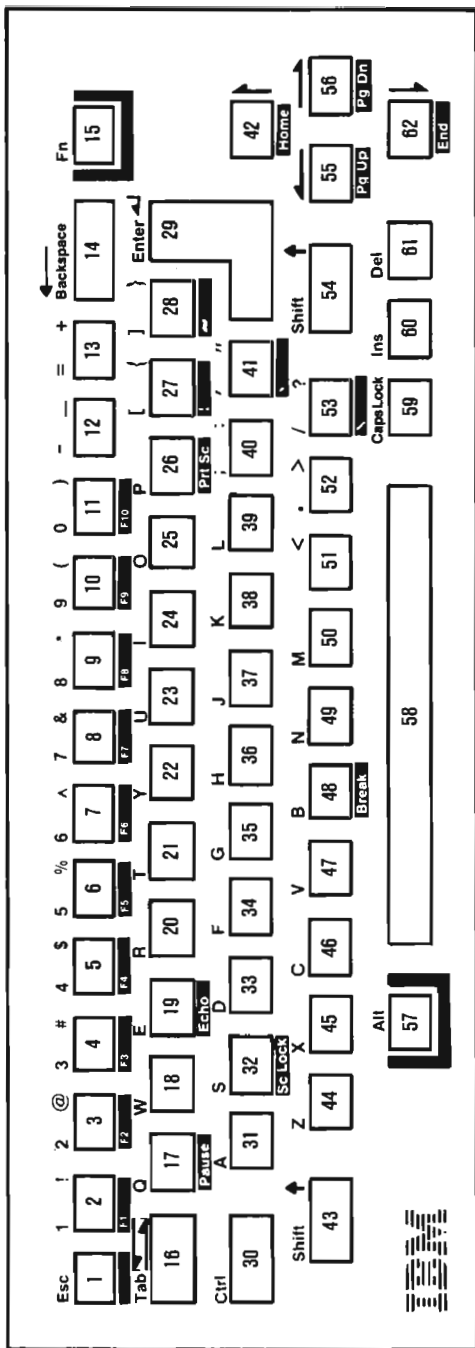
The following explains how the keyboard interacts with BIOS and how 83-key-keyboard functions are accomplished on the Cordless Keyboard.

Cordless Keyboard Encoding

The **KEYBOARD** routine provided by IBM in the ROM BIOS is responsible for converting the keyboard scan-codes into what is 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.

The following is the physical layout of the IBM PCjr Cordless Keyboard.



IBM PCjr Cordless Keyboard Diagram

The following are charts of the scan codes for the IBM PCjr Cordless Keyboard.

Key Position	Keyboard Characters	Make Code (Hex)	Break Code (Hex)
1	ESC	1	81
2	1/!	2	82
3	2/@	3	83
4	3/#	4	84
5	4/\$	5	85
6	5/%	6	86
7	6/≅	7	87
8	7/&	8	88
9	8/*	9	89
10	9/(A	8A
11	0/)	B	8B
12	-/_	C	8C
13	=/+	D	8D
14	BS←	E	8E
15	FN	54	D4
16	TAB	F	8F
17	q/Q	10	90
18	w/W	11	91
19	e/E	12	92
20	r/R	13	93
21	t/T	14	94
22	y/Y	15	95
23	u/U	16	96
24	i/I	17	97
25	o/O	18	98
26	p/P	19	99
27	[/{	1A	9A
28]}/	1B	9B
29	ENTER	1C	9C
30	CTRL	1D	9D
31	a/A	1E	9E

Cordless Keyboard Maxtrix Scan Codes (Part 1 of 2)

Key Position	Keyboard Characters	Make Code (Hex)	Break Code (Hex)
32	s/S	1F	9F
33	d/D	20	A0
34	f/F	21	A1
35	g/G	22	A2
36	h/H	23	A3
37	j/J	24	A4
38	k/K	25	A5
39	l/L	26	A6
40	;/:	27	A7
41	'/"	28	A8
42	CUR.UP	48	C8
43	LF.SHIFT	2A	AA
44	z/Z	2C	AC
45	x/X	2D	AD
46	c/C	2E	AE
47	v/V	2F	AF
48	b/B	30	B0
49	n/N	31	B1
50	m/M	32	B2
51	,/<	33	B3
52	./>	34	B4
53	//?	35	B5
54	RT.SHIFT	36	B6
55	CUR.LF.	4B	CB
56	CUR.RT.	4D	CD
57	ALT.	38	B8
58	SP.BAR	39	B9
59	CAPS LOCK	3A	BA
60	INSERT	52	D2
61	DELETE	53	D3
62	CUR.DWN.	50	D0
Phantom-Key Scan Code		55	

Cordless Keyboard Matrix Scan Codes (Part 2 of 2)

The Cordless Keyboard is unique to the PCjr. Even though it does not possess all 83 keys of the IBM Personal Computer keyboard, it does have a way in which you can cause all of the scan codes of the 83-key keyboard. The following chart shows the mapping of functions between both keyboards:

IBM Personal Computers 83-key Keyboard Function	IBM PCjr Cordless Keyboard Mapping
F1-F10 Ctrl Break Ctrl PrtSc (Echo Print) Shift PrtSc (Print Screen) Ctrl NumLock (Pause) Scroll Lock Numeric keypad region: Num Lock (Number keypad 1 through 10 becomes key scan codes.) PgUp key PgDn key Home key End key Numeric keypad - sign Numeric keypad + sign \ key ' key ! key ~ key * with PrtSc Numeric keypad . All 256 extended codes: Alt + numeric value from numeric keypad	Function key + 1-0 (F1-F10) Function key + B (Break) Function key + E (Echo) Function key + P (PrtSc) Function key + Q (Pause) Function key + S (ScLock) Alt + Function key + N (1 through 0 becomes numeric-key scan-codes) Function key + cursor left (PgUp) Function key + cursor right (PgDn) Function key + cursor up (Home) Function key + cursor down (End) Function key plus the - sign Function key + = sign Alt + / Alt + ' Alt + [Alt +] Alt + . Shift + Del NumLock then Alt + numeric value (1 through 0)

83-key-Keyboard Function to Cordless-Keyboard Mapping

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. See Appendix C, "Characters, Keystrokes, and Color" for the exact codes.

Key Number	Base Case	Upper Case	Ctrl	Alt	Fn
1	Esc	Esc	Esc	-1	**
2	1	!	-1	*,*****	(F1) *,***
3	2	@	Nul (000)	*,*****	(F2) *,***
4	3	#	-1	*,*****	(F3)
5	4	\$	-1	*,*****	(F4) *,***
6	5	%	-1	*,*****	(F5) *,***
7	6	^	RSO (030)	*,*****	(F6) *,***
8	7	&	-1	*,*****	(F7) *,***
9	8	*	-1	*,*****	(F8) *,***
10	9	(-1	*,*****	(F9) *,***
11	0)	-1	*,*****	(F10) *,***
12	—	-	US (031)	*	***
13	=	+	-1	*	***
14	Backspace (008)	Backspace (008)	DEL (127)	-1	-1
15 Fn	-1	-1	-1	-1	-1
16	—> (009)	<— *	-1	-1	-1
17	q	Q	DC1 (017)	*	**,*** (Pause)
18	w	W	ETB (023)	*	-1
19	e	E	ENQ (005)	*	**,*** (Echo)
20	r	R	DC2 (018)	*	-1
21	t	T	DC4 (020)	*	-1

- * - Refer to “Extended Codes” in this section.
- ** - Refer to “Special Handling” in this section.
- *** - Refer to “83-Key Keyboard functions to Cordless Keyboard Mapping Chart.”
- **** - Uppercase for cursor keys can be selected by pressing left or right shift or entering the Numlock state (Alt + Fn + N).
- ***** - When Alt is pressed and the keyboard is in the Numlock state, the upper row of digits is used to enter ASCII codes for generating any character from the extended ASCII character set.

Cordless-Keyboard Character Codes (Part 1 of 4)

Key Number	Base Case	Upper Case	Ctrl	Alt	Fn
22	y	Y	EM (025)	*	-1
23	u	U	NAK (021)	*	-1
24	i	I	HT (009)	*	-1
25	o	O	SI (015)	*	-1
26	p	P	DLE (016)	*	** , *** (PrtScreen)
27	[{	Esc (027)	() ***	-1
28]	}	GS (029)	(~) ***	-1
29	CR	CR	LF (010)	-1	-1
30 Ctrl	-1	-1	-1	-1	-1
31	a	A	SOH (001)	*	-1
32	s	S	DC3 (019)	*	** , *** (Scroll Lock)
33	d	D	EOT (004)	*	-1
34	f	F	ACK (006)	*	-1
35	g	G	BELL (007)	*	-1
36	h	H	BS (008)	*	-1
37	j	J	LF (010)	*	-1
38	k	K	VT (011)	*	-1
39	l	L	FF (012)	*	-1
40	;	:	-1	-1	-1
41	,	"	-1	(') ***	-1

- * - Refer to "Extended Codes" in this section.
- ** - Refer to "Special Handling" in this section.
- *** - Refer to "83-Key Keyboard functions to Cordless Keyboard Mapping Chart."
- **** - Uppercase for cursor keys can be selected by pressing left or right shift or entering the Numlock state (Alt + Fn + N).
- ***** - When Alt is pressed and the keyboard is in the Numlock state, the upper row of digits is used to enter ASCII codes for generating any character from the extended ASCII character set.

Cordless-Keyboard Character Codes (Part 2 of 4)

Key Number	Base Case	Upper Case	Ctrl	Alt	Fn	Alt + Ctrl
42	Cur.Up*	8 ****	-1	*	**,*** (Home)	
43 Left Shift	-1	-1	-1	-1	-1	
44	z	Z	SUB (026)	*	-1	
45	x	X	CAN (024)	*	-1	
46	c	C	EXT (003)	*	-1	
47	v	V	SYN (022)	*	-1	
48	b	B	STX (002)	*	**,*** (Break)	
49	n	N	SO (014)	*,***	***	
50	m	M	CR (013)	*	-1	
51	,	<	-1	-1	-1	
52	.	>	-1	(*) *	-1	
53	/	?	-1	\	-1	
54 Right Shift	-1	-1	-1	-1		
55	Cur.L *	4 ****	* Reverse Word	*	**,*** (PgUp)	**
56	Cur.R *	6 ****	* Advance Word	*	**,*** (PgDn)	**

- * - Refer to "Extended Codes" in this section.
- ** - Refer to "Special Handling" in this section.
- *** - Refer to "83-Key Keyboard functions to Cordless Keyboard Mapping Chart."
- **** - Uppercase for cursor keys can be selected by pressing left or right shift or entering the Numlock state (Alt + Fn + N).
- ***** - When Alt is pressed and the keyboard is in the Numlock state, the upper row of digits is used to enter ASCII codes for generating any character from the extended ASCII character set.

Cordless-Keyboard Character Codes (Part 3 of 4)

Key Number	Base Case	Upper Case	Ctrl	Alt	Fn	Alt + Ctrl
57 Alt	-1	-1	-1	-1	-1	
58	Space	Space	Space	Space	Space	
59 Caps Lock	-1	-1	-1	-1	-1	**
60	Ins.	0 ****	-1	*	-1	**
61	Del. *	. ****	-1	*	-1	**
62	Cur.Dn *	2 ****	-1	*	**,*** End	

- * - Refer to "Extended Codes" in this section.
- ** - Refer to "Special Handling" in this section.
- *** - Refer to "83-Key Keyboard functions to Cordless Keyboard Mapping Chart."
- **** - Uppercase for cursor keys can be selected by pressing left or right shift or entering the Numlock state (Alt + Fn + N).
- ***** - When Alt is pressed and the keyboard is in the Numlock state, the upper row of digits is used to enter ASCII codes for generating any character from the extended ASCII character set.

Cordless-Keyboard Character Codes (Part 4 of 4)

Extended Codes

An extended code is used for certain functions that cannot be represented in the standard ASCII code. A character code of 000 (Nul) is returned in AL. This indicates that the system or application program should examine a second code that indicates the actual function. This code is returned in AH. This is the same for both the Cordless Keyboard and 83-key keyboard.

Second Code	Function
3	Null Character
15	←
16 through 25	Alt Q, W, E, R, T, Y, U, I, O, P
30 through 38	Alt A, S, D, F, G, H, J, K, L
44 through 50	Alt Z, X, C, V, B, N; M
59 through 68	Fn + 1, 2, 3, 4, 5, 6, 7, 8, 9, 0 (Functions 1 through 10)
71	Home
72	Up Arrow
73	Page Up
75	← (Cursor Left)
77	→ (Cursor Right)
79	End
80	Down Arrow
81	Page Down
82	Ins (Insert)
83	Del (Delete)
84 through 93	F11 through F20 (Upper Case F1 through F10)
94 through 103	F21 through F30 (Ctrl F1 through F10)
104 through 113	F31 through F40 (Alt F1 through F10)
114	Fn/E or Ctrl/Fn/P (Start/Stop Echo to Printer)
115	Ctrl ← (Reverse Word)
116	Ctrl → (Advance Word)
117	Ctrl/End [Erase End of Line (EOL)]
118	Ctrl/PgDn [Erase to End of Screen (EOS)]
119	Ctrl/Home (Clear Screen and Home)
120 through 131	Alt/1, 2, 3, 4, 5, 6, 7, 8, 9, 0, -, = (Keys 2 through 13)
132	Ctrl/PgUp (Top 25 Lines of Text and Home Cur.)
133 through 149	Reserved
150 through 190	Reserved for Non-Keyboard Scan Codes

Cordless Keyboard Extended Functions

Shift States

Most shift states are handled within the **KEYBOARD** routine, transparent to the system or application

program. The current set of active shift states is available by 'calling' an entry point in the ROM **KEYBOARD**-routine. The following keys result in altered shift-states:

Shift

This key temporarily shifts keys 2 thru 13, 16 thru 28, 31 thru 41, and 44 thru 53 to upper case (base case if in Caps Lock state). The **Shift** key temporarily reverses the 'Num Lock' or 'non-Num-Lock' state of keys 42, 55, 56, and 60 thru 62.

Ctrl

This key temporarily shifts keys 3, 7, 12, 14, 16 thru 28, 30 thru 38, 42, 44 thru 50, 55, and 56 to the **Ctrl** state. 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, with the **Num Lock** key to cause the 'Pause' function, with the **Alt** and **Cursor Left** or **Right** for 'screen adjustment', with **Alt** and **Ins** to 'activate diagnostics', and with **Alt** and **CapsLock** to 'activate keyboard clicking'. These functions are described in "Special Handling" on the following pages.

Alt

The **Alt** key temporarily shifts keys 2 thru 13, 17 thru 26, 31 thru 39, and 44 thru 50 to the 'Alternate state'. 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 is also used with keys 27, 28, 41, and 53 to produce the characters under the key.

The **Alt** key has another use. This key allows the user to enter any character code from 0 to 255 into the system from the keyboard. The user must first put the keyboard in the 'Num Lock' state (concurrently press, first **Alt** then **Fn + n**). Then while holding down the **Alt** key type the decimal value of the character desired using keys 2 thru 11. 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 internal to the **KEYBOARD** routine.

Caps Lock

This key shifts keys 17 thru 25, 31 thru 39, and 44 thru 50 to 'upper case'. A second press of the **Caps Lock** key reverses the action. **Caps Lock** is handled internal to the **KEYBOARD** routine.

Shift-Key Priorities and Combinations

The following keys are listed in descending priority for translation in Interrupt Hex 48 and Interrupt hex 9 respectively:

1. Interrupt Hex 48
 - a. **Alt** key
 - b. **Ctrl** key
 - c. **Shift** key
2. Interrupt Hex 9
 - a. **Ctrl**
 - b. **Alt**
 - c. **Shift**

Of the three keys listed, only **Alt** and **Ctrl** are a valid combination. If any other combination of the three keys is used, only the key with the higher priority is recognized by the system.

Special Handling

System Reset

The combination of the **Alt**, **Ctrl**, and **Del** keys causes the **KEYBOARD** routine to initiate the equivalent of a 'System Reset'.

Break

The combination of the **Fn** and **B** keys results in the **KEYBOARD** routine signaling Interrupt Hex 1A. The extended characters (**AL** = hex 00, **AH** = hex 00) are returned.

Pause

The combination of the **Fn** and **Q** keys causes the **KEYBOARD**-interrupt routine to loop, waiting for any key to be pressed. This provides a system or application-transparent method of temporarily suspending an operation such as list or print and then resuming the operation by pressing any other key. The key pressed to exit the 'Pause' mode is unused otherwise.

Print Screen

The combination of the **Fn** and **P** keys results in an interrupt, invoking the **PRINT SCREEN** routine. This

routine works in the alphanumeric or graphics mode, with unrecognizable characters printing as blanks.

Scroll Lock

The combination of the **Fn** and **S** key is interpreted by appropriate application programs to indicate that the cursor-control keys should cause 'windowing' over the text rather than cursor movement. Pressing the 'Scroll Lock' combination a second time reverses the action. The **KEYBOARD** routine simply records the current shift state of 'Scroll Lock'. It is the responsibility of the system or application program to perform the function.

Functions 1 thru 10

The combination of the **Fn** key (15) and one of keys 2 thru 11 results in the corresponding 'Function' with key 2 being 'F1' up to key 11 being 'F10'.

Function Lock

Concurrently pressing first the **Fn** key and **Shift** key, and then pressing the **Esc** key causes keys 2 thru 11 to shift to their 'Function' states and remain there until the same combination is pressed again.

Screen Adjustment

The combination of the **Alt** key, **Ctrl** key, and either the **Left** or **Right** cursor movement key causes the screen to shift one character in the corresponding direction, up to a maximum of four.

Enable/Disable Keyboard Click

The combination of the **Alt**, **Ctrl**, and **Caps Lock** keys causes the keyboard audio feedback (click) to shift between 'on' and 'off'. The Power-On default is 'off'.

Run Diagnostics

The combination of the **Alt**, **Ctrl**, and **Ins** keys causes the system diagnostics stored in ROM to be initiated.

Phantom-Key Scan-Code (Hex 55)

The Phantom-Key scan-code is generated by the keyboard when an invalid combination of three or more keys is pressed. The keys pressed that caused the Phantom-Key scan-code are not put into the keyboard buffer, and are ignored by the keyboard microprocessor. The Phantom-Key scan-code is transmitted to BIOS where it is ignored.

Other Characteristics



The keyboard buffer is large enough to support a fast typist. If a key is pressed when the buffer is full, the character generated is ignored and the 'bell' is sounded. A larger buffer can be specified by modifying words at labels 'Buffer-Start' (hex 480) and 'Buffer-End' (hex 482) to point to another offset within segment hex 40.

The **KEYBOARD** routine suppresses the typematic action of the following keys: **Ctrl**, **Shift**, **Alt**, **Caps Lock**, **Insert**, and **Function**.









Function	Key Combinations	Description
System Reset	Alt + Ctrl + Del	Unconditional system reset
Break	Fn + B	Breaks program execution
Pause	Fn + Q	Resumable pause in program execution
Print Screen	Fn + P	
Function Lock	Fn and Shift then Esc (Held) concurrently)	Locks the number keys as Function keys (F1-F10) and B, Q, P, E, S, and the cursor control keys to their function states
Screen Adjustment	Alt + Ctrl + cursor right or cursor left	Allows the user to adjust the display's image left or right
Keyboard Click	Alt + Ctrl + CapsLock	Enables or disables the keyboard audio feedback click
Run Diagnostics	Alt + Ctrl + Ins	Initiates system ROM diagnostics
Keyboard Adventure Game	Esc	If the first key pressed after the system comes up in Cassette BASIC is Esc (key #1) then the Keyboard Adventure Game will be activated.
Cassette Autoload	Ctrl + Esc	If this is the first key sequence after the system comes up in Cassette BASIC then the screen will display 'Load "CAS1:"R followed by a Carriage Return. This allows a cassette program to be automatically loaded.

Keyboard Usage


“Keyboard Usage” is a set of guidelines of key-usage when performing commonly-used functions.

Function	Keys	Comment
Home Cursor	Fn Home	Editors; word processors
Return to outermost menu	Fn Home	Menu driven applications
Move cursor up	Up Arrow	Full screen editor, word processor
Page up, scroll backwards 25 lines	Fn PgUp	Editors; word processors
Move cursor left		Text, command entry
Move cursor right		Text, command entry
Scroll to end of text place cursor at end of line	Fn End	Editors; word processors
Move cursor down	Down Arrow	Full screen editor, word processor
Page down, scroll forwards 25 lines and home	Fn PgDn	Editors; word processors
Start/Stop insert text at cursor, shift text right in buffer	Ins	Text, command entry

Keyboard - Commonly Used Functions (Part 1 of 3)

Function	Keys	Comment
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 Fn Home	
Scroll up	Up Arrow	In scroll lock mode
Scroll down	Down Arrow	In scroll lock mode
Scroll left		In scroll lock mode
Scroll right		In scroll lock mode
Delete from cursor to EOL (end of line)	Ctrl Fn End	Text, command entry
Exit/Escape	Esc	Editor, 1 level of menu and so on
Start/Stop Echo screen to printer	Fn PrtSc	Any time
Delete from cursor to EOS (end of screen)	Ctrl Fn PgDn	Text, command entry
Advance word	Ctrl 	Text entry
Reverse word	Ctrl 	Text entry
Window Right	Ctrl 	When text is too wide to fit the screen.

Keyboard - Commonly Used Functions (Part 2 of 3)

Function	Keys	Comment
Window Left	Ctrl 	When text is too wide to fit the screen
Enter insert mode	Ins	Line Editor
Exit insert mode	Ins	Line Editor
Cancel current line	Esc	Command entry, text entry
Suspend system (Pause)	Ctrl Fn Pause	Stop list, stop program, and so on. Resumes on any key.
Break interrupt	Fn Break	Interrupt current process
System reset	Alt Ctrl Del	Reboot
Top of document and home cursor	Ctrl Fn PgUp	Editors, word processors
Standard function keys	Shift Fn/F1 through Fn/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 through 13 (1 through 9, 0) (-, =)	Line Editor
Extra function keys	Alt A through Z	Used when function starts with the same letter as one of the alpha keys.

Keyboard - Commonly Used Functions (Part 3 of 3)

Function	Key
Carriage return	↵ (Enter)
Line feed	Ctrl ↵ (Enter)
Bell	Ctrl G
Home	Fn Home
Cursor up	Up Arrow
Cursor down	Down Arrow
Cursor left	←
Cursor right	→
Advance one word	Ctrl →
Reverse one word	Ctrl ←
Insert	Ins
Delete	Del
Clear screen	Ctrl Fn Home
Freeze output	Fn Pause
Tab advance	→
Stop Execution (break)	Fn Break
Delete current line	Esc
Delete to end of line	Ctrl Fn End
Position cursor to end of line	Fn End

BASIC Screen Editor Special Functions

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

DOS Special Functions

Non-Keyboard Scan-code Architecture

The architecture of the IBM PC_{jr} BIOS is designed to also receive scan codes above those generated by the keyboard to accommodate any future device.

The keyboard generates scan codes from hex 1 to 55 and FF. Any scan codes above hex 55 (56 thru 7E for 'make' codes and D6 thru FE for 'break' codes) are processed by BIOS in the following manner:

1. If the incoming 'make' scan code falls within the range of the translate table, whose address is pointed to by BIOS Interrupt Hex 49, it is translated into the corresponding scan code. Any incoming 'break' codes above hex D5 are ignored.

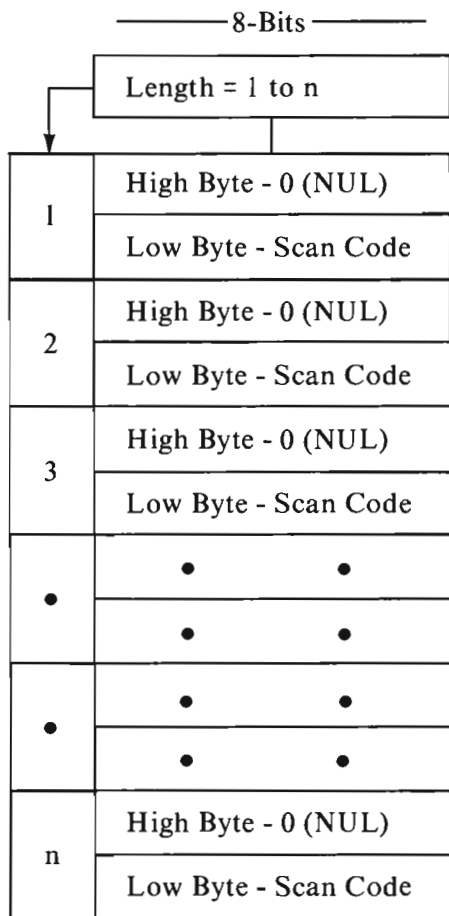
2. If the new translated scan code is less than hex 56, it is processed by BIOS as a keyboard scan-code and the same data is placed in the BIOS keyboard buffer.
3. If the translated scan-code is greater than hex 55 or the incoming scan-code is outside the range of the translate table, hex 40 is added, creating a new extended-scan-code. The new extended-scan-code is then placed in the BIOS keyboard buffer with the character code of 00(null). This utilizes the range hex 96 thru BE for scan codes hex 56 thru 7E respectively.

The default translate-table maps scan codes hex 56 thru 6A to existing keyboard-values. Scan codes hex 6B thru BE are mapped (by adding hex 40) to extended codes of hex AB thru FE, since these are out side the range of the default translate-table.

Users can modify Interrupt Hex 49 to address their own translate table if mapping differences are desired.

The translate table format is:

	Description
0	Length - The number of non-keyboard scan-codes that are mapped within the table (from 1 to n).
1 to n	Word with low-order byte representing the scan-code-mapped values relative to the input values in the range of hex 56 thru 7E.



Translate Table Format

With this architecture, all keyboard scan-codes can be intercepted thru Interrupt Hex 9 and all non-keyboard scan-codes can be intercepted thru Interrupt Hex 48.

The following is a chart showing the default values of the translate table in BIOS.

Length = 20 mapped values		
Input Scan Code	Mapped Value	Keyboard Character
86	72	(cursor up)
87	73	PgUp
88	77	(cursor right)
89	81	PgDn
90	80	(cursor down)
91	79	End
92	75	(cursor left)
93	71	Home
94	57	Space
95	28	Enter
96	17	W
97	18	E
98	31	S
99	45	X
100	44	Z
101	43	\
102	30	A
103	16	Q
104	15	Tab
105	1	Esc

Translate Table Default Values

Scan Codes (Hex)	Type of Scan Code
1 - 55	Normal Keyboard Scan Code (Make)
56 - 7E	Non-Keyboard Scan Code (Make)
81 - D5	Normal Keyboard Scan Code (Break)
D6 - FE	Non-Keyboard Scan Code (Break)
FF	Keyboard Buffer Full

Scan-Code Map

Notes:

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 by DS:BX and the number of bytes to be 'read' or 'written' is specified by CX. The actual 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 DS: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

AH Request Types

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

AH Cassette Status

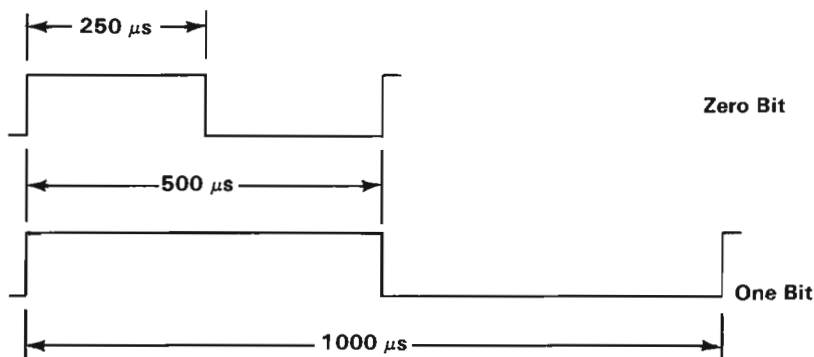
Cassette Write

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 'writes' the leader (256 bytes of all 1's) to the tape, 'writes' a synchronization bit (0), and then 'writes' a synchronization byte (ASCII character hex 16). Next, the routine 'writes' the number of data bytes 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 DS:BX.

The WRITE-BLOCK 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 millisecond for a 1 bit and 0.5 millisecond 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 byte '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. All 8259 interrupts are 'disabled' during cassette-write operations.



Cassette-Write Timing Chart

Cassette Read

The READ-BLOCK routine 'turns on' the cassette drive motor and then delays for approximately 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 DS:BX and BX is incremented by 1.

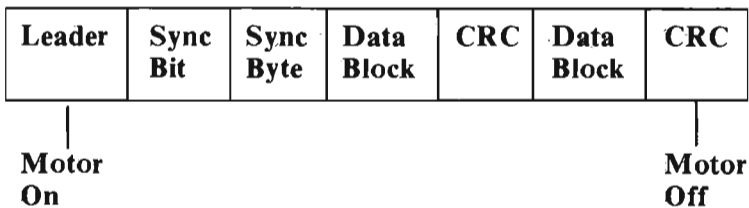
After each multiple of 256 data bytes is 'read', the CRC is 'read' and 'compared' to the CRC generated. If a CRC error is detected, the routine exits with the carry flag 'set' to indicate an error and the status of AH 'set' to hex 01. DX contains the number of bytes 'written' into memory.

All 8259 interrupts are 'disabled' during the cassette-'read' operations.

Data Record Architecture

The WRITE-BLOCK routine uses the following format to record a tape block onto a cassette tape:

(CASSETTE TAPE BLOCK)



Cassette Write-Block Format

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 Detection

Error detection 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. Upon 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. Also, the routine is exited on a CRC error.

Notes:



Appendixes

Contents

Appendix A. ROM BIOS LISTING	A-3
Appendix B. LOGIC DIAGRAMS	B-1
Appendix C. CHARACTERS, KEYSTROKES, and COLOR	C-1
Appendix D. UNIT SPECIFICATIONS	D-1
System Unit	D-1
Size:	D-1
Weight:	D-1
Transformer:	D-1
Environment:	D-1
Cordless Keyboard	D-2
Size:	D-2
Weight:	D-2
Optional Cable:	D-2
Diskette Drive	D-3
Size:	D-3
Weight:	D-3
Power:	D-3
Mechanical and Electrical	D-4
Color Display	D-5
Size:	D-5
Weight:	D-5
Heat Output:	D-5
Power Cables:	D-5
Graphics Printer	D-6
Size:	D-6
Weight:	D-6

Heat Output:	D-6
Power Cable:	D-6
Signal Cable:	D-6
Electrical:	D-6
Internal Modem	D-7
Power:	D-7
Interface	D-7
Compact Printer	D-8
Size	D-8
Weight	D-8
Heat Output	D-8
Power Cable	D-8
Signal Cable	D-8
Electrical	D-8

<CAVEAT EMPTR>:

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

EQUATES

```
= 0060 PORT_A EQU 50H ; 8255 PORT A ADDR
= 0038 CPUREG EQU 38H ; MASK FOR CPU REG BITS
= 0007 CRTREG EQU 7 ; MASK FOR CRT REG BITS
= 0061 PORT_B EQU 61H ; 8255 PORT B ADDR
= 0062 PORT_C EQU 52H ; 8255 PORT C ADDR
= 0063 CND_PORT EQU 63H
= 0089 MODE_8285 EQU 10001001B
= 0020 INTA00 EQU 20H ; 8259 PORT
= 0021 INTA01 EQU 21H ; 8259 PORT
= 0020 EOI EQU 20H
= 0040 TIMER EQU 40H
= 0043 TIM_CTL EQU 43H ; 8253 TIMER CONTROL PORT ADDR
= 0040 TIMERO EQU 40H ; 8253 TIMER/CNTER 0 PORT ADDR
= 0061 KB_CTL EQU 61H ; CONTROL BITS FOR KEYBOARD
= 03DA VGA_CTL EQU 3DAH ; VIDEO GATE ARRAY CONTROL PORT
= 00A0 NMI_PORT EQU 0A0H ; NMI CONTROL PORT
= 0080 PORT_BO EQU 080H
= 030F PAGREG EQU 030FH ; CRT/CPU PAGE REGISTER
= 0060 KBPORT EQU 060H ; KEYBOARD PORT
= 4000 DIAG_TABLE_PTR EQU 4000H
= 2000 MINI EQU 2000H
```

DISKETTE EQUATES

```
= 00F2 NEC_CTL EQU 0F2H ; CONTROL PORT FOR THE DISKETTE
= 0080 FDC_RESET EQU 80H ; RESETS THE NEC (FLOPPY DISK
; CONTROLLER). 0 RESETS,
; 1 RELEASES THE RESET
= 0020 WD_ENABLE EQU 20H ; ENABLES WATCH DOG TIMER IN NEC
= 0040 WD_STROBE EQU 40H ; STROBES WATCHDOG TIMER
= 0001 DRIVE_ENABLE EQU 01H ; SELECTS AND ENABLES DRIVE
= 00F4 NEC_STAT EQU 0F4H ; STATUS REGISTER FOR THE NEC
= 0020 BUSY_BIT EQU 20H ; BIT = 0 AT END OF EXECUTION PHASE
= 0040 DIO EQU 40H ; INDICATES DIRECTION OF TRANSFER
= 0080 ROM EQU 80H ; REQUEST FOR MASTER
= 00F5 NEC_DATA EQU 0F5H ; DATA PORT FOR THE NEC
```

8088 INTERRUPT LOCATIONS

```
0000 ABS0 SEGMENT AT 0
0008 ORG 2*4
0008 NMI_PTR LABEL WORD
000C ORG 3*4
000C INT3_PTR LABEL WORD
0014 ORG 5*4
0014 INT5_PTR LABEL WORD
0020 ORG B*4
0020 INT_PTR LABEL DWORD
0040 ORG 10H*4
0040 VIDEO_INT LABEL WORD
0070 ORG 1CH*4
0070 INT1C_PTR LABEL WORD
0074 ORG 10H*4
0074 PARM_PTR LABEL DWORD ; POINTER TO VIDEO.PARMS
0060 ORG 18H*4
0060 BASIC_PTR LABEL WORD ; ENTRY POINT FOR CASSETTE BASIC
0078 ORG 01EH*4
0078 DISK_POINTER LABEL DWORD ; INTERRUPT 1EH
007C ORG 01FH*4
007C EXT_PTR LABEL DWORD ; LOCATION OF POINTER
0110 ORG 044H*4
0110 CSET_PTR LABEL DWORD ; POINTER TO DOT PATTERNS
0120 ORG 048H*4
0120 KEY62_PTR LABEL WORD ; POINTER TO 62 KEY KEYBOARD CODE
0124 ORG 049H*4
0124 EXST LABEL WORD ; POINTER TO EXT. SCAN TABLE
0204 ORG 0B1H*4
0204 INT81 LABEL WORD
0208 ORG 0B2H*4
0208 INT82 LABEL WORD
0224 ORG 0BBH*4
0224 INT89 LABEL WORD
0400 ORG 400H
0400 DATA_AREA LABEL BYTE ; ABSOLUTE LOCATION OF DATA SEGMENT
0400 DATA_WORD LABEL WORD
7C00 ORG 7C00H
7C00 BOOT_LOCM LABEL FAR
7C00 ABS0 ENDS
```

```

;-----
; STACK -- USED DURING INITIALIZATION ONLY
;-----
0000          STACK SEGMENT AT 30H
0000 80 [      DW          128 DUP(?)
]

0100          TOS LABEL WORD
0100          STACK ENDS
;-----
; ROM BIOS DATA AREAS
;-----
0000          DATA SEGMENT AT 40H
0000 04 [      RS232_BASE DW          4 DUP(?) ; ADDRESSES OF RS232 ADAPTERS
]

000B 04 [      PRINTER_BASE DW          4 DUP(?) ; ADDRESSES OF PRINTERS
]

0010 7777     EQUIP_FLAG DW          ?          ; INSTALLED HARDWARE
0012 ??       KBD_ERR     DB          ?          ; COUNT OF KEYBOARD TRANSMIT ERRORS
0013 7777     MEMORY_SIZE DW          ?          ; USABLE MEMORY SIZE IN K BYTES
001B 7777     TRUE_MEM    DW          ?          ; REAL MEMORY SIZE IN K BYTES
;-----
; KEYBOARD DATA AREAS
;-----
0017 77       KB_FLAG     DB          ?
;----- SHIFT FLAG EQUATES WITHIN KB_FLAG
= 0040        CAPS_STATE   EQU        40H      ; CAPS LOCK STATE HAS BEEN TOGGLED
= 0020        NUM_STATE   EQU        20H      ; NUM LOCK STATE HAS BEEN TOGGLED
= 000B        ALT_SHIFT   EQU        08H      ; ALTERNATE SHIFT KEY DEPRESSED
= 0004        CTL_SHIFT   EQU        04H      ; CONTROL SHIFT KEY DEPRESSED
= 0002        LEFT_SHIFT  EQU        02H      ; LEFT SHIFT KEY DEPRESSED
= 0001        RIGHT_SHIFT EQU        01H      ; RIGHT SHIFT KEY DEPRESSED
001B 77       KB_FLAG_1   DB          ?          ; SECOND BYTE OF KEYBOARD STATUS
= 0080        INS_SHIFT   EQU        80H      ; INSERT KEY IS DEPRESSED
= 0040        CAPS_SHIFT  EQU        40H      ; CAPS LOCK KEY IS DEPRESSED
= 0020        NUM_SHIFT   EQU        20H      ; NUM LOCK KEY IS DEPRESSED
= 0010        SCROLL_SHIFT EQU        10H      ; SCROLL LOCK KEY IS DEPRESSED
= 000B        HOLD_STATE  EQU        08H      ; SUSPENS KEY HAS BEEN TOGGLED
= 0004        CLICK_ON    EQU        04H      ; INDICATES THAT AUDIO FEEDBACK IS
;----- ENABLED
= 0002        CLICK_SEQUENCE EQU        02H    ; OCCURRENCE OF ALT-CTRL-CAPSLOCK HAS
;----- OCCURED.
001B 77       ALT_INPUT   DB          ?          ; STORAGE FOR ALTERNATE KEYPAD
;----- ENTRY
001A 7777     BUFFER_HEAD DW          ?          ; POINTER TO HEAD OF KEYBOARD BUFF
001C 7777     BUFFER_TAIL DW          ?          ; POINTER TO TAIL OF KEYBOARD BUFF
001E 10 [     KB_BUFFER   DW          15 DUP(?) ; ROOM FOR 15 ENTRIES
]

;----- HEAD = TAIL INDICATES THAT THE BUFFER IS EMPTY
= 0045        NUM_KEY     EQU        69      ; SCAN CODE FOR NUMBER LOCK
= 0046        SCROLL_KEY  EQU        70      ; SCROLL LOCK KEY
= 0038        ALT_KEY     EQU        56      ; ALTERNATE SHIFT KEY SCAN CODE
= 001D        CTL_KEY     EQU        29      ; SCAN CODE FOR CONTROL KEY
= 003A        CAPS_KEY    EQU        58      ; SCAN CODE FOR SHIFT LOCK
= 002A        LEFT_KEY    EQU        42      ; SCAN CODE FOR LEFT SHIFT
= 0036        RIGHT_KEY   EQU        54      ; SCAN CODE FOR RIGHT SHIFT
= 0052        INS_KEY     EQU        82      ; SCAN CODE FOR INSERT KEY
= 0053        DEL_KEY     EQU        83      ; SCAN CODE FOR DELETE KEY
;-----
; DISKETTE DATA AREAS
;-----
003E 77       SEEK_STATUS DB          ?          ; DRIVE RECALIBRATION STATUS
;----- BIT 0 = DRIVE NEEDS RECAL BEFORE
003F 77       MOTOR_STATUS DB          ?          ; MOTOR STATUS
;----- NEXT SEEK IF BIT IS = 0
;----- BIT 0 = DRIVE 0 IS CURRENTLY
;----- RUNNING
0040 77       MOTOR_COUNT DB          ?          ; TIME OUT COUNTER FOR DRIVE
;----- TURN OFF
= 0025        MOTOR_WAIT  EQU        37      ; 2 SECS OF COUNTS FOR MOTOR
;----- TURN OFF
0041 77       DISKETTE_STATUS DB          ?      ; RETURN CODE STATUS BYTE
= 00B0        TIME_OUT    EQU        80H      ; ATTACHMENT FAILED TO RESPOND
= 0040        BAD_SEEK    EQU        40H      ; SEEK OPERATION FAILED
= 0020        BAD_NEC     EQU        20H      ; NEC CONTROLLER HAS FAILED
= 0010        BAD_CRC     EQU        10H      ; BAD CRC ON DISKETTE READ
= 0009        DMA_BOUNDARY EQU        09H      ; ATTEMPT TO DMA ACROSS 64K
;----- BOUNDARY
= 000B        BAD_DNA     EQU        08H      ; DMA OVERRUN ON OPERATION
= 0004        RECORD_NOT_FND EQU        04H    ; REQUESTED SECTOR NOT FOUND
= 0003        WRITE_PROTECT EQU        03H    ; WRITE ATTEMPTED ON WRITE
;----- PROTECTED DISK
= 0002        BAD_ADDR_MARK EQU        02H    ; ADDRESS MARK NOT FOUND
= 0001        BAD_CMD     EQU        01H      ; BAD COMMAND GIVEN TO DISKETTE I/O
0042 07 [     NEC_STATUS   DB          7 DUP(?) ; STATUS BYTES FROM NEC
]

= 0020        SEEK_END    EQU        20H
= 012C        THRESHOLD   EQU        300     ; NUMBER OF TIMER-0 TICKS TILL
;----- ENABLE
= 00AF        PARM0       EQU        0AFH    ; PARAMETER 0 IN THE DISK_PARM
;----- TABLE
= 0003        PARM1       EQU        3      ; PARAMETER 1
= 0019        PARM9       EQU        25     ; PARAMETER 9
= 0004        PARM10      EQU        4      ; PARAMETER 10

```



```

-----
VIDE DISPLAY DATA AREA
-----
0049 ?? CRT_MODE DB ? ; CURRENT CRT MODE
004A ???? CRT_COLS DW ? ; NUMBER OF COLUMNS ON SCREEN
004C ???? CRT_LEN DW ? ; LENGTH OF REGEN IN BYTES
004E ???? CRT_START DW ? ; STARTING ADDRESS IN REGEN BUFFER
0050 0B E ???? CURSOR_POSN DW 8 DUP(?) ; CURSOR FOR EACH OF UP TO 8 PAGES
                                     ]

0060 ???? CURSOR_MODE DW ? ; CURRENT CURSOR MODE SETTING
0062 ?? ACTIVE_PAGE DB ? ; CURRENT PAGE BEING DISPLAYED
0063 ???? ADDR_6B4B DW ? ; BASE ADDRESS FOR ACTIVE DISPLAY
                                     ]
0065 ?? CRT_MODE_SET DB ? ; CARD
0066 ?? CRT_PALLETTE DB ? ; CURRENT SETTING OF THE
                                     ]
                                     ]
                                     ]
CASSETTE DATA AREA
-----
0067 ???? EDGE_CNT DW 7 ; TIME COUNT AT DATA EDGE
0069 ???? CRC_REG DW ? ; CRC REGISTER
006B ?? LAST_VAL DB ? ; LAST INPUT VALUE
                                     ]
-----
TIMER DATA AREA
-----
006C ???? TIMER_LOW DW ? ; LOW WORD OF TIMER COUNT
006E ???? TIMER_HIGH DW ? ; HIGH WORD OF TIMER COUNT
0070 ?? TIMER_OFI DB ? ; TIMER HAS ROLLED OVER SINCE LAST
                                     ]
                                     ]
                                     ]
SYSTEM DATA AREA
-----
0071 ?? BIOS_BREAK DB ? ; BIT 7=1 IF BREAK KEY HAS BEEN HIT
0072 ???? RESET_FLAG DW ? ; WORD=1234H IF KEYBOARD RESET
                                     ]
                                     ]
EXTRA DIBKETTE DATA AREAS
-----
0074 ?? TRACK0 DB ?
0075 ?? TRACK1 DB ?
0076 ?? TRACK2 DB ?
0077 ?? DB ?
                                     ]
-----
PRINTER AND RS232 TIME-OUT VARIABLES
-----
007B 04 E ?? PRINT_TIM_OUT DB 4 DUP(?)
                                     ]
007C 04 E ?? RB232_TIM_OUT DB 4 DUP(?)
                                     ]
                                     ]
-----
ADDITIONAL KEYBOARD DATA AREA
-----
0080 ???? BUFFER_START DW ?
0082 ???? BUFFER_END DW ?
0084 ?? INTR_FLAG DB ? ; FLAG TO INDICATE AN INTERRUPT
                                     ]
                                     ]
62 KEY KEYBOARD DATA AREA
-----
0085 ?? CUR_CHAR DB ? ; CURRENT CHARACTER FOR TYRANATIC
0086 ?? VAR_DELAY DB ? ; DETERMINES WHEN INITIAL DELAY IS
                                     ]
                                     ]
= 000F DELAY_RATE EQU 0FH ; INCREASES INITIAL DELAY
0087 ?? CUR_FUNC DB ? ; CURRENT FUNCTION
0088 ?? KB_FLAG_2 DB ? ; 3RD BYTE OF KEYBOARD FLAGS
= 0004 RANGE EQU 4 ; NUMBER OF POSITIONS TO SHIFT
                                     ]
                                     ]
DISPLAY
-----
BIT ASSIGNMETS FOR KB_FLAG_2
-----
= 0080 FN_FLAG EQU 80H
= 0040 FN_BREAK EQU 40H
= 0020 FN_PENDING EQU 20H
= 0010 FN_LOCK EQU 10H
= 0008 TYPE_OFF EQU 08H
= 0004 HALF_RATE EQU 04H
= 0002 INIT_DELAY EQU 02H
= 0001 PUTCHAR EQU 01H
0089 ?? HORZ_POS DB ? ; CURRENT VALUE OF HORIZONTAL
                                     ]
                                     ]
008A ?? PAGDAT DB ? ; START PARM
008B DATA ENDS ; IMAGE OF DATA WRITTEN TO PAGREG
                                     ]
-----
EXTRA DATA AREA
-----
0000 XNDATA SEGMENT AT 50H
0000 ?? STATUS_BYTE DB ?
; THE FOLLOWING AREA IS USED ONLY DURING DIAGNOSTICS
; (POST AND ROM RESIDENT)
0001 ?? DCP_MENU_PAGE DB ? ; TO CURRENT PAGE FOR DIAG. MENU
0002 ???? DCP_ROW_COL DW ? ; CURRENT ROW/COLUMN COORDINATES
                                     ]
0004 ?? WRAP_FLAG DB ? ; FOR DIAG MENU
                                     ]
                                     ]
INTERNAL/EXTERNAL 8250 WRAP
INDICATOR

```

```

0005 ?? MFG_TST      OB  ? ; INITIALIZATION FLAG
0006 ???? MEM_TOT      DW  ? ; WORD EQUIV. TO HIGHEST SEGMENT IN
; MEMORY
0008 ???? MEM_00NES    OW  ? ; CURRENT SEGMENT VALUE FOR
; BACKGROUND MEM TEST
000A ???? MEM_DONE0    DW  ? ; CURRENT OFFSET VALUE FOR
; BACKGROUND MEM TEST
000C ???? INT1C0      OW  ? ; SAVE AREA FOR INTERRUPT 1C
; ROUTINE
000E ???? INTICS      DW  ?
0010 ?? MENU_UP      DB  ? ; FLAG TO INDICATE WHETHER MENU IS
; ON SCREEN (FF=YES, 0=NO)
0011 ?? DONE12B     OB  ? ; COUNTER TO KEEP TRACK OF 12B BYTE
; BLOCKS TESTED BY BGEM
0012 ???? KBOONE      DW  ? ; TOTAL K OF MEMORY THAT HAS BEEN
; TESTED BY BACKGROUND MEM TEST
;-----
; POST DATA AREA
0014 ???? IO_ROM_INIT  DW  ? ; POINTER TO OPTIONAL I/O ROM INIT
; ROUTINE
0016 ???? ID_ROM_SEG  DW  ? ; POINTER TO IO ROM SEGMENT
0018 ?? POST_ERR     DB  ? ; FLAG TO INDICATE ERROR OCCURRED
; DURING POST
0019 D9 [ ?? MO0EM_BUFFER DB  9 DUP(?) ; MO0EM RESPONSE BUFFER
; ]
;
; (MAX 9 CHARS)
0022 ???? MFG_RTN     DW  ? ; POINTER TO MFG. OUTPUT ROUTINE
0024 ????
;-----
; SERIAL PRINTER DATA
;-----
0026 ???? SP_FLAG      DW  ?
0028 ?? SP_CHAR       DB  ?
;
; THE FOLLOWING SIX ENTRIES ARE
; DATA PERTAINING TO NEW STICK
0029 ???? NEW_STICK_DATA DW  ? ; RIGHT STICK DELAY
002B ????           DW  ? ; RIGHT BUTTON A DELAY
002D ????           DW  ? ; RIGHT BUTTON B DELAY
002F ????           OW  ? ; LEFT STICK DELAY
0031 ????           DW  ? ; LEFT BUTTON A DELAY
0033 ????           DW  ? ; LEFT BUTTON B DELAY
0035 ????           DW  ? ; RIGHT STICK LOCATION
0037 ????           DW  ? ; UNUSED
0039 ????           DW  ? ; UNUSED
003B ????           DW  ? ; LEFT STICK POSITION
003D
;-----
; XXDATA ENDS
;-----
; DISKETTE DATA AREA
;-----
0000 DKDATA SEGMENT AT 60H
0000 ?? NUM_DRIVE    DB  ?
0001 ?? DUAL         DB  ?
0002 ?? OPERATION    DB  ?
0003 ?? DRIVE        DB  ?
0004 ?? TRACK        DB  ?
0005 ?? HEAD         DB  ?
0006 ?? SECTOR       DB  ?
0007 ?? NUM_SECTOR   DB  ?
0008 ?? SEC          DB  ?
; FDRMAT ID
0009 08 [ TK_HD_SC     DB  B DUP(0,0,0,0) ; TRACK, HEAD, SECTOR, NUM OF
; DO
; DO
; DO
; ]
; SECTOR
; BUFFER FOR READ AND WRITE OPERATION
= 0200
0029 0200 [ DK_BUF_LEN EQU 512 ; 512 BYTES/SECTOR
; READ_BUF DB DK_BUF_LEN DUP(0)
; ]
;
0229 0100 [ WRITE_BUF DB (DK_BUF_LEN/2) DUP(6DH,0BH)
; 6D
; 0B
; ]
;
; INFO FLAGS
042B ?? REQUEST_IN  DB  ? ; SELECTION CHARACTER
042A ?? DK_EXISTED DB  ?
042B ?? DK_FLAG     DB  ?
042C ???? RAN_NUM    DW  ?
042E ???? SEED       OW  ?
; SPEED TEST VARIABLES
0430 ???? DK_SPEED   OW  ?
0432 ???? TIM_1      DW  ?
0434 ???? TIM_L_1    DW  ?
0436 ???? TIM_2      DW  ?
0438 ???? TIM_L_2    DW  ?
043A ???? FRACT_H    DW  ?
043C ???? FRACT_L    DW  ?
043E ???? PART_CYCLE DW  ?
0440 ???? WHOLE_CYCLE DW  ?
0442 ???? HALF_CYCLE DW  ?

```

```

; ERROR PARAMETERS
0444 ?? DK_ER_OCCURED DB ? ; ERROR HAS OCCURRED
0445 ?? DK_ER_L1 DB ? ; CUSTOMER ERROR LEVEL
0446 ?? DK_ER_L2 DB ? ; SERVICE ERROR LEVEL
0447 ?? ER_STATUS_BYTE DB ? ; STATUS BYTE RETURN FROM INT 13H
; LANGUAGE TABLE
044B ?? LANG_BYTE DB ? ; PORT B0 TO DETERMINE WHICH
; LANGAGE TO USE
0449 DKDATA ENDS
;-----
; VIDEO DISPLAY BUFFER
;-----
0000 VIDEO_RAM SEGMENT AT 0B000H
0000 4000 I DB 16384 DUP(?)
;-----
0000 VIDEO_RAM ENDS
;-----
; ROM RESIDENT CODE
;-----
0000 CODE SEGMENT PAGE
ASSUME CS:CODE,DS:ABS0,ES:NOTHING,SS:STACK
0000 31 35 30 34 30 33 DB '1504036 COPR. 18M 1981,1983' ; COPYRIGHT NOTICE
36 20 43 4F 50 52
2E 20 49 42 4D 20
31 39 3B 31 2C 31
38 38 33
;-----
001B 0149 R Z1 DW L12 ; RETURN POINTERS FOR RTMS CALLED
001D 0157 R DW L14 ; BEFORE STACK INITIALIZED
001F 016D R DW L16
0021 0186 R DW L18
0023 018A R DW L24
0025 20 4B 42 F3B DB 'KB'
002B 0A47 R EX_0 DB OFFSET EBO
002A 0A47 R DW OFFSET EBO
002C 0A8B R DW OFFSET TOTLTP0
002E 0A84 R EX1 DW OFFSET M01
;-----
; MESSAGE AREA FOR POST
;-----
0030 45 52 52 4F 52 ERROR_ERR DB 'ERROR' ; GENERAL ERROR PROMPT
0035 41 MEM_ERR DB 'A' ; MEMORY ERROR
003B 42 KEY_ERR DB 'B' ; KEYBOARD ERROR MSG
0037 43 CASS_ERR DB 'C' ; CASSETTE ERROR MESSAGE
0038 44 COM1_ERR DB 'D' ; ON-BOARD SERIAL PORT ERR. MSG
0039 45 COM2_ERR DB 'E' ; SERIAL PORTION OF MODEM ERROR
003A 46 ROM_ERR DB 'F' ; OPTIONAL GENERIC BIOS ROM ERROR
003B 47 CART_ERR DB 'G' ; CARTRIDGE ERROR
003C 48 DISK_ERR DB 'H' ; DISKETTE ERR
;-----
0030 F4 LABEL WORD ; PRINTER SOURCE TABLE
0030 037B DW 37BH
003F 027B DW 27BH
F4E LABEL WORD
IMASKS LABEL BYTE ; INTERRUPT MASKS FOR 8259
0041 EF ; INTERRUPT CONTROLLER
0042 F7 DB 0EFH ; MODEM INTR MASK
DB 0F7H ; SERIAL PRINTER INTR MASK
;-----
; SETUP
;-----
; DISABLE NMI, MASKABLE INTS.
; SOUND CHIP, AND VIDEO.
; TURN DRIVE 0 MOTOR OFF
;-----
0043 RESET ASSUME CS:CODE,DS:ABS0,ES:NOTHING,SS:STACK
0043 80 00 LABEL FAR
0045 E6 A0 MOV AL,0
0047 FE CB DEC AL ; 01SABLES NMI
0049 E6 10 OUT 10H,AL ; SEND FF TO MFG_TESTER
004B E4 A0 IN AL,0A0H
004D FA CLI ; RESET NMI F/F
;-----
004E 8B 10BF MOV AX,10BFH ; DISABLES MASKABLE INTERRUPTS
; DISABLE ATTENUATION IN SOUND CHIP
; REG ADDRESS IN AH, ATTENUATOR OFF
; IN AL
0051 8A 00C0 MOV DX,00C0H ; ADDRESS OF SOUND CHIP
0054 89 0004 MOV CX,4 ; 4 ATTENUATORS TO DISABLE
0057 0A C4 L1: OR AL,AH ; COMBINE REG ADDRESS AND DATA
0059 EE OUT DX,AL
005A 60 C4 20 ADD AH,20H ; POINT TO NEXT REG
005D E2 FB LOOP L1
005F 80 A0 MOV AL,WD_ENABLE+FDC_RESET ; TURN DRIVE 0 MOTOR OFF,
; ENABLE TIMER
;-----
0061 E6 F2 OUT 0F2H,AL
0063 8A 030A MOV DX,VGA_CTL ; VIDEO GATE ARRAY CONTROL
0068 EC IN AL,DX ; SYNC VGA TO ACCEPT REG
0067 80 04 MOV AL,4 ; SET VGA RESET REG
0069 EE OUT DX,AL ; SELECT IT
006A 80 01 MOV AL,1 ; SET ASYNC RESET
006C EE OUT DX,AL ; RESET VIDEO GATE ARRAY
;-----
; TEST 1
;-----
; BOBB PROCESSOR TEST
; DESCRIPTION
; VERIFY BOBB FLAGS, REGISTERS
; AND CONDITIONAL JUMPS
;-----
; MFG. ERROR CODE 0001H
;-----

```

```

006D 84 D5      MOV     AH,005H      ; SET SF, CF, ZF, AND AF FLAGS ON
006F 9E          SAHF
0070 73 4C      JNC     L4           ; GO TO ERR ROUTINE IF CF NOT SET
0072 75 4A      JNZ     L4           ; GO TO ERR ROUTINE IF ZF NOT SET
0074 78 48      JNP     L4           ; GO TO ERR ROUTINE IF PF NOT SET
0076 79 46      JNS     L4           ; GO TO ERR ROUTINE IF SF NOT SET
0078 9F          LAHF
0079 81 05      MOV     CL,5         ; LOAD CNT REG WITH SHIFT CNT
0078 D2 EC      SHR     AH,CL        ; SHIFT AF INTO CARRY BIT POS
007D 73 3F      JNC     L4           ; GO TO ERR ROUTINE IF AF NOT SET
007F 80 40      MOV     AL,40H       ; SET THE OF FLAG ON
0081 D0 E0      SHL     AL,1         ; SETUP FOR TESTING
0083 71 39      JMO     L4           ; GO TO ERR ROUTINE IF OF NOT SET
0085 32 E4      XOR     AH,AH        ; SET AH = 0
0087 9E          SAHF
0088 76 34      JBE     L4           ; CLEAR SF, CF, ZF, AND PF
                                ; GO TO ERR ROUTINE IF CF ON
                                ; GO TO ERR ROUTINE IF ZF ON
                                ; GO TO ERR ROUTINE IF SF ON
008A 78 32      JS      L4           ; LOAD FLAG IMAGE TO AH
008C 7A 30      JP      L4           ; LOAD CNT REG WITH SHIFT CNT
008E 9F          LAHF
008F B1 05      MOV     CL,5         ; LOAD CNT REG WITH SHIFT CNT
0091 D2 EC      SHR     AH,CL        ; SHIFT 'AF' INTO CARRY BIT POS
0093 72 29      JC      L4           ; GO TO ERR ROUTINE IF ON
0095 D0 E4      SHL     AH,1         ; CHECK THAT 'OF' IS CLEAR
0097 70 25      JO      L4           ; GO TO ERR ROUTINE IF ON

```

```

;----- READ/WRITE THE 8088 GENERAL AND SEGMENTATION REGISTERS
; WITH ALL ONE'S AND ZEROES'S.

```

```

0099 8B FFFF     MOV     AX,0FFFFH   ; SETUP ONE'S PATTERN IN AX
009C F9        STC
009D 8E D8     L2:    MOV     DS,AX      ; WRITE PATTERN TO ALL REGS
009F 8C D8     MOV     BX,DS
00A1 8E C3     MOV     ES,BX
00A3 8C C1     MOV     CX,ES
00A5 8E D1     MOV     SS,CX
00A7 8C D2     MOV     DX,SS
00A9 8B E2     MOV     SP,DX
00AB 8B EC     MOV     BP,SP
00AD 8B F5     MOV     SI,BP
00AF 8B FE     MOV     DI,SI
00B1 73 07     JNC     L3
00B3 33 C7     XOR     AX,DI       ; PATTERN MAKE IT THRU ALL REGS
00B5 75 07     JNZ     L4         ; NO - GO TO ERR ROUTINE
00B7 F8        CLC
00B9 EB E3     JMP     L2
00BA 0B C7     L3:    OR      AX,DI       ; ZERO PATTERN MAKE IT THRU?
00BC 74 0C     JZ      L5         ; YEB - GO TO NEXT TEST
00BE 8A 0010   L4:    MOV     OX,0010H   ; HANDLE ERROR
00C1 80 00     MOV     AL,0
00C3 EE       OUT     DX,AL      ; ERROR 0001
00C4 42       INC     OX
00C5 EE       OUT     DX,AL
00C6 FE C0   INC     AL
00C8 EE       OUT     AL
00C9 F4       OUT     OX,AL
00CA 00CA     HLT
                                ; L5:

```

```

;----- TEST 2 -----
; DESCRIPTION
; B255 INITIALIZATION AND TEST
; FIRST INITIALIZE B255 PROG.
; PERIPHERAL INTERFACE. PORTS A#B
; ARE LATCHED OUTPUT
; BUFFERS. C IS INPUT.
; MFG. ERR. CODE =0002H
;-----

```

```

00CA 80 FE     MOV     AL,0FEH     ; SEND FE TO MFG
00CC E5 10   OUT     10H,AL
00CE 80 89   MOV     AL,MODE_B255
00D0 E6 63   OUT     CMD_PORT,AL ; CONFIGURES I/O PORTS
00D2 28 C0   SUB     AX,AX       ; TEST PATTERN SEED = 0000
00D4 8A C4   L6:    MOV     AL,AH
00D6 E6 60   OUT     PORT_A,AL   ; WRITE PATTERN TO PORT A
00D8 E4 60   IN      AL,PORT_A   ; READ PATTERN FROM PORT A
00DA E6 61   OUT     PORT_B,AL   ; WRITE PATTERN TO PORT B
00DC E4 61   IN      AL,PORT_B   ; READ OUTPUT PORT
00DE 3A C4   CMP     AL,AH       ; DATA AS EXPECTED?
00E0 75 06   JNE     L7         ; IF NOT, SOMETHING IS WRONG
00E2 FE C4   INC     AH          ; MAKE NEW DATA PATTERN
00E4 75 EE   JNZ     L8         ; LOOP TILL 255 PATTERNS DONE
00E6 EB 05   JMP     SHORT L8    ; CONTINUE IF DONE
00E8 B3 02   L7:    MOV     BL,02H     ; SET ERROR FLAG (BH=00 NOW)
00EA E9 09BC R JMP     E_MSG      ; GO ERROR ROUTINE
00ED 32 C0   L8:    XOR     AL,AL
00EF E5 80   OUT     KBPORT,AL  ; CLEAR KB PORT
00F1 E4 62   IN      AL,PORT_C
00F3 24 08   AND     AL,0001000B ; 64K CARD PRESENT?
00F5 80 1B   MOV     AL,1BH     ; PORT SETTING FOR 64K SYS
00F7 75 02   JNZ     L9
00F9 80 3F   MOV     AL,3FH     ; PORT SETTING FOR 128K SYS
00FB BA 03DF L9:    MOV     DX,PAGREG
00FE EE       OUT     DX,AL
00FF 80 0D   MOV     AL,0001101B ; INITIALIZE OUTPUT PORTS
0101 E6 61   OUT     PORT_B,AL

```

PART 3

SET UP VIDEO GATE ARRAY AND 6845 TO GET MEMORY WORKING

```

0103 B0 FD      MOV     AL, OFDH
0105 E6 10     OUT     10H, AL
0107 BA 03D4   MOV     DX, 0304H
010A BB FOA4 R  MOV     BX, OFFSET VIDEO_PARMS ; SET ADDRESS OF 6845
010D B9 0010 R  MOV     CX, M0040 ; SET PARM LEN
0111 32 EA     XOR     AH, AH ; AH IS REG #
0113 8A C4     L10:   MOV     AL, AH ; GET 6845 REG #
0115 EE        OUT     DX, AL
0116 42        INC     DX
0117 FE C4     INC     AH ; POINT TO DATA PORT
0119 2E: BA 07 MOV     AL, CS: [BX] ; NEXT REG VALUE
011C EE        OUT     DX, AL ; GET TABLE VALUE
011D 43        INC     BX ; OUT TO CHIP
011E 4A        DEC     DX ; NEXT IN TABLE
011F E2 F2     LOOP   L10 ; BACK TO POINTER REG

; START VGA WITHOUT VIDEO ENABLED
0121 BA 03DA   MOV     DX, VGA_CTL ; SET ADDRESS OF VGA
0124 EC        IN     AL, DX ; BE SURE ADDR/DATA FLAG IS
; IN THE PROPER STATE
0125 B9 0005   MOV     CX, 5 ; # OF REGISTERS
012B 32 EA     XOR     AH, AH ; AH IS REG COUNTER
012A 8A C4     L11:   MOV     AL, AH ; GET REG #
012C EE        OUT     DX, AL ; SELECT IT
012D 32 C0     XOR     AL, AL ; SET ZERO FOR DATA
012F EE        OUT     DX, AL
0130 FE C4     INC     AH ; NEXT REG
0132 E2 F6     LOOP   L11

```

TEST 4

PLANAR BOARD ROS CHECKSUM TEST

DESCRIPTION

A CHECKSUM TEST IS DONE FOR EACH ROS

MODULE ON THE PLANAR BOARD TO

MFG ERROR CODE =0003H MODULE AT ADDRESS :

F000: 0000 ERROR

0004H MODULE AT ADDRESS :

F800: 0000 ERROR

```

0134 B0 FC      MOV     AL, OFCH
0136 E6 10     OUT     10H, AL ; MFG OUT=FC
0138 33 F6     ; CHECK MODULE AT F000:0 (LENGTH 32K)
XOR     SI, SI ; FIRST BYTE
013A 8C C8     MOV     AX, CS ; SET UP STACK SEGMENT
013C 8E D0     MOV     SS, AX
013E 8E D8     MOV     DS, AX ; LOAD DS WITH SEGMENT OF ADDRESS
; SPACE OF BIOS/BASIC
0140 B9 8000   MOV     CX, 8000H ; NUMBER OF BYTES TO BE TESTED, 32K
0143 8C 0018 R MOV     SP, OFFSET Z1 ; SET UP STACK POINTER SO THAT
; RETURN WILL COME HERE
0146 E9 FEEB R JMP     ROS_CHECKSUM ; JUMP TO ROUTINE WHICH PERFORMS
; CRC CHECK
0149 74 06     L12:   JZ     L13 ; MODULE AT F000:0 OK, GO CHECK
; OTHER MODULE AT F000:8000
014B BB 0003   MOV     BX, 0003H ; SET ERROR CODE
014E E9 09BC R JMP     E_MSG ; INDICATE ERROR
0151 B9 8000   L13:   MOV     CX, 8000H ; LOAD COUNT (SI POINTING TO START
0154 E9 FEEB R JMP     ROS_CHECKSUM ; OF NEXT MODULE AT THIS POINT)
0157 74 06     L14:   JZ     L15 ; PROCEED IF NO ERROR
0159 BB 0004   MOV     BX, 0004H ; INDICATE ERROR
015C E9 09BC R JMP     E_MSG
015F
L15:

```

TEST 5

BASE 2K READ/WRITE STORAGE TEST

DESCRIPTION

WRITE/READ/VERIFY DATA PATTERNS

AA, 55, AND 00 TO 1ST 2K OF STORAGE

AND THE 2K JUST BELOW 64K (CRT BUFFER)

VERIFY STORAGE ADDRESSABILITY.

ON EXIT SET CRT PAGE TO 3. SET

TEMPORARY STACK ALSO.

MFG. ERROR CODE 04XX FOR SYSTEM BOARD MEM.

05XX FOR 64K ATTRIB. CD. MEM

06XX FOR ERRORS IN BOTH

(XX= ERROR BITS)

```

015F B0 FB      MOV     AL, OFBH
0161 E6 10     OUT     10H, AL ; SET MFG FLAG=FB
0163 B9 0400   MOV     CX, 0400H ; SET FOR 1K WORDS, 2K BYTES
0166 33 C0     XOR     AX, AX
0168 8E C0     MOV     ES, AX ; LOAD ES WITH 0000 SEGMENT
016A E9 0B59 R JNP     PODSTG
016D 75 19     L16:   JNZ    L20 ; BAD STORAGE FOUND
016F 90 FA     MOV     AL, OFAH ; MFG OUT=FA
0171 E6 10     OUT     10H, AL
0173 B9 0400   MOV     CX, 400H ; 1024 WORDS TO BE TESTED IN THE
; REGEN BUFFER
0176 E4 60     IN     AL, PORT_A ; WHERE IS THE REGEN BUFFER?
0178 3C 1B     CMP     AL, 1BH ; TOP OF 64K?
017A BB 0F80   MOV     AX, OF80H ; SET POINTER TO THERE IF IT IS
017D 74 02     JE     L18 ;
017F B4 1F     MOV     AH, 1FH ; OR SET POINTER TO TOP OF 128K
0181 8E C0     L18:   MOV     ES, AX
0183 E9 0B59 R JMP     PODSTG
0186 74 23     L19:   JZ     L23

```

```

0188 87 04
018A E4 62
018C 24 08
018E 74 06
0190 8A D9
0192 0A D0
0194 EB 12
0196 90 FC 02
0199 8A D9
019B 74 08
019D FE C7
019F 0A D0
01A1 80 FC 01
01A4 74 02
01A6 FE C7
01AB E9 09BC R
01AB 80 F9
01AD E6 10
01AF 89 0400
01B2 89 8880
01B5 8E C0
01B7 E9 0859 R
01BA 74 06
01BC 8B 0005
01BF E9 09BC R
01C2 8B 0030
01C5 8E D0
01C7 8C 0100 R
01CA 33 C0
01CC 8E D8
01CE C7 06 0462 R 0007
01D4 89 0040
01D7 E4 62
01D9 24 08
01DB 80 1B
01DD 75 05
01DF 83 C3 40
01E2 80 3F
01E4 99 1E 0415 R
01EB A2 048A R
L20: MOV BH,04H ; ERROR 04...
      IN AL,PORT_C ; GET CONFIG BITS
      AND AL,00001000H ; TEST FOR ATTRIB CARD PRESENT
      JZ L21 ; WORRY ABOUT ODD/EVEN IF IT IS
      MOV BL,CL
      OR BL,CH ; COMBINE ERROR BITS IF IT ISN'T
      JMP SHORT L22
L21: CMP AH,02 ; EVEN BYTE ERROR? ERR 04XX
      MOV BL,CL
      JE L22
      INC SH ; MAKE INTO 05XX ERR
      OR BL,CH ; MOVE AND POSSIBLY COMBINE
      ; ERROR BITS
      ; ODD BYTE ERROR
      ; MUST HAVE BEEN BOTH
      ; - MAKE INTO 06XX
      ; JUMP TO ERROR OUTPUT ROUTINE
L22: JMP E_MSG
      ; RETEST HIGH 2K USING B8000 ADDRESS PATH
L23: MOV AL,0F9H ; MFG OUT =F9
      OUT 10H,AL
      MOV CX,0400H ; 1K WORDS
      MOV AX,0880H ; POINT TO AREA JUST TESTED WITH
      ; DIRECT ADDRESSING
      MOV ES,AX
      JMP PODSTG
L24: JZ L25
      MOV BX,0005H ; ERROR 0005
      JMP E_MSG
      ;----- SETUP STACK SEG AND SP
L25: MOV AX,0030H ; GET STACK VALUE
      MOV SS,AX ; SET THE STACK UP
      MOV SP,OFFSET TOS ; STACK IS READY TO GO
      XOR AX,AX ; SET UP DATA SEG
      MOV DS,AX
      ;----- SETUP CRT PAGE
      MOV DATA_WORD[ACTIVE_PAGE-DATA],07
      ;----- SET PRELIMINARY MEMORY SIZE WORD
      MOV BX,64
      IN AL,PORT_C ;
      AND AL,09H ; 64K CARD PRESENT?
      MOV AL,1BH ; PORT SETTING FOR 64K SYSTEM
      JNZ L26 ; SET TO 64K IF NOT
      ADD BX,64 ; ELSE SET FOR 128K
      MOV AL,3FH ; PORT SETTING FOR 128K SYSTEM
L26: MOV DATA_WORD[TRUE_MEM-DATA],BX
      MOV DATA_AREA[PAGDAT-DATA],AL

```

PART 6

INTERRUPTS
DESCRIPTION

32 INTERRUPTS ARE INITIALIZED TO POINT TO A
DUMMY HANDLER. THE BIOS INTERRUPTS ARE LOADED.
DIAGNOSTIC INTERRUPTS ARE LOADED
SYSTEM CONFIGURATION WORD IS PUT IN MEMORY.
THE DUMMY INTERRUPT HANDLER RESIDES HERE.

```

ASSUME DS:BXDATA
01E9 8B ---- R      MOV AX,AXDATA
01EE 8E D8          MOV DS,AX
01F0 C6 06 0005 R F8 MOV MFG_TST,0F8H ; SET UP MFG CHECKPOINT FROM THIS
                        ; POINT
01F5 E6 E6D8 R     CALL MFG_UP ; UPDATE MFG CHECKPOINT
01F8 C7 06 0022 R 0A61 R MOV MFG_RTN,OFFSET MFG_OUT
01FE 8C C9          MOV AX,C6
0200 A3 0024 R     MOV MFG_RTN+2,AX ; SET DOUBLEWORD POINTER TO MFG.
                        ; ERROR OUTPUT ROUTINE 90 DIAG.
                        ; DON'T HAVE TO DUPLICATE CODE
ASSUME CS:CODE,DS:ABS0
0203 8B 0000       MOV AX,0
0206 8E D8         MOV DS,AX
;----- SET UP THE INTERRUPT VECTORS TO TEMP INTERRUPT
0208 8B 00FF       MOV CX,255 ; FILL ALL INTERRUPTS
0209 2B FF         SUB DI,D1 ; FIRST INTERRUPT LOCATION IS 0000
020D 8E C7         MOV ES,D1 ; SET ES=0000 ALSO
020F 8B FB16 R    D3: MOV AX,OFFSET D11 ; MOVE ADDR OF INTR PROC TO TBL
0212 AB           STOSW
0213 8C CB         MOV AX,C5 ; GET ADDR OF INTR PROC SEG
0215 AB           STOSW
0216 E2 F7        LOOP D3 ; VECTBLO
0218 C7 06 0124 R 109D R MOV EXST,OFFSET EXTAB ; SET UP EXT. SCAN TABLE
; SET UP BIOS INTERRUPTS
021E BF 0040 R     MOV DI,OFFSET VIDEO_INT ; SET UP VIDEO INT
0221 0E           PUSH CS
0222 1F           POP DS ; PLACE CS IN DS
0223 8E FF03 R     MOV SI,OFFSET VECTOR_TABLE+16
0226 89 0010       MOV CX,16
0229 A5           D4: MOVSW ; MOVE INTERRUPT VECTOR TO LOW
                        ; MEMORY
022A 47           INC D1
022B 47           INC D1 ; POINT TO NEXT VECTOR ENTRY
022C E2 FB        LOOP D4 ; REPEAT FOR ALL 16 BIOS INTERRUPTS
; SET UP DIAGNOSTIC INTERRUPTS
022E BF 0200       MOV DI,0200H ; START WITH INT. BOH
0230 BE 4000       MOV SI,DIAG_TABLE_PTR ; POINT TO ENTRY POINT TABLE
0234 8B 0010       MOV CX,16 ; 16 ENTRIES
0237 A5           D5: MOVSW ; MOVE INTERRUPT VECTOR TO LOW
                        ; MEMORY

```

```

0238 47          IMC  D1
0238 47          IMC  D1          ; POINT TO NEXT VECTOR ENTRY
023A E2 FB      LOOP  DS          ; REPEAT FOR ALL 16 BIOS INTERRUPTS
023C BE D9      MOV   DS,CX          ; SET 05 TO ZERO
023E C7 08 0204 R 1B63 R  MOV   INT$1,OFFSET LOCATE1
0244 C7 08 0208 R 1A2A R  MOV   INT$2,OFFSET PRNT3
024A C7 06 0224 R 1BA5 R  MOV   INT$9,OFFSET JOYSTICK

```

```

-----
SET UP DEFAULT EQUIPMENT DETERMINATION WORD
;
BIT 15,14 = NUMBER OF PRINTERS ATTACHED
BIT 13 = 1 = SERIAL PRINTER PRESENT
BIT 12 = GAME I/O ATTACHED
BIT 11,10,9 = NUMBER OF R6232 CARDS ATTACHED
BIT 8 = DMA 0=DMA PRESENT, 1=NO DMA ON SYSTEM
BIT 7,6 = NUMBER OF DISKETTE DRIVES
;
; 00=1, 01=2, 10=3, 11=4 ONLY IF BIT 0 = 1
BIT 5,4 = INITIAL VIDEO MODE
;
; 00 - UNUSED
; 01 - 40X25 BW USING COLOR CARD
; 10 - 80X25 BW USING COLOR CARD
; 11 - 80X25 BW USING BW CARD
BIT 3,2 = PLANAR RAM SIZE (10=48K,11=64K)
BIT 1 NOT USED
BIT 0 = 1 (1PL DISKETTE INSTALLED)
-----

```

```

0250 BB 1118      ASSUME CS:CODE,DS:ABS0
MOV             BX,1118H          ; DEFAULT GAME10,40X25,NO DMA,48K OM
;                               ; PLANAR
0253 E4 62          IN      AL,PORT_C
0255 24 08          AND     AL,08H          ; 64K CARD PRESENT
0257 75 03          JNZ     D55          ; NO, JUMP
0258 80 CB 04       OR      BL,4          ; SET 64K ON PLANAR
025C B9 1E 0410 R  DB5:  MOV     DATA_WORD[EQUIP_FLAG-DATA],BX
-----

```

```

; TEST 7
; INITIALIZE AND TEST THE B259 INTERRUPT CONTROLLER CHIP
; MFG ERR. CODE 07XX (XX=00, DATA PATH OR INTERNAL FAILURE,
; XX=ANY OTHER BITS ON=UNEXPECTED INTERRUPTS
-----

```

```

0260 E8 E8DB R     CALL    MFG_UP          ; MFG CODE=F7
0263 B0 13          ASSUME  DS:ABS0,CS:CODE
MOV             AL,13H          ; ICW1 - RESET EDGE SENSE CIRCUIT,
;                               ; SET SINGLE B259 CHIP AND ICW4 READ
0265 E6 20          OUT     INTA00,AL
0267 80 08          MOV     AL,B          ; ICW2 - SET INTERRUPT TYPE B (B-F)
0269 E6 21          OUT     INTA01,AL
026B 80 09          MOV     AL,9          ; ICW4 - SET BUFFERED MODE/SLAVE
;                               ; AND 8086 MODE
026D E6 21          OUT     INTA01,AL
-----

```

```

; TEST ABILITY TO WRITE/READ THE MASK REGISTER
-----

```

```

026F B0 00          MOV     AL,0          ; WRITE ZEROS TO IMR
0271 8A 08          MOV     BL,AL         ; PRESET ERROR INDICATOR
0273 E6 21          OUT     INTA01,AL   ; DEVICE INTERRUPTS ENABLED
0275 E4 21          IN      AL,INTA01  ; READ IMR
0277 0A C0          OR      AL,AL        ; IMR = 0?
0279 75 18          JNZ     GERROR        ; NO - GO TO ERROR ROUTINE
027B 80 FF          MOV     AL,OFFH      ; DISABLE DEVICE INTERRUPTS
027D E6 21          OUT     INTA01,AL   ; WRITE ONES TO IMR
027F E4 21          IN      AL,INTA01  ; READ IMR
0281 04 01          ADD     AL,1         ; ALL IMR BITS ON?
;                               ; (ADD SHOULD PRODUCE 0)
0283 75 0E          JNZ     GERROR        ; NO - GO TO ERROR ROUTINE
-----

```

```

; CHECK FOR HOT INTERRUPTS
-----

```

```

INTERRUPTS ARE MASKED OFF. NO INTERRUPTS SHOULD OCCUR.
;
; STI
; MOV
; CX,50H
; WAIT FDR ANY INTERRUPTS
; BL,DATA_AREA[INTR_FLAG-DATA] ; DID ANY INTERRUPTS
; OCCUR?
0285 FB          STI
0286 B9 0050      MOV     CX,50H
0288 E2 FE       LOOP  HOT1
028A 8A 1E 0484 R HOT1:  MOV     BL,DATA_AREA[INTR_FLAG-DATA]
;                               ;
028F 0A DB       OR      BL,BL
0291 74 05       JZ      END_TESTG    ; NO - GO TO NEXT TEST
0293 B7 07       MOV     BH,07H      ; SET 07 SECTION OF ERROR MSG
0295 E9 09BC R   JNP    E_MSG
0299
END_TESTG:
; FIRE THE DISKETTE WATCHDOG TIMER
029B B0 E0       MOV     AL,W0_ENABLE+W0_STROBE+FDC_RESET
029A E6 F2       OUT     OF2H,AL
029C B0 A0       MOV     AL,W0_ENABLE+FDC_RESET
029E E6 F2       OUT     OF2H,AL
ASSUME CS:CODE,DS:ABS0
-----

```

```

; B253 TIMER CHECKOUT
-----

```

```

DESCRIPTION
;
; VERIFY THAT THE TIMERS (0, 1, AND 2) FUNCTION PROPERLY.
; THIS INCLUDES CHECKING FOR STUCK BITS IN ALL THE TIMERS,
; THAT TIMER 1 RESPONDS TO TIMER 0 OUTPUTS, THAT TIMER 0
; INTERRUPTS WHEN IT SHOULD, AND THAT TIMER 2'S OUTPUT WORKS
; AS IT SHOULD.
;
; THERE ARE 7 POSSIBLE ERRORS DURING THIS CHECKOUT.
; BL VALUES FOR THE CALL TO E_MSG INCLUDE:
; 0) STUCK BITS IN TIMER 0
; 1) TIMER 1 DOES NOT RESPOND TO TIMER 0 OUTPUT
; 2) TIMER 0 INTERRUPT DOES NOT OCCUR
; 3) STUCK BITS IN TIMER 1
; 4) TIMER 2 OUTPUT INITIAL VALUE IS NOT LOW
; 5) STUCK BITS IN TIMER 2
; 6) TIMER 2 OUTPUT DOES NOT GO HIGH ON TERMINAL COUNT
-----

```

```

;-----
; INITIALIZE TIMER 1 AND TIMER 0 FOR TEST
;-----
02A0 EB E6DB R CALL MFG_UP ; MFG CKPOINT=F6
02A3 BB 0176 MOV AX,0176H ; SET TIMER 1 TO MODE 3 BINARY
02A6 BB FFFF MOV BX,OFFFH ; INITIAL COUNT OF FFFF
02A9 EB FFE0 R CALL INIT_TIMER ; INITIALIZE TIMER 1
02AC BB 0036 MOV AX,0036H ; SET TIMER 0 TO MODE 3 BINARY
; INITIAL COUNT OF FFFF
02AF E6 FFE0 R CALL INIT_TIMER ; INITIALIZE TIMER 0
;-----
; SET BIT 5 OF PORT A0 SO TIMER 1 CLOCK WILL BE PULSED BY THE
; TIMER 0 OUTPUT RATHER THAN THE SYSTEM CLOCK.
;-----
02B2 B0 20 MOV AL,00100000B
02B4 EB A0 OUT DA0H,AL
;-----
; CHECK IF ALL BITS GO ON AND OFF IN TIMER 0 (CHECK FOR STUCK
; BITS)
;-----
02B6 B4 00 MOV AH,0 ; TIMER 0
02B8 EB 036C R CALL BITS_ON_OFF ; LET SUBROUTINE CHECK IT
02BB 73 05 JNB TIMER1_NZ ; NO STUCK BITS (CARRY FLAG NOT SET)
02BD B3 00 MOV BL,0 ; STUCK BITS IN TIMER 0
02BF EB 0362 R JNP TIMER_ERROR
;-----
; SINCE TIMER 0 HAS COMPLETED AT LEAST ONE COMPLETE CYCLE,
; TIMER 1 SHOULD BE NON-ZERO. CHECK THAT THIS IS THE CASE.
;-----
02C2 E4 41 TIMER1_NZ: IN AL,TIMER+1 ; READ LSB OF TIMER 1
02C4 BA E0 MOV AH,AL ; SAVE LSB
02C6 E4 41 IN AL,TIMER+1 ; READ MSB OF TIMER 1
02C8 3D FFFF CMP AX,OFFFH ; STILL FFFF?
02CB 75 05 JNE TIMER0_INTR ; NO - TIMER 1 HAS BEEN BUMPED
02CD B3 01 MOV BL,1 ; TIMER 1 WAS NOT BUMPED BY TIMER 0
02CF E9 0362 R JMP TIMER_ERROR
;-----
; CHECK FOR TIMER 0 INTERRUPT
;-----
02D2 FB TIMER0_INTR: STI ; ENABLE MASKABLE EXT INTERRUPTS
02D3 E4 21 IN AL,INTA01
02D5 24 FE AND AL,0FEH ; MASK ALL INTRs EXCEPT LVL 0
02D7 20 06 04B4 R AND DATA_AREA[INTR_FLAG-DATA],AL ; CLEAR INT RECEIVED
02D8 E6 21 OUT INTA01,AL ; WRITE THE B25B INTR
02DA B9 FFFF MOV CX,OFFFH ; SET LOOP COUNT
02E0 WAIT_INTR_LOOP:
02E0 F6 06 04B4 R 01 TEST DATA_AREA[INTR_FLAG-DATA],1 ; TIMER 0 INT OCCUR?
02E5 75 06 JNE RESET_INTRS ; YES - CONTINUE
02E7 E2 F7 LOOP WAIT_INTR_LOOP ; WAIT FOR INTR FOR SPECIFIED TIME
02E9 B3 02 MOV BL,2 ; TIMER 0 INTR DIDN'T OCCUR
02EB EB 75 JMP SHORT_TIMER_ERROR
;-----
; HOUSEKEEPING FOR TIMER 0 INTERRUPTS
;-----
02ED FA RESET_INTRS: CLI
; SET TIMER INT. TO POINT TO MFG. HEARTBEAT ROUTINE IF IN MFG MDDE
02EE BA 0201 MOV DX,2D1H
02F1 EC IN AL,DX ; GET MFG. BITS
02F2 24 F0 AND AL,0F0H
02F4 3C 10 CMP AL,10H ; SYS TEST MODE?
02F6 74 04 JE DE
02F8 0A 0C OR AL,AL ; OR BURN-IN MODE
02FA 7B 11 JNZ TIME_1
02FC C7 06 0020 R 18BD R 06: MOV INT_PTR,OFFSET MFG_TICK ; SET TO POINT TO MFG.
; ROUTINE
0302 C7 06 0070 R 18BD R MOV INT_IC_PTR,OFFSET MFG_TICK ; ALSO SET USER TIMER INT
; FOR DIAGS. USE
0308 B0 FE MOV AL,0FEH
030A EB 21 OUT INTA01,AL
030C FB STI
;-----
; RESET D5 OF PORT A0 SO THAT THE TIMER 1 CLOCK WILL BE
; PULSED BY THE SYSTEM CLOCK.
;-----
030D B0 00 TIME_1: MOV AL,0 ; MAKE AL = 00
030F E6 A0 OUT OA0H,AL
;-----
; CHECK FOR STUCK BITS IN TIMER 1
;-----
0311 B4 01 MOV AH,1 ; TIMER 1
0313 EB 036C R CALL BITS_ON_OFF
0316 73 04 JNB TIMER2_INIT ; NO STUCK BITS
0318 B3 03 MOV BL,3 ; STUCK BITS IN TIMER 1
031A EB 46 JMP SHORT_TIMER_ERROR
;-----
; INITIALIZE TIMER 2
;-----
031C E4 41 TIMER2_INIT: IN AL,TIMER+1
031C BB 02B6 MOV AX,02B6H ; SET TIMER 2 TO MODE 3 BINARY
031F BB FFFF MOV BX,OFFFH ; INITIAL COUNT
0322 EB FFE0 R CALL INIT_TIMER
;-----
; SET PBD OF PORT_B OF 8255 (TIMER 2 GATE)
;-----
0323 E4 61 IN AL,PORT_B ; CURRENT STATUS
0327 0C 01 OR AL,00000010B ; SET BIT 0 - LEAVE OTHERS ALONE
0329 E6 61 OUT PORT_B,AL

```



```

-----
                                CHECK FOR STUCK BITS IN TIMER 2
-----
032B B4 02      MOV     AH,2          ; TIMER 2
032D EB 036C R  CALL    BITS_ON_OFF
0330 73 04      JNB    REINIT_T2      ; NO STUCK BITS
0332 B3 05      MOV     BL,5          ; STUCK BITS IN TIMER 2
0334 EB 2C      JMP     SHORT_TIMER_ERROR
-----
                                RE_INITIALIZE TIMER 2 WITH MODE 0 AND A SHORT COUNT
-----
0336      REINIT_T2:
; DROP GATE TO TIMER 2
0336 E4 81      IN     AL,PORT_B      ; CURRENT STATUS
0337 24 FE      AND    AL,1111110B   ; RESET BIT 0 - LEAVE OTHERS ALONE
033A E6 61      OUT    PORT_B,AL
033C BB 02B0    MOV    AX,02B0H      ; SET TIMER 2 TO MODE 0 BINARY
033F BB 000A    MOV    BX,000AH     ; INITIAL COUNT OF 10
0342 EB FFE0 R  CALL    INIT_TIMER
-----
                                CHECK PC5 OF PORT_C OF B255 TO SEE IF THE OUTPUT OF TIMER 2
                                IS LOW
-----
0345 E4 62      IN     AL,PORT_C      ; CURRENT STATUS
0347 24 20      AND    AL,00100000B ; MASK OFF OTHER BITS.
0349 74 04      JZ     CK2_ON        ; IT'S LOW
034B B3 04      MOV    BL,4          ; PCS OF PORT_C WAS HIGH WHEN IT
034D EB 13      JMP    SHORT_TIMER_ERROR ; SHOULD HAVE BEEN LOW
; TURN GATE BACK ON
034F E4 61      CK2_ON: IN    AL,PORT_B ; CURRENT STATUS
0351 0C 01      OR     AL,00000010B ; SET BIT 0 - LEAVE OTHERS ALONE
0353 EC 61      OUT    PORT_B,AL
-----
                                CHECK PC5 OF PORT_C TO SEE IF THE OUTPUT OF TIMER 2 GOES
                                HIGH
-----
0355 B8 000A    MOV    CX,000AH     ; WAIT FOR OUTPUT GO HIGH, SHOULD
035B E2 FE      CK2_LO: LOOP  CK2_LO ; BE LONGER THAN INITIAL COUNT
035A E4 62      IN     AL,PORT_C      ; CURRENT STATUS
035C 24 20      AND    AL,00100000B ; MASK OFF ALL OTHER BITS
035E 75 57      JNZ    POD13_END     ; IT'S HIGH - WE'RE DONE!
0360 B3 06      MOV    BL,6          ; TIMER 2 OUTPUT DID NOT GO HIGH
-----
                                B253 TIMER ERROR OCCURRED. SET BH WITH MAJOR ERROR
                                INDICATOR AND CALL E_MSG TO INFORM THE SYSTEM OF THE ERROR.
                                (BL ALREADY CONTAINS THE MINOR ERROR INDICATOR TO TELL
                                WHICH PART OF THE TEST FAILED.)
-----
0362      TIMER_ERROR:
0362 B7 08      MOV    BH,B          ; TIMER ERROR INDICATOR
0364 EB 09BC R  CALL    E_MSG
0367 EB 4E      JMP    SHORT_POD13_END
-----
                                BITS ON/OFF SUBROUTINE - USED FOR DETERMINING IF A
                                PARTICULAR TIMER'S BITS GO ON AND OFF AS THEY SHOULD.
                                THIS ROUTINE ASSUMES THAT THE TIMER IS USING BOTH THE LSB
                                AND THE MSB.
                                CALLING PARAMETER:
                                (AH) = TIMER NUMBER (0, 1, OR 2)
                                RETURNS:
                                (CF) = 1 IF FAILED
                                (CF) = 0 IF PASSED
                                REGISTERS AX, BX, CX, DX, DI, AND SI ARE ALTERED.
-----
0369      LATCHES LABEL BYTE
0369 00      DB    00H          ; LATCH MASK FOR TIMER 0
036A 40      DB    40H          ; LATCH MASK FOR TIMER 1
036B 80      DB    80H          ; LATCH MASK FOR TIMER 2
-----
036C      BITS_ON_OFF PROC NEAR
036C XOR     BX,BX      ; INITIALIZE BX REGISTER
036E XOR     SI,SI      ; 1ST PASS - SI = 0
0370 MOV    DX,TIMER   ; BASE PORT ADDRESS FOR TIMERS
0373 ADD    DL,AH
0375 MOV    DI,OFFSET LATCHES ; SELECT LATCH MASK
0378 XOR    AL,AL      ; CLEAR AL
037A XCHG  AL,AH      ; AH -> AL
037C ADD    DI,AX      ; TIMER LATCH MASK INDEX
037E      ; 1ST PASS - CHECKS FOR ALL BITS TO COME ON
037E      ; 2ND PASS - CHECKS FOR ALL BITS TO GO OFF
037E OUTER_LOOP:
037E MOV    CX,B        ; OUTER LOOP COUNTER
0381 INNER_LOOP:
0381 PUSH  CX          ; SAVE OUTER LOOP COUNTER
0382 MOV    CX,OFFFHH ; INNER LOOP COUNTER
0385 TST_BITS:
0385 MOV    AL,CS:[DI] ; TIMER LATCH MASK
0388 OUT    TIM_CTL,AL ; LATCH TIMER
038A PUSH  AX          ; PAUSE
038B POP   AX
038C IN    AL,DX      ; READ TIMER LSB
038D OR    SI,SI
038F JNE   .SECOND    ; SECOND PASS
0391 OR    AL,01H      ; TURN LS BIT ON
0393 OR    BL,AL      ; TURN 'ON' BITS ON
0395 IN    AL,DX      ; READ TIMER MSB
0396 OR    BH,AL      ; TURN 'ON' BITS ON
0398 CMP    BX,OFFFHH ; ARE ALL TIMER BITS ON?
039C JMP    SHORT_TST_CMP ; DON'T CHANGE FLAGS
-----

```

```

039E          SECOND:
03A0 22 DB      AND    BL,AL      ; CHECK FOR ALL BITS OFF
03A1 EC          IN     AL,DX      ; READ MSB
03A1 22 FB      AND    BH,AL      ; TURN OFF BITS
03A3 08 DB      OR     BX,BX      ; ALL OFF?
03A5
03A5 74 07      JE     CHK_END    ; YES - SEE IF DOME
03A7 E2 DC      LOOP   TST_BITS   ; KEEP TRYING
03A9 99          POP    CX         ; RESTORE OUTER LOOP COUNTER
03AA E2 DB      LOOP   INNER_LOOP ; TRY AGAIN
03AC F9          STC    STC        ; ALL TRIES EXHAUSTED - FAILED TEST
03AD C3          RET
03AE 59          POP    CX         ; POP FORMER OUTER LOOP COUNTER
03AF 46          INC    SI         ;
03B0 B3 FE 02   CMP    SI,2       ;
-03B3 75 CB     JNE    OUTER_LOOP ; CHECK FOR ALL BITS TO GO OFF
03B5 FB          CLC
03B6 C3          RET
03B7           BITS_ON_OFF  ENDP
03B7           POD15_END:

```

CRT ATTACHMENT TEST

```

; 1. INIT.CRT TO 40X25 - BW
; 2. CHECK FOR VERTICAL AND VIDEO ENABLES, AND CHECK
;    TIMING OF SAME
; 3. CHECK VERTICAL INTERRUPT
; 4. CHECK RED, BLUE, GREEN, AND INTENSIFY DOTS
; 5. INIT TO 40X25 - COLOR
; MFG. ERROR CODE 09XX (XX-SEE COMMENTS IN CODE)
-----

```

```

= 40AC          MAVT  EQU    0A0ACH    ; MAXIMUM TIME FOR VERT/VERT
;              ; (NOMINAL + 10%)
= C460          MIVT  EQU    0C460H    ; MINIMUM TIME FOR VERT/VERT
;              ; (NOMINAL - 10%)
= D0C8          ; NOMINAL TIME IS B2BGH FOR 60 Hz.
EPF            EQU    200             ; NUMBER OF ENABLES PER FRAME

03B7 E8 E6DB R  CALL   MFG_UP      ; MFG CHECKPOINT= F5
03BA FA          CLI
03BB 90 70      MOV    AL,0110000B    ; SET TIMER 1 TO MODE 0
03BD E6 43      OUT    TIM_CTL,AL
03BF B9 8000    MOV    CX,8000H
03C2 E2 FE      Q1:  LOOP   Q1         ; WAIT FOR MODE SET TO "TAKE"
03C4 80 00      MOV    AL,00H
03C6 E6 41      OUT    TIMER+1,AL ; SEND FIRST BYTE TO TIMER
03C8 2B C0      SUB    AX,AX          ; SET MODE 40X25 - BW
03CA CD 10      INT    10H
03CC B8 0507    MOV    AX,0507H        ; SET TO VIDEO PAGE 7
03CF CD 10      INT    10H
03D1 BA 03DA    MOV    DX,03DAH        ; SET ADDRESSING TO VIDEO ARRAY
03D4 2B C8      SUB    CX,CX
; LOOK FOR VERTICAL
03D6 EC          Q2:  IN     AL,DX      ; GET STATUS
03D7 AB 08      TEST   AL,00001000B    ; VERTICAL THERE YET?
-03D9 78 08     JNE    Q3         ; CONTINUE IF IT IS
03DB E2 F9      LOOP   Q2         ; KEEP LOOKING TILL COUNT EXHAUSTED
03DD B3 00      MOV    BL,00
03DF EB 4C      JMP    SHORT Q115    ; NO VERTICAL = ERROR 0900
; GOT VERTICAL - START TIMER
03E1 32 C0      Q3:  XOR    AL,AL
03E3 E6 41      OUT    TIMER+1,AL    ; SEND 2ND BYTE TO TIMER TO START
03E5 2B DB      SUB    BX,BX          ; INIT. ENABLE COUNTER
; WAIT FOR VERTICAL TO GO AWAY
03E7 33 C9      XOR    CX,CX
03E9 EC          Q4:  IN     AL,DX      ; GET STATUS
03EA AB 08      TEST   AL,00001000B    ; VERTICAL STILL THERE?
03EC 74 06      JZ     Q5         ; CONTINUE IF IT'S GONE
03EE E2 F9      LOOP   Q4         ; KEEP LOOKING TILL COUNT EXHAUSTED
03F0 B3 01      MOV    BL,01H
03F2 EB 39      JMP    SHORT Q115    ; VERTICAL STUCK ON = ERROR 0901
; NOW START LOOKING FOR ENABLE TRANSITIONS
03F4 2B C9      Q5:  SUB    CX,CX
03F6 EC          Q6:  IN     AL,DX      ; GET STATUS
03F7 AB 01      TEST   AL,00000001B    ; ENABLE ON YET?
03F9 75 0A      JNE    Q7         ; GO ON IF IT IS
03FB AB 08      TEST   AL,00001000B    ; VERTICAL ON AGAIN?
03FD 75 22      JNE    Q11         ; CONTINUE IF IT IS
03FF E2 F5      LOOP   Q6         ; KEEP LOOKING IF NOT
0401 B3 02      MOV    BL,02H
0403 EB 28      JMP    SHORT Q115    ; ENABLE STUCK OFF = ERROR 0902
; MAKE SURE VERTICAL WENT OFF WITH ENABLE GOING ON
0405 AB 06      Q7:  TEST   AL,00001000B    ; VERTICAL OFF?
0407 74 04      JZ     Q8         ; GO ON IF IT IS
0409 B3 03      MOV    BL,03H
040B EB 20      JMP    SHORT Q115    ; VERTICAL STUCK ON = ERROR 0903
; NOW WAIT FOR ENABLE TO GO OFF
040D 2B C9      Q8:  SUB    CX,CX
040F EC          Q9:  IN     AL,DX      ; GET STATUS
0410 AB 01      TEST   AL,00000001B    ; ENABLE OFF YET?
0412 74 06      JE     Q10        ; PROCEED IF IT IS
0414 E2 F9      LOOP   Q9         ; KEEP LOOKING IF NOT YET LOW
0416 B3 04      MOV    BL,04H
0418 EB 13      JMP    SHORT Q115    ; ENABLE STUCK ON = ERROR 0904
; ENABLE HAS TOGGLED, BUMP COUNTER AND TEST FOR NEXT VERTICAL
041A 43          Q10: INC    BX         ; BUMP ENABLE COUNTER
041B 74 04      JZ     Q11        ; IF COUNTER WRAPS, ERROR
041D AB 08      TEST   AL,00001000B    ; DID ENABLE GO LOW BECAUSE OF
; VERTICAL?
041F 74 D3      JZ     Q5         ; IF NOT, LOOK FOR ANOTHER ENABLE
; TOGGLE

```

```

; HAVE HAD COMPLETE VERTICAL-VERTICAL CYCLE, NOW TEST RESULTS
0421 B0 40      Q11:  MOV    AL,40H      ; LATCH TIMER1
0423 E6 43      OUT    TIM_CTL,AL
0425 81 FB 00CB CMP    BX,EPF      ; NUMBER OF ENABLES BETWEEN
; VERTICALS O.X.?

0429 74 04      JE     Q12      ;
042B B3 05      MOV    BL,05H
042D EB 74      Q115: JMP    SHORT Q22      ; WRONG # ENABLES = ERROR 0905
042F E4 41      Q12:  IN     AL,TIMER+1      ; GET TIMER VALUE LOW
0431 8A E0      MOV    AH,AL      ; SAVE IT
0433 90         NOP
0434 E4 41      IN     AH,TIMER+1      ; GET TIMER HIGH
0436 88 E0      KCHG  AL,AH      ;
0438 FB         STI
0439 90         NOP
043A 3D A0AC    CMP    AX,MAVT      ;
043D 7D 04      JGE    Q13      ;
043F B3 06      MOV    BL,06H
0441 E9 60      JMP    SHORT Q22      ; VERTICALS TOO FAR APART
; = ERROR 0906

0443 3D C4B0    Q13:  CMP    AX,HIVT      ;
0446 7E 04      JLE    Q14      ;
0448 B3 07      MOV    BL,07H
044A EB 57      JMP    SHORT Q22      ; VERTICALS TOO CLOSE TOGETHER
; = ERROR 0907

; TININGS SEEM O.X., NOW CHECK VERTICAL INTERRUPT (LEVEL 5)
044C 2B C9      Q14:  SUB    CX,CX      ; SET TIMEOUT REG
044E E4 21      IN     AL,INTA01
0450 24 DF      AND    AL,1101111B ; UNMASK INT. LEVEL 5
0452 E6 21      OUT    INTA01,AL
0454 20 06 0484 R AND    DATA_AREA(INTR_FLAG-DATA3),AL ; ENABLE INTS.
0455 FB         STI
0455 F6 06 0484 R 20 Q15:  TEST   DATA_AREA(INTR_FLAG-DATA3),00100000B ; SEE IF INTR.
; 5 HAPPENED YET
045E 75 06      JNZ    Q16      ; GO ON IF IT DID
0460 E2 F7      LOOP   Q15      ; KEEP LOOKING IF IT DIDN'T
0462 B3 08      MOV    BL,05H
0464 EB 3D      JMP    SHORT Q22      ; NO VERTICAL INTERRUPT
; = ERROR 0908

0466 E4 21      Q16:  IN     AL,INTA01      ; DISABLE INTERRUPTS FOR LEVEL 5
0469 0C 20      OR     AL,00100000B
046A E6 21      OUT    INTA01,AL

; SEE IF RED, GREEN, BLUE AND INTENSIFY DOTS WORK
; FIRST, SET A LINE OF REVERSE VIDEO, INTENSIFIED BLANKS INTO VIDEO
; BUFFER
046C BB 09DB    MOV    AX,09DBH      ; WRITE CHARS, BLOCKS
046F BB 077F    MOV    BX,077FH      ; PAGE 7, REVERSE VIDEO,
; HIGH INTENSITY
; 40 CHARACTERS

0472 BB 002B    MOV    CX,40
0475 CD 10      INT    10H
0477 33 C0      XOR    AX,AX      ; START WITH BLUE DOTS
0479 2B C9      Q17:  SUB    CX,CX
047B EE         OUT    DX,AL      ; SET VIDEO ARRAY ADDRESS FOR DOTS
; SEE IF DOT COMES ON
047C EC         ;
047D A9 10      Q18:  IN     AL,DX      ; GET STATUS
047F 75 08      TEST   AL,00010000B ; DOT THERE?
0491 E2 F9      JNZ    Q19      ; GO LOOK FOR DOT TO TURN OFF
0493 10 08      LOOP   Q18      ; CONTINUE TESTING FOR DOT ON
0495 B3 0A      MOV    BL,10H
0497 0A DC      OR     BL,AH      ; OR IN DOT BEING TESTED
0499 EB 1A      JMP    SHORT Q22      ; DOT NOT COMING ON = ERROR 091X
; (X=0, BLUE; X=1, GREEN;
; X=2, RED; X=3, INTENSITY)

; SEE IF DOT GOES OFF
049B 2B C9      Q19:  SUB    CX,CX
049B EC         ;
049C A9 10      Q20:  IN     AL,DX      ; GET STATUS
049E 74 08      TEST   AL,00010000B ; IS DOT STILL ON?
0499 E2 F9      JE     Q21      ; GO ON IF DOT OFF
0499 E2 F9      LOOP   Q20      ; ELSE, KEEP WAITING FOR DOT
; TO GO OFF

0492 B3 20      MOV    BL,20H
0494 0A DC      OR     BL,AH      ; OR IN DOT BEING TESTED
0496 E9 08      JMP    SHORT Q22      ; DOT STUCK ON = ERROR 092X
; (X=0, BLUE; X=1, GREEN;
; X=2, RED; X=3, INTENSITY)

; ADJUST TO POINT TO NEXT DOT
049B FE C4      Q21:  INC    AH
049A 80 FC 04    CMP    AH,4      ; ALL 4 DOTS DONE?
049D 74 09      JE     Q23      ; GO END
049F 8A C4      MOV    AL,AH
04A1 E9 D6      JMP    Q17      ; GO LOOK FOR ANOTHER DOT
04A3 B7 09      Q22:  MOV    BH,09H      ; SET MSB OF ERROR CODE
04A5 E9 09BC R  JNP    E_MSG
; DONE WITH TEST RESET TO 40X25 - COLOR

04A8 EB 13BB R   ASSUME DS:DATA
04A8 B8 0011    Q23:  CALL  DS
04AA CD 10      MOV    AX,0001H      ; INIT TO 40X25 - COLOR
04AB 10 10      INT    10H
04AD B8 0507    MOV    AX,0507H      ; SET TO VIDEO PAGE 7
04AD CD 10      INT    10H
04B5 B1 3E 0072 R 1234 CMP    RESET_FLAG,1234H ; WARM START?
04B8 74 03      JE     Q24      ; BYPASS PUTTING UP POWER-ON SCREEN
04BD EB 0C21 R   CALL  PUT_LOGO      ; PUT LOGO ON SCREEN

```

```

0480 EB 0C21 R      CALL PUT_LOGO          ; PUT LOGO ON SCREEN
04C0 80 76         MOV AL,0110110B    ; RE-INIT TIMER 1
04C2 E6 43        OUT TIM_CTL,AL
04C4 80 00        MOV AL,00H
04C6 E6 41        OUT TIMER+1,AL
04C8 90          NOP
04C9 90          NOP
04CA E6 41        OUT TIMER+1,AL
04CC EB E6DB R    CALL ASSUME          DS:ABS0
04CF 33 C0        XOR AX,AX
04D1 8E DB        MOV DS,AX
04D3 C7 06 000B R MOV NM1_PTR,OFFSET K8DNH1 ; SET INTERRUPT VECTOR
04D9 C7 06 0120 R MOV KEY62_PTR,OFFSET KEY_SCAN_SAVE ; SET VECTOR FOR
                                ; POD INT HANDLER

04DF 0E          PUSH CS
04E0 58          POP AX
04E1 A3 0122 R    MOV KEY62_PTR+2,AX
                                ASSUME DS:DATA
04E4 EB 138B R    CALL DDS            ; SET DATA SEGMENT
04E7 BE 001E R    MOV SI,OFFSET KB_BUFFER ; SET KEYBOARD PARMS
04EA 89 36 001A R MOV BUFFER_HEAD,SI
04EC 89 36 001C R MOV BUFFER_TAIL,SI
04F2 89 36 00B0 R MOV BUFFER_START,SI
04F6 83 C6 20    ADD SI,32          ; SET DEFAULT BUFFER OF 32 BYTES
04FB 89 36 00B2 R MOV BUFFER_END,SI
04FD E4 A0      IN AL,0A0H        ; CLEAR NMI F/F
04FF 80 B0      MOV AL,80H       ; ENABLE NMI
0501 E6 A0      OUT 0A0H,AL

; IF A KEY IS STUCK, THE BUFFER SHOULD FILL WITH THAT KEY'S CODE
; THIS WILL BE CHECKED LATER
-----
; MEMORY SIZE DETERMINE AND TEST
; THIS ROUTINE WILL DETERMINE HOW MUCH MEM
; IS ATTACHED TO THE SYSTEM (UP TO 640KB)
; AND SET "MEMORY_SIZE" AND "REAL_MEMORY"
; WORDS IN THE DATA AREA.
;
; AFTER THIS, MEMORY WILL BE EITHER TESTED
; OR CLEARED, DEPENDING ON THE CONTENTS OF
; "RESET_FLAG".
; MFG. ERROR CODES      -0AXX PLANAR BD ERROR
;                       -0BXX 64K CD ERROR
;                       -0CXX ERRORS IN BOTH
;                       00D AND EVEN BYTES
;                       IN A 128K SYS
;                       -1YXX MEMORY ABOVE 128K
;                       Y=SEGMENT HAVING TROUBLE
;                       XX= ERROR BITS
-----
0503 EB E6DB R    ASSUME DS:DATA
0506 B8 0040     CALL MFG_UP        ; MFG CHECKPOINT=F3
0509 E4 62      MOV BX,64         ; START WITH BASE 64K
050B A8 0B      IN AL,PORT_C     ; GET CONFIG BYTE
050D 75 03      TEST AL,00001000B ; SEE IF 64K CARD INSTALLED
050F 83 C3 40   JNE Q25          ; (BIT 4 WILL BE 0 IF CARD PLUGGED)
0512 53        ADD BX,64        ; ADD 64K
0513 53        PUSH BX         ; SAVE K COUNT
0515 83 EB 10   SUB BX,16        ; SUBTRACT 16K CRT REFRESH SPACE
0516 89 1E 0013 R MOV I(MEMORY_SIZE),BX ; LOAD "CONTIGUOUS MEMORY" WORD
051A 5B        POP BX
051B BA 2000   MOV DX,2000H    ; SET POINTER TO JUST ABOVE 128K
051E 2B FF     SUB DI,D1        ; SET DI TO POINT TO BEGINNING
0520 89 AA55   MOV CX,0AA55H   ; LOAD DATA PATTERN
0523 BE C2     MOV ES,DX        ; SET SEGMENT TO POINT TO MEMORY
                                SPACE
0525 26 89 0D   MOV ES:[DI],CX  ; SET DATA PATTERN TO MEMORY
0528 B0 0F     MOV AL,0FH      ; SET AL TO ODD VALUE
052A 26 B8 05   MOV AX,ES:[DI] ; GET DATA PATTERN BACK FROM MEM
052D 33 C1     XOR AX,CX       ; SEE IF DATA MADE IT BACK
052F 75 0C     JNZ Q27         ; NO? THEN END OF MEM HAS BEEN
                                REACHED
0531 81 C2 1000 ADD DX,1000H    ; POINT TO BEGINNING OF NEXT 64K
0535 83 C3 40   ADO BX,64      ; ADJUST TOTAL MEM. COUNTER
0538 80 FE A0   CMP DH,0A0H    ; PAST 640K YET?
053B 75 E6     JNE Q26        ; CHECK FOR ANOTHER BLOCK IF NOT
053D 89 1E 0015 R MOV I(TRUE_MEM),BX ; LOAD "TOTAL MEMORY" WORD
; SIZE HAS BEEN DETERMINED, NOW TEST OR CLEAR ALL OF MEMORY
; 4 KB KNOWN OK AT THIS POINT
0541 8B 0044   MOV AX,4
0544 EB 05BC R  CALL Q35
0547 BA 0080   MOV DX,0080H   ; SET POINTER TO JUST ABOVE
                                LOWER 2K
054A 89 7800   MOV CX,7800H   ; TEST 30K WORDS (60KB)
054D 8E C2     MOV ES,DX
054F 51        PUSH CX
0550 53        PUSH BX
0551 50        PUSH AX
0552 EB 085B R CALL PODSTG     ; TEST OR FILL MEM
0555 74 03     JZ Q29
0557 E9 0603 R JMP Q39        ; JUMP IF ERROR
055A 58        POP AX
055B 58        POP BX
055C 59        POP CX
055D 80 FD 78   CMP CH,78H    ; RECOVER
                                WAS THIS A 60 K PASS
0560 9C        PUSHF
0561 05 003C    ADD AX,60     ; BUMP GOOD STORAGE BY 60 KB
0564 9D        POPF
0566 74 03     JE Q30
0567 05 0002   ADD AX,2      ; ADD 2 FOR A 62K PASS
056A EB 05BC R CALL Q35
056D 38 C3     CMP AX,BX    ; ARE WE DONE YET?
056F 75 03     JNE Q31
0571 E9 0840 R  JMP Q43      ; ALL DONE, IF 50

```

```

0574 3D 0080
0577 74 1E
0579 BA 0F80
057C B9 0400
057F 8E C2
0581 50
0582 53
0583 52
0584 E8 0858 R
0587 75 7A
0588 5A
058A 58
058B 58
058C 05 0002
058F BA 1000
0592 B9 7C00
0595 E8 B6
0597 BA 2000
059A 3B DB
059C 75 03
059E E9 0640 R
05A1 B9 4000
05A4 8E C2
05A8 50
05A7 53
05A8 52
05A9 E8 0859 R
05AC 75 55
05AE 5A
05AF 58
05B0 58
05B1 05 0020
05B4 E8 058C R
05B7 80 C6 06
05BA E8 0E

```

```

Q31:  CMP AX,12B ; DONE WITH 1ST 128K?
      JE Q32 ; GO FINISH REST OF MEM.
      MOV DX,0F80H ; SET POINTER TO FINISH 1ST 64 KB
      MOV ES,DX
      PUSH AX
      PUSH BX
      PUSH DX
      CALL PODSTG ; GO TEST/FILL
      JNZ Q39
      POP DX
      POP BX
      POP AX
      ADD AX,2 ; UPDATE GOOD COUNT
      MOV DX,1000H ; SET POINTER TO 2ND 64K BLOCK
      MOV CX,7C00H ; 62K WORTH
      JMP Q29 ; GO TEST IT
      MOV DX,2000H ; POINT TO BLOCK ABOVE 128K
Q32:  CMP BX,AX ; COMPARE GOOD MEM TO TOTAL MEM
      JNE Q34
      JMP Q43
      MOV CX,4000H ; EXIT IF ALL DONE
      MOV ES,DX ; SET FOR 32KB BLOCK
      PUSH AK
      PUSH BX
      PUSH DX
      CALL POOSTG ; GO TEST/FILL
      JNZ Q39
      POP DX
      POP BX
      POP AX
      ADD AX,32 ; BUMP GOOD MEMORY COUNT
      CALL Q35 ; DISPLAY CURRENT GOOD MEM
      ADD DH,0BH ; SET POINTER TO NEXT 32K
      JMP Q33 ; AND MAKE ANOTHER PASS
Q33:
Q34:
Q35:

```

```

-----
SUBROUTINE FOR PRINTING TESTED
MEMORY OK MSG ON THE CRT
CALL PARAMS: AX = K OF GOOD MEMORY
              (IN HEX)
-----

```

```

05BC EB 138B R
05BF B1 3E 0072 R 1234
05C5 74 38
05C7 53
05CB 51
05C9 52
05CA 50
05CB 84 02
05CD 8A 1421
05D0 B7 07
05D2 CD 10
05D4 58
05D5 50
05D6 BB 000A
05D9 B5 0003
05DC 33 D2
05DE F7 F3
05E0 80 CA 30
05E3 52
05E4 E2 F6
05E6 B9 0003
05E9 58
05EA E8 18BA R
05ED E2 FA
05EF B9 0003
05F2 BE 0025 R
05FB 2E: BA 04
05FB 46
05F9 E8 18BA R
05FC E2 F7
05FE 58
05FF 5A
0600 59
0601 58
0602 C3
0603

```

```

Q35:  PROC NEAR
      CALL DDS ; ESTABLISH ADDRESSING
      CMP RESET_FLAG,1234H ; WARM START?
      JE Q35E ; NO PRINT ON WARM START
      PUSH BX
      PUSH CX
      PUSH DX
      PUSH AX ; SAVE WORK REGS
      MOV AH,2 ; SET CURSOR TOWARD THE END OF
      MOV DX,1421H ; ROW 20 (ROW 20, COL. 33)
      MOV BH,7 ; PAGE 7
      INT 10H
      POP AX
      PUSH AX ; SET UP FOR DECIMAL CONVERT
      MOV BX,10 ; OF 3 NIBBLES
      MOV CX,3
Q36:  XOR DX,DX
      DIV BX ; DIVIDE BY 10
      OR DL,30H ; MAKE INTO ASCII
      PUSH DX ; SAVE
      LOOP Q36
      MOV CX,3
Q37:  POP AX ; RECOVER A NUMBER
      CALL PRT_HEX
      LOOP Q37
      MOV CX,3
      MOV SI,OFFSET F3B ; PRINT " KB"
Q38:  MOV AL,CS:[SI]
      INC SI
      CALL PRT_HEX
      LOOP Q38
      POP AX
      POP DX
      POP CX
      POP BX
Q35E: RET
Q35: ENDP

```

```

0603 5A
0604 B1 FA 2000
0608 7C 0E
060A 8A D9
060C 0A DD
060E B1 04
0610 D2 EE
0612 B7 10
0614 0A FE
0616 E8 20
0618 B7 0A
061A E4 62
061C 24 08
061E 74 06
0620 8A D9
0622 0A DD
0624 EB 12

```

```

Q39:  POP DX ; POP SEGMENT POINTER TO DX
      ; (HEADING DOWNHILL, DON'T CARE
      ; ABOUT STACK)
      MOV DX,2000H ; ABOVE 128K (THE SIMPLE CASE)
      JL Q40 ; GO DO ODD/EVEN-LESS THAN 128K
      MOV BL,CL ; FORM ERROR BITS ("XX")
      OR BL,CH
      MOV CL,4 ; ROTATE MOST SIGNIFICANT
      ; NIBBLE OF SEGMENT
      SHR DH,CL ; TO LOW NIBBLE OF DH
      OR BH,DH ; FORM "IV" VALUE
      JMP SHORT Q42
Q40:  MOV BH,0AH ; ERROR 0A...
      IN AL,PORT_C ; GET CONFIG BITS
      AND AL,00001000H ; TEST FOR ATTRIB CARD PRESENT
      JZ Q41 ; WORRY ABOUT ODD/EVEN IF IT IS
      MOV BL,CL ; COMBINE ERROR BITS IF IT ISN'T
      OR BL,CH
      JMP SHORT Q42

```

```

0626 80 FC 02
0628 8A 09
062B 74 08
062D FE C7
062F 0A DD
0631 80 FC 01
0634 74 02
0636 FE C7

063B BE 0035 R
063B EB 09BC R

063E FA
063F F4
0640

Q41:  CMP    AH,02          ; EVEN BYTE ERROR? ERR OAXX
      MOV    BL,CL
      JNC    042
      INC    BH            ; MAKE INTO 08XX ERR
      OR     BL,CH        ; HAVE AND COMBINE ERROR BITS
      CMP    AH,1        ; ODD BYTE ERROR
      JE     042
      INC    BH            ; MUST HAVE BEEN BOTH
                          ; - MAKE INTO 0CXX

Q42:  MOV    SI,OFFSET MEM_ERR
      CALL  E_MSG        ; LET ERROR ROUTINE FIGURE OUT
                          ; WHAT TO DO

Q43:  -----
      ;
      ; KEYBOARD TEST
      ; DESCRIPTION
      ; NMI HAS BEEN ENABLED FOR QUITE A FEW
      ; SECONDS NOW. CHECK THAT NO SCAN CODES
      ; HAVE SHOWN UP IN THE BUFFER. (STUCK
      ; KEY) IF THEY HAVE, DISPLAY THEM AND
      ; POST ERROR.
      ; MFG ERR CODE
      ; 2000 STRAY NMI INTERRUPTS OR KEYBOARD
      ; RECEIVE ERRORS
      ; 21XX CARO FAILURE
      ; XX=01, KB DATA STUCK HIGH
      ; XX=02, KB DATA STUCK LOW
      ; XX=03, NO NMI INTERRUPT
      ; 22XX STUCK KEY (XX=SCAN CODE)
      ;-----
      ; ASSUME DS:DATA
      ;----- CHECK FOR STUCK KEYS
0640  EB E608 R
0643  EB 1388 R
0646  8B 001E R
0649  8A 07
064B  0A C0
064D  74 08
064F  87 22
0651  8A DB
0653  EB 0A
0655  80 3E 0012 R 00

065A  74 1C
065C  8B 2000
065F  BE 0036 R
0662  81 3E 0072 R 4321
066B  74 08
066A  81 3E 0072 R 1234
0670  74 03
0672  EB 09BC R
0675  E9 067F R

067B  8A 0201
067B  EC
067C  24 F0
067E  74 7F
0680  E4 62
0682  24 80
0684  74 79
0686  E4 61
0688  24 FC
068A  E6 61
068C  80 B6
068E  E5 43
0690  80 40
0692  E6 A0
0694  80 20

0696  8A 0042
0699  EE
069A  2B C0
069C  8B C8
069E  EE
069F  E4 61
06A1  0C 01
06A3  E6 61
06A5  E4 62
06A7  24 40
06A9  75 06
06AB  E2 F8
06AD  83 02
06AF  EB 49
06B1  06
06B2  2B C0
06B4  BE C0
06B6  26 C7 06 000B R FB15 R
06BD  A2 006A R
06C0  E4 61
06C2  0C 30
06C4  E6 61
06C6  80 C0
06CB  E6 A0
06CA  8B 0100

F6_Y:  CMP    KBD_ERR,00H    ; DID NMI'S HAPPEN WITH NO SCAN
      ; CODE PASSED?
      JE     F7            ; (STRAYS) - CONTINUE IF NONE
      MOV    BX,2000H     ; SET ERROR CODE 2000
      MOV    SI,OFFSET KEY_ERR ; GET MSG ADDR
      CMP    RESET_FLAG,4321H ; WARN START TO DIAGS
      JE     F6_Z         ; DO NOT PUT UP MESSAGE
      CMP    RESET_FLAG,1234H ; WARN SYSTEM START
      JE     F6_Z         ; DO NOT PUT UP MESSAGE
      CALL  E_MSG        ; PRINT MSG ON SCREEN

F6_Z:  JMP    F6_X
      ; CHECK LINK CARD, IF PRESENT
F7:    MOV    DX,0201H
      IN     AL,DX        ; CHECK FOR BURN-IN MODE
      AND   AL,0F0H
      JZ    F6_X         ; BYPASS CHECK IN BURN-IN MODE
      AL,PORT_C          ; GET CONFIG. PORT DATA
      AND   AL,10000000B ; KEYBOARD CABLE ATTACHED?
      JZ    F6_X         ; BYPASS TEST IF IT IS
      IM    AL,PORT_B
      AND   AL,1111100B ; DROP SPEAKER DATA
      OUT   PORT_B,AL
      MOV   AL,086H     ; MODE SET TIMER 2
      OUT   TIM_CTL,AL
      MOV   AL,040H     ; DISABLE NMI
      OUT   0A0H,AL    ;
      MOV   AL,32      ; LSB TO TIMER 2
                          ; (APPROX. 40Khz VALUE)

      MOV   DX,TIMER+2
      OUT  DX,AL
      SUB  AX,AX
      MOV  CX,AX
      OUT  DX,AL
      ; MSB TO TIMER 2 (START TIMER)
      IN  AL,PORT_B
      OR  AL,1
      OUT PORT_B,AL
      ; ENABLE TIMER 2
      IM  AL,PORT_C
      AND AL,01000000B
      ; SEE IF KEYBOARD DATA ACTIVE
      JNZ F7_1
      ; EXIT LOOP IF DATA SHOWED UP
      LOOP F7_0
      MOV  BL,02H
      ; SET NO KEYBOARD DATA ERROR
      JMP  SHORT F6_1

F7_1:  PUSH  ES
      SUB  AX,AX
      MOV  ES,AX
      ; SAVE ES
      ; SET UP SEGMENT REG
      MOV  ES:[NMI_PTR],OFFSET D11
      ; SET UP NEW NMI VECTOR
      MOV  INTR_FLAG,AL
      ; RESET INTR FLAG
      IN  AL,PORT_B
      ; DISABLE INTERNAL BEEPER TO
      OR  AL,00110000B
      ; PREVENT ERROR BEEP
      OUT PORT_B,AL
      MOV  AL,0C0H
      ; ENABLE NMI
      OUT  0A0H,AL
      MOV  CX,0100H

```

```

06CD E2 FE
06CF E4 61
06D1 24 CF
06D3 E6 61
06D5 A0 0084 R
06DB 0A 0C
08DA B3 03
06DC 26: C7 06 0006 R OF78 R
06E3 07
06E4 74 14
06E6 B0 00
06EB E6 A0
06EA E4 61
06EC 24 FE
06EE E6 61
06FO E4 62
06F2 24 40
06F4 74 09
06F8 E2 F8
06F9 B3 01
06FA B7 21
06FC E9 065F R
06FF B0 00
0701 E6 A0
F6_0: LOOP F6_0 ; WAIT A BIT
      IN AL,PORT_B ; RE-ENABLE BEEPER
      AND AL,1100111B
      OUT PORT_B,AL
      MOV AL,INTR_FLAG ; GET INTR FLAG
      OR AL,AL ; WILL BE NON-ZERO IF NMI HAPPENED
      MOV BL,03H ; SET POSSIBLE ERROR CODE
      MOV ES:(NMI_PTR),OFFSET KBONMI ; RESET NMI VECTOR
      POP ES ; RESTORE ES
      JZ F6_1 ; JUMP IF NO NMI
      MOV AL,00H ; DISABLE FEEDBACK CKT
      OUT OAOH,AL
      IN AL,PORT_B
      AND AL,1111110B ; DROP GATE TO TIMER 2
      OUT PORT_B,AL
F6_2: IN AL,PORT_C ; SEE IF KEYBOARD DATA ACTIVE
      AND AL,01000000B
      JZ F6_X ; EXIT LOOP IF DATA WENT LOW
      LOOP F6_2
      MOV BL,01H ; SET KEYBOARD DATA STUCK HIGH ERR
      MOV BH,21H ; POST ERROR "21XX"
F6_1: JMP F6
F6_X: MOV AL,00H ; DISABLE FEEDBACK CKT
      OUT OAOH,AL

```

```

-----
; CASSETTE INTERFACE TEST
; DESCRIPTION
; TURN CASSETTE MOTOR OFF. WRITE A BIT OUT TO THE
; CASSETTE DATA BUS. VERIFY THAT CASSETTE DATA
; READ IS WITHIN A VALID RANGE
; MFG. ERROR CODE=2300H (DATA PATH ERROR)
; 23FF (RELAY FAILED TO PICK)
-----

```

```

= 0A9A
= 0BAD
MAX_PERIOD EQU 0A9AH ; NOM.+10%
MIN_PERIOD EQU 0BADH ; NOM.-10%
;----- TURN THE CASSETTE MOTOR OFF
0703 EB E6DB R CALL MFG_UP ; MFG CODE=F1
0706 E4 61 IN AL,PORT_B
0709 0C 09 OR AL,00001001B ; SET TIMER 2 SPK OUT, AND CASSETTE
070A E6 61 OUT PORT_B,AL ; OUT BITS-ON, CASSETTE NOT OFF
;----- WRITE A BIT
070C E4 21 IN AL,INTA01
070E 0C 01 OR AL,01H ; DISABLE TIMER INTERRUPTS
0710 E6 21 OUT INTA01,AL
0712 B0 86 MOV AL,086H ; SEL TIM 2, LSB, MSB, MD 3
0714 E6 43 OUT TIMER+3,AL ; WRITE B253 CMD/MODE REG
0716 BB 04D2 MOV AX,1234 ; SET TIMER 2 CNT FOR 1000 USEC
0719 E6 42 OUT TIMER+2,AL ; WRITE TIMER 2 COUNTER REG
071B 9A C4 MOV AL,AH ; WRITE MSB
071D E6 42 OUT TIMER+2,AL
071F 2B C9 SUB CX,CX ; CLEAR COUNTER FOR LONG DELAY
0721 E2 FE LOOP $ ; WAIT FOR COUNTER TO INIT
;----- READ CASSETTE INPUT
0723 E4 62 IN AL,PORT_C ; READ VALUE OF CASS IN BIT
0725 24 10 AND AL,10H ; ISOLATE FROM OTHER BITS
0727 A2 0068 R MOV LAST_VAL,AL
072A EB F96F R CALL READ_HALF_BIT ; TO SET UP CONDITIONS FOR CHECK
072D EB F96F R CALL READ_HALF_BIT
0730 E3 3E JCXZ F8 ; CAS_ERR
0732 53 PUSH BX ; SAVE HALF BIT TIME VALUE
0733 EB F96F R CALL READ_HALF_BIT
0736 59 POP AX ; GET TOTAL TIME
0737 E3 37 JCXZ F8 ; CAS_ERR
0739 03 C3 ADD AX,BX
073B 3D 0A9A CMP AX,MAX_PERIOD ; CAS_ERR
073E 73 30 JNC F8
0740 3D 0BAD CMP AX,MIN_PERIOD
0743 72 2B JC F8
0746 BA 0201 MOV DX,201H
0748 EC IN AL,DX
0749 24 F0 AND AL,0F0H ; DETERMINE MODE
074B 3C 10 CMP AL,00010000B ; MFG?
074D 74 04 JE F9
074F 3C 40 CMP AL,01000000B ; SERVICE?
0751 75 26 JNE T13_END ; GO TO NEXT TEST IF NOT
; CHECK THAT CASSETTE RELAY IS PICKING (CAN'T DO TEST IN NORMAL
; MODE BECAUSE OF POSSIBILITY OF WRITING ON CASSETTE IF "RECORD"
; BUTTON IS DEPRESSED.)
F9: IN AL,PORT_B
      MOV DL,AL ; SAVE PORT B CONTENTS
      AND AL,1100101B ; SET CASSETTE MOTOR ON
      OUT PORT_B,AL
      XOR CX,CX
F91: LOOP F91 ; WAIT FOR RELAY TO SETTLE
      CALL READ_HALF_BIT
      CALL READ_HALF_BIT
      MOV AL,DL ; DROP RELAY
      OUT PORT_B,AL
      JCXZ T13_END ; READ_HALF_BIT SHOULD TIME OUT IN
; THIS SITUATION
; ERROR 23FF
F8: MOV BX,23FFH
      JMP SHORT F81 ; CAS_ERR
; ERR. CODE 2300H
F81: MOV SI,OFFSET CAS_ERR ; CASSETTE WRAP FAILED
      CALL E_MSG ; GO PRINT ERROR MSG
T13_END: IN AL,INTA01
        AND AL,0FEH ; ENABLE TIMER INTS
        OUT INTA01,AL
        IN AL,NMI_PORT ; CLEAR NMI FLIP/FLOP
        MOV AL,B0H ; ENABLE NMI INTERRUPTS
        OUT NMI_PORT,AL

```

 SERIAL PRINTER AND MODEM POWER ON DIAGNOSTIC
 DESCRIPTION:

VERIFIES THAT THE SERIAL PRINTER UART FUNCTIONS PROPERLY.
 CHECKS IF THE MODEM CARD IS ATTACHED. IF IT'S NOT, EXITS.
 VERIFIES THAT THE MODEM UART FUNCTIONS PROPERLY.
 ERROR CODES RETURNED BY 'UART' RANGE FROM 1 TO 1FH AND ARE
 REPORTED VIA REGISTER BL. SEE LISTING OF 'UART' (POD27)
 FOR POSSIBLE ERRORS.
 MFG. ERR. CODES 23XX FOR SERIAL PRINTER
 24XX FOR MODEM

 ASSUME CS:CODE,DS:DATA

TEST SERIAL PRINTER INSB250 UART

```

0785 EB E608 R CALL MFG_UP ; MFG ROUTINE INDICATOR=FO
0788 9A 02F8 MOV DX,02F8H ; ADDRESS OF SERIAL PRINTER CARD
0789 EB E631 R CALL UART ; ASYNCH. COMM. ADAPTER POD
078E 73 06 JNC TM ; PASSED
0780 BE 0038 R MOV SI,OFFSET COM1_ERR ; CODE FOR DISPLAY
0793 EB 098C R CALL E_MSG ; REPORT ERROR
  
```

TEST MODEM INSB250 UART

```

0796 EB E606 R TH: CALL MFG_UP ; MFG ROUTINE INDICATOR = EF
0798 E4 62 IN AL,F0RT_C ; TEST FOR MODEM CARD PRESENT
0798 24 02 AND AL,0000010B ; ONLY CONCERNED WITH BIT 1
0790 75 0E JNE TM1 ; IT'S NOT THERE - DONE WITH TEST
079F BA 03FB MOV DX,03FBH ; ADDRESS OF MODEM CARD
07A2 EB E831 R CALL UART ; ASYNCH. COMM. ADAPTER POD
07A5 73 06 JNC TM1 ; PASSED
07A7 BE 0039 R MOV SI,OFFSET COM2_ERR ; MODEM ERROR
07AA EB 098C R CALL E_MSG ; REPORT ERROR
07AD
  
```

TM1:

SETUP HARDWARE INT. VECTOR TABLE

```

07AD 2B C0 ASSUME CS:CODE,OS:ABSO
07AF BE C0 SUB AX,AX
07B1 B9 0008 MOV ES,AX
07B4 0E MOV CX,08 ; GET VECTOR CNT
07B5 1F PUSH CS ; SETUP DS SEG REG
07B6 BE FEF3 R POP DS
07B9 BF 0020 R MOV SI,OFFSET VECTOR_TABLE
07BC A5 MOV DI,OFFSET INT_PTR
F7A: 07B0 47 INC DI ; SKIP OVER SEGMENT
07BE 47 INC DI
07BF E2 FB LOOP F7A
  
```

SET UP OTHER INTERRUPTS AS NECESSARY

```

07C1 BE D9 ASSUME DS:ABSO
07C3 C7 06 0014 R FF54 R MOV INTS_PTR,OFFSET PRINT_SCREEN ; PRINT SCREEN
07C9 C7 06 0120 R 10C6 R MOV KEY62_PTR,OFFSET KEY62_INT ; 62 KEY CONVERSION
; ROUTINE
07CF C7 06 0110 R FAGE R MOV CSET_PTR,OFFSET CRT_CHAR_GEN ; DOT TABLE
07D5 C7 06 0060 R FFCB R MOV BASIC_PTR,OFFSET BAS_ENT ; CASSETTE BASIC ENTRY
07DB 0E PUSH CS
07DC 98 POP AX
07DD A3 0062 R MOV WORD PTR BASIC_PTR+2,AX ; CODE SEGMENT FOR CASSETTE
  
```

 CHECK FOR OPTIONAL ROM FROM C000 TO F000 IN 2K BLOCKS
 (A VALID MODULE HAS '55AA' IN THE FIRST 2 LOCATIONS,
 LENGTH INDICATOR (LENGTH/512) IN THE 3D LOCATION AND
 TEST/INIT. CODE STARTING IN THE 4TH LOCATION.)
 MFG ERR CODE 25XX (XX=MSB OF SEGMENT THAT HAS CRC CHECK)

```

07E0 B0 01 MOV AL,01H
07E2 E6 13 OUT 13H,AL
07E4 EB E6D8 R CALL MFG_UP ; MFG ROUTINE = EE
07E7 BA C000 MOV DX,0C000H ; SET BEGINNING ADDRESS
ROM_SCAN_1:
07EA 8E DA MOV DS,DX
07EC 2B DB SUB BX,BX ; SET BX=0000
07EE BB 07 MOV AX,[BX] ; GET 1ST WORD FROM MODULE
07F0 53 PUSH BX
07F1 5B POP BX ; BUS SETTLING
07F2 3D AAB5 CMP AX,0AAB5H ; = TO 1D WORD?
07F5 75 05 JNZ NEXT_ROM ; PROCEED TO NEXT ROM IF NOT
07F7 EB EB51 R CALL ROM_CHECK ; GO CHECK OUT MODULE
07FA EB 04 JMP SHORT ARE_WE_DONE ; CHECK FOR END OF ROM SPACE
NEXT_ROM:
07FC B1 C2 00B0 ADD DX,00B0H ; POINT TO NEXT 2K ADDRESS
ARE_WE_DONE:
0800 81 FA F000 CMP DX,0F000H ; AT F0000.YET?
0804 7C E4 JL ROM_SCAN_1 ; GO CHECK ANOTHER ADD. IF NOT
  
```


DISKETTE ATTACHMENT TEST

DESCRIPTION

CHECK IF IPL DISKETTE DRIVE IS ATTACHEO TO SYSTEM. IF ATTACHED, VERIFY STATUS OF NEC FDC AFTER A RESET. ISSUE A RECAL AND SEEK CMD TO FDC AND CHECK STATUS. COMPLETE SYSTEM INITIALIZATION THEN PASS CONTROL TO THE BOOT LOADER PROGRAM.

MFG ERR CODES: 2601 RESET TO DISKETTE CONTROLLER CD. FAILED
2602 RECALIBRATE TO DISKETTE DRIVE FAILED
2603 WATCHDOG TIMER FAILED

```

0806 EB E6D8 R          ASSUME CS:CODE,DS:DATA
0808 EB 138B R          CALL MFG_UP           ; MFG ROUTINE = ED
080C 80 FF             CALL DDS            ; POINT TO DATA AREA
080E A2 0074 R         MOV AL,OFFH
0811 A2 0075 R         MOV TRACK0,AL      ; INIT DISKETTE SCRATCHPADS
0814 A2 0076 R         MOV TRACK1,AL
0817 E4 E2           IN AL,PORT_C      ; DISKETTE PRESENT?
0818 24 04           ANO AL,00000100B
081B 74 03           JZ F10_0
081D E9 08A3 R        JMP F15             ; NO - BYPASS DISKETTE TEST
0820 80 0E 0010 R 01  F10_0: OR BYTE PTR EQUIP_FLAG,01H ; SET IPL DISKETTE
                                ; INDICATOR IN EQUIP. FLAG
0825 83 3E 0072 R 00  CMP RESET_FLAG,0    ; RUNNING FROM POWER-ON STATE?
082A 75 0E           JNE F10             ; BYPASS WATCHDOG TEST
082C 80 04           MOV AL,00000100B   ; READ INT. REQUEST REGISTER CMD
082E E8 20           OUT INTA00,AL
0830 E4 20           IN AL,INTA00
0832 24 40           ANO AL,01000000B   ; HAS WATCHDOG GONE OFF?
0834 75 04           JNZ F10             ; PROCEED IF IT HAS
0838 83 03           MOV BL,03H         ; SET ERROR CODE
0839 E8 33           JMP SHORT F13
083A 80 80           F10: MOV AL,FDC_RESET
083C E6 F2           OUT OF2H,AL        ; DISABLE WATCHDOG TIMER
083E 84 00           MOV AH,0           ; RESET NEC FDC
0840 8A D4           MOV DL,AH          ; SET FOR DRIVE 0
0842 CD 13           INT 13H            ; VERIFY STATUS AFTER RESET
0844 F6 C4 FF        TEST AH,OFFH       ; STATUS OK?
0847 83 01           MOV BL,01H         ; SET UP POSSIBLE ERROR CODE
0849 75 22           JNZ F13            ; NO - FDC FAILED
                                ;----- TURN DRIVE 0 MOTOR ON
084B 80 81           MOV AL,DRIVE_ENABLE+FDC_RESET ; TURN MOTOR ON,DRIVE 0
084D E6 F2           OUT OF2H,AL        ; WRITE FDC CONTROL REG
084F 28 C8           SUB CX,CX
0851 E2 FE           F11: LOOP F11        ; WAIT FOR 1 SECOND
0853 E2 FE           F12: LOOP F12
0855 33 D2           XOR DX,DX          ; SELECT DRIVE 0
0857 85 01           MOV CH,1           ; SELECT TRACK 1
0859 8B 16 003E R    MOV DL,SEEK_STATUS,DL
085D EB EBF8 R        CALL SEEK          ; RECALIBRATE DISKETTE
0860 83 02           MOV BL,02H        ; ERROR CODE
0862 72 08           JC F13             ; GO TO ERR SUBROUTINE IF ERR
0864 85 22           MOV CH,34          ; SELECT TRACK 34
0866 EB E9FB R        CALL SEEK          ; SEEK TO TRACK 34
0869 73 0A           JNC F14            ; OK, TURN MOTOR OFF
086B 83 02           MOV BL,02H
086D 87 26           F13: MOV BH,26H        ; DSK_ERR:(26XX)
086F 8E 003C R       MOV SI,OFFSET DISK_ERR ; GET ADDR OF MSG
0872 E8 09BC R        CALL E_MSG         ; GO PRINT ERROR MSG
0875 80 82           F14: MOV AL,FDC_RESET+02H
0877 E6 F2           OUT OF2H,AL
0879 E4 E2           IN AL,0E2H
087B 24 06           ANO AL,00000110B
087D 3C 02           CMP AL,00000010B
087F 75 1E           JNE F14_1
0881 80 84           MOV AL,FDC_RESET+04H
0883 E6 F2           OUT OF2H,AL
0885 E4 E2           IN AL,0E2H
0887 24 06           ANO AL,00000110B
0889 3C 04           CNP AL,00000100B
088B 75 12           JNE F14_1
088D E4 E2           IN AL,0E2H
088F 24 30           AND AL,00110000B
0891 74 0C           JZ F14_1
0893 3C 10           CMP AL,00010000B
0895 84 40           MOV AH,01000000B
0897 74 02           JE F14_2
0899 84 80           MOV AH,10000000B
089B 08 26 0010 R   F14_2: OR BYTE PTR EQUIP_FLAG,AH
                                ;----- TURN DRIVE 0 MOTOR OFF
089F 80 80           F14_1: MOV AL,FDC_RESET ; TURN DRIVE 0 MOTOR OFF
08A1 E6 F2           OUT OF2H,AL
08A3 C6 06 0084 R 00  F15: MOV INTR_FLAG,00H ; SET STRAY INTERRUPT FLAG = 00
08A8 BF 0078 R        MOV DI,OFFSET PRINT_TIM_OUT ; SET DEFAULT PRT TIMEOUT
08AB 1E           PUSH DS
08AC 07           POP ES
08AD 8B 1414         MOV AX,1414H      ; DEFAULT=20
08B0 AB           STOSB
08B1 AB           STOSW
08B2 8B 0101         MOV AX,0101H      ; RS232 DEFAULT=01
08B5 AB           STOSB
08B6 AB           STOSW
08B7 E4 21           IN AL,INTA01
08B9 24 FE           AND AL,0FEH      ; ENABLE TIMER INT. (LVL 0)
08BB E6 21           OUT INTA01,AL
                                ASSUME DS:XXDATA
08BD 1E           PUSH DS
08BE 8B           MOV AX,XXDATA
08C1 9E D8           MOV DS,AX

```

```

08C3 80 3E 001B R 00      CMP      POST_ERR,00H ; CHECK FOR "POST_ERR" NON-ZERO
                                ASSUME DS:DATA
08C8 1F                    POP      DS
08C8 74 10                  JE       F15A_0 ; CONTINUE IF NO ERROR
08C8 B2 02                  MOV      DL,2 ; 2 SHORT BEEPS (ERROR)
08CD EB 1A0C R              CALL     ERR_BEEP
08D0                        ERR_WAIT:
08D0 B4 00                  MOV      AH,00
08D2 CD 16                  INT      16H ; WAIT FOR "ENTER" KEY
08D4 90 FC 1C              CMP      AH,ICH
08D7 75 F7                  JNE     ERR_WAIT
08D8 EB 05                  JMP      SHORT F15C
08DB B2 01                  F15A_0: MOV  DL,1 ; 1 SHORT BEEP (NO ERRORS)
08DD EB 1A0C R              CALL     ERR_BEEP
08E0 BD 003D R              ;----- SETUP PRINTER AND RS232 BASE ADDRESSES IF DEVICE ATTACHED
08E3 33 F6                  F15:    MOV  BP,OFFSET F4 ; PRT_SRC_TBL
                                XOR   SI,SI
08E5                        F16:    MOV  DX,CS:[BP] ; PRT_BASE:
                                MOV  AL,0AAH ; GET PRINTER BASE ADDR
                                OUT  DX,AL ; WRITE DATA TO PORT A
08EC 1E                    PUSH   DS ; BUS SETTling
08ED EC                    IN     AL,DX ; READ PORT A
08EE 1F                    POP    DS
08EF 3C AA                  CMP    AL,0AAH ; DATA PATTERN SAME
08F1 75 06                  JNE   F17 ; NO - CHECK NEXT PRT CD
08F3 B9 94 000B R          MOV    PRINTER_BASE[SI],DX ; YES - STORE PRT BASE ADDR
08F7 46                    INC   SI ; INCREMENT TO NEXT WORD
08F8 46                    INC   SI
08F9 45                    F17:   INC   BP ; POINT TO NEXT BASE ADDR
08FA 45                    INC   BP
08FB 93 FD 41              CMP    BP,OFFSET F4E ; ALL POSSIBLE ADDRS CHECKED?
08FE 75 E5                  JNE   F16 ; PRT_BASE
0900 33 DB                  XOR    BX,BX ; SET ADDRESS BASE
0902 BA 03FA                MOV    DX,03FAH ; POINT TO INT ID REGISTER
0905 EC                    IN     AL,DX ; READ PORT
0908 AB F8                  TEST   AL,0FBH ; SEEM TO BE AN B250
0909 76 08                  JNZ   F18
090A C7 87 0000 R 03FB    MOV    RS232_BASE[8X],3FBH ; SETUP RS232 CD #1 ADDR
0910 43                    INC   BX
0911 43                    INC   BX
0912 C7 87 0000 R 02FB    MOV    RS232_BASE[8X],2FBH ; SETUP RS232 #2
0918 43                    INC   BX ; (ALWAYS PRESENT)
0919 43                    INC   BX
                                ;----- SET UP EQUIP FLAG TO INDICATE NUMBER OF PRINTERS AND RS232
                                ; CARDS
091A 8B C6                  MOV    AX,SI ; SI HAS 2# NUMBER OF PRINTERS.
091C 81 03                  MOV    CL,3 ; SHIFT COUNT
091E D2 C8                  ROR    AL,CL ; ROTATE RIGHT 3 POSITIONS
0920 0A C3                  OR     AL,BL ; OR IN THE RS232 COUNT
0922 0B 06 0011 R          OR     BYTE PTR EQUIP_FLAG+1,AL ; STORE AS SECOND BYTE
                                ;----- SET EQUIP. FLAG TO INDICATE PRESENCE OF SERIAL PRINTER
                                ; ATTACHED TO ON BOARD RS232 PORT. ---ASSUMPTION---"RTS" IS TIED TO
                                ; "CARRIER DETECT" IN THE CABLE PLUG FOR THIS SPECIFIC PRINTER.
0926 BB CB                  MOV    CX,AX ; SAVE PRINTER COUNT IN CX
0928 BB 02FE                MOV    BX,2FEH ; SET POINTER TO MODEM STATUS REG
0929 BA 02FC                MOV    DX,2FCH ; POINT TO MODEM CONTROL REG
092E 2A C0                  SUB    AL,AL
0930 EE                    OUT    OX,AL ; CLEAR IT
0931 EB 00                  JMP    $+2 ; DELAY
0933 B7 D3                  XCHG  OX,BX ; POINT TO MODEM STATUS REG
0935 EC                    IM     AL,DX ; CLEAR IT
0936 EB 00                  JMP    $+2 ; DELAY
0938 B0 02                  MOV    AL,02H ; BRING UP RTS
093A B7 D3                  XCHG  DX,BX ; POINT TO MODEM CONTROL REG
093C EE                    OUT    DX,AL
093D EB 00                  JMP    $+2 ; DELAY
093F B7 D3                  XCHG  DX,BX ; POINT TO MODEM STATUS REG
0941 EC                    IN     AL,DX ; GET CONTENTS
0942 AB 08                  TEST   AL,00001000B ; HAS CARRIER DETECT CHANGED?
0944 74 23                  JZ     F19_A ; NO, THEN NO PRINTER
0946 AB 01                  TEST   AL,00000001B ; DID CTS CHANGE? (A9 WITH WRAP
                                ; CONNECTOR INSTALLED)
0948 75 1F                  JNZ   F19_A ; WRAP-CONNECTOR ON IF IT DID
094A 2A C0                  SUB    AL,AL ; SET RTS OFF
094C B7 D3                  XCHG  DX,BX ; POINT TO MODEM CONTROL REG
094E EE                    OUT    DX,AL ; DROP RTS
094F EB 00                  JMP    $+2 ; DELAY
0951 B7 D3                  XCHG  DX,BX ; MODEM STATUS REG
0953 EC                    IN     AL,DX ; GET STATUS
0954 AB 08                  AND    AL,00001000B ; HAS CARRIER DETECT CHANGED?
0956 74 11                  JZ     F19_A ; NO, THEN NO PRINTER
0958 80 C9 20              ; CARRIER DETECT IS FOLLOWING RTS- INDICATE SERIAL PRINTER ATTACHED
0959 F6 C1 C0              OR     CL,00100000B
095E 75 09                  TEST   CL,11000000B ; CHECK FOR NO PARALLEL PRINTERS
                                ; DO NOTHING IF PARALLEL PRINTER
                                ; ATTACHED
0960 B0 C9 40              OR     CL,01000000B ; INDICATE 1 PRINTER ATTACHED.
0963 C7 06 000B R 02F8    MOV    PRINTER_BASE,2F9H ; STORE ON-BOARD RS232 BASE IN
                                ; PRINTER BASE
0969 08 0E 0011 R          F19_A: OR     BYTE PTR EQUIP_FLAG+1,CL ; STORE AS SECOND BYTE
096B 33 D2                  XOR    DX,DX ; POINT TO FIRST SERIAL PORT
096F F6 C1 40              TEST   CL,040H ; SERIAL PRINTER ATTACHED?
0972 74 18                  JZ     F19_C ; NO, SKIP INIT
0974 B1 3E 0000 R 02FB    CMP    RS232_BASE,02F9H ; PRINTER IN FIRST SERIAL PORT
097A 74 01                  JE     F19_B ; YES, JUMP
097C 42                    INC   DX ; NO POINT TO SECOND SERIAL PORT
097D B8 00B7              F19_B: MOV  AX,87H ; INIT SERIAL PRINTER
0980 CD 14                  INT      14H
0982 F6 C4 1E              TEST   AH,1EH ; ERROR?
0985 75 05                  JNZ   F19_C ; YES - JUMP
0987 B8 0118                MOV    AX,0118H ; SEND CANCEL COMMAND TO
098A CD 14                  INT      14H ; SERIAL PRINTER

```

```

098C BA 0201
098F EC
0990 24 F0
0992 75 03
0994 E9 0043 R
0997 3C 20
0999 74 F9
099B B1 3E 0072 R 4321
09A1 74 0C
09A3 3C 10
09A5 74 08
09A7 C7 06 0072 R 1234

09AD CD 19

09AF FA
09B0 28 C0
09B2 8E D8
09BA C7 06 0020 R FE45 R
09B4 CD 80

```

```

F19_C: MOV DX,0201H
        IN AL,DX ; GET MFG. / SERVICE MODE INFO
        AND AL,0FOH ; 15 HIGH ORDER NIBBLE = 0?
        JNZ F19_1 ; (BURN-IN MODE)
F19_0: JMP START ; ELSE GO TO BEGINNING OF POST
F19_1: CMP AL,00100000B ; SERVICE MODE LOOP?
        JE F19_0 ; BRANCH TO START
        CMP RESET_FLAG,4321H ; DIAG. CONTROL PROGRAM RESTART?
        JE F19_3 ; NO, GO BOOT
        CMP AL,00010000B ; MFG DCP RUN REQUEST
        JE F19_3
        MOV RESET_FLAG,1234H ; SET WARM START INDICATOR IN CASE
        INT 19H ; OF CARTRIDGE RESET
        ASSUME DS:ABS0 ; GD TO THE BOOT LOADER

F19_3: CLI
        SUB AX,AX
        MOV DS,AX ; RESET TIMER INT.
        MOV INT_PTR,OFFSET TIMER_INT
        INT 80H ; ENTER DCP THROUGH INT. 80H

```

----- THIS SUBROUTINE IS THE GENERAL ERROR HANDLER FOR THE POST -----

```

; ENTRY REQUIREMENTS:
; S1 = OFFSET(ADDRESS) OF MESSAGE BUFFER
; BX= ERROR CODE FOR MANUFACTURING OR SERVICE MODE
; REGISTERS ARE NOT PRESERVED
; LOCATION "POST_ERR" IS SET NON-ZERO IF AN ERROR OCCURS IN
; CUSTOMER MODE
; SERVICE/MANUFACTURING FLAGS AS FOLLOWS: (HIGH NIBBLE OF
; PORT 201)
; 0000 = MANUFACTURING (BURN-IN) MODE
; 0001 = MANUFACTURING (SYSTEM TEST) MODE
; 0010 = SERVICE MODE (LOOP POST)
; 0100 = SERVICE MODE (SYSTEM TEST)
;-----

```

```

098C BA 0201
098F EC
09C0 24 F0
09C2 75 03
09C4 E9 0A61 R
09C7 3C 10
09C9 75 03
09CB E9 0A61 R
09CE BA F0
09D0 80 FF 0A

08D3 7C 63
09D5 53
09D6 56
09D7 52
09D8 84 02
09DA BA 1521
09DD B7 07
09DF C0 10
09E1 8E 0030 R
09E4 89 0005
09E7 2E: 8A 04
09EA 46
09EB E8 18BA R
09EE E2 F7

```

```

E_MSG PROC NEAR
        MOV DX,201H
        IN AL,DX ; GET MODE BITS
        AND AL,0FOH ; ISOLATE BITS OF INTEREST
        JNZ EM0
        JMP MFG_OUT ; MANUFACTURING MODE (BURN-IN)
EM0: CMP AL,00010000B
        JNE EM1
        JMP MFG_OUT ; MFG. MODE (SYSTEM TEST)
EM1: MOV DH,AL ; SAVE MODE
        MOV BH,0AH ; ERROR CODE ABOVE 0AH (CRT STARTED
        ; DISPLAY POSSIBLE)?
        JL BEEPS ; DO BEEP OUTPUT IF BELOW 10H
        PUSH BX ; SAVE ERROR AND MODE FLAGS
        PUSH SI
        PUSH DX
        MOV AH,2 ; SET CURSOR
        MOV DX,1B21H ; ROW 21, COL.33
        MOV BH,7 ; PAGE 7
        INT 10H
        MOV SI,OFFSET ERROR_ERR
        MOV CX,5 ; PRINT WORD "ERROR"
EM_0: MOV AL,CS:[SI]
        INC SI
        CALL PRT_HEX
        LOOP EM_0
; LOOK FOR A BLANK SPACE TO POSSIBLY PUT CUSTOMER LEVEL ERRORS (IN
; CASE OF MULTI ERROR)
        MOV DH,16H
EM_1: MOV AH,2 ; SET CURSOR
        INT 10H ; ROW 22, COL.33 (OR ABOVE, IF
        ; MULTIPLE ERRS)
        MOV AH,B ; READ CHARACTER THIS POSITION
        INT 10H
        INC DL ; POINT TO NEXT POSITION
        CMP AL,' ' ; BLANK?
        JNE EM_1 ; GO CHECK NEXT POSITION, IF NOT
        POP DX ; RECOVER ERROR POINTERS
        POP BX
        CMP DH,00100000B ; SERVICE MODE?
        JE SERV_OUT
        CMP DH,01000000B
        JE SERV_OUT
        MOV AL,CS:[SI] ; GET ERROR CHARACTER
        CALL PRT_HEX ; DISPLAY IT
        CMP BH,20H ; ERROR BELOW 20? (MEM TROUBLE?)
        JNL EM_2
        JMP TOTLTP0 ; HALT SYSTEM IF SO.
        ASSUME DS:XXDATA
EM_2: PUSH DS
        PUSH AX
        MOV AX,XXDATA
        MOV DS,AX
        MOV POST_ERR,BH ; SET ERROR FLAG NON-ZERO
        POP AX
        POP DS
        ASSUME DS:NOTHING
        RET ; RETURN TO CALLER

```

```

09F0 86 16
09F2 84 02
09F4 CD 10

09F6 84 08
09FB CD 10
09FA FE C2
09FC 3C 20
09FE 75 F2
0A00 5A
0A01 5E
0A02 58
0A03 80 FE 20
0A08 74 21
0A0B 80 FE 40
0A08 74 1C
0A0D 2E: 8A 04
0A10 E8 18BA R
0A13 90 FF 20
0A16 7D 03
0A18 E9 0ABB R

0A18 1E
0A1C 50
0A1D B8 ---- R
0A20 8E D8
0A22 98 3E 0018 R
0A26 58
0A27 1F
0A28 C3

```

```

        MOV DH,16H
EM_1: MOV AH,2 ; SET CURSOR
        INT 10H ; ROW 22, COL.33 (OR ABOVE, IF
        ; MULTIPLE ERRS)
        MOV AH,B ; READ CHARACTER THIS POSITION
        INT 10H
        INC DL ; POINT TO NEXT POSITION
        CMP AL,' ' ; BLANK?
        JNE EM_1 ; GO CHECK NEXT POSITION, IF NOT
        POP DX ; RECOVER ERROR POINTERS
        POP BX
        CMP DH,00100000B ; SERVICE MODE?
        JE SERV_OUT
        CMP DH,01000000B
        JE SERV_OUT
        MOV AL,CS:[SI] ; GET ERROR CHARACTER
        CALL PRT_HEX ; DISPLAY IT
        CMP BH,20H ; ERROR BELOW 20? (MEM TROUBLE?)
        JNL EM_2
        JMP TOTLTP0 ; HALT SYSTEM IF SO.
        ASSUME DS:XXDATA
EM_2: PUSH DS
        PUSH AX
        MOV AX,XXDATA
        MOV DS,AX
        MOV POST_ERR,BH ; SET ERROR FLAG NON-ZERO
        POP AX
        POP DS
        ASSUME DS:NOTHING
        RET ; RETURN TO CALLER

```

```

0A29
0A29 BA C7          SERV_OUT: MOV AL,BH          ; PRINT MSB
0A28 53             PUSH BX
0A2C E9 18A9 R      CALL XPC_BYTE     ; DISPLAY IT
0A2F 5B             POP BX
0A30 BA C3          MOV AL,BL
0A32 E9 18A9 R      CALL XPC_BYTE     ; PRINT LSB
0A35 E9 0A8B R      JMP TOTLTP0
0A38 FA
0A39 BC C8          BEEPS: CLM
0A3B 8E D0          MOV AX,C9        ; SET CODE SEG= STACK SEG
; (STACK IS LOST, BUT THINGS ARE
; OVER, ANYWAY)
; 2 BEEPS
0A3D B2 02          MOV DL,2
0A3F BC 0028 R      MOV SP,OFFSET EX_0 ; SET DUMMY RETURN
0A42 53 01          EB: MOV BL,1       ; SHORT BEEP
0A44 E9 FF31 R      JMP BEEP
0A47 E2 FE          EBO: LOOP EBO     ; WAIT (BEEPER OFF)
0A49 FE CA          DEC DL           ; DONE YET?
0A4B 75 F5          JNZ EB          ; LOOP IF NOT
0A4D 80 FF 05       CMP BH,05H      ; 64K CARD ERROR?
0A50 75 89          JNE TOTLTP0    ; END IF NOT
0A52 80 FE 20       CMP DH,00100000B ; SERVICE MODE?
0A55 74 05          JE EB1
0A57 80 FE 40       CMP DH,01000000B ;
0A5A 75 5F          JNE TOTLTP0
0A5C B3 01          EB1: MOV BL,1     ; END IF NOT
; ONE MORE BEEP FOR 64K ERROR IF IN
; SERVICE MODE
0A5E E9 FF31 R      JMP BEEP
0A61
0A61 FA             MFG_OUT: CLM
0A62 E4 81          IN AL,PORT_B
0A64 24 FC          AND AL,0FCH
0A66 E6 81          OUT PORT_B,AL
0A6B BA 0011        MOV AL,BH       ; SEND DATA TO ADDRESSES 11, 12
0A6B BA C7          MOV AL,BH
0A6D EE           OUT DX,AL       ; SEND HIGH BYTE
0A6E 42            INC DX
0A6F BA C3          MOV AL,BL
0A71 EE           OUT DX,AL       ; SEND LOW BYTE
; INIT. ON-BOARD RS232 PORT FOR COMMUNICATIONS W/MFG MONITOR
0A72 BB ---- R     ASSUME DS:XXDATA
0A75 8E D8          MOV AX,XXDATA
; POINT TO DATA SEGMENT CONTAINING
; CHECKPOINT #
0A77 BC C8          MOV AX,CS
0A79 BE D0          MOV SS,AX       ; SET STACK FOR RTN
0A7B BC 002E R     MOV SP,OFFSET EX1
0A7E BA 02FB        MOV DX,02FBH
0A81 E9 F085 R      JMP $B250
0A84 BB CA          M01: MOV CX,DX
; ADDRESS IN IT
; MODEM CONTROL REG
; SET DTR AND RTS LOW SO POSSIBLE
; WRAP PLUG WON'T CONFUSE THINGS
0A86 BA 02FC        MOV DX,02FCH
0A89 2A C0          SUB AL,AL
; MODEM STATUS REG
0A8B EE           OUT DX,AL
0A8C BA 02FE        MOV DX,02FEH
0A8F EC           M02: IN AL,DX
0A90 24 10          AND AL,00010000B ; CTS UP YET?
0A92 74 FB          JZ M02         ; LOOP TILL IT IS
0A94 4A            DEC DX         ; SET DX=2FD (LINE STATUS REG)
0A95 87 D1          XCHG DX,CX    ; POINT TO XMIT. DATA REG
0A97 A0 0005 R     MOV AL,MFG_TST ; GET MFG ROUTINE ERROR INDICATOR
0A9A EE           OUT DX,AL     ; (MAY BE WRONG FOR EARLY ERRORS)
0A9B EB 00          JMP $+2       ; DELAY
0A9D 87 D1          XCHG DX,CX    ; POINT DX=2FD
0A9F EC           M03: IN AL,DX  ; TRANSMIT EMPTY?
0AA0 24 20          AND AL,00100000B ;
0AA2 EB 00          JMP $+2       ; DELAY
0AA4 74 F9          JZ M03        ; LOOP TILL IT IS
0AA6 87 D1          XCHG DX,CX
0AA8 BA C7          MOV AL,BH     ; GET MSB OF ERROR WORD
0AAA EE           OUT DX,AL
0AAB EB 00          JMP $+2       ; DELAY
0AAD 87 D1          XCHG DX,CX
0AAF EC           M04: IN AL,DX  ; WAIT FOR XMIT EMPTY
0AB0 24 20          AND AL,00100000B ;
0AB2 EB 00          JMP $+2       ; DELAY
0AB4 74 F9          JZ M04        ;
0AB6 BA C3          MOV AL,BL     ; GET LSB OF ERROR WORD
0AB8 87 D1          XCHG DX,CX
0ABA EE           OUT DX,AL
0ABB
0ABB FA             TOTLTP0: CLM
0ABC 2A C0          SUB AL,AL     ; DISABLE INTS.
0ABE E6 F2          OUT OF2H,AL   ; STOP DISKETTE MOTOR
0AC0 E6 A0          OUT 0A0H,AL   ; DISABLE NMI
0AC2 F4            HLT           ; HALT
0AC3 C3            RET
0AC4 E_MSG ENDP

```

```

-----
SUBROUTINE TO INITIALIZE INS8250 PORTS TO THE MASTER RESET
STATUS. THIS ROUTINE ALSO TESTS THE PORTS' PERMANENT
ZERO BITS.
EXPECTS TO BE PASSED:
(DX) = ADDRESS OF THE 8250 TRANSMIT/RECEIVE BUFFER
UPON RETURN:
(CF) = 1 IF ONE OF THE PORTS' PERMANENT ZERO BITS WAS NOT
ZERO (ERR)
(DX) = PORT ADDRESS THAT FAILED TEST
(AL) = MEANINGLESS
(BL) = 2 INTR ENBL REG BITS NOT 0
3 INTR ID REG BITS NOT 0
4 MODEM CTRL REG BITS NOT 0
5 LINE STAT REG BITS NOT 0
0 IF ALL PORTS' PERMANENT ZERO BITS WERE ZERO
(DX) = TRANSMIT/RECEIVE BUFFER ADDRESS
(AL) = LAST VALUE READ FROM RECEIVER BUFFER
(BL) = 5 (MEANINGLESS)
PORTS SET UP AS FOLLOWS ON ERROR-FREE RETURN:
XFB - INTR ENBL REG = 0 ALL INTERRUPTS DISABLED
XFA - INTR ID REG = 00000001B NO INTERRUPTS PENDING
XFB - LINE CTRL REG = 0 ALL BITS LOW
XFC - MODEM CTRL REG = 0 ALL BITS LOW
XFD - LINE STAT REG = 01100000B TRANSMITTER HOLDING
REGISTER AND TRANSMITTER EMPTY ON
XFE - MODEM STAT REG = XXXX0000B WHERE X 'S REPRESENT
INPUT SIGNALS
REGISTERS DX, AL, AND BL ARE ALTERED. NO OTHER REGISTERS USED.
-----

```

```

OAC4      18250 PROC      NEAR
OAC4      EC          IN      AL,DX          ; READ RCVR BUFFER BUT IGNORE
;          ;          ;          ;          ;          ;          ;          ;
;          ;          ;          ;          ;          ;          ;          ;
OAC5      83 02      MOV     BL,2          ;          ;          ;          ;          ;
OAC7      EB FEF R   CALL    RR2          ;          ;          ;          ;          ;
OACA      24 F0      AND     AL,11110000B ;          ;          ;          ;          ;
OACC      75 2B      JNE     AT20         ;          ;          ;          ;          ;
OACE      EB FEBA R  CALL    RRI          ;          ;          ;          ;          ;
OAD1      24 FB      AND     AL,11110000B ;          ;          ;          ;          ;
OAD3      75 21      JNE     AT20         ;          ;          ;          ;          ;
OAD5      42         INC     DX           ;          ;          ;          ;          ;
OAD6      EB FEFA R  CALL    RRI          ;          ;          ;          ;          ;
OAD9      24 E0      AND     AL,11100000B ;          ;          ;          ;          ;
OADB      75 1B      JNE     AT20         ;          ;          ;          ;          ;
OADD      EB FEBA R  CALL    RRI          ;          ;          ;          ;          ;
OAE0      24 80      AND     AL,10000000B ;          ;          ;          ;          ;
OAE2      75 12      JNE     AT20         ;          ;          ;          ;          ;
OAE4      80 60      MOV     AL,60H        ;          ;          ;          ;          ;
OAE6      EE         OUT     DX,AL        ;          ;          ;          ;          ;
OAE7      EB 00      JNP     8+2          ;          ;          ;          ;          ;
OAE9      42         INC     DX           ;          ;          ;          ;          ;
OAEA      32 C0      XOR     AL,AL        ;          ;          ;          ;          ;
OAEC      EE         OUT     DX,AL        ;          ;          ;          ;          ;
OAEF      EB FEAO R  CALL    RR3          ;          ;          ;          ;          ;
;          ;          ;          ;          ;          ;          ;          ;
OAF0      83 EA 06   SUB     DX,6         ;          ;          ;          ;          ;
OAF3      EC         IN      AL,DX         ;          ;          ;          ;          ;
;          ;          ;          ;          ;          ;          ;          ;
;          ;          ;          ;          ;          ;          ;          ;
OAF4      FB         CLC                    ;          ;          ;          ;          ;
OAF5      C3         RET                    ;          ;          ;          ;          ;
OAF6      F9         STC                    ;          ;          ;          ;          ;
OAF7      C3         RET                    ;          ;          ;          ;          ;
OAF9      18250     ENDP                   ;          ;          ;          ;          ;
;          ;          ;          ;          ;          ;          ;          ;
AT20:     STC                    ;          ;          ;          ;          ;
;          ;          ;          ;          ;          ;          ;          ;
;          ;          ;          ;          ;          ;          ;          ;
18250     ENDP
-----

```

```

-----
SUBROUTINE TO TEST A PARTICULAR 8250 INTERRUPT. PASS IT THE
(BIT # + 1) OF THE STATUS REGISTER THAT IS TO BE TESTED.
THIS ROUTINE SETS THAT BIT AND CHECKS TO SEE IF THE CORRECT
8250 INTERRUPT IS GENERATED.
IT EXPECTS TO BE PASSED:
(AH) = BIT # TO BE TESTED
(BL) = INTERRUPT IDENTIFIER
(0) = RECEIVED DATA AVAILABLE OR TRANSMITTER HOLDING
REGISTER EMPTY INTERRUPT TEST
(1) = RECEIVER LINE STATUS OR MODEM STATUS INTERRUPT
TEST
(BH) = BITS WHICH DETERMINE WHICH INTERRUPT IS TO BE
CHECKED
(0) = MODEM STATUS
(2) = TRANSMITTER HOLDING REGISTER EMPTY
(4) = RECEIVED DATA AVAILABLE
(6) = RECEIVER LINE STATUS
(CX) = VALUE TO SUBTRACT AND ADD IN ORDER TO REFERENCE THE
INTERRUPT IDENTIFICATION REGISTER
(3) = RECEIVED DATA AVAILABLE, TRANSMITTER HOLDING
REGISTER AND RECEIVER LINE STATUS INTERRUPTS
(4) = MODEM STATUS INTERRUPT
(DX) = ADDRESS OF THE LINE STATUS OR MODEM STATUS REGISTER
IT RETURNS:
(AL) = OFFH IF TEST FAILS - EITHER NO INTERRUPT OCCURRED OR
THE WRONG INTERRUPT OCCURRED
OR
(AL) = CONTENTS OF THE INTERRUPT ID REGISTER FOR RECEIVED
DATA AVAILABLE AND TRANSMITTER HOLDING REGISTER
EMPTY INTERRUPTS
-OR-
CONTENTS OF THE LINE STATUS OR MODEM STATUS REGISTER
DEPENDING ON WHICH ONE WAS TESTED
(DX) = ADDRESS OF INTERRUPT ID REGISTER FOR RECEIVED DATA
AVAILABLE OR TRANSMITTER HOLDING REGISTER EMPTY
INTERRUPTS
OR
(DX) = ADDRESS OF THE LINE STATUS OR DATA SET STATUS
REGISTER (DEPENDING ON WHICH INTERRUPT WAS TESTED)
NO OTHER REGISTERS ARE ALTERED.
-----

```

```

0AFB
0AFB EC
0AF9 EB 00
0AF6 OA C4
0AFD EE
0AFE 2B D1
0B00 51
0B01 2B CB
0B03 EC
0B04 AB 01
0B06 74 02
0B08 E2 F9
0B0A 59
0B0B 3A C7
0B0D 75 09
0B0F 0A DB
0B11 74 07
0B13 03 D1
0B15 EC
0B16 EB 02
0B18 B0 FF
0B1A C3
0B1B

```

```

ICT PROC NEAR
IN AL,DX ; READ STATUS REGISTER
JMP 9*2 ; I/O DELAY
OR AL,AH ; SET TEST BIT
OUT DX,AL ; WRITE IT TO THE STATUS REGISTER
SUB DX,CX ; POINT TO INTERRUPT ID REGISTER
PUSH CX
SUB CX,CX ; WAIT FOR B250 INTERRUPT TO OCCUR
AT21: IN AL,DX ; READ INTR ID REG
TEST AL,1 ; INTERRUPT PENDING?
JE AT22 ; YES -RETURN W/ INTERRUPT ID IN AL
LOOP AT21 ; NO - TRY AGAIN
AT22: POP CX ; AL = 1 IF NO INTERRUPT OCCURRED
CMP AL,BH ; INTERRUPT WE'RE LOOKING FOR?
JNE AT23 ; NO
OR BL,BL ; DONE WITH TEST FOR THIS INTERRUPT
JE AT24 ; RETURN W/ CONTENTS OF INTR ID REG
ADD DX,CX ; READ STATUS REGISTER TO CLEAR THE
IN AL,DX ; INTERRUPT (WHEN BL=1)
JMP SHORT AT24 ; RETURN CONTENTS OF STATUS REG
AT23: MOV AL,OFFH ; SET ERROR INDICATOR
AT24: RET
ICT ENDP

```

----- INT 19 -----

```

BOOT STRAP LOADER
; TRACK 0 SECTOR 1 IS READ INTO THE
; BOOT LOCATION (SEGMENT 0, OFFSET 7C00)
; AND CONTROL 19 TRANSFERRED THERE.
;
; IF THE DISKETTE IS NOT PRESENT OR HAS A
; PROBLEM LOADING (E.G., NOT READY), AN INT.
; 1BH IS EXECUTED. IF A CARTRIDGE HAS VECTORED
; INT. 1BH TO ITSELF, CONTROL WILL BE PASSED TO
; THE CARTRIDGE.

```

----- ASSUME CS:CODE,DS:ABS0 -----

```

0B1B FB
0B1C 2B C0
0B1E CD 10
0B20 2B C0
0B22 BE DB
0B24 E4 82
0B26 24 04
0B2B 75 2B
0B2A C7 06 007B R EFC7 R
0B30 BC 0E 007A R
0B34 B9 0004
0B37 51
0B3B 84 00
0B3A CD 13
0B3C 72 0F
0B3E BB 0201
0B41 2B D2
0B43 BE C2
0B45 BB 7C00 R
0B4B B9 0001
0B4B CD 13
0B4D 59
0B4E 73 04
0B50 E2 E5
0B52 CD 1B
0B54 EA 7C00 ---- R
0B59

```

```

BOOT_STRAP PROC NEAR
STI ; ENABLE INTERRUPTS
SUB AX,AX ; SET 40X25 BBW MODE ON CRT
INT 10H ; ESTABLISH ADDRESSING
SUB AX,AX
MOV DS,AX
;----- SEE IF DISKETTE PRESENT
IN AL,PORT_C ; GET CONFIG BITS
AND AL,0000100B ; IS DISKETTE PRESENT?
JNZ H3 ; NO, THEN ATTEMPT TO GO TO CART.
;----- RESET THE DISK PARAMETER TABLE VECTOR
MOV WORD PTR DISK_POINTER, OFFSET DISK_BASE
MOV WORD PTR DISK_POINTER+2, CS
;----- LOAD SYSTEM FROM DISKETTE -- CX HAS RETRY COUNT
MOV CX,4 ; SET RETRY COUNT
H1: PUSH CX ; SAVE RETRY COUNT
MOV AH,0 ; RESET THE DISKETTE SYSTEM
INT 13H ; DISKETTE_10
JC H2 ; IF ERROR, TRY AGAIN
MOV AX,201H ; READ IN THE SINGLE SECTOR
SUB DX,DX ; TO THE BOOT LOCATION
MOV ES,DX
MOV BX,OFFSET BOOT_LOCM
; DRIVE 0, HEAD 0
; SECTOR 1, TRACK 0
MOV CX,1 ; DISKETTE_10
INT 13H ; DISKETTE_10
H2: POP CX ; RECOVER RETRY COUNT
JNC H3A ; CF SET BY UNSUCCESSFUL READ
LOOP H1 ; DO IT FOR RETRY TIMES
;----- UNABLE TO IPL FROM THE DISKETTE
H3: INT 1BH ; GO TO BASIC OR CARTRIDGE
;----- IPL WAS SUCCESSFUL
H3A: JMP BOOT_LOCM
BOOT_STRAP ENDP

```

```

; THIS ROUTINE PERFORMS A READ/WRITE TEST ON A BLOCK OF
; STORAGE (MAX. SIZE = 32KB). IF "WARM START", FILL
; BLOCK WITH 0000 AND RETURN.
; DATA PATTERNS USED:
; 0->FF ON ONE BYTE TO TEST DATA BUS
; AAAA,5555,00FF,FF00 FOR ALL WORDS
; FILL WITH 0000 BEFORE EXIT
; ON ENTRY:
; ES = ADDRESS OF STORAGE TO BE TESTED
; DS = ADDRESS OF STORAGE TO BE TESTED
; CX = WORD COUNT OF STORAGE BLOCK TO BE TESTED
; (MAX. = 8000H (32K WORDS))
; ON EXIT:
; ZERO FLAG = OFF IF STORAGE ERROR
; IF ZERO FLAG = OFF, THEN CX = XOR'ED BIT PATTERN
; OF THE EXPECTED DATA PATTERN VS. THE ACTUAL DATA
; READ. (I.E., A BIT "ON" IN AL IS THE BIT IN ERROR)
; AH=03 IF BOTH BYTES OF WORD HAVE ERRORS
; AH=02 IF LOW (EVEN) BYTE HAS ERROR
; AH=01 IF HI (ODD) BYTE HAS ERROR
; AX,BX,CX,DX,DI,SI ARE ALL DESTROYED.

```

```

0859          P0DSTG  PROC      NEAR
                ASSUME   DS:ABS0
0859 FC          CLD
085A 28 FF      SUB      DI,DI          ; SET DIRECTION TO INCREMENT
085C 28 C0      SUB      AX,AX          ; INITIAL DATA PATTERN FOR 00-FF
                                ; TEST
085E 8E DB      MOV      DS,AX          ; SET DS TO ABS0
0860 BB 1E 0472 R  MOV      BX,DATA_WORD[RESET_FLAG-DATA] ; WARM START?
0864 81 FB 1234  CMP      BX,1234H
0866 9C C2      MOV      DX,ES
086A 8E DA      MOV      DS,DX          ; RESTORE DS
086C 75 08      JNE      P1
086E F3/ AB     REP      STOSW          ; SIMPLE FILL WITH 0 ON WARM-START
0870 8E D8      MOV      DS,AX
0872 89 1E 0472 R  MOV      DATA_WORD[RESET_FLAG-DATA],BX
0876 8E DA      MOV      DS,DX          ; RESTORE DS
0878 C3        RET
0879 81 FB 4321  P1:  CMP      BX,4321H          ; DIAG. RESTART?
087D 74 EF      JE      P12              ; DO FILL WITH ZEROS
087F 8B 05      P2:  MOV      [DI],AL          ; WRITE TEST DATA
0881 8A 05      MOV      AL,[DI]          ; GET IT BACK
0883 32 C4      XOR      AL,AH            ; COMPARE TO EXPECTED
0885 74 03      JZ      P3
0887 E3 0C0C R   JMP      PB              ; ERROR EXIT IF MISCOMPARE
088A FE C4      PY:  INC      AH              ; FORM NEW DATA PATTERN
088C 8A C4      MOV      AL,AH
088E 75 EF      JNZ      P2              ; LOOP TILL ALL 288 DATA PATTERNS
                                ; DONE
0890 8B E9      MOV      BP,CX          ; SAVE WORD COUNT
0892 8B AAAA     MOV      AX,0AAAAH       ; LOAD DATA PATTERN
0895 8B 8B      MOV      BX,AX
0897 8A 5555     MOV      DX,05555H       ; LOAD OTHER DATA PATTERN
089A F3/ AB     REP      STOSW          ; FILL WORDS FROM LOW TO HIGH
                                ; WITH AAAA
089C 4F        DEC      DI              ; POINT TO LAST WORD WRITTEN
089D 4F        DEC      DI
089E FD        STO
089F 8B F7      MOV      SI,DI          ; SET DIRECTION FLAG TO GO DOWN
08A1 8B CD      MOV      CX,8P          ; SET INDEX REGS. EQUAL
                                ; RECOVER WORD COUNT
08A3          P3:  LODSW
08A3 AD        XOR      AX,BX          ; GO FROM HIGH TO LOW
08A4 33 C3      XOR      PB,PB          ; EQUAL WHAT FROM MEMORY
08A6 75 64      JNZ      PB            ; EQUAL WHAT S/B THERE?
08A8 8B C2      MOV      AX,DX          ; GO ERROR EXIT IF NOT
                                ; GET 88 DATA PATTERN
08AA AB        STOSW          ; STORE IT IN LOCATION JUST READ
08AB E2 F6      LOOP     P3            ; LOOP TILL ALL BYTES DONE
08AD 8B CD      MOV      CX,8P          ; RECOVER WORD COUNT
08AF FC        BACK TO INCREMENT
08B0 46        CLD          ; ADJUST PTRS
08B1 46        INC      SI
08B2 8B FE      MOV      DI,SI
08B4 8B DA      MOV      BX,DX          ; S/B DATA PATTERN TO BX
08B6 8A 00FF    MOV      DX,00FFH       ; DATA FOR CHECKERBOARD PATTERN
08B8 AD        PX:  LOOSW
08B8 33 C3      XOR      AX,BX          ; GET WORD FROM MEMORY
08BC 75 4E      JNZ      PB            ; EQUAL WHAT S/B THERE?
08BE 8B C2      MOV      AX,DX          ; GO ERROR EXIT IF NOT
                                ; GET OTHER PATTERN
08C0 AB        STOSW          ; STORE IT IN LOCATION JUST READ
08C1 E2 F6      LOOP     PX            ; LOOP TILL ALL BYTES DONE
08C3 8B CD      MOV      CX,8P          ; RECOVER WORD COUNT
08C5 FD        DECREMENT
08C6 4E        DEC      SI          ; ADJUST PTRS
08C7 4E        DEC      SI
08C8 8B FE      MOV      DI,SI
08CA 8B DA      MOV      BX,DX          ; S/B DATA PATTERN TO BX
08CC F7 D2      NOT      DX            ; MAKE PATTERN FFO0
08CE 0A D2      OR       DL,DL         ; FIRST PASS?
08D0 74 E7      JZ      PX            ; INCREMENT
08D2 FC        CLD
08D3 83 C8 04     ADD     SI,4
08D6 F7 D2      NOT      DX
08D8 8B FE      MOV      DI,SI
08DA 8B CD      MOV      CX,8P
08DC          P4:  ; LOW TO HIGH
08DC AD        LODSW          ; GET A WORD
08DD 33 C2      XOR      AX,DX          ; SHOULD COMPARE TO DX
08DF 75 28      JNZ      P8            ; GO ERROR IF NOT
08E1 AB        STOSW          ; WRITE 0000 BACK TO LOCATION
                                ; JUST READ
08E2 E2 F8      LOOP     P4            ; LOOP TILL DONE
08E4 FD        STD          ; BACK TO DECREMENT
08E5 4E        DEC      SI          ; ADJUST POINTER DOWN TO LAST WORD
08E6 4E        DEC      SI          ; WRITTEN
                                ; CHECK IF IN SERVICE/MFG MODES, IF SO, PERFORM REFRESH CHECK
08E7 8A 0201    MOV      DX,201H
08EA EC        IN       AL,DX          ; GET OPTION BITS
08EB 24 F0      AND     AL,0F0H
08ED 3C F0      CMP     AL,0F0H
08EF 74 10      JE      P8            ; ALL BITS HIGH-NORMAL MODE
08F1 8C C9      MOV     CX,CS
08F3 8C D3      MOV     BX,BS
08F5 3B CB      CMP     CX,BX          ; SEE IF IN PRE-STACK MODE
08F7 74 08      JE      P6            ; BYPASS RETENTION TEST IF SO
08F9 80 18      MOV     AL,24          ; SET OUTER LOOP COUNT
                                ; WAIT ABOUT 6-8 SECONDS WITHOUT ACCESSING MEMORY
                                ; IF REFRESH IS NOT WORKING PROPERLY THIS SHOULD
                                ; BE ENOUGH TIME FOR SOME DATA TO GO SOUR.

```

```

08FB E2 FE      P5:  LOOP      P5
08FD FE CB      DEC        AL
08FF 75 FA      JNZ        P5
OC01 89 CD      P6:  MOV        CX, BP      ; RECOVER WORD COUNT
OC03 AD         P7:  LODSW       ; GET WORD
OC04 0B C0      OR         AX, AX        ; = TO 0000
OC06 75 04      JNZ        PB           ; ERROR IF NOT
OC08 E2 F9      LOOP       P7           ; LOOP TILL DONE
OC0A EB 13      JMP        SHORT P11    ; THEN EXIT
OC0C BB CB      P8:  MOV        CX, AX      ; SAVE BITS IN ERROR
OC0E 32 E4      XOR        AH, AH
OC10 0A ED      OR         CH, CH        ; HIGH BYTE ERROR?
OC12 74 02      JZ         P9           ;
OC14 FE C4      INC        AH           ; SET HIGH BYTE ERROR
OC16 0A C9      P9:  OR         CL, CL    ; LOW BYTE ERROR?
OC18 74 03      JZ         P10
OC1A 80 C4 02   ADD        AH, 2
OC1D 0A E4      P10: OR        AH, AH    ; SET ZERO FLAG=0 (ERROR INDICATION)
OC1F FC        P11: CLD           ; SET DIR FLAG BACK TO INCREMENT
OC20 C3        RET            ; RETURN TO CALLER
OC21

```

```

PODSTG ENDP.
;*****
; PUT_LOGO PROCEDURE
; THIS PROC SETS UP POINTERS AND CALLS THE SCREEN
; OUTPUT ROUTINE SO THAT THE IBM LOGO, A MESSAGE,
; AND A COLOR BAR ARE PUT UP ON THE SCREEN.
; AX, BX, AND DX ARE DESTROYED. ALL OTHERS ARE SAVED
;*****
PUT_LOGO PROC NEAR

```

```

OC21 IE        PUSH     DS
OC22 85        PUSH     BP
OC23 50        PUSH     AX
OC24 53        PUSH     BX
OC25 51        PUSH     CX
OC26 52        PUSH     DX
OC27 BD OC4A R MOV     BP, OFFSET LOGO ; POINT DH DL AT ROW,COLUMN 0,0
OC2A BA B000   MOV     DX, B000H      ; ATTRIBUTE OF CHARACTERS TO BE
OC2D 83 1F     MOV     BL, 00011111B ; WRITTEN
OC2F CD B2     INT     B2H           ; CALL OUTPUT ROUTINE
OC31 B3 00     MOV     BL, 00000000B ; CALL OUTPUT ROUTINE
OC32 00        MOV     DL, 0         ; INITIALIZE COLUMN
OC35 B6 94     AGAIN: MOV    OH, 94H    ; SET LINE
OC37 8D OCBD. R MOV    BP, OFFSET COLOR ; OUTPUT GIVEN COLOR BAR
OC3A CD 62     INT     B2H           ; CALL OUTPUT ROUTINE
OC3C FE C3     INC     BL           ; INCREMENT ATTRIBUTE
OC3E 80 FA 20  CMP    OL, 32        ; IS THE COLUMN COUNTER POINTING
OC41 7C F2     JL     AGAIN         ; PAST 40?
OC43 5A        POP     DX           ; IF NOT, DO IT AGAIN
OC44 59        POP     CX
OC45 5B        POP     BX
OC46 58        POP     AX
OC47 5D        POP     BP           ; RESTORE BP
OC48 1F        POP     OS           ; RESTORE DS
OC49 C3        RET

```

```

PUT_LOGO ENDP
LOGO DB LOGO_E - LOGO
      DB 4, 220
LOGO_E =
      $
      DB 40, -5
      DB 40, -5
      DB 2, 7, 1, 9, 3, 4, 9, 4, 1, -5
      DB 2, 7, 1, 10, 2, 5, 7, 5, 1, -5
      DB 2, 7, 1, 11, 1, 6, 5, 6, 1, -5
      DB 4, 3, 5, 3, 3, 3, 3, 5, 3, 5, 3, -5
      DB 4, 3, 5, 3, 3, 3, 6, 1, 6, 3, -5
      DB 4, 3, 5, 8, 4, 13, 3, -5
      DB 4, 3, 5, 7, 5, 13, 3, -5
      DB 4, 3, 5, 8, 4, 13, 3, -5
      DB 4, 3, 5, 3, 3, 3, 13, 3, -5
      DB 4, 3, 5, 3, 3, 3, 3, 1, 5, 1, 3, 3, -5
      DB 2, 7, 1, 11, 1, 5, 2, 3, 2, 5, 1, -5
      DB 2, 7, 1, 10, 2, 5, 3, 1, 3, 5, 1, -5
      DB 2, 7, 1, 9, 3, 5, 7, 5, 1, -5
      DB 40, -5
      DB 40, -4
COLOR DB COLOR_E - COLOR
      DB 219
COLOR_E =
      $
      DB 2, 121-2, 2, 121-2, 2, 121-2, 2, 121-2, 2, -4

```

ASSUME DS:DATA

----- INT 10
VIDEO_10

THESE ROUTINES PROVIDE THE CRT INTERFACE
THE FOLLOWING FUNCTIONS ARE PROVIDED:

(AH)=0 SET MODE (AL) CONTAINS MODE VALUE
(AL)=0 40X25 BW (POWER ON DEFAULT)
(AL)=1 40X25 COLOR
(AL)=2 80X25 BW
(AL)=3 80X25 COLOR
GRAPHICS MODES
(AL)=4 320X200 4 COLOR
(AL)=5 320X200 BW 4 SHADES
(AL)=8 640X200 BW 2 SHADES
(AL)=7 NOT VALID
**** EXTENDED MODES ****
(AL)=8 160X200 16 COLOR
(AL)=9 320X200 16 COLOR
(AL)=A 640X200 4 COLOR
*** NOTE BW MODES OPERATE SAME AS COLOR MODES, BUT
COLOR BURST IS NOT ENABLED
*** NOTE IF HIGH ORDER BIT IN AL IS SET, THE REGEN
BUFFER IS NOT CLEARED.

(AH)=1 SET CURSOR TYPE
(CH) = BITS 4-0 = START LINE FOR CURSOR
** HARDWARE WILL ALWAYS CAUSE BLINK
** SETTING BIT 5 OR 6 WILL CAUSE ERRATIC
BLINKING OR NO CURSOR AT ALL
** IN GRAPHICS MODES, BIT 5 IS FORCED ON TO
DISABLE THE CURSOR
(CL) = BITS 4-0 = END LINE FOR CURSOR

(AH)=2 SET CURSOR POSITION
(DH, DL) = ROW, COLUMN (0,0) IS UPPER LEFT
(BH) = PAGE NUMBER (MUST BE 0 FOR GRAPHICS MODES)

(AH)=3 READ CURSOR POSITION
(BH) = PAGE NUMBER (MUST BE 0 FOR GRAPHICS MODES)
ON EXIT (DH, DL) = ROW, COLUMN OF CURRENT CURSOR
(CH, CL) = CURSOR MODE CURRENTLY SET

(AH)=4 READ LIGHT PEN POSITION
ON EXIT:
(AH) = 0 -- LIGHT PEN SWITCH NOT DOWN/NOT TRIGGERED
(AH) = 1 -- VALID LIGHT PEN VALUE IN REGISTERS
(DH, DL) = ROW, COLUMN OF CHARACTER LP POSN
(CH) = RASTER LINE (0-199)
(BX) = PIXEL COLUMN (0-319, 639)

(AH)=5 SELECT ACTIVE DISPLAY PAGE (VALID ONLY FOR
ALPHA MODES)
(AL)=NEW PAGE VALUE (0-7 FOR MODES 0B1, 0-3 FOR
MODES 2B3)
IF BIT 7 (80H) OF AL=1
READ/WRITE CRT/CPU PAGE REGISTERS
(AL) = 80H READ CRT/CPU PAGE REGISTERS
(AL) = 81H SET CPU PAGE REGISTER
(BL) = VALUE TO SET
(AL) = 82H SET CRT PAGE REGISTER
(BH) = VALUE TO SET
(AL) = 83H SET BOTH CRT AND CPU PAGE REGISTERS
(BL) = VALUE TO SET IN CPU PAGE REGISTER
(BH) = VALUE TO SET IN CRT PAGE REGISTER
IF BIT 7 (80H) OF AL=1
ALWAYS RETURNS (BH) = CONTENTS OF CRT PAGE REG
(BL) = CONTENTS OF CPU PAGE REG

(AH)=6 SCROLL ACTIVE PAGE UP
(AL) = NUMBER OF LINES, INPUT LINES BLANKED AT
BOTTOM OF WINDOW, AL = 0 MEANS BLANK
ENTIRE WINDOW
(CH, CL) = ROW, COLUMN OF UPPER LEFT CORNER OF
SCROLL
(DH, DL) = ROW, COLUMN OF LOWER RIGHT CORNER OF
SCROLL
(BH) = ATTRIBUTE TO BE USED ON BLANK LINE

(AH)=7 SCROLL ACTIVE PAGE DOWN
(AL) = NUMBER OF LINES, INPUT LINES BLANKED AT TOP
OF WINDOW, AL=0 MEANS BLANK ENTIRE WINDOW
(CH, CL) = ROW, COLUMN OF UPPER LEFT CORNER OF
SCROLL
(DH, DL) = ROW, COLUMN OF LOWER RIGHT CORNER OF
SCROLL
(BH) = ATTRIBUTE TO BE USED ON BLANK LINE

CHARACTER HANDLING ROUTINES

(AH) = 8 READ ATTRIBUTE/CHARACTER AT CURRENT CURSOR POSITION
(BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY)
ON EXIT:
(AL) = CHAR READ
(AH) = ATTRIBUTE OF CHARACTER READ (ALPHA MODES
ONLY)

(AH) = 9 WRITE ATTRIBUTE/CHARACTER AT CURRENT CURSOR
POSITION
(BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY)
(CX) = COUNT OF CHARACTERS TO WRITE
(AL) = CHAR TO WRITE
(BL) = ATTRIBUTE OF CHARACTER (ALPHA)/COLOR OF
CHARACTER (GRAPHICS). SEE NOTE ON WRITE
DOT FOR BIT 7 OF BL = 1.

(AH) = 10 (0AH) WRITE CHARACTER ONLY AT CURRENT CURSOR
POSITION
(BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY)
(CX) = COUNT OF CHARACTERS TO WRITE
(AL) = CHAR TO WRITE
(BL) = COLOR OF CHAR (GRAPHICS)
SEE NOTE ON WRITE DOT FOR BIT 7 OF BL = 1.

FOR READ/WRITE CHARACTER INTERFACE WHILE IN GRAPHICS MODE, THE CHARACTERS ARE FORMED FROM A CHARACTER GENERATOR IMAGE MAINTAINED IN THE SYSTEM ROM. INTERRUPT 4AH (LOCATION 00110H) IS USED TO POINT TO THE 1K BYTE TABLE CONTAINING THE FIRST 128 CHARS (0-127). INTERRUPT 1FH (LOCATION 0007CH) IS USED TO POINT TO THE 1K BYTE TABLE CONTAINING THE SECOND 128 CHARS (128-255).

FOR WRITE CHARACTER INTERFACE IN GRAPHICS MODE, THE REPLICATION FACTOR CONTAINED IN (CX) ON ENTRY WILL PRODUCE VALID RESULTS ONLY FOR CHARACTERS CONTAINED ON THE SAME ROW. CONTINUATION TO SUCCEEDING LINES WILL NOT PRODUCE CORRECTLY.

GRAPHICS INTERFACE

(AH) = 11 (0BH) SET COLOR PALETTE
 (BH) = PALETTE COLOR ID BEING SET (0-127)
 (BL) = COLOR VALUE TO BE USED WITH THAT COLOR ID
 COLOR ID = 0 SELECTS THE BACKGROUND COLOR (0-15)
 COLOR ID = 1 SELECTS THE PALETTE TO BE USED:
 2 COLOR MODE:
 0 = WHITE FOR COLOR 1
 1 = BLACK FOR COLOR 1
 4 COLOR MODES:
 0 = GREEN, RED, BROWN FOR COLORS 1,2,3
 1 = CYAN, MAGENTA, WHITE FOR COLORS 1,2,3
 16 COLOR MODES:
 ALWAYS SETS UP PALETTE AS:
 BLUE FOR COLOR 1
 GREEN FOR COLOR 2
 CYAN FOR COLOR 3
 RED FOR COLOR 4
 MAGENTA FOR COLOR 5
 BROWN FOR COLOR 6
 LIGHT GRAY FOR COLOR 7
 DARK GRAY FOR COLOR 8
 LIGHT BLUE FOR COLOR 9
 LIGHT GREEN FOR COLOR 10
 LIGHT CYAN FOR COLOR 11
 LIGHT RED FOR COLOR 12
 LIGHT MAGENTA FOR COLOR 13
 YELLOW FOR COLOR 14
 WHITE FOR COLOR 15

IN 40X25 OR 80X25 ALPHA MODES, THE VALUE SET FOR PALETTE COLOR 0 INDICATES THE BORDER COLOR TO BE USED IN GRAPHIC MODES, IT INDICATES THE BORDER COLOR AND THE BACKGROUND COLOR.

(AH) = 12 (0CH) WRITE DOT
 (DX) = ROW NUMBER
 (CX) = COLUMN NUMBER
 (AL) = COLOR VALUE
 IF BIT 7 OF AL = 1, THEN THE COLOR VALUE IS EXCLUSIVE OR'D WITH THE CURRENT CONTENTS OF THE DOT

(AH) = 13 (0DH) READ DOT
 (DX) = ROW NUMBER
 (CX) = COLUMN NUMBER
 (AL) RETURNS THE DOT READ

ASCII TELETYPE ROUTINE FOR OUTPUT

(AH) = 14 (0EH) WRITE TELETYPE TO ACTIVE PAGE
 (AL) = CHAR TO WRITE
 (BL) = FOREGROUND COLOR IN GRAPHICS MODE
 NOTE -- SCREEN WIDTH IS CONTROLLED BY PREVIOUS MODE SET

(AH) = 15 (0FH) CURRENT VIDEO STATE
 RETURNS THE CURRENT VIDEO STATE
 (AL) = MODE CURRENTLY SET (SEE AH=0 FOR EXPLANATION)

(AH) = NUMBER OF CHARACTER COLUMNS ON SCREEN
 (BH) = CURRENT ACTIVE DISPLAY PAGE

(AH) = 16 (10H) SET PALETTE REGISTERS
 (AL) = 0 SET PALETTE REGISTER
 (BL) = PALETTE REGISTER TO SET (00H - 0FH)
 (BH) = VALUE TO SET
 (AL) = 1 SET BORDER COLOR REGISTER
 (SH) = VALUE TO SET
 (AL) = 2 SET ALL PALETTE REGISTERS AND BORDER REGISTER
 ES:DX POINTS TO A 17 BYTE LIST
 BYTES 0 THRU 15 ARE VALUES FOR PALETTE REGISTERS 0 THRU 15
 BYTE 16 IS THE VALUE FOR THE BORDER REGISTER

NOTE:

IN MODES USING A 32K REGEN (9 AND A), ACCESS THROUGH THE CPU REGISTER BY USE OF B800H SEGMENT VALUE ONLY REACHES THE FIRST 16K. BIOS USES THE CONTENTS OF THE CPU PAGE REG (BITS 3, 4, & 5 OF PAGDAT IN BIOS DATA AREA) TO DERIVE THE PROPER SEGMENT VALUE.

CS, SS, DS, ES, BX, CX, DX PRESERVED DURING CALL
 ALL OTHERS DESTROYED

VIDEO GATE ARRAY REGISTERS

```

PORT 3DA OUTPUT
REG 0  MODE CONTROL 1 REGISTER
01H  +HI BANDWIDTH/-LOW BANDWIDTH
02H  +GRAPHICS/-ALPHA
04H  +8BW
08H  +VIDEO ENABLE
10H  +16 COLOR GRAPHICS

REG 1  PALETTE MASK REISTER
01H  PALETTE MASK 0
02H  PALETTE MASK 1
04H  PALETTE MASK 2
08H  PALETTE MASK 3

REG 2  BORDER COLOR REGISTER
01H  BLUE
02H  GREEN
04H  RED
08H  INTENSITY

REG 3  MODE CONTROL 2 REGISTER
01H  RESERVED -- MUST BE ZERO
02H  +ENABLE BLINK
04H  RESERVED -- MUST BE ZERO
08H  +2 COLOR GRAPHICS (640X200 2 COLOR ONLY)

REG 4  RESET REGISTER
01H  +ASYNCHRONOUS RESET
02H  +SYNCHRONOUS RESET

REGS 10 TO 1F PALETTE REGISTERS
01H  BLUE
02H  GREEN
04H  RED
08H  INTENSITY

VIDEO GATE ARRAY STATUS
PORT 3DA INPUT
01H  +DISPLAY ENABLE
02H  +LIGHT PEN TRIGGER SET
04H  -LIGHT PEN SWITCH MADE
08H  +VERTICAL RETRACE
10H  +VIDEO DOTS
ASSUME CS:CODE,DS:DATA,ES:VIDEO_RAM
M0010 LABEL WORD ; TABLE OF ROUTINES WITHIN VIDEO I/O
0CE9 00A5 R DW OFFSET SET_MODE
0CE8 E45E R DW OFFSET SET_CTYPE
0CED E48B R DW OFFSET SET_CP05
0CEF E82D R DW OFFSET READ_CURSOR
OCF1 F7B1 R DW OFFSET READ_LPEN
OCF3 E4B3 R DW OFFSET ACT_DISP_PAGE
OCF8 E5D3 R DW OFFSET SCROLL_UP
OCF7 E63F R DW OFFSET SCROLL_DOWN
OCF9 F0E4 R DW OFFSET READ_AC_CURRENT
OCFB F113 R DW OFFSET WRITE_AC_CURRENT
OCFD F12C R DW OFFSET WRITE_C_CURRENT
OCFF E543 R DW OFFSET SET_COLOR
0D01 F187 R DW OFFSET WRITE_DOT
0D03 F146 R DW OFFSET READ_DOT
0D05 1992 R DW OFFSET WRITE_TTY
0D07 E581 R DW OFFSET VIDEO_STATE
0D09 E685 R DW OFFSET SET_PALETTE
= 0022 MOD10L EQU B-M0010

0D0B VIDEO_I0 PROC NEAR
0D0B FB STI ; INTERRUPTS BACK ON
0D0C FC CLD ; SET DIRECTION FORWARD
0D0D 06 ;
0D0E 1E PUSH ES ;
0D0F 52 PUSH DS ; SAVE SEGMENT REGISTERS
0D10 51 PUSH DX
0D11 53 PUSH CX
0D12 56 PUSH BX
0D13 57 PUSH SI
0D14 50 PUSH DI
0D15 8A C4 MOV AL,AH ; SAVE AX VALUE
0D17 32 E4 MOV AH,AH ; GET INTO LOW BYTE
0D19 91 E0 SAL AX,1 ; ZERO TO HIGH BYTE
0D1B 8B F0 MOV SI,AX ; #2 FOR TABLE LOOKUP
0D1D 3D 0022 CMP AX,M0010L ; PUT INTO SI FOR BRANCH
0D20 72 04 JB C1 ; TEST FOR WITHIN RANGE
0D22 58 POP C1 ; BRANCH AROUND BRANCH
0D23 E8 OF70 R JMP VIDEO_RETURN ; THROW AWAY THE PARAMETER
0D26 E8 138B R ; DO NOTHING IF NOT IN RANGE
0D29 8B 8B00 C1: CALL DDS ; SEGMENT FOR COLOR CARD
0D2C 80 3E 0049 R 09 MOV AX,08B00H ; IN MODE USING 32K REGEN
0D31 72 09 CMP CRT_MODE,9 ; ND, JUMP
0D33 8A 26 00BA R JC C2 ; GET COPY OF PAGE REGS
0D37 80 E4 38 AND AH,CPUREG ; ISOLATE CPU REG
0D3A 0A EC SHR AH,1 ; SHIFT TO MAKE INTO SEGMENT VALUE
0D3C 8E C0 MOV ES,AX ; SET UP TO POINT AT VIDEO RAM AREA
0D3E 58 POP AX ; RECOVER VALUE
0D3F 8A 26 0049 R MOV AH,CRT_MODE ; GET CURRENT MODE INTO AH
0D43 2E FF A4 0CE9 R JMP WORD PTR CS:[SI+OFFSET M0010]
0D4B VIDEO_I0 ENDP

```

```

;-----
; SET_MODE
; THIS ROUTINE INITIALIZES THE ATTACHMENT TO
; THE SELECTED MODE. THE SCREEN IS BLANKED.
; INPUT
; (AL) = MODE SELECTED (RANGE 0-8)
; OUTPUT
; NONE
;-----
0D48 M0050 LABEL WORD ; TABLE OF REGEN LENGTHS
0D48 0800 DW 2048 ; MODE 0 40X25 BW
0D4A 0800 DW 2048 ; MODE 1 40X25 COLOR
0D4C 1000 DW 4096 ; MODE 2 80X25 BW
0D4E 1000 DW 4096 ; MODE 3 80X25 COLOR
0D50 4000 DW 16384 ; MODE 4 320X200 4 COLOR
0D52 4000 DW 16384 ; MODE 5 320X200 4 COLOR
0D54 4000 DW 16384 ; MODE 6 640X200 BW
0D56 0000 DW 0 ; MODE 7 INVALID
0D58 4000 DW 16384 ; MODE 8 160X200 16 COLOR
0D5A 8000 DW 32768 ; MODE 9 320X200 16 COLOR
0D5C 8000 DW 32768 ; MODE A 640X200 4 COLOR
;-----
0D5E M0060 COLUMNS LABEL BYTE
0D5E 28 28 50 50 28 28 LABEL 40,40,80,80,40,80,0,20,40,80
50 00 14 28 50 DB
;-----
;----- TABLE OF GATE ARRAY PARAMETERS FOR MODE SETTING
0D69 M0070 LABEL BYTE
0D69 0C 0F 00 02 ; SET UP FOR 40X25 BW MODE 0
= 0004 EQU $-M0070 ; GATE ARRAY PARMS
;-----
0D6D ; SET UP FOR 40X25 COLOR MODE 1
0D6D 0B 0F 00 02 DB 0BH,0FH,0,2 ; GATE ARRAY PARMS
;-----
0D71 ; SET UP FOR 80X25 BW MODE 2
0D71 0D 0F 00 02 DB 0DH,0FH,0,2 ; GATE ARRAY PARMS
;-----
0D75 ; SET UP FOR 80X25 COLOR MODE 3
0D75 0B 0F 00 02 DB 0BH,0FH,0,2 ; GATE ARRAY PARMS
;-----
0D79 ; SET UP FOR 320X200 4 COLOR MODE 4
0D79 0A 03 00 00 DB 0AH,03H,0,0 ; GATE ARRAY PARMS
;-----
0D7D ; SET UP FOR 320X200 BW MODE 5
0D7D 0E 03 00 00 DB 0EH,03H,0,0 ; GATE ARRAY PARMS
;-----
0D81 ; SET UP FOR 640X200 BW MODE 6
0D81 0E 01 00 08 DB 0EH,01H,0,8 ; GATE ARRAY PARMS
;-----
0D85 ; SET UP FOR INVALID MODE 7
0D85 00 00 00 00 DB 00H,00H,0,0 ; GATE ARRAY PARMS
;-----
0D89 ; SET UP FOR 160X200 16 COLOR MODE 8
0D89 1A 0F 00 00 DB 1AH,0FH,0,0 ; GATE ARRAY PARMS
;-----
0D8D ; SET UP FOR 320X200 16 COLOR MODE 9
0D8D 1B 0F 00 00 DB 1BH,0FH,0,0 ; GATE ARRAY PARMS
;-----
0D91 ; SET UP FOR 640X200 4 COLOR MODE A
0D91 0B 03 00 00 DB 0BH,03H,0,0 ; GATE ARRAY PARMS
;-----
;----- TABLES OF PALETTE COLORS FOR 2 AND 4 COLOR MODES
0D95 M0072 2 COLOR, SET 0 LABEL BYTE
0D95 00 0F 00 00 DB 0,0FH,0,0
= 0004 EQU $-M0072 ; ENTRY LENGTH
;-----
0D99 M0072L 2 COLOR, SET 1 LABEL BYTE
0D99 0F 00 00 00 DB 0FH,0,0,0
;-----
0D9D M0074 4 COLOR, SET 0 LABEL BYTE
0D9D 00 02 04 06 DB 0,2,4,6
;-----
0DA1 M0075 4 COLOR, SET 1 LABEL BYTE
0DA1 00 03 05 0F DB 0,3,5,0FH
;-----
0DA5 SET_MODE PROC NEAR
0DA5 50 PUSH AX ; SAVE INPUT MODE ON STACK
0DA6 24 7F AND AL,7FH ; REMOVE CLEAR REGEN SWITCH
0DA8 3C 07 CMP AL,7 ; CHECK FOR VALID MODES
0DAA 74 04 JE C3 ; MODE 7 IS INVALID
0DAC 3C 08 CMP AL,08H ; GREATER THAN A IS INVALID
0DAE 72 02 JC C3 ; DEFAULT TO MODE 0
0DB0 80 00 MOV AL,0 ; ADDRESS OF COLOR CARD
0DB2 3C 02 CMP AL,2 ; SAVE MODE IN AH
0DB4 74 08 JE C5 ; SAVE IN GLOBAL VARIABLE
0DB6 3C 03 CMP AL,3 ; SAVE ADDRESS OF BASE
0DB8 74 04 JE C5 ; SAVE MODE IN DI
0DBA 3C 09 CMP AL,09H ; POINT TO CONTROL REGISTER
0DBC 72 0A JC C6 ; SYNC CONTROL REG TO ADDRESS
0DBE 81 3E 0015 R 0080 C5: CMP TRUE_MEM,128 ; DO WE HAVE 128K?
0DC4 73 02 JNC C6 ; YES, JUMP
0DC6 80 00 MOV AL,0 ; NO, DEFAULT TO MODE 0
0DC8 BA 03D4 C6: MOV DX,03D4H ; ADDRESS OF COLOR CARD
0DC8 8A E0 MOV AH,AL ; SAVE MODE IN AH
0DCD A2 0049 R MOV CRT_MODE,AL ; SAVE IN GLOBAL VARIABLE
0DD0 B8 16 0063 R MOV ADDR_BB45,DX ; SAVE ADDRESS OF BASE
0DD4 8B FB MOV DI,AX ; SAVE MODE IN DI
0DD6 BA 03DA MOV DX,VGA_CTL ; POINT TO CONTROL REGISTER
0DD9 EC IN AL,DX ; SYNC CONTROL REG TO ADDRESS
0DDA 32 C0 XOR AL,AL ; SET VGA REG 0
0DDC EE OUT DX,AL ; SELECT 1T
0DDD A0 0065 R MOV AL,CRT_MODE_SET ; GET LAST MODE SET
0DE0 24 F7 AND AL,07H ; TURN OFF VIDEO
0DE2 EE OUT DX,AL ; SET IN GATE ARRAY

```

```

;----- SET DEFAULT PALETTES
0DE3 88 C7      MOV AX,01      ; GET MODE
0DE5 B4 10      MOV AH,10H     ; SET PALETTE REG 0
0DE7 BB 0DB5 R  MOV BX,OFFSET M0072 ; POINT TO TABLE ENTRY
0DEA 3C 06      CMP AL,6       ; 2 COLOR MODE?
0DEC 74 0F      JE C7          ; YES, JUMP
0DEE BB 0DA1 R  MOV BX,OFFSET M0075 ; POINT TO TABLE ENTRY
0DF1 3C 05      CMP AL,5       ; CHECK FOR 4 COLOR MODE
0DF3 74 08      JE C7          ; YES, JUMP
0DF5 3C 04      CMP AL,4       ; CHECK FOR 4 COLOR MODE
0DF7 74 04      JE C7          ; YES JUMP
0DF9 3C 04      CMP AL,0AH    ; CHECK FOR 4 COLOR MODE
0DFB 75 11      JNE C10        ; NO JUMP
0DFD B9 0004    C7: MOV CX,4   ; NUMBER OF REGS TO SET
0E00 BA C4      CB: MOV AL,AH  ; GET REG NUMBER
0E02 EE        OUT DX,AL     ; SELECT IT
0E03 2E: 8A 07  MOV AL,CS:[BX]; GET DATA
0E06 EE        OUT DX,AL     ; SET IT
0E07 FE C4     INC AH       ; NEXT REG
0E09 43       INC BX       ; NEXT TABLE VALUE
0E0A E2 F4     LOOP CB
0E0C EB 08     JMP SHORT C11

;----- SET PALETES FOR DEFAULT 16 COLOR
0E0E B9 0010    C9: MOV CX,16  ; NUMBER OF PALETES, AH IS REG
                                ; COUNTER
0E11 BA C4     C10: MOV AL,AH  ; GET REG NUMBER
0E13 EE      OUT DX,AL  ; SELECT IT
0E14 EE      OUT DX,AL  ; SET PALETTE VALUE
0E15 FE C4     INC AH     ; NEXT REG
0E17 E2 F8     LOOP C10

;----- SET UP MO & M1 IN PAGREG
0E19 88 C7     C11: MOV AX,01  ; GET CURRENT MODE
0E1B 32 DB     XOR BL,BL  ; SET UP FOR ALPHA MODE
0E1D 3C 04     CMP AL,4   ; IN ALPHA MODE
0E1F 72 08     JC C12     ; YES, JUMP
0E21 83 40     MOV BL,40H ; SET UP FOR 16K REGEN
0E23 3C 09     CMP AL,09H ; MODE USE 16K
0E25 72 02     JC C12     ; YES, JUMP
0E27 83 C0     MOV BL,00H ; SET UP FOR 32K REGEN
0E29 BA 03DF   C12: MOV DX,PAGREG ; SET PORT ADDRESS OF PAGREG
0E2C A0 00BA R MOV AL,PAGDAT ; GET LAST DATA OUTPUT
0E2F 2A 3F     AND AL,3FH ; CLEAR MO & M1 BITS
0E31 0A C3     OR AL,BL   ; SET NEW BITS
0E33 EE      OUT DX,AL  ; STUFF BACK IN PORT
0E34 A2 00BA R MOV PAGOAT,AL ; SAVE COPY IN RAM

;----- ENABLE VIDEO AND CORRECT PORT SETTING
0E37 88 C7     MOV AX,01  ; GET CURRENT MODE
0E39 32 E4     XOR AH,AH  ; INTO AX REG
0E3B B9 0004    MOV CX,M0070L ; SET TABLE ENTRY LENGTH
0E3E F7 E1     MUL CX    ; TIMES MODE FOR OFFSET INTO TABLE
0E40 8B DB     MOV BX,AX  ; TABLE OFFSET IN BX
0E42 81 C3 0069 R ADD BX,OFFSET M0070 ; ADD TABLE START TO OFFSET
0E46 2E: BA 27  MOV AH,CS:[BX]; SAVE MODE SET AND PALETTE
0E49 2E: 6A 47 02 MOV AL,CS:[BX + 2] ; TILL WE CAN PUT THEM IN RAM
0E4D 88 F0     MOV SI,AX
0E4F FA       CLI        ; DISABLE INTERRUPTS
0E50 EB E675 R CALL MODE_ALIVE ; KEEP MEMORY DATA VALID
0E53 B0 10     MOV AL,10H ; DISABLE NMI AND HOLD REQUEST
0E55 E6 A0     OUT NMI_PORT,AL
0E57 BA 03DA R MOV DX,VGA_CTL
0E5A B0 04     MOV AL,4   ; POINT TO RESET REG
0E5C EE      OUT DX,AL  ; SEND TO GATE ARRAY
0E5D B0 02     MOV BL,2   ; SET SYNCHRONOUS RESET
0E5F EE      OUT DX,AL  ; DO IT

; WHILE THE GATE ARRAY IS IN RESET STATE, WE CANNOT ACCESS RAM
0E60 88 C6     MOV AX,SI  ; RESTORE NEW MODE SET
0E62 80 E4 F7 AMD AH,0F7H ; TURN OFF VIDEO ENABLE
0E65 32 C0     XOR AL,AL  ; SET UP TO SELECT VGA REG 0
0E67 EE      OUT DX,AL  ; SELECT IT
0E69 86 E0     XCHG AH,AL ; AH IS VGA REG COUNTER
0E6A EE      OUT DX,AL  ; SET MODE
0E6B B0 04     MOV BL,4   ; SET UP TO SELECT VGA REG 4
0E6D EE      OUT DX,AL  ; SELECT IT
0E6E 32 C0     XOR AL,AL  ; REMOVE RESET FROM VGA
0E70 EE      OUT DX,AL

; NOW OKAY TO ACCESS RAM AGAIN
0E71 B0 B0     MOV AL,B0H ; ENABLE NMI AGAIN
0E73 E6 A0     OUT NMI_PORT,AL
0E75 E9 E675 R CALL MODE_ALIVE ; KEEP MEMORY DATA VALID
0E78 FB       STI        ; ENABLE INTERRUPTS
0E79 EB 07     JMP SHORT C14
0E7B BA C4     C13: MOV AL,AH  ; GET VGA REG NUMBER
0E7D EE      OUT DX,AL  ; SELECT REG
0E7E 2E: 8A 07  MOV AL,CS:[BX]; GET TABLE VALUE
0E81 EE      OUT DX,AL  ; PUT IN VGA REG
0E82 43       INC BX     ; NEXT IN TABLE
0E83 FE C4     CB: MOV AH  ; NEXT REG
0E85 E2 F4     LOOP C13  ; DO ENTIRE ENTRY

;----- SET UP CRT AND CPU PAGE REGS ACCORDING TO MODE & MEMORY SIZE
0E87 BA 03DF   MOV DX,PAGREG ; SET 10 ADDRESS OF PAGREG
0E8A A0 00BA R MOV AL,PAGDAT ; GET LAST DATA OUTPUT
0E8D 24 C0     AND AL,00H  ; CLEAR REG BITS
0E8F B3 36     MOV BL,36H  ; SET UP FOR GRAPHICS MODE WITH 32K
                                ; REGEN
0E91 AB 80     TEST AL,B0H ; IN THIS MODE?
0E93 75 0C     JNZ C15    ; YES, JUMP
0E95 B3 3F     MOV BL,3FH ; SET UP FOR 16K REGEN AND 128K
                                ; MEMORY
0E97 81 3E 0015 R 000 CMP TRUE_MEM,128 ; DO WE HAVE 128K?
0E9D 73 02     JNC C15    ; YES, JUMP
0E9F B3 1B     MOV BL,1BH ; SET UP FOR 16K REGEN AND 64K
                                ; MEMORY

```

```

0EA1 0A C3          C15:  OR    AL,BL          ; COMBINE MODE BITS AND REG VALUES
0EA3 EE           OUT    DX,AL          ; SET PORT
0EA4 A2 00BA R    MOV    PAGDAT,AL     ; SAVE COPY IN RAM
0EA7 8B C6        MOV    AX,SI         ; PUT MODE SET & PALETTE IN RAM
0EA9 8B 26 0065 R MOV    CRT_MODE_SET,AH
0EAD A2 0065 R    MOV    CRT_PALETTE,AL
0EB0 E4 61        IN     AL,PORT_B     ; GET CURRENT VALUE OF 8255 PORT B
0EB2 24 FB        AND    AL,OFBh      ; SET UP GRAPHICS MODE
0EB4 F6 C4 02    TEST   AH,2         ; JUST SET ALPHA MODE IN VGA?
0EB7 75 02       JNZ   C16           ; YES, JUMP
0EB9 0C 04       OR    AL,4          ; SET UP ALPHA MODE
0EBB E6 81       C16:  OUT    PORT_B,AL  ; STUFF BACK IN 8255
;----- SET UP 6845
0EBD 1E           PUSH   DS            ; SAVE DATA SEGMENT VALUE
0EBE 33 C0        XOR    AX,AX         ; SET UP FOR A850 SEGMENT
0EC0 8E D8        MOV    DS,AX        ; ESTABLISH VECTOR TABLE ADDRESSING
0EC2 C5 1E 0074 R LDS    BX,PARAM_PTR ; GET POINTER TO VIDEO PARAMS
;-----
0EC6 8B C7        MOV    AX,DI         ; GET CURRENT MODE IN AX
0ECB 89 0010 90  MOV    CX,MO040     ; LENGTH OF EACH ROW OF TABLE
0ECC 80 FC 02    CMP    AH,2         ; DETERMINE WHICH TO USE
0ECF 72 10       JC    C17           ; MODE IS 0 OR 1
0ED1 03 D9        ADD    BX,CX         ; MOVE TO NEXT ROW OF INIT TABLE
0ED3 80 FC 04    CMP    AH,4         ; MODE IS 2 OR 3
0ED8 03 D9        ADD    BX,CX         ; MOVE TO GRAPHICS ROW OF
;----- INIT_TABLE
0EDA 80 FC 09    CMP    AH,9         ;
0EDD 72 02       JC    C17           ; MODE IS 4, 5, 8, 9, OR 9
0EDF 03 D9        ADD    BX,CX         ; MOVE TO NEXT GRAPHICS ROW OF
;----- INIT_TABLE
;----- BX POINTS TO CORRECT ROW OF INITIALIZATION TABLE
0EE1 50           ;-----
0EE2 BA 47 02    C17:  PUSH   AX            ; SAVE MODE IN AH
0EE5 8B 7F 0A    MOV    DI,WORD PTR DS:[BX+10] ; GET HORZ. SYNC POSITION
0EE8 1E           PUSH   DS            ; GET CURSOR TYPE
0EE9 EB 1389 R    CALL  DDS           ;
;-----
0EEC A2 0089 R    ASSUME DS:DATA      ;
0EEF 89 3E 0060 R MOV    HORZ_POS,AL  ; SAVE HORZ. SYNC POSITION VARIABLE
0EF3 50           MOV    CURSOR_MODE,DI ; SAVE CURSOR MODE
0EF4 A0 0086 R    PUSH   AX            ;
0EF7 24 0F        MOV    AL,VAR_DELAY ; SET DEFAULT OFFSET
0EF9 72 0086 R    AND    AL,OFh       ;
0EFC 58           MOV    VAR_DELAY,AL ;
0EFD 1F           POP    AX            ;
0EFE 32 E4        ASSUME DS:CODE      ;
0F00 8A 03D4      POP    DS            ;
;-----
0F03 84 C4        XOR    AH,AH         ; AH WILL SERVE AS REGISTER NUMBER
0F05 EE           MOV    DX,03D4h     ; DURING LOOP
0F06 42           MOV    DX,03D4h     ; POINT TO 6845
;-----
0F07 FE C4      ; LOOP THROUGH TABLE, OUTPUTTING REG ADDRESS, THEN VALUE FROM TABLE
0F09 BA 07      C18:  MOV    AL,AH         ; GET 6845 REGISTER NUMBER
0F0B EE           OUT    DX,AL        ;
0F0C 42           INC    DX            ; POINT TO DATA PORT
0F07 FE C4      INC    AH            ; NEXT REGISTER VALUE
0F09 BA 07      MOV    AL,[BX]      ; GET TABLE VALUE
0F0B EE           OUT    DX,AL        ; OUT TO CHIP
0F0C 43           INC    BX            ; NEXT IN TABLE
0F0D 4A           DEC    DX            ; BACK TO POINTER REGISTER
0F0E E2 F3      LOOP  C18           ; DO THE WHOLE TABLE
0F10 58           POP    AX            ; GET MODE BACK
0F11 1F           POP    DS            ; RECOVER SEGMENT VALUE
;-----
0F12 33 FF        ASSUME DS:DATA      ;
0F14 89 3E 004E R XOR    DI,DI         ; FILL REGEN AREA WITH BLANK
0F18 C6 06 0062 R 00 MOV    CRT_START,DI ; SET START ADDRESS SAVED IN GLOBAL
0F1D 5A           MOV    ACTIVE_PAGE,0 ; SET PAGE VALUE
0F1E 80 E2 80    POP    DX            ; GET ORIGINAL INPUT BACK
0F21 75 1C        AND    DL,80h       ; NO CLEAR OF REGEN ?
0F23 BA B800    JNZ   C21           ; SKIP CLEARING REGEN
0F26 89 2000    MOV    DX,0B800h    ; SET UP SEGMENT FOR 16K REGEN AREA
0F29 3C 09      MOV    CX,B192      ; NUMBER OF WORDS TO CLEAR
0F2B 72 05      CMP    AL,09h       ; REQUIRE 32K BYTE REGEN ?
0F2D 01 E1      JC    C19           ; NO, JUMP
0F2F 8A 1800    SHL   CX,1          ; SET 16K WORDS TO CLEAR
0F32 BE C2      MOV    DX,1800h     ; SET UP SEGMENT FOR 32K REGEN AREA
0F34 3C 04      C19:  MOV    ES,DX        ; SET REGEN SEGMENT
0F36 B8 0F20    CMP    AL,4         ; TEST FOR GRAPHICS
0F39 72 02      MOV    AX,' '+15*256 ; FILL CHAR FOR ALPHA
0F3B 33 C0      JC    C20           ; NO GRAPHICS INIT
0F3D F3/ AB      XOR    AX,AX        ; FILL FOR GRAPHICS MODE
;-----
0F3F BA 03DA      C20:  REP    STOSW        ; FILL THE REGEN BUFFER WITH BLANKS
0F42 32 C0      ;----- ENABLE VIDEO
0F44 EE           MOV    DX,VGA_CTL   ; GET PORT ADDRESS OF VGA
0F45 A0 0065 R    XOR    AL,AL        ;
0F48 EE           OUT    DX,AL        ; SELECT VGA REG 0
0F49 32 FF      MOV    AL,CRT_MODE_SET ; GET MODE SET VALUE
0F4B 72 02      OUT    DX,AL        ; SET MODE
;-----
0F49 32 FF      ;----- DETERMINE NUMBER OF COLUMNS, BOTH FOR ENTIRE DISPLAY
0F4B 72 02      ;----- AND THE NUMBER TO BE USED FOR TTY INTERFACE
0F4D 8A 1E 0049 R XOR    BH,BH        ;
0F4F 2E: 8A 87 005E R MOV    BL,CRT_MODE  ;
0F54 32 E4      MOV    AL,CS:[BX + OFFSET MO060] ;
0F56 A3 004A R    XOR    AH,AH        ;
;-----
;----- CRT_COLS,AX ; NUMBER OF COLUMNS IN THIS SCREEN

```

```

;----- SET CURSOR POSITIONS
OF59 D1 E3          SHL     BX,1          ; WORD OFFSET INTO CLEAR LENGTH
;
;
OF5B 2E: BB BF 0D4B R  MOV     CX,CS:EBX + OFFSET M0050J ; LENGTH TO CLEAR
OF60 89 0E 004C R    MOV     CRT_LEN,CX          ; SAVE LENGTH OF CRT
OF64 B9 000B        MOV     CX,8              ; CLEAR ALL CURSOR POSITIONS
OF67 BF 0050 R      MOV     DI,OFFSET CURSOR_POSN ; ESTABLISH SEGMENT
OF6A 1E             PUSH    DS                ; ADDRESSING
OF6B 07             POP     ES
OF6C 33 C0          XOR     AX,AX
OF6E F3/ AB        REP     STOSW           ; FILL WITH ZEROES
;----- NORMAL RETURN FROM ALL VIDEO RETURNS
VIDEO_RETURN:
OF70 0F70 5F        POP     DI
OF71 0F71 5E        POP     SI
OF72 0F72 5B        POP     BX
OF73 0F73 59        POP     CX
C22:  OF74 5A        POP     DX
OF75 0F75 5F        POP     DS
OF76 0F76 07        POP     ES          ; RECOVER SEGMENTS
OF77 0F77 CF        IRET                    ; ALL DONE
OF78 0F78          SET_MODE  ENDP
;-----
;
; KBDNMI -- KEYBOARD NMI INTERRUPT ROUTINE
;
; THIS ROUTINE OBTAINS CONTROL UPON AN NMI INTERRUPT, WHICH
; OCCURS UPON A KEYSTROKE FROM THE KEYBOARD.
;
; THIS ROUTINE WILL DE-SERIALIZE THE BIT STREAM IN ORDER TO
; GET THE KEYBOARD SCAN CODE ENTERED. IT THEN ISSUES INT 41
; PASSING THE SCAN CODE IN AL TO THE KEY PROCESSOR. UPON RETURN
; IT RE-ENABLES NMI AND RETURNS TO SYSTEM (IRET).
;-----
;
; ASSUME CS: CODE, DS: DATA
KBDNMI PROC FAR
;-----DISABLE INTERRUPTS
OF78 FA             CLI
;-----SAVE REGS & DISABLE NMI
OF79 56             PUSH    SI
OF7A 57             PUSH    DI
OF7B 50             PUSH    AX          ; SAVE REGS
OF7C 53             PUSH    BX
OF7D 51             PUSH    CX
OF7E 52             PUSH    DX
OF7F 1E             PUSH    DS
OF80 06             PUSH    ES
;-----INIT COUNTERS
OF81 8E 000B        MOV     SI,8          ; SET UP # OF DATA BITS
OF84 32 DB          XOR     BL,BL         ; INIT. PARITY COUNTER
;-----SAMPLE 5 TIMES TO VALIDATE START BIT
OF86 32 E4          XOR     AH,AH
OF8B B9 0005        MOV     CX,5          ; SET COUNTER
OF8B B9 0005        IN     AL,PORT_C     ; GET SAMPLE.
I1:  OF8D AB 40        TEST    AL,40H       ; TEST IF 1
OF8F 74 02          JZ     12            ; JMP IF 0
OF91 FE C4          INC     AH           ; KEEP COUNT OF 1'S
OF93 E2 F6          LOOP   11           ; KEEP SAMPLING
I2:  OF95 B0 FC 03        CMP     AH,3         ; VALID START BIT ?
OF9B 73 03          JNB   125          ; JMP IF OK
OF9A EB 5D 90        JMP    18           ; INVALID (SYNC ERROR) NO AUDIO
;-----VALID START BIT, LOOK FOR TRAILING EDGE
I25: MOV     CX,50      ; SET UP WATCHDOG TIMEOUT
I3:  OFA0 E4 62          IN     AL,PORT_C     ; GET SAMPLE
OFA2 AB 40          TEST    AL,40H       ; TEST IF 0
OFA4 74 05          JZ     15           ; JMP IF TRAILING EDGE FOUND
OFA6 E2 FB          LOOP   13           ; KEEP LOOKING FOR TRAILING EDGE
OFA8 EB 4F 90        JMP    18           ; SYNC ERROR (STUCK ON 1'S)
;-----READ CLOCK TO SET START OF BIT TIME
I5:  OFAB 80 40          MOV     AL,40H       ; READ CLOCK
OFAD E8 43          OUT    TIM_CTL,AL   ;
OFAF 50             NOP                    ;
OFB0 90             NOP                    ;
OFB1 E4 41          IN     AL,TIMER+1   ;
OFB3 8A E0          MOV     AH,AL        ;
OFB5 E4 41          IN     AL,TIMER+1   ;
OFB7 86 E0          XCHG   AH,AL        ;
OFB9 6B F8          MOV     DI,AX        ; SAVE CLOCK TIME IN DI
;-----VERIFY VALID TRANSITION
OFBB B9 0004        MOV     CX,4          ; SET COUNTER
OFBE E4 62          IN     AL,PORT_C     ; GET SAMPLE
I6:  OFC0 AB 40          TEST    AL,40H       ; TEST IF 0
OFC2 75 35          JNZ   18           ; JMP IF INVALID TRANSITION (SYNC)
OFC4 E2 FB          LOOP   16           ; KEEP LOOKING FOR VALID TRANSITION
;-----SET UP DISTANCE TO MIDDLE OF 1ST DATA BIT
OFC6 BA 0220        MOV     DX,544        ; 310 USEC AWAY (.838 US / CT)
;-----START LOOKING FOR TIME TO READ DATA BITS AND ASSEMBLE BYTE.
I7:  OFC9 EB 1031 R      CALL   130           ;
OFCB BA 020E        MOV     DX,526        ; SET NEW DISTANCE TO NEXT HALF BIT
OFCF 50             PUSH    AX           ; SAVE 1ST HALF BIT
OFD0 E8 1031 R      CALL   130           ;
OFD3 BA C8          MOV     CL,AL        ; PUT 2ND HALF BIT IN CL
OFD5 5B             POP     AX           ; RESTORE 1ST HALF BIT
OFD6 3A C8          CMP     CL,AL        ; ARE THEY OPPOSITES ?
OFD8 74 2A          JE     19           ; NO, PHASE ERROR

```



```

-----
;KEYB2_INT
; THE PURPOSE OF THIS ROUTINE IS TO TRANSLATE SCAN CODES AND
; SCAN CODE COMBINATIONS FROM THE 82 KEY KEYBOARD TO THEIR
; EQUIVLENTS ON THE 83 KEY KEYBOARD. THE SCAN CODE IS
; PASSED IN AL. EACH SCAN CODE PASSED EITHER TRIGGERS ONE OR
; MORE CALLS TO INTERRUPT 9 OR SETS FLAGS TO RETAIN KEYBOARD
; STATUS. WHEN INTERRUPT 9 IS CALLED THE TRANSLATED SCAN
; CODES ARE PASSED TO IT IN AL. THE INTENT OF THIS CODE WAS
; TO KEEP INTERRUPT 9 INTACT FROM ITS ORIGIN IN THE PC FAMILY
; THIS ROUTINE IS IN THE FRONT END OF INTERRUPT 9 AND
; TRANSFORMS A 82 KEY KEYBOARD TO LOOK AS IF IT WERE AN 83
; KEY VERSION.
; IT IS ASSUMED THAT THIS ROUTINE IS CALLED FROM THE NM1
; DESERIALIZATION ROUTINE AND THAT ALL REGISTERS WERE SAVED
; IN THE CALLING ROUTINE. AS A CONSEQUENCE ALL REGISTERS ARE
; DESTROYED.
-----
;EQUATES
= 0080 BREAK_BIT EQU 80H
= 0054 FN_KEY EQU 54H
= 0055 PHK EQU FN_KEY+1
= 0056 EXT_SCAN EQU PHK+1 ; BASE CODE FOR SCAN CODES
; ; EXTENDING BEYOND 83
= 00FF AND_MASK EQU OFFH ; USED TO SELECTIVELY REMOVE BITS
= 001F CLEAR_FLAGS EQU AND_MASK - (FN_FLAG+FN_BREAK+FN_PENDING)
; SCAN CODES
= 0030 B_KEY EQU 48
= 0010 Q_KEY EQU 16
= 0019 P_KEY EQU 25
= 0012 E_KEY EQU 18
= 001F S_KEY EQU 31
= 0031 N_KEY EQU 49
= 0048 UP_ARROW EQU 72
= 0050 DOWN_ARROW EQU 80
= 004B LEFT_ARROW EQU 75
= 004D RIGHT_ARROW EQU 77
= 000C MINUS EQU 12
= 000D EQUALS EQU 13
= 000B NUM_0 EQU 11
; NEW TRANSLATED SCAN CODES
-----
; NOTE:
; BREAK, PAUSE, ECHO, AND PRT_SCREEN ARE USED AS DFFSETS
; INTO THE TABLE 'SCAN'. OFFSET = TABLE POSITION + 1.
-----
= 0001 ECHO EQU 01
= 0002 BREAK EQU 02
= 0003 PAUSE EQU 03
= 0004 PRT_SCREEN EQU 04
= 0046 SCROLL_LOCK EQU 70
= 0045 NUM_LOCK EQU 69
= 0047 HOME EQU 71
= 004F END_KEY EQU 79
= 0049 PAGE_UP EQU 73
= 0051 PAGE_DOWN EQU 81
= 004A KEYPAD_MINUS EQU 74
= 004E KEYPAD_PLUS EQU 78
; ASSUME CS: CODE, DS: DATA
; ----TABLE OF VALID SCAN CODES
KBO LABEL BYTE
DB B_KEY, Q_KEY, E_KEY, P_KEY, S_KEY, N_KEY
DB UP_ARROW, DOWN_ARROW, LEFT_ARROW, RIGHT_ARROW, MINUS
DB EQUALS
KBOLEN EQU $ - KBO
; ----TABLE OF NEW SCAN CODES
KB1 LABEL BYTE
DB BREAK, PAUSE, ECHO, PRT_SCREEN, SCROLL_LOCK, NUM_LOCK
DB HOME, END_KEY, PAGE_UP, PAGE_DOWN, KEYPAD_MINUS, KEYPAD_PLUS
-----
; NOTE: THERE IS A ONE TO ONE CORRESPONDENCE BETWEEN
; THE SIZE OF KBO AND KB1.
-----
; TABLE OF NUMERIC KEYPAD SCAN CODES
; THESE SCAN CODES WERE NUMERIC KEYPAD CODES ON
; THE 83 KEY KEYBOARD.
-----
NUM_CODES LABEL BYTE
DB 79, 80, 81, 75, 76, 77, 71, 72, 73, 82
-----
; TABLE OF SIMULATED KEYSTROKES
; THIS TABLE REPRESENTS A 4*2 ARRAY. EACH ROW
; CONSISTS OF A SEQUENCE OF SCAN CODES WHICH
; WOULD HAVE BEEN GENERATED ON AN 83 KEY KEYBOARD
; TO CAUSE THE FOLLOWING FUNCTIONS:
; ROW 1=ECHO CRT OUTPUT TO THE PRINTER
; ROW 2=BREAK
; THE TABLE HAS BOTH MAKE AND BREAK SCAN CODES.
-----
SCAN LABEL BYTE
DB 29, 55, 183, 157 ; CTRL + PRTSC
DB 29, 70, 188, 157 ; CTRL + SCROLL-LOCK

```

```

-----
TABLE OF VALID ALT SHIFT SCAN CODES
THIS TABLE CONTAINS SCAN CODES FOR KEYS ON THE
62 KEY KEYBOARD. THESE CODES ARE USED IN
COMBINATION WITH THE ALT KEY TO PRODUCE SCAN CODES
FOR KEYS NOT FOUND ON THE 62 KEY KEYBOARD.
-----

```

```

1093
1093 35 28 34 1A 1B
= 0005

```

```

ALT_TABLE LABEL BYTE
DB 53,40,52,26,27
ALT_LEN EQU $ - ALT_TABLE

```

```

-----
TABLE OF TRANSLATED SCAN CODES WITH ALT SHIFT
THIS TABLE CONTAINS THE SCAN CODES FOR THE
KEYS WHICH ARE NOT ON THE 62 KEY KEYBOARD AND
WILL BE TRANSLATED WITH ALT SHIFT. THERE IS A
ONE TO ONE CORRESPONDENCE BETWEEN THE SIZES
OF ALT_TABLE AND NEW_ALT.
THE FOLLOWING TRANSLATIONS ARE MADE:
ALT+ / = \
ALT+ ' = `
ALT+ [ = {
ALT+ ] = }
ALT+ . = ~
-----

```

```

1098
1098 28 29 37 2B 29

```

```

NEW_ALT LABEL BYTE
DB 43,41,55,43,41

```

```

-----
;EXTAB

```

```

TABLE OF SCAN CODES FOR MAPPING EXTENDED SET
OF SCAN CODES (SCAN CODES > 85). THIS TABLE
ALLOWS OTHER DEVICES TO USE THE KEYBOARD INTERFACE.
IF THE DEVICE GENERATES A SCAN CODE > 85 THIS TABLE
CAN BE USED TO MAP THE DEVICE TO THE KEYBOARD. THE
DEVICE ALSO HAS THE OPTION OF HAVING A UNIQUE SCAN
CODE PUT IN THE KEYBOARD BUFFER (INSTEAD OF MAPPING
TO THE KEYBOARD). THE EXTENDED SCAN CODE PUT IN THE
BUFFER WILL BE CONTINUOUS BEGINNING AT 150. A ZERO
WILL BE USED IN PLACE OF AN ASCII CODE. (E.G. A
DEVICE GENERATING SCAN CODE 85 AND NOT MAPPING 86
TO THE KEYBOARD WILL HAVE A [150,0] PUT IN THE
KEYBOARD BUFFER)

```

```

TABLE FORMAT:
THE FIRST BYTE IS A LENGTH INDICATING THE NUMBER
OF SCAN CODES MAPPED TO THE KEYBOARD. THE REMAINING
ENTRIES ARE WORDS. THE FIRST BYTE (LOW BYTE) IS A
SCAN CODE AND THE SECOND BYTE (HIGH BYTE) IS ZERO.
A DEVICE GENERATING N SCAN CODES IS ASSUMED TO GENERATE THE
FOLLOWING STREAM 88,87,88,...,86+(N-1). THE SCAN CODE BYTES
IN THE TABLE CORRESPOND TO THIS SET WITH THE FIRST DATA
BYTE MATCHING 86, THE SECOND MATCHING 87 ETC.

```

```

NOTES:

```

- (1) IF A DEVICE GENERATES A BREAK CODE, NOTHING IS PUT IN THE BUFFER.
- (2) A LENGTH OF 0 INDICATES THAT ZERO SCAN CODES HAVE BEEN MAPPED TO THE KEYBOARD AND ALL EXTENDED SCAN CODES WILL BE USED.
- (3) A DEVICE CAN MAP SOME OF ITS SCAN CODES TO THE KEYBOARD AND HAVE SOME ITS SCAN CODES IN THE EXTENDED SET.

```

1090
1090 14
109E 004B 0049 0040 0051
0050 004F 004B 0047
0039 001C
1092 0011 0012 001F 0020
002C 002B 001E 0010
000F 0001

```

```

EXTAB LABEL BYTE
DB 20 ; LENGTH OF TABLE
DW 72,73,77,B1,80,79,75,71,57,28
DW 17,18,31,45,44,43,30,16,15,1

```

```

KEY62_INT PROC FAR

```

```

10C6 FB ; STI
10C7 FC ; CLD
10C8 EB 13BB R ; CALL DDS ; FORWARD DIRECTION
10CB 8A E0 ; MOV AH,AL ; SET UP ADDRESSING
10CD EB 131E R ; CALL TPM ; SAVE SCAN CODE
; ADJUST OUTPUT FOR USER
; MODIFICATION
10D0 73 01 ; JNC KBX0 ; JUMP IF OK TO CONTINUE
10D2 CF ; IRET ; RETURN FROM INTERRUPT.
;---EXTENDED SCAN CODE CHECK
KBX0: CMP AL,OFFH ; IS THIS AN OVERRUN CHAR?
JE KBX1 ; PASS IT TO INTERRUPT 9
AND AL,AND_MASK-BREAK_BIT ; TURN OFF BREAK BIT
CMP AL,EXT_SCAN ; IS THIS A SCAN CODE > 83
JL KBX4 ; REPLACE BREAK BIT
;---SCAN CODE IS IN EXTENDED SET
PUSH DS
XOR SI,SI
MOV DS,SI
ASSUME DS:ABS0
LES DI,DWORD PTR EXT ; GET THE POINTER TO THE EXTENDED
; SET
10E2 C4 3E 0124 R ; MOV CL,BYTE PTR ES:[DI] ; GET LENGTH BYTE
10E6 26: 8A 0D ; POP DS
10E9 1F ; ASSUME DS:DATA
;---DOES SCAN CODE GET MAPPED TO KEYBOARD OR TO NEW EXTENDED SCAN
CODES?
10EA 2C 56 ; SUB AL,EXT_SCAN ; CONVERT TO BASE OF NEW SET
10EC FE C9 ; DEC CL ; LENGTH - 1
10EE 3A C1 ; CMP AL,CL ; IS CODE IN TABLE?
10F0 7F 10 ; JG KBX1 ; JUMP IF SCAN CODE IS NOT IN TABLE

```

```

10F2 47          ;----GET SCAN CODE FROM TABLE
10F3 88 DB      INC D1          ; POINT D1 PAST LENGTH BYTE
10F4 32 FF      MOV BX,AX          ;
10F5 32 FF      XOR BH,BH          ; PREPARE FOR ADDING TO 16 BIT
                                ; REGISTER
10F7 01 E3      SHL BX,1          ;
10F9 03 FB      ADD D1,BX          ; OFFSET TO CORRECT TABLE ENTRY
10FB 26: BA 05  MOV AL,BYTE PTR ES:[D1] ; TRANSLATED SCAN CODE IN AL
10FC 3C 56      CMP AL,EXT_SCAN    ; IS CODE IN KEYBOARD SET?
1100 7C 3A      JL KBX4          ; IN KEYBOARD SET, CHECK FOR BREAK
                                ;
;----SCAN CODE GETS MAPPED TO EXTENDED SCAN CODES
KBX1: TEST AH,BREAK_BIT ; IS THIS A BREAK CODE?
      JZ KBX2          ; MAKE CODE, PUT IN BUFFER
1107 CF         IRET          ; BREAK CODE, RETURN FROM INTERRUPT
1108 80 C4 40   KBX2: ADD AH,64      ; EXTENDED SET CODES BEGIN AT 150
1109 32 C0      XOR AL,AL        ; ZERO OUT ASCII VALUE (NUL)
110D BB 1E 001C R MOV BX,BUFFER_TAIL ; GET TAIL POINTER
1111 88 F3      MOV SI,BX         ; SAVE POINTER TO TAIL
1113 EB 144F R   CALL K4          ; INCREMENT TAIL VALUE
1116 3B 1E 001A R CMP BX,BUFFER_HEAD ; IS BUFFER FULL?
111A 75 19      JNE KBX3          ; PUT CONTENTS OF AX IN BUFFER
                                ;
;----BUFFER IS FULL, BEEP AND CLEAR FLAGS
111C BB 0080     MOV BX,80H        ; FREQUENCY OF BEEP
111D B9 0048     MOV CX,48H        ; DURATION OF BEEP
1122 EB E038 R   CALL KB_NOISE    ; BUFFER FULL BEEP
1125 80 26 0017 R FO AND KB_FLAG,0F0H ; CLEAR ALT, CTRL, LEFT AND RIGHT
                                ; SHIFTS
112A 80 26 0018 R OF AND KB_FLAG_1,0FH ; CLEAR MAKE OF INS,CAPS_LOCK,NUM
                                ; AND SCROLL
112F 80 26 00BB R 1F AND KB_FLAG_2,1FH ; CLEAR FUNCTION STATES
1134 CF         IRET          ; DONE WITH INTERRUPT
1135 89 04      MOV [SI],AX        ; PUT CONTENTS OF AX IN BUFFER
1137 89 1E 001C R MOV MOV          ; ADVANCE BUFFER TAIL
113B CF         IRET          ; RETURN FROM INTERRUPT
113C 80 E4 80   KBX4: AND AH,BREAK_BIT ; MASK BREAK BIT ON ORIGINAL SCAN
113F 0A C4      OR AL,AH          ; UPDATE NEW SCAN CODE
1141 BA E0      MOV AH,AL         ; SAVE AL IN AH AGAIN
                                ;
;----B3 KEYBOARD FUNCTIONS SHIFT+PRTSC AND CTRL+NUMLOCK
KBO_1: CMP AL,NUM_KEY ; IS THIS A NUMLOCK?
      JNE KBO_3      ; CHECK FOR PRTSC
1143 F6 06 0017 R O4 TEST KB_FLAG,CTL_SHIFT ; IS CTRL KEY BEING HELD DOWN?
1144 74 0A      JZ KBO_2          ; NUMLOCK WITHOUT CTRL, CONTINUE
114E F6 06 0017 R OB TEST KB_FLAG,ALT_SHIFT ; IS ALT KEY HELD CONCURRENTLY?
1153 75 03      JNZ KBO_2          ; PASS IT ON
1155 E9 12EB R   JMP KBIS_1        ; PUT KEYBOARD IN HOLD STATE
1158 E9 125C R   KBO_2: JMP CONT_INT ; CONTINUE WITH INTERRUPT 48H
                                ;
;----CHECK FOR PRTSC
KBO_3: CMP AL,56      ; IS THIS A PRTSC KEY?
      JNZ KB1_1      ; NOT A PRTSC KEY
115F F6 06 0017 R O3 TEST KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT ; EITHER SHIFT
                                ; ACTIVE?
1164 74 F2      JZ KBO_2          ; PROCESS SCAN IN INT9
1166 F6 06 0017 R O4 TEST KB_FLAG,CTL_SHIFT ; IS THE CTRL KEY PRESSED?
1168 75 EB      JNZ KBO_2          ; NOT A VALID PRTSC (PC COMPATIBLE)
116D E9 1301 R   JMP PRTSC         ; HANDLE THE PRINT SCREEN FUNCTION
                                ;
;----ALTERNATE SHIFT TRANSLATIONS
KB1_1: MOV AH,AL        ; SAVE CHARACTER
      AND AL,AND_MASK - BREAK_BIT ; MASK BREAK BIT
1174 F6 06 0017 R OB TEST KB_FLAG,ALT_SHIFT ; IS THIS A POTENTIAL TRANSLATION
1179 74 39      JZ KB2          ;
                                ;
;----TABLE LOOK UP
117B 0E         PUSH CS          ;
117C 07         POP ES           ; INITIALIZE SEGMENT FOR TABLE LOOK
                                ; UP
117D BF 1093 R   MOV D1,OFFSET ALT_TABLE
1180 B9 0005     MOV CX,ALT_LEN    ; GET READY FOR TABLE LOOK UP
1183 F2/ AE     REPNE SCASB      ; SEARCH TABLE
1185 75 2D      JNE KB2          ; JUMP IF MATCH IS NOT FOUND
1187 B9 1094 R   MOV CX,OFFSET ALT_TABLE + 1
118A 29 F9      SUB D1,CX          ; UPDATE D1 TO INDEX SCAN CODE
118C 2E: 8A B5 1098 R MOV AL,CS:NEW_ALT[D1] ; TRANSLATE SCAN CODE
                                ;
;----CHECK FOR BREAK CODE
1191 8A 1E 0017 R MOV BL,KB_FLAG    ; SAVE KB_FLAG STATUS
1195 80 36 0017 R OB XOR KB_FLAG,ALT_SHIFT ; MASK OFF ALT SHIFT
119A F6 C4 80   TEST AH,BREAK_BIT ; IS THIS A BREAK CHARACTER?
119D 74 02      JZ KB1_2          ; JUMP IF SCAN IS A MAKE
119F 0C 80      OR AL,BREAK_BIT ; SET BREAK BIT
                                ;
;----MAKE CODE, CHECK FOR SHIFT SEQUENCE
KB1_2: CMP D1,3        ; IS THIS A SHIFT SEQUENCE
      JL KB1_3        ; JUMP IF NOT SHIFT SEQUENCE
11A6 80 0E 0017 R O2 OR KB_FLAG,LEFT_SHIFT ; TURN ON SHIFT FLAG
11AB E6 60      MOV KBPORT,AL    ;
11AD CD 09      INT 9H           ; ISSUE INT TO PROCESS SCAN CODE
11AF BB 1E 0017 R MOV KB_FLAG,BL    ; RESTORE ORIGINAL FLAG STATES
11B3 CF         IRET          ;
                                ;
;----FUNCTION KEY HANDLER
KB2:  CMP AL,FN_KEY    ; CHECK FOR FUNCTION KEY
      JNZ KB4          ; JUMP IF NOT FUNCTION KEY
11B8 F6 C4 80   TEST AH, BREAK_BIT ; IS THIS A FUNCTION BREAK
11BB 75 08      JNZ KB3          ; JUMP IF FUNCTION BREAK
11BD 80 26 008B R 1F AND KB_FLAG_2,CLEAR_FLAGS ; CLEAR ALL PREVIOUS
                                ; FUNCTIONS
11C2 80 0E 008B R AO OR KB_FLAG_2, FN_FLAG + FN_PENDING
11C7 CF         IRET          ; RETURN FROM INTERRUPT
                                ;
;----FUNCTION BREAK
KB3:  TEST KB_FLAG_2, FN_PENDING
      JNZ KB3_1        ; JUMP IF FUNCTION IS PENDING
11CF 80 26 008B R 1F AND KB_FLAG_2,CLEAR_FLAGS ; CLEAR ALL FLAGS
11D4 CF         IRET          ;
11D5 80 0E 008B R AO KB3_1: OR KB_FLAG_2, FN_BREAK ; SET BREAK FLAG
11DA CF         IRET          ; RETURN FROM INTERRUPT

```

```

;----CHECK IF FUNCTION FLAG ALREADY SET
110B 3C 55      KB4:  CMP     AL,PHK          ; IS THIS A PHANTOM KEY?
110D 74 FB      ;         KB3_2          ; JUMP IF PHANTOM SEQUENCE
11DF F6 06 00B9 R 80  KB4_0: TEST    KB_FLAG_2,FN_FLAG+FN_LOCK ; ARE WE IN FUNCTION
;         ; STATE?

11E4 75 21      ;         JNZ     K85
;----CHECK IF NUM_STATE IS ACTIVE
11E6 F6 06 0017 R 20  TEST    KB_FLAG,NUM_STATE
11EB 74 16      ;         JZ      KB4_1          ; JUMP IF NOT IN NUM_STATE
11ED 3C 08      CMP     AL,NUM_0          ; ARE WE IN NUMERIC KEYPAD REGION?
11EF 77 12      JNA    KB4_1          ; JUMP IF NOT IN KEYPAD
11F1 FE CB      DEC     AL                ; CHECK LOWER BOUND OF RANGE
11F3 74 0E      JZ      KB4_1          ; JUMP IF NOT IN RANGE (ESC KEY)
;----TRANSLATE SCAN CODE TO NUMERIC KEYPAD
11F5 FE C8      DEC     AL                ; AL IS OFFSET INTO TABLE
11F7 BB 10B1 R   MOV     BX,OFFSET NUM_CODES
11FA 2E: D7     XLAT   CS:NUM_CODES      ; NEW SCAN CODE IS IN AL
11FC 80 E4 80   AND     AH,BREAK_BIT     ; ISOLATE BREAK BIT ON ORIGINAL
;         ; SCAN CODE
11FF 0A C4      OR      AL,AH            ; UPDATE KEYPAD SCAN CODE
1201 E8 59      JMP     SHORT CONT_INT   ; CONTINUE WITH INTERRUPT
1203 8A C4      KB4_1: MOV     AL,AH        ; GET BACK BREAK BIT IF SET
1205 E8 55      JMP     SHORT CONT_INT
;----CHECK FOR VALID FUNCTION KEY
1207 3C 08      K85:  CMP     AL, NUM_0      ; CHECK FOR RANGE OF INTEGERS
1208 77 2D      JA     KB7              ; JUMP IF NOT IN RANGE
120B FE CB      DEC     AL                ; CHECK FOR ESC KEY (=1)
120D 75 25      JNZ    KB6              ; NOT ESCAPE KEY, RANGE OF INTEGERS
;----ESCAPE KEY, LOCK KEYBOARD IN FUNCTION LOCK
120F F6 C4 80   TEST    AH,BREAK_BIT     ; IS THIS A BREAK CODE?
1212 75 30      JNZ    K8B              ; NO PROCESSING FOR ESCAPE BREAK
1214 F6 06 00B8 R 80  TEST    KB_FLAG_2,FN_FLAG ; TOGGLES ONLY WHEN FN HELD
;         ; CONCURRENTLY
1219 74 29      JZ      K8B              ; NOT HELD CONCURRENTLY
121B F6 06 00B8 R 40  TEST    KB_FLAG_2,FN_BREAK ; HAS THE FUNCTION KEY BEEN
;         ; RELEASED?
1220 75 22      JNZ    K8B              ; CONTINUE IF RELEASED. PROCESS AS
;         ; ESC
1222 F6 06 0017 R 03  TEST    KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT ; EITHER SHIFT?
1227 74 19      JZ      K8B              ; NOT HELD DOWN
1229 80 36 00B8 R 10  XOR     KB_FLAG_2,FN_LOCK ; TOGGLE STATE
122E 80 26 00B8 R 1F  AND     KB_FLAG_2,CLEAR_FLAGS ; TURN OFF OTHER STATES
1233 CF         IRET          ; RETURN FROM INTERRUPT
;----SCAN CODE IN RANGE 1 -> 0
1234 04 3A      K86:  ADD     AL, 5B          ; GENERATE CORRECT SCAN CODE
1236 EB 3E      JMP     SHORT KB12       ; CLEAN-UP BEFORE RETURN TO KB_INT
;----CHECK TABLE FOR OTHER VALID SCAN CODES
123B 0E         K87:  PUSH   CS                ; ESTABLISH ADDRESS OF TABLE
1239 07         POP    ES                ; BASE OF TABLE
123A BF 1069 R   MOV     OI, OFFSET KBO   ; BASE OF TABLE
123D B9 000C     MOV     CX, KBOLEN       ; LENGTH OF TABLE
1240 F2/ AE     REPNE  SCASB             ; SEARCH TABLE FOR A MATCH
1242 74 1D      JE      KB10             ; JUMP IF MATCH
;----ILLEGAL CHARACTER
1244 F6 06 00B8 R 40  K88:  TEST    KB_FLAG_2,FN_BREAK ; HAS BREAK OCCURED?
1249 74 0F      JZ      K89              ; FUNCTION KEY HAS NOT BEEN
;         ; RELEASED
124B F6 C4 80   TEST    AH,BREAK_BIT     ; IS THIS A BREAK OF AN ILLEGAL
124E 75 0A      JNZ    K89              ; DON'T RESET FLAGS ON ILLEGAL
;         ; BREAK
1250 80 26 00B8 R 1F  K89:  AND     KB_FLAG_2,CLEAR_FLAGS ; NORMAL STATE
1255 C8 06 00B7 R 00  MOV     CUR_FUNC,0       ; RETRIEVE ORIGINAL SCAN CODE
;----FUNCTION BREAK IS NOT SET
125A 8A C4      K89:  MOV     AL,AH            ; RETRIEVE ORIGINAL SCAN CODE
125C          CONT_INT:
125C E8 80      OUT    KBPORT,AL        ;
125E CD 09      INT    9H                ; ISSUE KEYBOARD INTERRUPT
1260 CF         RET_INT:
;         IRET
;----BEFORE TRANSLATION CHECK FOR ALT+FN+NUM_KEY AS NUM LOCK
1261 3C 31      K810: CMP     AL,NUM_KEY      ; IS THIS A POTENTIAL NUMLOCK?
1263 75 07      JNE    KB10_1           ; NOT A NUMKEY, TRANSLATE IT
1265 F6 06 0017 R 0B  TEST    KB_FLAG,ALT_SHIFT ; ALT HELD DOWN ALSO?
126A 74 D8      JZ      K8B              ; TREAT AS ILLEGAL COMBINATION
126C B9 106A R   KB10_1: MOV     CX, OFFSET KBO + 1 ; GET OFFSET TO TABLE
126F 28 F9      SUB     DI, CX            ; UPDATE INDEX TO NEW SCAN CODE
;         ; TABLE
1271 2E: BA B5 1075 R  MOV     AL, CS:KB1[DI]   ; MOV NEW SCAN CODE INTO REGISTER
;----TRANSLATED CODE IN AL OR AN OFFSET TO THE TABLE "SCAN"
1276 F6 C4 80   K812: TEST    AH,BREAK_BIT     ; IS THIS A BREAK CHAR?
1279 74 35      JZ      KB13             ; JUMP IF MAKE CODE
;----CHECK FOR TOGGLE KEY
127B 3C 45      CMP     AL,NUM_LOCK      ; IS THIS A NUM LOCK?
127D 74 04      JZ      KB12_1           ; JUMP IF TOGGLE KEY
127F 3C 45      CMP     AL,SCROLL_LOCK  ; IS THIS A SCROLL LOCK?
1281 75 08      JNZ    KB12_2           ; JUMP IF NOT A TOGGLE KEY
1283 0C 80      KB12_1: OR      AL,BOH          ; TURN ON BREAK BIT
1285 E6 60      OUT    KBPORT,AL
1287 CD 09      INT    9H                ; TOGGLE STATE
1289 24 7F      AND    AL,AND_MASK-BREAK_BIT ; TURN OFF BREAK BIT
128B F6 06 00B8 R 40  K812_2: TEST    KB_FLAG_2,FN_BREAK ; HAS FUNCTION BREAK OCCURED?
1290 74 11      JZ      KB12_3           ; JUMP IF BREAK HAS NOT OCCURED
1292 3A 06 00B7 R   CMP     AL,CUR_FUNC      ; IS THIS A BREAK OF OLD VALID
;         ; FUNCTION
1296 75 C8      JNE    RET_INT          ; ALLOW FURTHER CURRENT FUNCTIONS
129B 80 26 00B8 R 1F  AND    KB_FLAG_2,CLEAR_FLAGS
129D          KB12_20:
129D C8 06 00B7 R 00  MOV     CUR_FUNC,0       ; CLEAR CURRENT FUNCTION
12A2 CF         IRET          ; RETURN FROM INTERRUPT

```

```

12A3 3A 06 00B7 R
12A7 75 87
12A9 80 26 00B8 R DF
12AE EB ED
12B0 FB 08 00B8 R 40
12B5 74 0D
12B7 80 3E 00B7 R 00
12B8 74 06
12BE 38 06 00B7 R
12C2 75 8C
12C4 A2 00B7 R
12C7 3C 04
12C9 7F 91
12CB 74 34
12CD 3C 03
12CF 74 1A
12D1 FE CB
12D3 00 E0
12D5 00 E0
12D7 9B
12DB 2E: 8D 36 10BB R
12DD 03 F0
12DF 89 0004
12E2
12E2 2E: AC
12E4 E6 60
12E8 CD 09
12EB E2 FB
12EA CF
12EB FB 05 0018 R 0B
12F0 75 0E
12F2 80 0E 001B R 0B
12F7 E4 A0
12F9 F6 08 001B R 0B
12FE 75 F9
1300 CF
1301 FB 06 001B R 0B
1306 74 06
1308 80 26 001B R F7
130D CF
130E 83 C4 06
1311 07
1312 1F
1313 5A
1314 59
1315 5B
1316 E4 A0
1318 CD 05
131A 58
131B 5F
131C 5E
131D CF
131E
131E 53
131F 38 06 00B5 R
1323 74 31
1325 AB 80
1327 74 12
1329 24 7F
132B 38 06 00B5 R
132F 8A C4
1331 75 05
1333 C6 06 00B5 R 00
1338 F8
1339 8B
133A C3
KB12_3: CMP AL,CUR_FUNC ; IS THIS BREAK OF FIRST FUNCTION?
JNE RET_INT ; IGNORE
AND KB_FLAG_2,AND_MASK-FM_PENDING ; TURN OFF PENDING
FUNCTION
JMP KB12_20 ; CLEAR CURRENT FUNCTION AND RETURN
;----VALID MAKE KEY HAS BEEN PRESSED
KB13: TEST KB_FLAG_2,FM_BREAK ; CHECK IF FUNCTION KEY HAS BEEN
PRESSED
JZ KB14_1 ; JUMP IF NOT SET
;----FUNCTION BREAK HAS ALREADY OCCURED
CMP CUR_FUNC,0 ; IS THIS A NEW FUNCTION?
JZ KB14_1 ; INITIALIZE NEW FUNCTION
CMP CUR_FUNC,AL ; IS THIS NON-CURRENT FUNCTION
JNZ KB85 ; JUMP IF NO FUNCTION IS PENDING
; TO RETRIEVE ORIGINAL SCAN CODE
;----CHECK FOR SCAN CODE GENERATION SEQUENCE
KB14_1: MOV CUR_FUNC,AL ; INITIALIZE CURRENT FN
KB16: CMP AL,PRT_SCREEN ; IS THIS A SIMULATED SEQUENCE?
JG CONT_INT ; JUMP IF THIS IS A SIMPLE
TRANSLATION
JZ PRTSC ; DO THE PRINT SCREEN FUNCTION
CMP AL,PAUSE ; IS THIS THE HOLD FUNCTION?
JZ KB16_1 ; DO THE PAUSE FUNCTION
;----BREAK OR ECHO
DEC AL ; POINT AT BASE
SHL AL,1 ; MULTIPLY BY 4
SHL AL,1
CBW
LEA SI,SCAN ; ADDRESS SEQUENCE OF SIMULATED
KEYSTROKES
ADD SI,AX ; UPDATE TO POINT AT CORRECT SET
MOV CX,4 ; LOOP COUNTER
GENERATE:
LODS SCAN ; GET SCAN CODE FROM TABLE
OUT KBPORT,AL
INT 9H ; PROCESS IT
LOOP GENERATE ; GET NEXT
IRET
;----PUT KEYBOARD IN HOLD STATE
KB16_1: TEST KB_FLAG_1,HOLD_STATE ; CANNOT GO IN HOLD STATE IF
ITS ACTIVE
JNZ KB16_2 ; DONE WITH INTERRUPT
OR KB_FLAG_1,HOLD_STATE ; TURN ON HOLD FLAG
IN AL,NMI_PORT ; RESET KEYBOARD LATCH
HOLD: TEST KB_FLAG_1,HOLD_STATE ; STILL IN HOLD STATE?
JNZ HOLD ; CONTINUE LOOPING UNTIL KEY IS
PRESSED
KB16_2: IRET ; RETURN FROM INTERRUPT 4BH
;----PRINT SCREEN FUNCTION
PRTSC: TEST KB_FLAG_1,HOLD_STATE ; IS HOLD STATE IN PROGRESS?
JZ KB16_3 ; OK TO CONTINUE WITH PRTSC
AND KB_FLAG_1,OFFH-HOLD_STATE ; TURN OFF FLAG
IRET
KB16_3: AOD SP,3*2 ; GET RID OF CALL TO INTERRUPT 4BH
POP ES ; POP REGISTERS THAT AREN'T
MODIFIED IN INT5
POP DS
POP DX
POP CX
POP BX
IN AL,NMI_PORT ; RESET KEYBOARD LATCH
INT 5H ; ISSUE INTERRUPT
POP AX
POP DI
POP SI ; POP THE REST
IRET
KEY62_INT ENDP
;-----
; TYPAMATIC
; THIS ROUTINE WILL CHECK KEYBOARD STATUS BITS IN KB_FLAG_2
; AND DETERMINE WHAT STATE THE KEYBOARD IS IN. APPROPRIATE
; ACTION WILL BE TAKEN.
; INPUT
AL= SCAN CODE OF KEY WHICH TRIGGERED NON-MASKABLE INTERRUPT
; OUTPUT
CARRY BIT = 1 IF NO ACTION IS TO BE TAKEN.
CARRY BIT = 0 MEANS SCAN CODE IN AL SHOULO BE PROCESSED
FURTHER.
; MODIFICATIONS TO THE VARIABLES CUR_CHAR AND VAR_DELAY ARE
; MADE. ALSO THE PUTCHAR BIT IN KB_FLAG_2 IS TOGGLED WHEN
; THE KEYBOARD IS IN HALF RATE MODE.
;-----
TPM PROC NEAR
PUSH BX
CMP CUR_CHAR,AL ; IS THIS A NEW CHARACTER?
JZ TP2 ; JUMP IF SAME CHARACTER
;----NEW CHARACTER CHECK FOR BREAK SEQUENCES
TEST AL,BREAK_BIT ; IS THE NEW KEY A BREAK KEY?
JZ TP0 ; JUMP IF NOT A BREAK
AND AL,07FH ; CLEAR BREAK BIT
CMP CUR_CHAR,AL ; IS NEW CHARACTER THE BREAK OF
LAST MAKE?
MOV AL,AH ; RETRIEVE ORIGINAL CHARACTER
JNZ TP ; JUMP IF NOT THE SAME CHARACTER
MOV CUR_CHAR,00 ; CLEAR CURRENT CHARACTER
TP: CLC ; CLEAR CARRY BIT
POP BX
RET ; RETURN

```

```

;----INITIALIZE A NEW CHARACTER
133B A2 00B5 R TP0: MOV CUR_CHAR,AL ; SAVE NEW CHARACTER
133E 80 26 00B6 R FO AND VAR_DELAY,OF0H ; CLEAR VARIABLE DELAY
1343 80 26 00B8 R FE AND KB_FLAG_2,OF0H ; INITIAL PUTCHAR BIT AS ZERO
1348 F6 06 00B8 R O2 TEST KB_FLAG_2,INIT_DELAY ; ARE WE INCREASING THE
; INITIAL DELAY?
134D 74 E9 JZ TP ; INITIAL DELAY?
134F 80 0E 00B6 R OF OR VAR_DELAY,DELAY_RATE ; INCREASE DELAY BY 2X
1354 EB E2 JMP SHORT TP
;----CHECK IF WE ARE IN TYPAMATIC MODE AND IF DELAY IS OVER
1356 F6 06 00B8 R O8 TP2: TEST KB_FLAG_2,TYPE_OFF ; IS TYPAMATIC TURNED OFF?
135B 76 28 JNZ TP4 ; JUMP IF TYPAMATIC RATE IS OFF
136D 8A 1E 00B6 R MOV BL,VAR_DELAY ; GET VAR_DELAY
1361 80 E3 0F AND BL,OFH ; MASK OFF HIGH ORDER(SCREEN RANGE)
1364 0A DB OR BL,BL ; IS INITIAL DELAY OVER?
1366 74 00 JZ TP3 ; JUMP IF DELAY IS OVER
1368 FE C8 DEC BL ; DECREASE DELAY WAIT BY ANOTHER
; CHARACTER
136A 80 26 00B8 R FO AND VAR_DELAY,OF0H
136F 08 1E 00B6 R OR VAR_DELAY,BL
1373 EB 13 JMP SHORT TP4
;----CHECK IF TIME TO OUTPUT CHAR
1375 F6 06 00B8 R O4 TP3: TEST KB_FLAG_2,HALF_RATE ; ARE WE IN HALF RATE MODE
137A 74 BC JZ TP ; JUMP IF WE ARE IN NORMAL MODE
137C 80 36 00B8 R O1 XOR KB_FLAG_2,PUTCHAR ; TOGGLE BIT
1381 F6 06 00B8 R O1 TEST KB_FLAG_2,PUTCHAR ; IS IT TIME TO PUT OUT A CHAR
1386 75 B0 JNZ TP ; NOT TIME TO OUTPUT CHARACTER
TP4: ; SKIP THIS CHARACTER
; SET CARRY FLAG
1388 F9 STC
1389 5B POP BX
138A C3 RET
138B ENDP
TPN ENDP
;-----
; THIS SUBROUTINE SETS DS TO POINT TO THE BIOS DATA AREA
; INPUT: NONE
; OUTPUT: DS IS SET
;-----
138B DD5 PROC NEAR
138B 50 PUSH AX
138C 88 0040 MOV AX,40H
138F BE DB MOV DS,AX
1391 5B POP AX
1392 C3 RET
1393 DD5 ENDP
;-----
; INT 1A
; TIME_OF_DAY/SOUND SOURCE SELECT
; THIS ROUTINE ALLOWS THE CLOCK TO BE SET/READ.
; AM INTERFACE FOR SETTING THE MULTIPLEXER FOR
; AUDIO SOURCE IS ALSO PROVIDED
;
; INPUT
; (AH) = 0 READ THE CURRENT CLOCK SETTING
; RETURNS CX = HIGH PORTION OF COUNT
; DX = LOW PORTION OF COUNT
; AL = 0 IF TIMER HAS NOT PASSED 24 HOURS
; SINCE LAST READ. < 0 IF ON ANOTHER DAY
; (AH) = 1 SET THE CURRENT CLOCK
; CX = HIGH PORTION OF COUNT
; DX = LOW PORTION OF COUNT
; (AH) = 80H SET UP SOUND MULTIPLEXER
; AL =(SOURCE OF SOUND) --> "AUDIO OUT" OR RF MODULATOR
; 00 = 8253 CHANNEL 2
; 01 = CASSETTE INPUT
; 02 = "AUDIO IN" LINE ON I/O CHANNEL
; 03 = COMPLEX SOUND GENERATOR CHIP
;
; NOTE: COUNTS OCCUR AT THE RATE OF 1193180/85536 COUNTS/SEC
; (OR ABOUT 16.2 PER SECOND -- SEE EQUATES BELOW)
;-----
ASSUME CS:CODE,DS:DATA
1393 TIME_OF_DAY PROC FAR
1393 F8 STI ; INTERRUPTS BACK ON
1394 1E PUSH DS ; SAVE SEGMENT
1395 EB 138B R CALL DD5
1398 80 FC 80 CMP AH,80H ; AH=80
1398 74 2E JE T4A ; MUX_SET-UP
139D 0A E4 OR AH,AH ; AH=0
139F 74 07 JZ T2 ; READ_TIME
13A1 FE CC DEC AH ; AH=1
13A3 74 16 JZ T3 ; SET_TIME
13A5 FB STI ; INTERRUPTS BACK ON
13A6 1F POP OS ; RECOVER SEGMENT
13A7 CF IRET ; RETURN TO CALLER
13A8 FA CLI ; NO TIMER INTERRUPTS WHILE READING
13A9 A0 0070 R MOV AL,TIMER_OFL
13AC C8 08 0070 R 00 MOV CX,TIMER_OFL,0 ; GET OVERFLOW, AND RESET THE FLAG
13B1 88 0E 006E R MOV CX,TIMER_HIGH
13B5 8B 16 006C R MOV 0X,TIMER_LOW
13B9 E8 EA JMP T1 ; TOD_RETURN
13BB FA CLI ; NO INTERRUPTS WHILE WRITING
13BC 89 16 006C R MOV TIMER_LOW,DX
13C0 89 0E 006E R MOV TIMER_HIGH,CX ; SET THE TIME
13C4 C8 08 0070 R 00 MOV TIMER_OFL,0 ; REBET OVERFLOW
13C9 E8 DA JMP T1 ; TOD_RETURN

```

```

13CB 81
13CC 81 05
13CE D2 E0
13D0 86 C4
13D2 E4 61
13D4 24 9F
13D6 0A C4
13DB E6 81
13DA 59
13DB EB C8
13DD

```

```

T4A:  PUSH  CX
      MOV  CL,5
      SAL  AL,CL
      XCHG AL,AX
      IN   AL,PORT_B
      AND  AL,1001111B
      OR   AL,AX
      OUT  PORT_B,AL
      POP  CX
      JMP  T1
TIME_OF_DAY INT1
INT1  INT 16
KEYBOARD  I/O

```

THESE ROUTINES PROVIDE KEYBOARD SUPPORT

```

INPUT
(AH)=0  READ THE NEXT ASCII CHARACTER STRUCK FROM THE
        KEYBOARD, RETURN THE RESULT IN (AL), SCAN CODE IN
        (AH)
(AH)=1  SET THE Z FLAG TO INDICATE IF AN ASCII CHARACTER IS
        AVAILABLE TO BE READ.
        (ZF)=1 -- NO CODE AVAILABLE
        (ZF)=0 -- CODE IS AVAILABLE
        IF ZF = 0, THE NEXT CHARACTER IN THE BUFFER TO BE
        READ IS IN AX, AND THE ENTRY REMAINS IN THE BUFFER
(AH)=2  RETURN THE CURRENT SHIFT STATUS IN AL REGISTER
        THE BIT SETTINGS FOR THIS CODE ARE INDICATED IN THE
        THE EQUATES FOR KB_FLAG
(AH)=3  SET TYPAMATIC RATES. THE TYPAMATIC RATE CAN BE
        CHANGED USING THE FOLLOWING FUNCTIONS:
        (AL)=0  RETURN TO DEFAULT. RESTORES ORIGINAL
        STATE. I.E. TYPAMATIC ON; NORMAL INITIAL
        DELAY, AND NORMAL TYPAMATIC RATE.
        (AL)=1  INCREASE INITIAL DELAY. THIS IS THE
        DELAY BETWEEN THE FIRST CHARACTER AND
        THE BURST OF TYPAMATIC CHARS.
        (AL)=2  HALF_RATE. SLOWS TYPAMATIC CHARACTERS
        BY ONE HALF.
        (AL)=3  COMBINES AL=1 AND AL=2. INCREASES
        INITIAL DELAY AND SLOWS TYPAMATIC
        CHARACTERS BY ONE HALF.
        (AL)=4  TURN OFF TYPAMATIC CHARACTERS. ONLY THE
        FIRST CHARACTER IS HONORED. ALL OTHERS
        ARE IGNORED.
        AL IS RANGE CHECKED. IF AL<0 OR AL>4 THE STATE
        REMAINS THE SAME.
        ***** EACH TIME THE TYPAMATIC RATES ARE
        CHANGED ALL PREVIOUS STATES ARE REMOVED. I.E. IF
        THE KEYBOARD IS IN THE HALF RATE MODE AND YOU WANT
        TO ADD AN INCREASE IN TYPAMATIC DELAY, YOU MUST
        CALL THIS ROUTINE WITH AH=3 AND AL=3.
        ADJUST KEYBOARD BY THE VALUE IN AL AS FOLLOWS:
        (AL)=0  TURN OFF KEYBOARD CLICK.
        (AL)=1  TURN ON KEYBOARD CLICK.
        AL IS RANGE CHECKED. THE STATE IS UNALTERED IF
        AL <> 1,0.

```

OUTPUT

AS NOTED ABOVE, ONLY AX AND FLAGS CHANGED
ALL REGISTERS RETAINED

```

13DD
13DD FB
13DE 1E
13DF 53
13E0 EB 13BB R
13E3 0A E4
13E5 74 0A
13E7 FE CC
13E9 74 1E
13EB FE CC
13ED 74 2B
13EF EB 2E
13F1
13F1 FB
13F2 90
13F3 FA
13F4 8B 1E 001A R
13FB 3B 1E 001C R
13FC 74 F3
13FE 8B 07
1400 EB 144F R
1403 89 1E 001A R
1407 EB 43

```

KEYBOARD_ID PROC FAR
ASSUME CS:CODE,DS:DATA

```

13DD  STI
13DE  PUSH DS
13DF  PUSH BX
13E0  CALL DDS
13E3  OR AH,AX
13E5  JZ K1
13E7  DEC AH
13E9  JZ K2
13EB  DEC AH
13ED  JZ K3
13EF  JNP SHORT K3_1

```

READ THE KEY TO FIGURE OUT WHAT TO DO

```

K1:  STI
      NOP
      CLI
      MOV BX,BUFFER_HEAD
      CMP BX,BUFFER_TAIL
      JZ K1
      MOV AX,[BX]
      CALL K4
      MOV BUFFER_HEAD,BX
      JMP SHORT RET_INT16

```

ASCII STATUS

```

K2:  CLI
      MOV BX,BUFFER_HEAD
      CMP BX,BUFFER_TAIL
      MOV AX,[BX]
      STI
      POP BX
      POP DS
      RET 2

```

SHIFT STATUS

```

K3:  MOV AL,KB_FLAG
      JMP SHORT RET_INT16

```

```

1409 FA
140A 8B 1E 001A R
140E 3B 1E 001C R
1412 8B 07
1414 FB
1415 5B
1416 1F
1417 CA 0002
141A A0 0017 R
141D EB 2D

```

```

----- ADJUST KEY CLICK
K3_1: DEC AH ; AH=3, ADJUST TYPAMATIC
      JZ K3_3 ; RANGE CHECK FOR AH=4
      DEC AH ; ILLEGAL FUNCTION CALL
      JNZ RET_INT16 ; TURN OFF KEYBOARD CLICK?
      OR AL,AL ; JUMP FOR RANGE CHECK
      JNZ K3_2 ; NOT IN RANGE, RETURN
      AND KB_FLAG_1,AND_MASK-CLICK_ON ; TURN OFF CLICK
      JMP SHORT RET_INT16
K3_2: CMP AL,1 ; RANGE CHECK
      JNE RET_INT16 ; NOT IN RANGE, RETURN
      OR KB_FLAG_1,CLICK_ON ; TURN ON KEYBOARD CLICK
      JMP SHORT RET_INT16
----- SET TYPAMATIC
K3_3: CMP AL,4 ; CHECK FOR CORRECT RANGE
      JG RET_INT16 ; IF ILLEGAL VALUE IN AL IGNORE
      AND KB_FLAG_2,OF1H ; MASK OFF ANY OLD TYPAMATIC STATES
      SHL AL,1 ; SHIFT TO PROPER POSITION
      OR KB_FLAG_2,AL
RET_INT16: POP BX ; RECOVER REGISTER
         POP OS ; RECOVER REGISTER
         IRET ; RETURN TO CALLER
KEYBOARD_IO ENDP
----- INCREMENT A BUFFER POINTER
K4 PROC NEAR ; MOVE TO NEXT WORD IN LIST
      INC BX
      IMC BX
      CMP BX,BUFFER_END ; AT END OF BUFFER?
      JNE KS ; NO, CONTINUE
      MOV BX,BUFFER_START ; YES, RESET TO BUFFER BEGINNING
K5: RET
K4 ENDP
----- TABLE OF SHIFT KEYS AND MASK VALUES
K6 LABEL BYTE
      DB INS_KEY ; INSERT KEY
      DB CAPS_KEY,NUM_KEY,SCROLL_KEY,ALT_KEY,CTL_KEY
      DB LEFT_KEY,RIGHT_KEY
K6L EQU $-K6
----- SHIFT_MASK_TABLE
K7 LABEL BYTE
      DB INS_SHIFT ; INSERT MODE SHIFT
      DB CAPS_SHIFT,NUM_SHIFT,SCROLL_SHIFT,ALT_SHIFT,CTL_SHIFT
      DB LEFT_SHIFT,RIGHT_SHIFT
----- SCAN CODE TABLES
K8 DB 27,-1,0,-1,-1,-1,30,-1
      DB -1,-1,-1,31,-1,127,-1,17
      DB 23,5,18,20,25,21,9,15
      DB 16,27,29,10,-1,1,19
      DB 4,6,7,8,10,11,12,-1,-1
      DB -1,-1,28,26,24,3,22,2
      DB 14,13,-1,-1,-1,-1,-1,-1
      DB ' ',-1
----- CTL TABLE SCAN
K9 LABEL BYTE
      DB 94,95,96,97,98,99,100,101
      DB 102,103,-1,-1,119,-1,132,-1
      DB 115,-1,116,-1,117,-1,118,-1
      DB -1
----- LC TABLE
K10 LABEL BYTE
      DB 01BH,'1234567890','=','0BH,09H
      DB 'qwertyuiop[]',0DH,-1,'asdfghjkl;',027H
      DB 60H,-1,5CH,'zxcvbnm, /',-1,'*','-1,' '
      DB -1
----- UC TABLE
K11 LABEL BYTE
      DB 27,'!@#$%',37,05EH,'&()*_+',0BH,0
      DB 'QWERTYUIOP()',0DH,-1,'ASDFGHJKL:'
      DB 07EH,-1,':ZXCVBNM<?>',-1,0,-1,' ',-1

```



```

----- UC TABLE SCAN
K12 LABEL BYTE
      DB      B4,85,86,B7,88,89,90
      DB      91,92,93
----- ALT TABLE SCAN
K13 LABEL BYTE
      DB      104,108,106,107,108
      DB      109,110,111,112,113
----- NUM STATE TABLE
K14 LABEL BYTE
      DB      '789-456+1230.'
----- BASE CASE TABLE
K15 LABEL BYTE
      DB      71,72,73,-1,78,-1,77
      DB      -1,79,80,81,82,63
----- KEYBOARD INTERRUPT ROUTINE
KB_INT PROC FAR
      STI
      PUSH AX
      PUSH BX
      PUSH CX
      PUSH DX
      PUSH SI
      PUSH DI
      PUSH DS
      PUSH ES
      CLD
      CALL DDS
      MOV AH,AL
----- TEST FOR OVERRUN SCAN CODE FROM KEYBOARD
      CMP AL,OFFH
      JNZ K16
      MOV BX,80H
      MOV CX,48H
      CALL KB_NOISE
      AND KB_FLAG,OF0H
----- SHIFT KEY FOUND
K17: SUB D1,OFFSET K6+1
      MOV AH,CS:K7ID1J
      TEST AL,80H
      JNZ K23
----- SHIFT MAKE FOUND, DETERMINE SET OR TOGGLE
      CMP AH,SCROLL_SHIFT
      JAE K18
----- PLAIN SHIFT KEY, SET SHIFT ON
      OR KB_FLAG,AH
      JMP K26
----- TOGGLED SHIFT KEY, TEST FOR 1ST MAKE OR NOT
K18: TEST KB_FLAG,CTL_SHIFT
      JNZ K25
      CMP AL,INS_KEY
      JNZ K22
      TEST KB_FLAG,ALT_SHIFT
      JNZ K25
      TEST KB_FLAG,NUM_STATE
      JNZ K21
      TEST KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT
      JZ K22
----- K20: MOV AX,5230H
      JMP K57
----- K21: TEST KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT
      JZ K20
----- K22: TEST AH,KB_FLAG_1
      JNZ K26
      OR KB_FLAG_1,AH
----- XOR KB_FLAG,AH
      CMP AL,INS_KEY
      JNE K26
      MOV AX,INS_KEY*256
      JMP K57
      ; ALLOW FURTHER INTERRUPTS
      ; FORWARD DIRECTION
      ; SAVE SCAN CODE IN AH
      ; IS THIS AN OVERRUN CHAR?
      ; NO, TEST FOR SHIFT KEY
      ; DURATION OF ERROR BEEP
      ; FREQUENCY OF TONE
      ; BUFFER FULL BEEP
      ; CLEAR ALL, CLRRL, LEFT AND RIGHT
      ; SHIFTS
      ; CLEAR POTENTIAL BREAK OF INS,CAPS
      ; NUM AND SCROLL SHIFT
      ; CLEAR FUNCTION STATES
      ; END OF INTERRUPT
      ; TEST SHIFT
      ; TURN OFF THE BREAK BIT
      CS
      ES
      ; ESTABLISH ADDRESS OF SHIFT TABLE
      ; SHIFT KEY TABLE
      ; LENGTH
      ; LOOK THROUGH THE TABLE FOR A
      ; MATCH
      ; RECOVER SCAN CODE
      ; JUMP IF MATCH FOUND
      ; IF NO MATCH, THEN SHIFT NOT FOUND
      ; ADJUST PTR TO SCAN CODE MATCH
      ; GET MASK INTO AH
      ; TEST FOR BREAK KEY
      ; BREAK_SHIFT_FOUND
      ; IF SCROLL SHIFT OR ABOVE, TOGGLE
      ; KEY
      ; TURN ON SHIFT BIT
      ; INTERRUPT_RETURN
      ; SHIFT+TOGGLE
      ; CHECK CTL_SHIFT STATE
      ; JUMP IF CTL_SHIFT
      ; CHECK FOR INSERT KEY
      ; JUMP IF NOT INSERT KEY
      ; CHECK FOR ALTERNATE SHIFT
      ; JUMP IF ALTERNATE SHIFT
      ; CHECK FOR BASE STATE
      ; JUMP IF NUM LOCK IS ON
      ; JUMP IF BASE STATE
      ; NUMERIC ZERO, NOT INSERT KEY
      ; PUT OUT AN ASCII ZERO
      ; BUFFER_FILL
      ; MIGHT BE NUMERIC
      ; SHIFT TOGGLE KEY HIT; PROCESS IT
      ; IS KEY ALREADY DEPRESSED
      ; JUMP IF KEY ALREADY DEPRESSED
      ; INDICATE THAT THE KEY IS
      ; DEPRESSED
      ; TOGGLE THE SHIFT STATE
      ; TEST FOR 1ST MAKE OF INSERT KEY
      ; JUMP IF NOT INSERT KEY
      ; SET SCAN CODE INTO AH, 0 INTO AL
      ; PUT INTO OUTPUT_BUFFER

```

```

;----- BREAK SHIFT FOUND
1600                                K23:
1600      80 FC 10      CMP      AH,SCROLL_SHIFT ; BREAK-SHIFT-FOUND
1603      73 1A      JAE      K24 ; IS THIS A TOGGLE KEY
1605      F6 04      NOT      AH ; YES, HANDLE BREAK TOGGLE
1607      20 26 0017 R AND      KB_FLAG,AH ; INVERT MASK
160B      3C 88      CMP      AL,ALT_KEY+80H ; TURN OFF SHIFT BIT
160D      75 38      JNE      K26 ; IS THIS ALTERNATE SHIFT RELEASE
;----- ALTERNATE SHIFT KEY RELEASED, GET THE VALUE INTO BUFFER
160F      A0 0019 R   MOV      AL,ALT_INPUT ; INTERRUPT_RETURN
1612      32 E4      XOR      AH,AH ; SCAN CODE OF 0
1614      8B 26 0019 R MOV      ALT_INPUT,AH ; ZERO OUT THE FIELD
1618      0A C0      OR      AL,AL ; WAS THE INPUT=0?
161A      74 2E      JE      K26 ; INTERRUPT_RETURN
161C      E9 17F5 R   JMP      K5B ; IT WASN'T, SO PUT IN BUFFER
161F                                K24:
181F      3C 8A      CMP      AL,CAPS_KEY+BREAK_BIT ; SPECIAL CASE OF TOGGLE KEY
1821      75 0F      JNE      K24_1 ; JUMP AROUND POTENTIAL UPDATE
1823      F6 06 0018 R O2 TEST     KB_FLAG_1,CLICK_SEQUENCE
1828      74 08      JZ      K24_1 ; JUMP IF NOT SPECIAL CASE
182A      80 26 0018 R FD AND      KB_FLAG_1,AND_MASK-CLICK_SEQUENCE ; MASK OFF MAKE
;----- BREAK OF NORMAL TOGGLE
182F      E8 19 90      JMP      K26 ; OF CLICK
;----- BREAK OF NORMAL TOGGLE
1832      F6 04      K24_1: NOT      AH ; INTERRUPT IS OVER
1834      20 26 0018 R AND      KB_FLAG_1,AH ; INVERT MASK
1838      EB 10      JMP      SHORT K26 ; INDICATE NO LONGER DEPRESSED
;----- TEST FOR HOLD STATE
183A                                K25:
183A      3C 80      CMP      AL,60H ; NO-SHIFT-FOUND
183C      73 0C      JAE      K26 ; TEST FOR BREAK KEY
;----- TEST FOR HOLD STATE
183E      F6 06 0018 R O8 TEST     KB_FLAG_1,HOLD_STATE ; ARE WE IN HOLD STATE?
1843      74 0E      JZ      K2B ; BRANCH AROUND TEST IF NOT
1845      80 26 0018 R F7 AND      KB_FLAG_1,NOT_HOLD_STATE ; TURN OFF THE HOLD STATE
;----- TEST FOR HOLD STATE
184A                                K26:
184A      07      POP      ES ; INTERRUPT-RETURN
184B      1F      POP      DS
184C      8F      POP      DI
184D      9E      POP      SI
184E      5A      POP      DX
184F      59      POP      CX
1850      58      POP      BX
1851      58      POP      AX ; RESTORE STATE
1852      CF      IRET ; RETURN, INTERRUPTS BACK ON WITH
;----- NOT IN HOLD STATE, TEST FOR SPECIAL CHARS
1853                                K2B:
1853      F6 06 0017 R O8 TEST     KB_FLAG,ALT_SHIFT ; NO-HOLD-STATE
1855      75 03      JNZ      K25 ; ARE WE IN ALTERNATE SHIFT
185A      E9 1749 R   JMP      K3B ; JUMP IF ALTERNATE SHIFT
;----- TEST FOR ALT+CTRL KEY SEQUENCES
1860                                K29:
1860      F6 06 0017 R O4 TEST     KB_FLAG,CTL_SHIFT ; TEST-RESET
1862      74 69      JZ      K31 ; ARE WE IN CONTROL SHIFT ALSO
1864      3C 53      CMP      AL,DEL_KEY ; NO_RESET
1868      75 09      JNE      K29_1 ; SHIFT STATE IS THERE, TEST KEY
;----- CTL-ALT-DEL HAS BEEN FOUND, DO I/O CLEANUP
1868      C7 06 0072 R 1234 MOV      RESET_FLAG,1234H ; SET FLAG FOR RESET FUNCTION
186E      E9 0043 R   JMP      NEAR PTR RESET ; JUMP TO POWER ON DIAGNOSTICS
1871      3C 52      CMP      AL,INS_KEY ; CHECK FOR RESET WITH DIAGNOSTICS
1873      75 09      JNE      K29_2 ; CHECK FOR OTHER
;----- ALT-CTRL-INS HAS BEEN FOUND
1875      C7 06 0072 R 4321 MOV      RESET_FLAG,4321H ; SET FLAG FOR DIAGNOSTICS
187B      E9 0043 R   JMP      NEAR PTR RESET ; LEVEL 1 DIAGNOSTICS
187E      3C 3A      CMP      AL,CAPS_KEY ; CHECK FOR KEYBOARD CLICK TOGGLE
1880      75 13      JNE      K29_3 ; CHECK FOR SCREEN ADJUSTMENT
;----- ALT+CTRL+CAPSLOCK HAS BEEN FOUND
1882      F6 06 0018 R O2 TEST     KB_FLAG_1,CLICK_SEQUENCE
1887      75 C1      JNZ      K26 ; JUMP IF SEQUENCE HAS ALREADY
;----- ALT+CTRL+CAPSLOCK HAS BEEN FOUND
1889      80 36 0018 R O4 XOR      KB_FLAG_1,CLICK_ON ; OCCURED
;----- ALT+CTRL+CAPSLOCK HAS BEEN FOUND
188E      80 0E 0018 R O2 OR      KB_FLAG_1,CLICK_SEQUENCE ; FEEDBACK
1893      EB 85      JMP      SHORT K26 ; SET CLICK_SEQUENCE STATE
1895      3C 40      CMP      AL,RIGHT_ARROW ; INTERRUPT IS OVER
1897      75 12      JNE      K29_4 ; LEVEL 1 DIAGNOSTICS
1899      EB 188E R   CALL     GET_POS ; LOOK FOR RIGHT ADJUSTMENT
;----- ALT+CTRL+CAPSLOCK HAS BEEN FOUND
189C      3C FC      CMP      AL,0-RANGE ; GET THE # OF POSITIONS SCREEN IS
;----- ALT+CTRL+CAPSLOCK HAS BEEN FOUND
189E      7C AA      JL      K26 ; SHIFTD
18A0      FE 0E 0089 R DEC     HORZ_POS ; IS SCREEN SHIFTED AS FAR AS
18A4      FE CB      DEC     AL ; POSSIBLE?
18A8      EB 187A R   CALL     PUT_POS ; OUT OF RANGE
18AA      EB 14      CALL     PUT_POS ; SHIFT VALUE TO THE RIGHT
18AB      EB 14      CALL     PUT_POS ; DECREASE RANGE VALUE
18AC      EB 14      CALL     PUT_POS ; RESTORE STORAGE LOCATION
18AD      EB 14      CALL     PUT_POS ; ADJUST
18AE      EB 14      CALL     PUT_POS ; ADJUST SCREEN TO THE LEFT?
18AF      EB 188E R   CALL     GET_POS ; NOT AN ALT_CTRL SEQUENCE
;----- ALT+CTRL+CAPSLOCK HAS BEEN FOUND
18B2      3C 04      CMP      AL,RANGE ; GET NUMBER OF POSITIONS SCREEN IS
;----- ALT+CTRL+CAPSLOCK HAS BEEN FOUND
18B4      7F 94      JC      K26 ; SHIFTD
18B6      FE 06 0089 R INC     HORZ_POS ; IS SCREEN SHIFTED AS FAR AS
18BA      FE C0      INC     AL ; POSSIBLE?
;----- ALT+CTRL+CAPSLOCK HAS BEEN FOUND
18BC      EB 187A R   CALL     PUT_POS ; SHIFT SCREEN TO THE LEFT
;----- ALT+CTRL+CAPSLOCK HAS BEEN FOUND
18BE      FE 06 0089 R INC     HORZ_POS ; INCREASE NUMBER OF POSITIONS
18C0      FE C0      INC     AL ; SCREEN IS SHIFTED
;----- ALT+CTRL+CAPSLOCK HAS BEEN FOUND
18C2      EB 187A R   CALL     PUT_POS ; PUT POSITION BACK IN STORAGE

```

```

168F 80 02          K29_5: MOV     AL,2           ; ADJUST
16C1 BA 03D4       MOV     DX,3D4H        ; ADDRESS TO CRT CONTROLLER
16C4 EE           OUT     DX,AL
16C5 A0 0089 R    MOV     AL,HORZ_POS    ; COLUMN POSITION
16C8 42           INC     DX              ; POINT AT DATA REGISTER
16C9 EE           OUT     DX,AL          ; MOV POSITION
16CA E9 164A R    JMP     K26

;-----
16CD             IN ALTERNATE SHIFT, RESET NOT FOUND
16CD             K31: CMP     AL,57         ; NO-RESET
16CE             JNE     K32         ; TEST FOR SPACE KEY
16CF             MOV     AL,' '      ; NOT THERE
16D0             JMP     K57         ; SET SPACE CHAR
16D3             JNE     K57         ; BUFFER_FILL

;-----
16D6             ALT-INPUT-TABLE
16D6             K30 LABEL BYTE
16D6             DB     82,79,80,81,75,76,77

16DD             DB     71,72,73           ; 10 NUMBERS ON KEYPAD

;-----
16E0             SUPER-SHIFT-TABLE
16E0             DB     16,17,18,19,20,21,22,23 ; A-Z TYPEWRITER CHARS

16E8             DB     24,25,30,31,32,33,34,35

16F0             DB     36,37,38,44,45,46,47,48

16FB             DB     49,50

;-----
16FA             LOOK FOR KEY PAD ENTRY
16FA             K32: MOV     DI,OFFSET K30 ; ALT-KEY-PAD
16FB             MOV     CX,10        ; ALT-INPUT-TABLE
1700             REPNE SCASB        ; LOOK FOR ENTRY USING KEYPAD
1702             JNE     K33         ; LOOK FOR MATCH
1704             SUB     DI,OFFSET K30+1 ; NO ALT-KEYPAD
1704             MOV     AL,ALT_INPUT ; D1 NOW HAS ENTRY VALUE
1708             MOV     AH,10        ; GET THE CURRENT BYTE
1708             MUL     AH           ; MULTIPLY BY 10
170D             ADD     AX,D1        ; ADD IN THE LATEST ENTRY
1711             MOV     AL,ALT_INPUT,AL ; STORE IT AWAY
1714             JMP     K26         ; THROW AWAY THAT KEYSTROKE

;-----
1717             LOOK FOR SUPERSHIFT ENTRY
1717             K33: MOV     ALT_INPUT,0 ; NO-ALT-KEYPAD
1717             MOV     CX,26        ; ZERO ANY PREVIOUS ENTRY INTO
1717             REPNE SCASB        ; INPUT
171C             JNE     K34         ; D1,ES ALREADY POINTING
171F             MOV     CX,10        ; LOOK FOR MATCH IN ALPHABET
1721             JNE     K34         ; NOT FOUND, FUNCTION KEY OR OTHER
1723             XOR     AL,AL        ; ASCII CODE OF ZERO
1725             JMP     K57         ; PUT IT IN THE BUFFER

;-----
1728             LOOK FOR TOP ROW OF ALTERNATE SHIFT
1728             K34: CMP     AL,2           ; ALT-TOP-ROW
1728             JB     K35           ; KEY WITH '1' ON IT
172A             CMP     AL,14        ; NOT ONE OF INTERESTING KEYS
172C             JAE     K36         ; IS IT IN THE REGION?
172E             JAE     K36         ; ALT-FUNCTION
1730             ADD     AH,11B       ; CONVERT PSEUDO SCAN CODE TO
1730             XOR     AL,AL        ; RANGE
1733             JMP     K57         ; INDICATE AS SUCH
1735             JMP     K57         ; BUFFER_FILL

;-----
1738             TRANSLATE ALTERNATE SHIFT PSEUDO SCAN CODES
1738             K35: CMP     AL,59        ; ALT-FUNCTION
173A             JAE     K37         ; TEST FOR IN TABLE
173C             JMP     K26         ; ALT-CONTINUE
173C             K36: JMP     K26         ; CLOSE-RETURN
173F             CMP     AL,71        ; IGNORE THE KEY
1741             JAE     K36         ; ALT-CONTINUE
1743             MOV     BX,OFFSET K13 ; IN KEYPAD REGION
1746             JMP     K63         ; IF SO, IGNORE
1746             NOT IN ALTERNATE SHIFT ; ALT SHIFT PSEUDO SCAN TABLE
1746             ; TRANSLATE THAT

;-----
1749             NOT-ALT-SHIFT
1749             K38: TEST     KB_FLAG,CTL_SHIFT ; NOT-ALT-SHIFT
174E             JZ     K44         ; ARE WE IN CONTROL SHIFT?
174E             ; NOT-CTL-SHIFT

;-----
1750             CONTROL SHIFT, TEST SPECIAL CHARACTERS
1750             ; TEST FOR BREAK AND PAUSE KEYS
1750             CMP     AL,SCROLL_KEY ; TEST FOR BREAK
1752             JNE     K41         ; NO-BREAK
1754             MOV     BX,BUFFER_HEAD ; GET CURRENT BUFFER HEAD
1758             MOV     BIOS_BREAK,80H ; TURN ON BIOS_BREAK BIT
175D             INT     1BH         ; BREAK INTERRUPT VECTOR
175F             SUB     AX,AX        ; PUT OUT DUMMY CHARACTER
1761             MOV     [BX],AX      ; PUT DUMMY CHAR AT BUFFER HEAD
1763             CALL    K4          ; UPDATE BUFFER POINTER
1766             MOV     BUFFER_TAIL,BX ; UPDATE TAIL
176A             JMP     K26         ; DONE WITH INTERRUPT
176D             K41:             ; NO-PAUSE

;-----
176D             TEST SPECIAL CASE KEY 55
176D             CMP     AL,55
176F             JNE     K42         ; NOT-KEY-55
1771             MOV     AX,114*256 ; START/STOP PRINTING SWITCH
1774             JMP     K57         ; BUFFER_FILL

```

```

;----- SET UP TO TRANSLATE CONTROL SHIFT
1777 BB 146C R K42: MOV BX,OFFSET K8 ; NOT-KEY-55
177A 3C 3B ; SET UP TO TRANSLATE CTL
177C 72 6A ; IS IT IN TABLE?
;----- MOV BX,OFFSET K9 ; YES, GO TRANSLATE CHAR
177E BB 14A8 R ; CTL-TABLE-TRANSLATE
1781 E9 1863 R ; CTL TABLE SCAN
;----- JMP K63 ; TRANSLATE_SCAN
;----- NOT IN CONTROL SHIFT
1784 ; NOT-CTL-SHIFT
1784 3C 47 CMP AL,71 ; TEST FOR KEYPAD REGION
1786 73 1F JAE K48 ; HANDLE KEYPAD REGION
1789 F6 06 0017 R 03 TEST KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT ; TEST FOR SHIFT STATE
178D 74 4E JZ K54 ; TEST FOR SHIFT STATE
;----- UPPER CASE, HANDLE SPECIAL CASES
178F 3C 0F CMP AL,15 ; BACK TAB KEY
1791 75 05 JNE K46 ; NOT-BACK-TAB
1793 8B 0F00 MOV AX,15*256 ; SET PSEUDO SCAN CODE
1796 EB 54 JMP SHORT K57 ; BUFFER_FILL
1798 ; NOT-PRINT-SCREEN
1798 3C 3B CMP AL,59 ; FUNCTION KEYS
179A 72 06 JB K47 ; NOT-UPPER-FUNCTION
179C BB 1533 R MOV BX,OFFSET K12 ; UPPER CASE PSEUDO SCAN CODES
179F E9 1863 R JMP K63 ; TRANSLATE_SCAN
17A2 ; NOT-UPPER-FUNCTION
17A2 BB 14F9 R K47: MOV BX,OFFSET K11 ; POINT TO UPPER CASE TABLE
17A5 EB 41 JMP SHORT K56 ; OK, TRANSLATE THE CHAR
;----- KEYPAD KEYS, MUST TEST NUM LOCK FOR DETERMINATION
17A7 ; KEYPAD-REGION
17A7 F6 06 0017 R 20 TEST KB_FLAG,NUM_STATE ; ARE WE IN NUM_LOCK?
17AC 75 21 JNZ K52 ; TEST FOR SURE
17AE F6 06 0017 R 03 TEST KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT ; ARE WE IN SHIFT
;----- STATE
17B3 75 21 JNZ K53 ; IF SHIFTED, REALLY NUM STATE
;----- BASE CASE FOR KEYPAD
17B5 ; BASE-CASE
17B5 3C 4A CMP AL,74 ; SPECIAL CASE FOR A COUPLE OF KEYS
17B7 74 0C JE K50 ; MINUS
17B9 3C 4E CMP AL,78
17BB 74 0D JE K51
17BD 2C 47 SUB AL,71 ; CONVERT DRIGIN
17BF BB 1854 R MOV BX,OFFSET K15 ; BASE CASE TABLE
17C2 E9 1865 R JMP K64 ; CONVERT TO PSEUDO SCAN
17C5 BB 4A20 K50: MOV AX,74*256+'-' ; MINUS
17C8 EB 22 JMP SHORT K57 ; BUFFER_FILL
17CA BB 4E2B K51: MOV AX,78*256+'+' ; PLUS
17CD EB 1D JMP SHORT K57 ; BUFFER_FILL
;----- MIGHT BE NUM LOCK, TEST SHIFT STATUS
17CF F6 06 0017 R 03 K52: TEST KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT ; ALMOST-NUM-STATE
17D4 75 DF JNZ K49 ; SHIFTED TEMP OUT OF NUM STATE
17D6 ; REALLY_NUM_STATE
17D6 2C 46 SUB AL,70 ; CONVERT ORIGIN
17D8 BB 1547 R MOV BX,OFFSET K14 ; NUM STATE TABLE
17DB EB 08 JMP SHORT K56 ; TRANSLATE_CHAR
;----- PLAIN OLD LOWER CASE
17DD ; NOT-SHIFT
17DD 3C 3B CMP AL,59 ; TEST FOR FUNCTION KEYS
17DF 72 04 JB K55 ; NOT-LOWER-FUNCTION
17E1 32 C0 XOR AL,AL ; SCAN CODE IN AH ALREADY
17E3 EB 07 JMP SHORT K57 ; BUFFER_FILL
17E5 ; NOT-LOWER-FUNCTION
17E5 BB 148F R K55: MOV BX,OFFSET K10 ; LC TABLE
;----- TRANSLATE THE CHARACTER
17E8 ; TRANSLATE-CHAR
17E8 FE C8 DEC AL ; CONVERT ORIGIN
17EA 2E: D7 XLAT CS:K11 ; CONVERT THE SCAN CODE TO ASCII
;----- PUT CHARACTER INTO BUFFER
17EC ; BUFFER-FILL
17EC 3C FF CMP AL,-1 ; IS THIS AN IGNORE CHAR?
17EE 74 1F JE K59 ; YES, DO NOTHING WITH IT
17F0 90 FC FF CMP AH,-1 ; LOOK FOR -1 PSEUDO SCAN
17F3 74 1A JE K59 ; NEAR_INTERRUPT_RETURN
;----- HANDLE THE CAPS LOCK PROBLEM
17F5 ; BUFFER-FILL-NOTEST
17F5 F6 06 0017 R 40 K58: TEST KB_FLAG,CAPS_STATE ; ARE WE IN CAPS LOCK STATE?
17FA 74 20 JZ K61 ; SKIP IF NOT
;----- IN CAPS LOCK STATE
17FC F6 06 0017 R 03 TEST KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT ; TEST FOR SHIFT
;----- STATE
1801 74 0F JZ K60 ; IF NOT SHIFT, CONVERT LOWER TO
;----- UPPER
1803 ; CONVERT ANY UPPER CASE TO LOWER CASE
1803 3C 41 CMP AL,'A' ; FIND OUT IF ALPHABETIC
1805 72 15 JB K61 ; NOT_CAPS_STATE
1807 3C 5A CMP AL,'Z'
1809 77 11 JA K61 ; NOT_CAPS_STATE
180B 04 20 ADD AL,'a'-'A' ; CONVERT TO LOWER CASE
180D EB 00 JMP SHORT K61 ; NOT_CAPS_STATE
180F ; NEAR-INTERRUPT_RETURN
180F E9 164A R K59: JMP K26 ; INTERRUPT_RETURN
;----- CONVERT ANY LOWER CASE TO UPPER CASE
1812 ; LOWER-TO-UPPER
1812 3C 61 CMP AL,'a' ; FIND OUT IF ALPHABETIC
1814 72 06 JB K61 ; NOT_CAPS_STATE
1816 3C 7A CMP AL,'z'
1818 77 02 JA K61 ; NOT_CAPS_STATE
181A 2C 20 SUB AL,'a'-'A' ; CONVERT TO UPPER CASE

```

```

181C
181C 9B 1E 001C R
1820 9B F3
1822 EB 144F R
1825 3E 1E 001A R
1829 75 1D
182B 53
182C 8B 0080
182F 8B 004B
1832 EB E035 R
1835 80 26 0017 R FO

163A 80 28 0018 R OF
183F 80 26 008B R IF
1844 8B
1845 E9 164A R
1848 F6 06 0018 R O4
184D 74 08
184F 53
1850 8B 0001
1853 89 0010
1856 EB E035 R

1859 5B
185A 89 04
185C 89 1E 001C R
1860 E9 164A R

1863
1863 2C 3B
1865
1865 2E: D7
1867 8A E0
1869 32 C0
186B E9 17EC R
186E

K61:
MOV BX,BUFFER_TAIL ; NOT-CAPS-STATE
MOV SI,BX ; GET THE END POINTER TO THE BUFFER
CALL K4 ; SAVE THE VALUE
CMP BX,BUFFER_HEAD ; ADVANCE THE TAIL
JNE K61_1 ; HAS THE BUFFER WRAPPED AROUND?
PUSH BX ; BUFFER_FULL,BEEP
MOV BX,080H ; SAVE BUFFER_TAIL
MOV CX,48H ; DURATION OF ERROR BEEP
CALL KB_NOISE ; FREQUENCY OF ERROR BEEP HALF TONE
AND KB_FLAG,0FOH ; OUTPUT NOISE
; CLEAR ALT, CLR_L, LEFT AND RIGHT
; SHIFTS
AND KB_FLAG_1,0FH ; CLEAR POTENTIAL BREAK OF INS, CAPS
; NUM AND SCROLL SHIFTS
AND KB_FLAG_2,1FH ; CLEAR FUNCTION STATES
POP BX ; RETRIEVE BUFFER_TAIL
JMP K26 ; RETURN FROM INTERRUPT
K61_1: TEST KB_FLAG_1,CLICK_ON ; IS AUDIO FEEDBACK ENABLED?
JZ K61_2 ; NO, JUST PUT IN BUFFER
PUSH BX ; SAVE BUFFER_TAIL VALUE
MOV BX,1H ; DURATION OF CLICK
MOV CX,10H ; FREQUENCY OF CLICK
CALL KB_NOISE ; OUTPUT AUDIO FEEDBACK OF KEY
; STROKE
POP BX ; RETRIEVE BUFFER_TAIL VALUE
K61_2: MOV [SI],AX ; STORE THE VALUE
MOV BUFFER_TAIL,BX ; MOVE THE POINTER UP
JMP K26 ; INTERRUPT_RETURN
;----- TRANSLATE SCAN FOR PSEUDO SCAN CODES
K63: SUB AL,59 ; TRANSLATE-SCAN
; CONVERT ORIGIN TO FUNCTION KEYS
K64: XLAT CS:K9 ; TRANSLATE-SCAN-ORGD
MOV AH,AL ; CTL TABLE SCAN
XOR AH,AL ; PUT VALUE INTO AH
JMP K67 ; ZERO ASCII CODE
; PUT IT INTO THE BUFFER
KB_INT ENDP
;-----
;GET_POS
; THIS ROUTINE WILL SHIFT THE VALUE STORED IN THE HIGH NIBBLE
; OF THE VARIABLE VAR_DELAY TO THE LOW NIBBLE.
;INPUT
; NONE. IT IS ASSUMED THAT OS POINTS AT THE BIOS DATA AREA
;OUTPUT
; AL CONTAINS THE SHIFTED VALUE.
;-----
186E
186E 51
186F A0 0086 R
1872 24 F0
1874 B1 04
1876 D2 FB
1878 59
1879 C3
187A

GET_POS PROC NEAR
PUSH CX ; SAVE SHIFT REGISTER
MOV AL,BYTE PTR VAR_DELAY ; GET STORAGE LOCATION
AND AL,0FOH ; MASK OFF LOW NIBBLE
MOV CL,4 ; SHIFT OF FOUR BIT POSITIONS
SAR AL,CL ; SHIFT THE VALUE SIGN EXTENDED
POP CX ; RESTORE THE VALUE
GET_POS ENDP
;-----
;PUT_POS
; THIS ROUTINE WILL TAKE THE VALUE IN LOW ORDER NIBBLE IN
; AL AND STORE IT IN THE HIGH ORDER OF VAR_DELAY
;INPUT
; AL CONTAINS THE VALUE FOR STORAGE
;OUTPUT
; NONE.
;-----
187A
187A 51
187B B1 04
187D D2 E0
187F 8A 0E 0086 R
1883 80 E1 0F
188B 0A C1
188B A2 0086 R
188B 59
188C C3
189D

PUT_POS PROC NEAR
PUSH CX ; SAVE REGISTER
MOV CL,4 ; SHIFT COUNT
SHL AL,CL ; PUT IN HIGH ORDER NIBBLE
MOV CL,BYTE PTR VAR_DELAY ; GET DATA BYTE
AND CL,0FH ; CLEAR OLD VALUE IN LOW NIBBLE
OR AL,CL ; COMBINE HIGH AND LOW NIBBLES
MOV BYTE PTR VAR_DELAY,AL ; PUT IN POSITION
POP CX ; RESTORE REGISTER
PUT_POS ENDP
;-----
; MANUFACTURING ACTIVITY SIGNAL ROUTINE - INVOKED THROUGH THE TIMER
; TICK ROUTINE DURING MANUFACTURING ACTIVITIES. (ACCESSED THROUGH
; INT 1CH)
;-----
189D
189D 50
189E 2B C0
1899 E6 13
1892 E4 61
1894 8A E0
1898 80 E4 9D
1899 F6 D0
1898 24 02
189D 0A C4
189F 0C 10
18A1 E6 81
18A3 B0 20
18A5 E6 20
18A7 58
18AB CF
18A9

MFG_TICK PROC FAR
PUSH AX
SUB AX,AX ; SEND A 00 TO PORT 13 AS A
; ACTIVITY SIGNAL
OUT 13H,AL
IN AL,PORT_B ; FLIP SPEAKER DATA TO OPPOSITE
; SENSE
MOV AH,AL ; SAVE ORIG SETTING
AND AH,10011101B ; MAKE SURE MUX IS -> RIGHT AND
; ISOLATE SPEAKER BIT
NOT AL ; FLIP ALL BITS
AND AL,0000010B ; ISOLATE SPEAKER DATA BIT (NOW IN
; OPPOSITE SENSE)
OR AL,AH ; COMBINE WITH ORIG. DATA FROM
; PORT B
OR AL,00010000B ; AND DISABLE INTERNAL SPEAKER
OUT PORT_B,AL
MOV AL,20H ; EO1 TO INTR. CHIP
OUT 20H,AL
POP AX
IRET
MFG_TICK ENDP

```

 CONVERT AND PRINT ASCII CODE

AL MUST CONTAIN NUMBER TO BE CONVERTED.
 AX AND BX DESTROYED

```

18A9          XPC_BYTE      PROC    NEAR
18A9 50        PUSH        AX          ; SAVE FOR LOW NIBBLE DISPLAY
18AA 81 04    MOV          CL,4         ; SHIFT COUNT
18AC D2 EB    SHR          AL,CL         ; NIBBLE SWAP
18AE E8 18B4 R CALL        XLAT_PR      ; DO THE HIGH NIBBLE DISPLAY
18B1 58        POP          AX          ; RECOVER THE NIBBLE
18B2 24 0F    AND          AL,0FH         ; ISOLATE TO LOW NIBBLE
18B4          XLAT_PR      PROC    NEAR      ; FALL INTO LOW NIBBLE CONVERSION
18B4 04 90    ADD          AL,090H       ; CONVERT 00-0F TO ASCII CHARACTER
18B6 27        DAA          ; ADD FIRST CONVERSION FACTOR
18B7 14 40    ADC          AL,040H       ; ADJUST FOR NUMERIC AND ALPHA
18B9 27        DAA          ; RANGE
18BA          PRT_HEX     PROC    NEAR      ; ADD CONVERSION AND ADJUST LOW
18BA 53        PUSH        BX          ; NIBBLE
18BB 84 0E    MOV          AH,14         ; DISPLAY CHARACTER IN AL
18BD B7 00    NOV          BH,0         ;
18BF CD 10    INT          10H         ; CALL VIDEO_10
18C1 58        POP          BX          ;
18C2 C3        RET
18C3          PRT_HEX     ENDP
18C3          XLAT_PR     ENDP
18C3          XPC_BYTE     ENDP
; CONTROL IS PASSED HERE WHEN THERE ARE NO PARALLEL PRINTERS
; ATTACHED. CX HAS EQUIPMENT FLAG,DS POINTS AT DATA (40H)
; DETERMINE WHICH RS232 CARD (0,1) TO USE
18C3          REPRINT    PROC    NEAR
18C3 28 D2    SUB          DX,DX        ; ASSUME TO USE CARD 0
18C5 F6 C5 04 TEST       CH,00000100B  ; UNLESS THERE ARE TWO CARDS
18C8 74 01    JE          B10_1        ; IN WHICH CASE,
18CA 42        INC          DX          ; USE CARD 1
; DETERMINE WHICH FUNCTION IS BEING CALLED
18CB 0A E4    OR          AH,AH        ; TEST FOR AH = 0
18CC 74 A1    JZ          B12          ; GO PRINT CHAR
18CF FE CC    DEC       AH          ; TEST FOR AH = 1
18D1 74 1D    JZ          B11          ; GO DO INIT
18D3 FE CC    DEC       AH          ; TEST FOR AH = 2
18D5 75 16    JNZ        SHORT B10_3  ; IF NOT VALID, RETURN
; ELSE...
18D7 50        PUSH        AX          ; SAVE AL
18D8 84 03    MOV       AH,03H       ; USE THE GET COMMO PORT
18DA CD 14    INT       014H        ; STATUS FUNCTION OF INT14
18DC E8 1925 R CALL      FAKE         ; FAKE WILL MAP ERROR BITS FROM
; RS232 TO CORRESPONDING ONES
; FOR THE PRINTER
18DF 58        POP          AX          ; RESTORE AL
18E0 0A F6    OR          DH,DH       ; CHECK IF ANY FLAGS WERE SET
18E2 74 07    JZ          B10_2       ;
18E4 8A E6    MOV       AH,DH        ; MOVE FAKED ERROR CONDITION TO AH
18E6 80 E4 FE AND       AH,0FEH      ;
18E9 EB 02    JMP       SHORT B10_3  ; THEN RETURN
18EB 84 90    MOV       AH,090H     ; MOVE IN STATUS FOR 'CORRECT'
; RETURN
18ED E9 F0D0 R JMP       B10_3        ;
; INIT COMMO PORT --- DX HAS WHICH CARD TO INIT.
; MOVE TIME OUT VALUE FROM PRINTER TO RS232 TIME OUT VALUE
18F0 88 F2    MOV       SI,DX        ; SI GETS OFFSET INTO THE TABLE
18F2 A0 007B R MOV       AL,PRINT_TIM_OUT
18F5 04 0A    ADD       AL,0AH       ; INCREASE DELAY
18F7 8B 84 007C R MOV      RS232_TIM_OUT[SI],AL
18F8 50        PUSH        AX          ; SAVE AL
18FC 80 87    MOV       AL,087H     ; SET INIT FOR: 1200 BAUD
; 8 BIT WRD LNG
; NO PARITY
; 2 STOP BITS
18FE 2A E4    SUB       AH,AH        ; AH=0 IS COMMO INIT FUNCTION
1900 CD 14    INT       014H        ; DO INIT
1902 E8 1925 R CALL      FAKE         ; FAKE WILL MAP ERROR BITS FROM
; RS232 TO CORRESPONDING ONES
; FOR THE PRINTER
1905 58        POP          AX          ; RESTORE AL
1906 8A E6    MOV       AH,DH        ; IF DH IS RETURNED ZERO, MEANING
1908 0A E4    OR        AH,AH        ; NO ERRORS RETURN IT FOR THAT'S THE
; 'CORRECT' RETURN FROM AN ERROR
; FREE INIT
190A 74 E1    JE          B10_3
190C B4 A8    MOV       AH,D8BH
190E E8 DD    JMP       SHORT B10_3  ; THEN RETURN

```



```

-----
WRITE_TTY
THIS INTERFACE PROVIDES A TELETYPE-LIKE INTERFACE TO THE
VIDEO CARD. THE INPUT CHARACTER IS WRITTEN TO THE CURRENT
CURSOR POSITION, AND THE CURSOR IS MOVED TO THE NEXT POSITION.
IF THE CURSOR LEAVES THE LAST COLUMN OF THE FIELD, THE COLUMN
IS SET TO ZERO, AND THE ROW VALUE IS INCREMENTED. IF THE ROW
ROW VALUE LEAVES THE FIELD, THE CURSOR IS PLACED ON THE LAST
ROW, FIRST COLUMN, AND THE ENTIRE SCREEN IS SCROLLED UP ONE
LINE. WHEN THE SCREEN IS SCROLLED UP, THE ATTRIBUTE FOR FILLING
THE NEWLY BLANKED LINE IS READ FROM THE CURSOR POSITION ON THE
PREVIOUS LINE BEFORE THE SCROLL. IN CHARACTER MODE. IN
GRAPHICS MODE, THE 0 COLOR IS USED.
ENTRY --
(AH) = CURRENT CRT MODE
(AL) = CHARACTER TO BE WRITTEN
NOTE THAT BACK SPACE, CAR RET, BELL AND LINE FEED ARE
HANDLED AS COMMANDS RATHER THAN AS DISPLAYABLE GRAPHICS
(BL) = FOREGROUND COLOR FOR CHAR WRITE IF CURRENTLY IN A
GRAPHICS MODE
EXIT --
ALL REGISTERS SAVED
-----
ASSUME CS:CODE,DS:DATA
WRITE_TTY PROC NEAR
PUSH AX ; SAVE REGISTERS
PUSH AX ; SAVE CHAR TO WRITE
MOV BH,ACTIVE_PAGE ; GET CURRENT PAGE SETTING
PUSH BX ; SAVE IT
MOV BL,BH ; IN BL
XOR BH,BH
SAL BX,1 ; CONVERT TO WORD OFFSET
MOV DX,[BX+OFFSET CURSOR_POSNJ] ; GET CURSOR POSITION
POP BX ; RECOVER CURRENT PAGE
POP AX ; RECOVER CHAR
;----- DX NOW HAS THE CURRENT CURSOR POSITION
CMP AL,B ; IS IT A BACKSPACE?
JE UB ; BACK SPACE
CMP AL,0DH ; IS IT A CARRIAGE RETURN?
JE UR ; CAR_RET
CMP AL,0AH ; IS IT A LINE FEED
JE U10 ; LINE_FEED
CMP AL,07H ; IS IT A BELL
JE U11 ; BELL
;----- WRITE THE CHAR TO THE SCREEN
MOV AH,10 ; WRITE CHAR ONLY
MOV CX,1 ; ONLY ONE CHAR
INT 10H ; WRITE THE CHAR
;----- POSITION THE CURSOR FOR NEXT CHAR
INC DL
CMP DL,BYTE PTR CRT_COLS ; TEST FOR COLUMN OVERFLOW
JNZ U7 ; SET_CURSOR
XOR DL,DL ; COLUMN FOR CURSOR
;----- LINE FEED
U10: CMP DH,24
JNZ U6 ; SET_CURSOR_INC
;----- SCROLL REQUIRED
MOV AH,2
INT 10H ; SET THE CURSOR
;----- DETERMINE VALUE TO FILL WITH DURING SCROLL
MOV AL,CRT_MODE ; GET THE CURRENT MODE
CMP AL,4
JC U2 ; READ-CURSOR
XOR BH,BH ; FILL WITH BACKGROUND
JMP SHORT U3 ; SCROLL-UP
U2: MOV AH,B
INT 10H ; READ CHAR/ATTR AT CURRENT CURSOR
MOV BH,AH ; STORE IN BH
MOV AX,601H ; SCROLL ONE LINE
SUB CX,CX ; UPPER LEFT CORNER
MOV DH,24 ; LOWER RIGHT ROW
MOV DL,BYTE PTR CRT_COLS ; LOWER RIGHT COLUMN
DEC DL
U4: INT 10H ; SCROLL UP THE SCREEN
U5: POP AX ; RESTORE THE CHARACTER
JMP VIDEO_RETURN ; RETURN TO CALLER
U6: INC DH ; NEXT ROW
U7: MOV AH,2
JMP U4 ; ESTABLISH THE NEW CURSOR
;----- BACK SPACE FOUND
U8: OR DL,DL ; ALREADY AT END OF LINE
JE U7 ; SET_CURSOR
DEC DL ; NO -- JUST MOVE IT BACK
JMP U7 ; SET_CURSOR
;----- CARRIAGE RETURN FOUND
U9: XOR DL,DL ; MOVE TO FIRST COLUMN
JMP U7 ; SET_CURSOR
;----- BELL FOUND
U11: MOV BL,2 ; SET UP COUNT FOR BEEP
CALL BEEP ; SOUND THE POD BELL
JMP US ; TTY_RETURN
WRITE_TTY ENDP

```



```

;-----
; THIS PROCEDURE WILL ISSUE SHORT TONES TO INDICATE FAILURES
; THAT 1: OCCUR BEFORE THE CRT IS STARTED, 2: TO CALL THE
; OPERATORS ATTENTION TO AN ERROR AT THE END OF POST, OR
; 3: TO SIGNAL THE SUCCESSFUL COMPLETION OF POST
; ENTRY PARAMETERS:
; DL = NUMBER OF APPROX. 1/2 SEC TONES TO SOUND
;-----

```

```

IAOC
IAOC 9C
IAOD 53
IAOE FA
IAOF B3 01
IA11 E9 FF31 R
IA14 E2 FE
IA16 FE CA
IA18 75 F5
IA1A E2 FE
IA1C E2 FE
IA1E 58
IA1F 80
IA20 C3
IA21

```

```

ERR_BEEP PROC NEAR
; SAVE FLAGS
PUSHF
PUSH BX
CLI
; DISABLE SYSTEM INTERRUPTS
; SHORT_BEEP:
; COUNTER FOR A SHORT BEEP
MOV BL, 1
CALL BEEP
; DO THE SOUND
; DELAY BETWEEN BEEPS
; ONE WITH SHORTS
; LONG SOME MORE
; DO DELAY BEFORE RETURN
G3: MOV BL, 1
CALL BEEP
; DO THE SOUND
; DELAY BETWEEN BEEPS
; ONE WITH SHORTS
; LONG SOME MORE
; DO DELAY BEFORE RETURN
G4: LOOP G4
; RESTORE ORIG CONTENTS OF BX
; RESTORE FLAGS TO ORIG SETTINGS
; RETURN TO CALLER
G5: LOOP G5
G6: LOOP G6
POP BX
POPF
RET
ERR_BEEP ENOP
LIST

```

```

E000
E000 31 35 30 34 30 33
37 20 43 4F 50 52
2E 20 49 42 40 20
31 39 3B 31 2C 31
39 38 33

```

```

ASSUME CS:CODE,DS:DATA
ORG 0E000H
DB '1504037 COPR. IBM 1981, 1983' ; COPYRIGHT NOTICE

```

REAL_VECTOR_SETUP

```

;-----
; THIS ROUTINE WILL INITIALIZE THE INTERRUPT 9 VECTOR TO
; POINT AT THE REAL INTERRUPT ROUTINE.
;-----

```

```

E01B
E01B 50
E01C 53
E01D 06
E01E 33 C0
E020 BE C0
E022 BB 0024
E025 26: C7 07 1561 R

```

```

REAL_VECTOR_SETUP PROC NEAR
; SAVE THE SCAN CODE
PUSH AX
PUSH BX
PUSH ES
XOR AX,AX
; INITIALIZE TO POINT AT VECTOR
; VECTOR(0)
MOV ES,AX
MOV BX,BH*4H
MOV WORD PTR ES:[BX],OFFSET KB_INT ; MOVE IN OFFSET OF
; ROUTINE
; ADD 2 TO BX
INC BX
INC BX
; GET CODE SEGMENT OF BIOS (SEGMENT
; RELOCATEABLE)
PUSH CS
POP AX
MOV WORD PTR ES:[BX],AX ; MOVE IN SEGMENT OF ROUTINE
POP ES
POP BX
POP AX
RET
REAL_VECTOR_SETUP ENDP

```

```

E02A 43
E02B 43
E02C 0E
E02D 58
E02E 26: 89 07
E031 07
E032 58
E033 58
E034 C3
E035

```

KB_NOISE

```

;-----
; THIS ROUTINE IS CALLED WHEN GENERAL BEEPS ARE REQUIRED FROM
; THE SYSTEM.
;-----

```

```

; INPUT BX=LENGTH OF THE TONE
; CX=CONTAINS THE FREQUENCY
; OUTPUT ALL REGISTERS ARE MAINTAINED.
; HINTS AS CX GETS LARGER THE TONE PRODUCED GETS LOWER IN PITCH.
;-----

```

```

E035
E035 FB
E036 50
E037 53
E038 51
E039 E4 61
E03B 50
E03C 24 FC
E03E E6 61
E040 51
E041 E2 FE
E043 0C 02
E045 E6 81
E047 59
E048 51
E049 E2 FE
E04B 4B
E04C 59
E04D 75 ED
E04F 58
E050 E6 61
E052 59
E053 58
E054 58
E055 C3
E056
E05B
E058 E9 0043 R

```

```

KB_NOISE PROC NEAR
STI
PUSH AX
PUSH BX
PUSH CX
IN AL,061H
PUSH AX
; GET CONTROL INFO
; SAVE
LOOP01: AND AL,0FCH
; TURN OFF TIMER GATE AND SPEAKER
; DATA
; OUTPUT TO CONTROL
; HALF CYCLE TIME FOR TONE
OUT 061H,AL
PUSH CX
; SPEAKER OFF
; TURN ON SPEAKER BIT
; OUTPUT TO CONTROL
LOOP02: LOOP LOOP02
OR AL,2
OUT 061H,AL
POP CX
PUSH CX
; RETRIEVE FREQUENCY
; ANOTHER HALF CYCLE
; TOTAL TIME COUNT
; RETRIEVE FREQ.
; DO ANOTHER CYCLE
; RECOVER CONTROL
; OUTPUT THE CONTROL
LOOP03: LOOP LOOP03
DEC BX
POP CX
JNZ LOOP01
POP AX
OUT 061H,AL
POP CX
POP BX
POP AX
RET
KB_NOISE ENDP
ORG 0E05BH
NEAR PTR RESET

```

CHARACTER GENERATOR GRAPHICS FOR 320X200 AND 640X200
 GRAPHICS FOR CHARACTERS 80H THROUGH FFH

E05E	CRT_CHARH	LABEL	BYTE
E05E 78 CC 00 CC 78 18 0C 78	DB	07BH, 0CCH, 0C0H, 0CCH, 07BH, 01BH, 00CH, 07BH	; D_80
E066 00 CC 00 CC CC CC 7E 00	DB	000H, 0CCH, 000H, 0CCH, 0CCH, 0CCH, 07EH, 000H	; D_81
E06E 1C 00 78 CC FC C0 78 00	DB	01CH, 000H, 07BH, 0CCH, 0FCH, 0C0H, 07BH, 000H	; D_82
E076 7E C3 3C 06 3E 66 3F 00	DB	07EH, 0C3H, 03CH, 066H, 03EH, 066H, 03FH, 000H	; D_83
E07E CC 00 78 0C 7C CC 7E 00	DB	0CCH, 000H, 07BH, 00CH, 07CH, 0CCH, 07EH, 000H	; D_84
E086 E0 00 78 0C 7C CC 7E 00	DB	0E0H, 000H, 07BH, 00CH, 07CH, 0CCH, 07EH, 000H	; D_85
E08E 30 30 78 0C 7C CC 7E 00	DB	030H, 030H, 07BH, 00CH, 07CH, 0CCH, 07EH, 000H	; D_86
E096 00 00 78 C0 C0 78 0C 38	DB	000H, 000H, 07BH, 0C0H, 0C0H, 07BH, 00CH, 03BH	; D_87
E09E 7E C3 3C 66 7E 60 3C 00	DB	07EH, 0C3H, 03CH, 066H, 07EH, 060H, 03CH, 000H	; D_88
E0A6 CC 00 78 CC FC C0 78 00	DB	0CCH, 000H, 07BH, 0CCH, 0FCH, 0C0H, 07BH, 000H	; D_89
E0AE E0 00 78 CC FC C0 78 00	DB	0E0H, 000H, 07BH, 0CCH, 0FCH, 0C0H, 07BH, 000H	; D_8A
E0B6 CC 00 70 30 30 30 78 00	DB	0CCH, 000H, 070H, 030H, 030H, 030H, 07BH, 000H	; D_8B
E0BE 7C C6 38 18 18 18 3C 00	DB	07CH, 0C6H, 03BH, 01BH, 01BH, 01BH, 03CH, 000H	; D_8C
E0C6 E0 00 70 30 30 30 78 00	DB	0E0H, 000H, 070H, 030H, 030H, 030H, 07BH, 000H	; D_8D
E0CE C6 38 6C C6 FE C6 C6 00	DB	0C6H, 03BH, 06CH, 0C6H, 0FEH, 0C6H, 0C6H, 000H	; D_8E
E0D6 30 30 00 78 CC FC CC 00	DB	030H, 030H, 000H, 07BH, 0CCH, 0FCH, 0CCH, 000H	; D_8F
E0DE 1C 00 FC 60 78 60 FC 00	DB	01CH, 000H, 0FCH, 060H, 07BH, 060H, 0FCH, 000H	; D_90
E0E6 00 00 7F 0C 7F CC 7F 00	DB	000H, 000H, 07FH, 00CH, 07FH, 0CCH, 07FH, 000H	; D_91
E0EE 3E 8C CC FE CC CC CE 00	DB	03EH, 06CH, 0CCH, 0FEH, 0CCH, 0CCH, 0CEH, 000H	; D_92
E0F6 78 CC 00 78 CC CC 78 00	DB	07BH, 0CCH, 000H, 07BH, 0CCH, 0CCH, 07BH, 000H	; D_93
E0FE 00 CC 00 78 CC CC 78 00	DB	000H, 0CCH, 000H, 07BH, 0CCH, 0CCH, 07BH, 000H	; D_94
E106 00 E0 00 78 CC CC 78 00	DB	000H, 0E0H, 000H, 07BH, 0CCH, 0CCH, 07BH, 000H	; D_95
E10E 78 CC 00 CC CC CC 7E 00	DB	07BH, 0CCH, 000H, 0CCH, 0CCH, 0CCH, 07EH, 000H	; D_96
E116 00 E0 00 CC CC CC 7E 00	DB	000H, 0E0H, 000H, 0CCH, 0CCH, 0CCH, 07EH, 000H	; D_97
E11E 00 CC 00 CC CC 7C 0C F8	DB	000H, 0CCH, 000H, 0CCH, 0CCH, 07CH, 00CH, 0FBH	; D_98
E126 C3 18 3C 66 66 3C 18 00	DB	0C3H, 01BH, 03CH, 066H, 066H, 03CH, 01BH, 000H	; D_99
E12E CC 00 CC CC CC CC 78 00	DB	0CCH, 000H, 0CCH, 0CCH, 0CCH, 0CCH, 07BH, 000H	; D_9A
E136 18 18 7E 0C C0 7E 18 18	DB	01BH, 01BH, 07EH, 0C0H, 0C0H, 07EH, 01BH, 01BH	; D_9B
E13E 38 6C 64 F0 60 6E FC 00	DB	03BH, 06CH, 064H, 0F0H, 060H, 0E6H, 0FCH, 000H	; D_9C
E146 CC CC 76 FC 30 FC 30 30	DB	0CCH, 0CCH, 07BH, 0FCH, 030H, 0FCH, 030H, 030H	; D_9D
E14E FB CC CC FA C6 CF C6 C7	DB	0FBH, 0CCH, 0CCH, 0FAH, 0C6H, 0CFH, 0C6H, 0C7H	; D_9E
E156 0E 18 18 3C 18 18 DB 70	DB	00EH, 01BH, 01BH, 03CH, 01BH, 01BH, 0DBH, 070H	; D_9F
E15E 1C 00 78 0C 7C CC 7E 00	DB	01CH, 000H, 07BH, 00CH, 07CH, 0CCH, 07EH, 000H	; D_A0
E166 38 00 70 30 30 30 78 00	DB	03BH, 000H, 070H, 030H, 030H, 030H, 07BH, 000H	; D_A1
E16E 00 1C 00 78 CC CC 78 00	DB	000H, 01CH, 000H, 07BH, 0CCH, 0CCH, 07BH, 000H	; D_A2
E176 00 1C 00 CC CC CC 7E 00	DB	000H, 01CH, 000H, 0CCH, 0CCH, 0CCH, 07EH, 000H	; D_A3
E17E 00 F8 00 F8 CC CC CC 00	DB	000H, 0FBH, 000H, 0FBH, 0CCH, 0CCH, 0CCH, 000H	; D_A4
E186 FC 00 CC EC FC DC CC 00	DB	0FCH, 000H, 0CCH, 0ECH, 0FCH, 0DCH, 0CCH, 000H	; D_A5
E18E 3C 6C 6C 3E 00 7E 00 00	DB	03CH, 06CH, 06CH, 03EH, 000H, 07EH, 000H, 000H	; D_A6
E196 38 6C 6C 38 00 7C 00 00	DB	03BH, 06CH, 06CH, 03BH, 000H, 07CH, 000H, 000H	; D_A7
E19E 30 00 30 C0 CC 78 00	DB	030H, 000H, 030H, 060H, 0C0H, 0CCH, 07BH, 000H	; D_A8
E1A6 00 00 00 FC C0 C0 00 00	DB	000H, 000H, 000H, 0FCH, 0C0H, 0C0H, 000H, 000H	; D_A9
E1AE 00 00 00 FC 0C 0C 00 00	DB	000H, 000H, 000H, 0FCH, 00CH, 00CH, 000H, 000H	; D_AA
E1B6 C3 C6 CC DE 33 66 CC 0F	DB	0C3H, 0C6H, 0CCH, 0DEH, 033H, 066H, 0CCH, 00FH	; D_AB
E1BE C3 C6 CC DB 37 6F CF 03	DB	0C3H, 0C6H, 0CCH, 0DBH, 037H, 06FH, 0CFH, 003H	; D_AC
E1C6 18 18 00 18 18 18 18 00	DB	01BH, 01BH, 000H, 01BH, 01BH, 01BH, 01BH, 000H	; D_AD
E1CE 00 33 66 CC 68 33 00 00	DB	000H, 033H, 066H, 0CCH, 066H, 033H, 000H, 000H	; D_AE
E1D6 00 CC 66 33 66 CC 00 00	DB	000H, 0CCH, 066H, 033H, 066H, 0CCH, 000H, 000H	; D_AF

E1DE	22 88 22 88 22 88 22 88	DB	022H, 088H, 022H, 088H, 022H, 088H, 022H, 088H ;	D_B0
E1E6	55 AA 55 AA 55 AA 55 AA	DB	055H, 0AAH, 055H, 0AAH, 055H, 0AAH, 055H, 0AAH ;	D_B1
E1EE	D8 77 D8 EE D8 77 D8 EE	DB	0D8H, 077H, 0D8H, 0EEH, 0D8H, 077H, 0D8H, 0EEH ;	D_B2
E1F6	18 18 18 18 18 18 18 18	DB	018H, 018H, 018H, 018H, 018H, 018H, 018H, 018H ;	D_B3
E1FE	18 18 18 18 F8 18 18 18	DB	018H, 018H, 018H, 018H, 0F8H, 018H, 018H, 018H ;	D_B4
E206	18 18 F8 18 F8 18 18 18	DB	018H, 018H, 0F8H, 018H, 0F8H, 018H, 018H, 018H ;	D_B5
E20E	36 36 36 36 F6 36 36 36	DB	036H, 036H, 036H, 036H, 0F6H, 036H, 036H, 036H ;	D_B6
E216	00 00 00 00 FE 36 36 36	DB	000H, 000H, 000H, 000H, 0FEH, 036H, 036H, 036H ;	D_B7
E21E	00 00 F8 18 F8 18 18 18	DB	000H, 000H, 0F8H, 018H, 0F8H, 018H, 018H, 018H ;	D_B8
E226	36 36 F6 06 F6 36 36 36	DB	036H, 036H, 0F6H, 006H, 0F6H, 036H, 036H, 036H ;	D_B9
E22E	36 36 36 36 F6 36 36 36	D6	036H, 036H, 036H, 036H, 036H, 036H, 036H, 036H ;	D_BA
E236	00 00 FE 06 F6 36 36 36	DB	000H, 000H, 0FEH, 006H, 0F6H, 036H, 036H, 036H ;	D_BB
E23E	36 36 F6 06 FE 00 00 00	DB	036H, 036H, 0F6H, 006H, 0FEH, 000H, 000H, 000H ;	D_BC
E246	36 36 36 36 FE 00 00 00	DB	036H, 036H, 036H, 036H, 0FEH, 000H, 000H, 000H ;	D_BD
E24E	18 18 F8 18 F8 00 00 00	DB	018H, 018H, 0F8H, 018H, 0F8H, 000H, 000H, 000H ;	D_BE
E256	00 00 00 00 F8 18 18 18	DB	000H, 000H, 000H, 000H, 0F8H, 018H, 018H, 018H ;	D_BF
E25E	18 18 18 18 1F 00 00 00	DB	018H, 018H, 018H, 018H, 01FH, 000H, 000H, 000H ;	D_C0
E266	18 18 18 18 FF 00 .00 00	DB	018H, 018H, 018H, 018H, 0FFH, 000H, 000H, 000H ;	D_C1
E26E	00 00 00 00 FF 18 18 18	DB	000H, 000H, 000H, 000H, 0FFH, 018H, 018H, 018H ;	D_C2
E276	18 18 18 18 1F 18 18 18	DB	018H, 018H, 018H, 018H, 01FH, 018H, 018H, 018H ;	D_C3
E27E	00 00 00 00 FF 00 00 00	DB	000H, 000H, 000H, 000H, 0FFH, 000H, 000H, 000H ;	D_C4
E286	18 18 18 18 FF 18 18 18	DB	018H, 018H, 018H, 018H, 0FFH, 018H, 018H, 018H ;	D_C5
E28E	18 18 1F 18 1F 18 18 18	DB	018H, 018H, 01FH, 018H, 01FH, 018H, 018H, 018H ;	D_C6
E296	36 36 36 36 37 36 36 36	DB	036H, 036H, 036H, 036H, 037H, 036H, 036H, 036H ;	D_C7
E29E	36 36 37 30 3F 00 00 00	DB	036H, 036H, 037H, 030H, 03FH, 000H, 000H, 000H ;	D_C8
E2A6	00 00 3F 30 37 36 36 36	DB	000H, 000H, 03FH, 030H, 03FH, 036H, 036H, 036H ;	D_C9
E2AE	36 36 F7 00 FF 00 00 00	DB	036H, 036H, 0F7H, 000H, 0FFH, 000H, 000H, 000H ;	D_CA
E2B6	00 00 FF 00 F7 36 36 36	DB	000H, 000H, 0FFH, 000H, 0F7H, 036H, 036H, 036H ;	D_CB
E2BE	36 36 37 30 37 36 36 36	DB	036H, 036H, 037H, 030H, 037H, 036H, 036H, 036H ;	D_CC
E2C6	00 00 FF 00 FF 00 00 00	DB	000H, 000H, 0FFH, 000H, 0FFH, 000H, 000H, 000H ;	D_CD
E2CE	36 36 F7 00 F7 36 36 36	DB	036H, 036H, 0F7H, 000H, 0F7H, 036H, 036H, 036H ;	D_CE
E2D6	18 18 FF 00 FF 00 00 00	DB	018H, 018H, 0FFH, 000H, 0FFH, 000H, 000H, 000H ;	D_CF
E2DE	36 36 36 36 FF 00 00 00	DB	036H, 036H, 036H, 036H, 0FFH, 000H, 000H, 000H ;	D_D0
E2E6	00 00 FF 00 FF 18 18 18	DB	000H, 000H, 0FFH, 000H, 0FFH, 018H, 018H, 018H ;	D_D1
E2EE	00 00 00 00 FF 36 36 36	DB	000H, 000H, 000H, 000H, 0FFH, 036H, 036H, 036H ;	D_D2
E2F6	36 36 36 36 3F 00 00 00	DB	036H, 036H, 036H, 036H, 03FH, 000H, 000H, 000H ;	D_D3
E2FE	18 18 1F 18 1F 00 00 00	DB	018H, 018H, 01FH, 018H, 01FH, 000H, 000H, 000H ;	D_D4
E306	00 00 1F 18 1F 18 18 18	DB	000H, 000H, 01FH, 018H, 01FH, 018H, 018H, 018H ;	D_D5
E30E	00 00 00 00 3F 36 36 36	DB	000H, 000H, 000H, 000H, 03FH, 036H, 036H, 036H ;	D_D6
E316	36 36 36 36 FF 36 36 36	DB	036H, 036H, 036H, 036H, 0FFH, 036H, 036H, 036H ;	D_D7
E31E	18 18 FF 18 FF 18 18 18	DB	018H, 018H, 0FFH, 018H, 0FFH, 018H, 018H, 018H ;	D_D8
E326	18 18 18 18 F8 00 00 00	DB	018H, 018H, 018H, 018H, 0F8H, 000H, 000H, 000H ;	D_D9
E32E	00 00 00 00 1F 18 18 18	DB	000H, 000H, 000H, 000H, 01FH, 018H, 018H, 018H ;	D_DA
E336	FF FF FF FF FF FF FF FF	DB	0FFH, 0FFH, 0FFH, 0FFH, 0FFH, 0FFH, 0FFH, 0FFH ;	D_DB
E33E	00 00 00 00 FF FF FF FF	DB	000H, 000H, 000H, 000H, 0FFH, 0FFH, 0FFH, 0FFH ;	D_DC
E346	F0 F0 F0 F0 F0 F0 F0 F0	DB	0F0H, 0F0H, 0F0H, 0F0H, 0F0H, 0F0H, 0F0H, 0F0H ;	D_DD
E34E	0F 0F 0F 0F 0F 0F 0F 0F	DB	00FH, 00FH, 00FH, 00FH, 00FH, 00FH, 00FH, 00FH ;	D_DE
E356	FF FF FF FF 00 00 00 00	DB	0FFH, 0FFH, 0FFH, 0FFH, 000H, 000H, 000H, 000H ;	D_DF

```

E35E 00 00 76 DC CB DC DB 000H,000H,076H,0DCH,0CBH,0DCH,076H,000H ; D_E0
      76 00
E368 00 78 CC F8 CC F8 DB 000H,078H,0CCH,0F8H,0CCH,0FBH,0C0H,0C0H ; D_E1
      CO CO
E36E 00 FC CC CO CO CO DB 000H,0FCH,0CCH,0C0H,0C0H,0C0H,0C0H,000H ; D_E2
      CO CO
E376 00 FE 6C 6C 8C 6C DB 000H,0FEH,06CH,06CH,06CH,06CH,06CH,000H ; D_E3
      8C 00
E37E FC CC 60 30 60 CC DB 0FCH,0CCH,060H,030H,060H,0CCH,0FCH,000H ; D_E4
      FC 00
E388 00 00 7E D8 D8 D8 DB 000H,000H,07EH,0D6H,0DBH,0DBH,070H,000H ; D_E5
      70 00
E38E 00 66 66 66 66 7C DB 000H,066H,066H,066H,066H,07CH,060H,0C0H ; D_E6
      60 CO
E396 00 76 DC 18 18 18 DB 000H,076H,0DCH,018H,018H,018H,018H,000H ; D_E7
      18 00
E39E FC 30 78 CC CC 78 DB 0FCH,030H,078H,0CCH,0CCH,078H,030H,0FCH ; D_E8
      30 FC
E3A6 38 6C C6 FE C6 6C DB 038H,06CH,0C6H,0FEH,0C6H,06CH,038H,000H ; D_E9
      38 00
E3AE 38 6C C6 C6 6C 6C DB 038H,06CH,0C6H,0C6H,06CH,06CH,0EEH,000H ; D_EA
      EE 00
E3B6 1C 30 18 7C CC CC DB 01CH,030H,016H,07CH,0CCH,0CCH,076H,000H ; D_EB
      78 00
E3BE 00 00 7E D8 D8 7E DB 000H,000H,07EH,0DBH,0DBH,07EH,000H,000H ; D_EC
      00 00
E3C6 06 0C 7E D8 D8 7E DB 006H,00CH,07EH,0DBH,0DBH,07EH,060H,0C0H ; D_ED
      60 CO
E3CE 38 60 C0 F8 C0 60 DB 038H,060H,0C0H,0FBH,0C0H,060H,038H,000H ; D_EE
      38 00
E3D6 78 CC CC CC CC CC DB 078H,0CCH,0CCH,0CCH,0CCH,0CCH,0CCH,000H ; D_EF
      CC 00

E3DE 00 FC 00 FC 00 FC DB 000H,0FCH,000H,0FCH,000H,0FCH,000H,000H ; D_F0
      00 00
E3E6 30 30 FC 30 30 00 DB 030H,030H,0FCH,030H,030H,000H,0FCH,000H ; D_F1
      FC 00
E3EE 60 30 18 30 60 00 DB 060H,030H,018H,030H,060H,000H,0FCH,000H ; D_F2
      FC 00
E3F6 18 30 60 30 18 00 DB 018H,030H,060H,030H,018H,000H,0FCH,000H ; D_F3
      FC 00
E3FE 0E 18 18 18 18 18 DB 00EH,018H,018H,018H,018H,018H,018H,018H ; D_F4
      18 18
E406 18 18 18 18 18 D6 DB 018H,018H,018H,018H,018H,0DBH,0DBH,070H ; D_F5
      D8 70
E40E 30 30 00 FC 00 30 DB 030H,030H,000H,0FCH,000H,030H,030H,000H ; D_F6
      30 00
E416 00 76 0C 00 76 DC DB 000H,076H,0DCH,000H,076H,0DCH,000H,000H ; D_F7
      00 00
E41E 38 6C 6C 38 00 00 DB 038H,06CH,06CH,038H,000H,000H,000H,000H ; D_F8
      00 00
E426 00 00 00 18 18 00 DB 000H,000H,000H,018H,018H,000H,000H,000H ; D_F9
      00 00
E42E 00 00 00 00 18 00 DB 000H,000H,000H,000H,018H,000H,000H,000H ; D_FA
      00 00
E436 0F 0C 0C EC 6C DB 00FH,00CH,00CH,00CH,0ECH,06CH,03CH,01CH ; D_FB
      3C 1C
E43E 78 6C 6C 6C 6C 00 DB 078H,06CH,06CH,06CH,06CH,000H,000H,000H ; D_FC
      00 00
E446 70 18 30 60 78 00 DB 070H,018H,030H,060H,078H,000H,000H,000H ; D_FD
      00 00
E44E 00 00 3C 3C 3C 3C DB 000H,000H,03CH,03CH,03CH,03CH,000H,000H ; D_FE
      00 00
E456 00 00 00 00 00 00 DB 000H,000H,000H,000H,000H,000H,000H,000H ; D_FF
      00 00

```

ASSUME CS: CODE, DS: DATA

```

-----
; SET_CTYPE
; THIS ROUTINE SETS THE CURSOR VALUE
; INPUT (CX) HAS CURSOR VALUE CH-START LINE, CL-STOP LINE
; OUTPUT
; NONE
-----
SET_CTYPE PROC NEAR
E45E CMP AH,4 ; IN GRAPHICS MODE?
E45E JC C23X ; NO, JUMP
E463 OR CH,20H ; YES, DISABLE CURSOR
E466 C23X: MOV AH,10 ; 6845 REGISTER FOR CURSOR SET
E468 MOV CURSOR_MODE,CX ; SAVE IN DATA AREA
E46C CALL C23 ; OUTPUT CX REG
E46F JMP VIDEO_RETURN
; THIS ROUTINE OUTPUTS THE CX REGISTER TO THE 6845 REGS NAMED IN AH
C23: MOV DX,ADDR_6845 ; ADDRESS REGISTER
MOV AL,AH ; GET VALUE
OUT DX,AL ; REGISTER SET
INC DX ; DATA REGISTER
MOV AL,CH ; DATA
OUT DX,AL
DEC DX
MOV AL,AH
INC AL ; POINT TO OTHER DATA REGISTER
OUT DX,AL ; SET FOR SECOND REGISTER
INC DX
MOV AL,CL ; SECOND DATA VALUE
OUT DX,AL
RET ; ALL DONE
SET_CTYPE ENDP

```

```

E488
E489 8A CF
E48A 32 ED
E48C 01 E1
E48E 88 F1
E490 89 94 0050 R
E494 38 3E 0052 R
E498 75 05
E49A 9B C2
E49C E8 E4A2 R
E49F E9 0F70 R
E4A2

```

```

-----
SET_CPOS
THIS ROUTINE SETS THE CURRENT CURSOR POSITION TO THE
NEW X-Y VALUES PASSED
INPUT
DX - ROW, COLUMN OF NEW CURSOR
BH - DISPLAY PAGE OF CURSOR
OUTPUT
CURSOR IS SET AT 6B45 IF DISPLAY PAGE IS CURRENT DISPLAY
-----

```

```

SET_CPOS      PROC    NEAR
MOV          CL, BH
XOR         CH, CH
SAL        CX, 1
MOV         SI, CX
MOV        [SI+OFFSET_CURSOR_POSN], DX
CMP        ACTIVE_PAGE, BH
JNZ        C24
MOV        AX, DX
CALL       C25
JMP        VIDEO_RETURN
C24:         JMP        VIDEO_RETURN
SET_CPOS      ENDP
-----
C25:         SET CURSOR POSITION, AX HAS ROW/COLUMN FOR CURSOR
PROC        NEAR
CALL       POSITION
MOV        CX, AX
ADD        CX, CRT_START
SAR        CX, 1
MOV        AH, 14
CALL       C23
RET
C25         ENDP

```

```

E4A2
E4A2 E8 E5C2 R
E4A5 8B CB
E4A7 03 0E 004E R
E4AB D1 F9
E4AD B4 0E
E4AF EB E472 R
E4B2 C3
E4B3
E4B3
E4B3 A9 80
E4B5 75 24
E4B7 A2 0082 R
E4BA 8B 0E 004C R
E4BE 3B
E4BF 50
E4C0 F7 E1
E4C2 A3 004E R
E4C5 8B CB
E4C7 01 F9
E4C9 B4 0C
E4CB EB E472 R
E4CE 58
E4CF D1 E3
E4D1 8B 87 0050 R
E4D5 EB E4A2 R
E4D8 E9 0F70 R

```

```

-----
ACT_DISP_PAGE
THIS ROUTINE SETS THE ACTIVE DISPLAY PAGE, ALLOWING
THE FULL USE OF THE RAM SET ASIOE FOR THE VIDEO ATTACHMENT
INPUT
AL HAS THE NEW ACTIVE DISPLAY PAGE
OUTPUT
THE 8B45 IS RESET TO DISPLAY THAT PAGE
-----

```

```

ACT_DISP_PAGE  PROC    NEAR
TEST         AL, 0B0H
JNZ         SET_CRTCPU
MOV         ACTIVE_PAGE, AL
MOV         CX, CRT_LEN
CBW
PUSH        AX
MUL        CX
MOV        CRT_START, AX
MOV        CX, AX
SAR        CX, 1
MOV        AH, 12
CALL       C23
POP        BX
SAL        BX, 1
MOV        AX, [BX + OFFSET_CURSOR_POSN]
CALL       C25
JMP        VIDEO_RETURN

```

```

-----
SET_CRTCPU
THIS ROUTINE READS OR WRITES THE CRT/CPU PAGE REGISTERS
INPUT
(AL) = 83H SET BOTH CRT AND CPU PAGE REGS
(BH) = VALUE TO SET IN CRT PAGE REG
(BL) = VALUE TO SET IN CPU PAGE REG
(AL) = 82H SET CRT PAGE REG
(BH) = VALUE TO SET IN CRT PAGE REG
(AL) = 81H SET CPU PAGE REG
(BL) = VALUE TO SET IN CPU PAGE REG
(AL) = 80H READ CURRENT VALUE OF CRT/CPU PAGE REGS
OUTPUT
ALL FUNCTIONS RETURN
(BH) = CURRENT CONTENTS OF CRT PAGE REG
(BL) = CURRENT CONTENTS OF CPU PAGE REG
-----

```

```

E4DB 8A E0
E4DD 8A 03DA
E4E0 EC
E4E1 24 08
E4E3 74 FB
E4E5 8A 03DF
E4EB A0 008A R
E4ED 80 FC 80
E4EE 74 27
E4F0 80 FC 84
E4F3 73 22
E4F5 F8 C4 01
E4F8 74 0D
E4FA D0 E3
E4FC D0 E3
E4FE D0 E3
E500 24 C7
E502 80 E3 38
E505 0A C3

```

```

SET_CRTCPU:
MOV        AH, AL
MOV        DX, VGA_CTL
C26:      IN         AL, DX
AND        AL, 08H
JZ         C26
MOV        DX, PAGREG
MOV        AL, PAGDAT
CMP        AH, 80H
JZ         C29
CMP        AH, 84H
JNC        C29
TEST       AH, 1
JZ         C27
SHL        BL, 1
SHL        BL, 1
AND        AL, NOT CPUREG
AND        BL, CPUREG
OR         AL, BL

```

```

E507 F6 C4 02
E50A 74 07
E50C 24 FB
E50E 80 E7 07
E511 0A C7
E513 EE
E514 A2 008A R
E517 8A 0B
E519 80 E3 3B
E51C 00 FB
E51E 00 FB
E520 00 FB
E522 8A FB
E524 80 E7 07
E527 5F
E528 5E
E529 5B
E52A E9 0F73 R
E52D

```

```

C27: TEST AH,2 ; SET CRT REG7
      JZ C28 ; NO, GO RETURN CURRENT SETTINGS
      AND AL,NOT CRTREG ; CLEAR OLD CRT VALUE
      AND BH,CRTREG ; BE SURE UNRELATED BITS ARE ZERO
      OR AL,BH ; OR IN NEW VALUE
C28: OUT DX,AL ; SET NEW VALUES
      MOV PAGE0AT,AL ; SAVE COPY IN RAM
C29: MOV BL,AL ; GET CPU REG VALUE
      AND BL,CPUREG ; CLEAR EXTRA BITS
      SAR BL,1 ; RIGHT JUSTIFY IN BL
      SAR BL,1
      SAR BL,1
      MOV BH,AL ; GET CRT REG VALUE
      AND BH,CRTREG ; CLEAR EXTRA BITS
      POP DI ; RESTORE SOME REGS
      POP SI
      POP AX ; DISCARD SAVED BX
      JMP C22 ; RETURN

```

ACT_DISP_PAGE ENDP

```

-----
; READ_CURSOR
; THIS ROUTINE READS THE CURRENT CURSOR VALUE FROM THE
; 6845, FORMATS IT, AND SENDS IT BACK TO THE CALLER
; INPUT
; BH - PAGE OF CURSOR
; OUTPUT
; DX - ROW, COLUMN OF THE CURRENT CURSOR POSITION
; CX - CURRENT CURSOR MODE
-----

```

```

E52D
E52D 8A DF
E52F 32 FF
E531 D1 E3
E533 8B 97 0050 R
E537 9B 0E 0060 R
E53B 5F
E53C 5E
E53D 5B
E53E 5B
E53F 5B
E540 1F
E541 07
E542 CF
E543

```

```

READ_CURSOR PROC NEAR
      MOV BL,BH
      XOR BH,BH
      SAL BX,1 ; WORD OFFSET
      MOV DX,[BX+OFFSET CURSOR_POSN]
      MOV CX,CURSOR_MODE
      POP DI
      POP SI
      POP BX ; DISCARD SAVED CX AND DX
      POP AX
      POP AX
      POP DS
      POP ES
      IRET
READ_CURSOR ENDP

```

```

-----
; SET_COLOR
; THIS ROUTINE WILL ESTABLISH THE BACKGROUND COLOR, THE
; OVERSCAN COLOR, AND THE FOREGROUND COLOR SET FOR GRAPHICS
; INPUT
; (BH) HAS COLOR ID
; IF BH=0, THE BACKGROUND COLOR VALUE IS SET
; FROM THE LOW BITS OF BL (0-31)
; IN GRAPHIC MODES, BOTH THE BACKGROUND AND
; BORDER ARE SET. IN ALPHA MODES, ONLY THE
; BORDER IS SET.
; IF BH=1, THE PALETTE SELECTION IS MADE
; BASED ON THE LOW BIT OF BL:
; 2 COLOR MODE:
; 0 = WHITE FOR COLOR 1
; 1 = BLACK FOR COLOR 1
; 4 COLOR MODES:
; 0 = GREEN, RED, YELLOW FOR
; COLORS 1,2,3
; 1 = BLUE, CYAN, MAGENTA FOR
; COLORS 1,2,3
; 16 COLOR MODES:
; ALWAYS SETS UP PALETTE AS:
; BLUE FOR COLOR 1
; GREEN FOR COLOR 2
; CYAN FOR COLOR 3
; RED FOR COLOR 4
; MAGENTA FOR COLOR 5
; BROWN FOR COLOR 6
; LIGHT GRAY FOR COLOR 7
; DARK GRAY FOR COLOR 8
; LIGHT BLUE FOR COLOR 9
; LIGHT GREEN FOR COLOR 10
; LIGHT CYAN FOR COLOR 11
; LIGHT RED FOR COLOR 12
; LIGHT MAGENTA FOR COLOR 13
; YELLOW FOR COLOR 14
; WHITE FOR COLOR 15
; (BL) HAS THE COLOR VALUE TO BE USED
; OUTPUT
; THE COLOR SELECTION IS UPDATED
-----

```

```

E543
E543 8A 03DA
E546 EC
E547 AB 0B
E549 74 FB
E54B 0A FF
E54D 75 19

```

```

SET_COLOR PROC NEAR
      MOV DX,VGA_CTL ; I/O PORT FOR PALETTE
C30: IN AL,DX ; SYNC UP VGA FOR REG ADDRESS
      TEST AL,B ; IS VERTICAL RETRACE ON?
      JZ C30 ; NO, WAIT UNTIL IT IS
      OR BH,BH ; IS THIS COLOR 0?
      JNZ C31 ; OUTPUT COLOR 1

```

```

;----- HANDLE COLOR 0 BY SETTING THE BACKGROUND COLOR
; AND BORDER COLOR
E54F 80 3E 0049 R 04    CMP     CRT_MODE, 4    ; IN ALPHA MODE?
E554 72 06              JC      C305          ; YES, JUST SET BORDER REG
E556 80 10              MOV     AL, 10H       ; SET PALETTE REG 0
E558 EE                OUT     DX, AL        ; SELECT VGA REG
E559 8A C3              MOV     AL, BL        ; GET COLOR
E55B EE                OUT     DX, AL        ; SET IT
E55C 80 02              MOV     AL, 2         ; SET BORDER REG
E55E EE                OUT     DX, AL        ; SELECT VGA BORDER REG
E55F 8A C3              MOV     AL, BL        ; GET COLOR
E561 EE                OUT     DX, AL        ; SET IT
E562 A2 0068 R          MOV     CRT_PALETTE, AL ; SAVE THE COLOR VALUE
E565 E9 0F70 R          JMP     VIDEO_RETURN  ;

;----- HANDLE COLOR 1 BY CHANGING PALETTE REGISTERS
E568 A0 0049 R          MOV     AL, CRT_MODE  ; GET CURRENT MODE
E56B 89 0D98 R          MOV     CX, OFFSET M0072 ; POINT TO 2 COLOR TABLE ENTRY
E56E 3C 06              CMP     AL, 6         ; 2 COLOR MODE?
E570 74 0F              JE      C33           ; YES, JUMP
E572 3C 04              CMP     AL, 4         ; 4 COLOR MODE?
E574 74 08              JE      C32           ; YES, JUMP
E576 3C 06              CMP     AL, 5         ; 4 COLOR MODE?
E578 74 04              JE      C32           ; YES, JUMP
E57A 3C 0A              CMP     AL, 0AH      ; 4 COLOR MODE?
E57C 75 20              JNE     C36           ; NO, GO TO 16 COLOR SET UP
E57E 89 0D9D R          MOV     CX, OFFSET M0074 ; POINT TO 4 COLOR TABLE ENTRY
E581 00 CB              ROR     BL, 1         ; SELECT ALTERNATE SET?
E583 73 03              JNC     C34           ; NO, JUMP
E585 83 C1 04           ADD     CX, M0072L    ; POINT TO NEXT ENTRY
E588 8B D9              MOV     BX, CX        ; TABLE ADDRESS IN BX
E58A 43              INC     BX            ; SKIP OVER BACKGROUND COLOR
E58B 89 0003            MOV     CX, M0072L-1 ; SET NUMBER OF REGS TO FILL
E58E 84 11              MOV     AH, 11H      ; AH IS REGISTER COUNTER
E590 8A C4              MOV     AL, AH        ; GET REG NUMBER
E592 EE                OUT     DX, AL        ; SELECT IT
E593 2E: 8A 07          MOV     AL, CS:[BX]  ; GET DATA
E596 EE                OUT     DX, AL        ; SET IT
E597 FE C4              INC     AH            ; NEXT REG
E599 43              INC     BX            ; NEXT TABLE VALUE
E59A E2 F4              LOOP   JMP           ; SHORT C38
E59C EB 0D              MOV     AH, 11H      ; AH IS REGISTER COUNTER
E59E 84 11              MOV     CX, 15       ; NUMBER OF PALETTES
E5A0 89 000F            MOV     AL, AH        ; GET REG NUMBER
E5A3 8A C4              MOV     DX, AL        ; SELECT IT
E5A5 EE                OUT     DX, AL        ; SET PALETTE VALUE
E5A7 FE C4              INC     AH            ; NEXT REG
E5A8 E2 F0              LOOP   C37           ;
E5AB 32 C0              XOR     AL, AL        ; SELECT LOW REG TO ENABLE VIDEO
; AGAIN
E5AD EE                OUT     DX, AL
E5AE E9 0F70 R          JMP     VIDEO_RETURN
E5B1                    SET_COLOR            ENDP

```

```

; VIDEO STATE
; RETURNS THE CURRENT VIDEO STATE IN AX
; AH = NUMBER OF COLUMNS ON THE SCREEN
; AL = CURRENT VIDEO MODE
; BH = CURRENT ACTIVE PAGE

```

```

E5B1                    VIDEO_STATE            PROC    NEAR
E5B1 8A 26 004A R      MOV     AH, BYTE PTR CRT_COLS ; GET NUMBER OF COLUMNS
E5B5 A0 0048 R          MOV     AL, CRT_MODE      ; CURRENT MODE
E5B8 8A 3E 0062 R      MOV     BH, ACTIVE_PAGE   ; GET CURRENT ACTIVE PAGE
E5BC 5F                POP     DI                ; RECOVER REGISTERS
E5BD 5E                POP     SI
E5BE 59                POP     CX                ; DISCARD SAVED BX
E5BF E9 0F73 R          JMP     C22               ; RETURN TO CALLER
E5C2                    VIDEO_STATE            ENDP

```

```

; POSITION
; THIS SERVICE ROUTINE CALCULATES THE REGEN BUFFER ADDRESS
; OF A CHARACTER IN THE ALPHA MODE

```

```

; INPUT
; AX = ROW, COLUMN POSITION
; OUTPUT
; AX = OFFSET OF CHAR POSITION IN REGEN BUFFER

```

```

E5C2                    POSITION            PROC    NEAR
E5C2 53                PUSH    BX               ; SAVE REGISTER
E5C3 8B DB            MOV     BX, AX
E5C5 8A C4            MOV     AL, AH           ; ROWS TO AL
E5C7 F6 26 004A R    MUL     BYTE PTR CRT_COLS ; DETERMINE BYTES TO ROW
E5C8 32 FF            XOR     BH, BH
E5CD 03 C3            ADD     AX, BX           ; ADD IN COLUMN VALUE
E5CF D1 E0            SAL     AX, 1           ; * 2 FOR ATTRIBUTE BYTES
E5D1 5B                POP     BX
E5D2 C9                RET
E5D3                    POSITION            ENDP

```

```

; SCROLL UP
; THIS ROUTINE MOVES A BLOCK OF CHARACTERS UP
; ON THE SCREEN

```

```

; INPUT
; (AH) = CURRENT CRT MODE
; (AL) = NUMBER OF ROWS TO SCROLL
; (CX) = ROW/COLUMN OF UPPER LEFT CORNER
; (DX) = ROW/COLUMN OF LOWER RIGHT CORNER
; (BH) = ATTRIBUTE TO BE USED ON BLANKED LINE
; (DS) = DATA SEGMENT
; (ES) = REGEN BUFFER SEGMENT
; OUTPUT
; NONE -- THE REGEN BUFFER IS MODIFIED

```

```

ASSUME CS:COOE,DS:DATA,ES:DATA
E5D3 SCROLL_UP PROC NEAR
E5D3      MOV     BL,AL          ; SAVE LINE COUNT IN BL
E5D5      CMP     AH,4          ; TEST FOR GRAPHICS MODE
E5D8      JC      C39          ; HANDLE SEPARATELY
E5DA      JMP     GRAPHICS_UP
E5DD
C39:      ; UP_CONTINUE
E5DD      PUSH  BX             ; SAVE FILL ATTRIBUTE IN BH
E5DE      MOV     AX,CX        ; UPPER LEFT POSITION
E5E0      CALL  SCROLL_POSITION ; DO SETUP FOR SCROLL
E5E3      JZ      C44          ; BLANK_FIELD
E5E5      ADD     SI,AX        ; FROM ADDRESS
E5E7      MOV     AH,DH        ; # ROWS IN BLOCK
E5E9      SUB     AH,BL        ; # ROWS TO BE MOVED
E5EB      CALL  C45            ; MOVE ONE ROW
E5EE      ADD     SI,BP        ;
E5F0      ADD     DI,BP        ; POINT TO NEXT LINE IN BLOCK
E5F2      DEC     AH           ; COUNT OF LINES TO MOVE
E5F4      JNZ    C40          ; ROW_LOOP
E5F6      POP     AX           ; RECOVER ATTRIBUTE IN AH
E5F7      MOV     AL,' '       ; FILL WITH BLANKS
E5F9      CALL  C46            ; CLEAR THE ROW
E5FC      ADD     DI,BP        ; POINT TO NEXT LINE
E5FE      DEC     BL           ; COUNTER OF LINES TO SCROLL
E600      JNZ    C42          ; CLEAR_LOOP
E602      JMP     VIDEO_RETURN
E605      MOV     BL,DH         ; GET ROW COUNT
E607      JMP     C41          ; GO CLEAR THAT AREA
E609
SCROLL_UP ENDP
;----- HANDLE COMMON SCROLL SET UP HERE
SCROLL_POSITION PROC NEAR
E609      CALL  POSITION        ; CONVERT TO REGEN POINTER
E60C      ADD     AX,CRT_START  ; OFFSET OF ACTIVE PAGE
E610      MOV     DI,AX         ; TO ADDRESS FOR SCROLL
E612      MOV     SI,AX         ; FROM ADDRESS FOR SCROLL
E614      SUB     DX,CX         ; DX = #ROWS, #COLS IN BLOCK
E616      INC     DH           ;
E618      FE     C2            ; INCREMENT FOR 0 ORIGIN
E61A      XOR     CH,CH        ; SET HIGH BYTE OF COUNT TO ZERO
E61C      MOV     BP,CRT_COLS  ; GET NUMBER OF COLUMNS IN DISPLAY
E620      ADD     BP,SP        ; TIMES 2 FOR ATTRIBUTE BYTE
E622      MOV     AL,BL         ; GET LINE COUNT
E624      MUL     BYTE PTR CRT_COLS ; DETERMINE OFFSET TO FROM
; ADDRESS
E628      ADD     AX,AX         ; *2 FOR ATTRIBUTE BYTE
E62A      PUSH   ES            ; ESTABLISH ADDRESSING TO REGEN
; BUFFER
E62B      POP     DS           ; FOR BOTH POINTERS
E62C      OR     BL,BL         ; 0 SCROLL MEANS BLANK FIELD
E62E      RET                  ; RETURN WITH FLAGS SET
E62F
SCROLL_POSITION ENDP
;----- MOVE_ROW
E62F      PROC  NEAR
E62F      MOV     CL,DL        ; GET # OF COLS TO MOVE
E631      PUSH  SI             ;
E632      PUSH  DI             ; SAVE START ADDRESS
E633      REP   MOVSW          ; MOVE THAT LINE ON SCREEN
E635      POP   DI             ;
E636      POP   SI             ; RECOVER ADDRESSES
E637      RET
E638
C45      ENDP
;----- CLEAR_ROW
E638      PROC  NEAR
E638      MOV     CL,DL        ; GET # COLUMNS TO CLEAR
E63A      PUSH  DI             ;
E63B      PUSH  STOSW          ; STORE THE FILL CHARACTER
E63D      POP   DI             ;
E63E      RET
E63F
C46      ENDP
;-----
; SCROLL_DOWN
; THIS ROUTINE MOVES THE CHARACTERS WITHIN A DEFINED
; BLOCK DOWN ON THE SCREEN, FILLING THE TOP LINES
; WITH A DEFINED CHARACTER
; INPUT
; (AH) = CURRENT CRT MODE
; (AL) = NUMBER OF LINES TO SCROLL
; (CX) = UPPER LEFT CORNER OF REGION
; (DX) = LOWER RIGHT CORNER OF REGION
; (BH) = FILL CHARACTER
; (DS) = DATA SEGMENT
; (ES) = REGEN SEGMENT
; OUPUT
; NONE -- SCREEN IS SCROLLED
;-----
SCROLL_DOWN PROC NEAR
E63F      STD             ; DIRECTION FOR SCROLL DOWN
E640      MOV     BL,AL        ; LINE COUNT TO BL
E642      CMP     AH,4          ; TEST FOR GRAPHICS
E645      JC      C47          ;
E647      JMP     GRAPHICS_DOWN
E64A      PUSH  BX             ; SAVE ATTRIBUTE IN BH
E648      MOV     AX,DX        ; LOWER RIGHT CORNER
E649      CALL  SCROLL_POSITION ; GET REGEN LOCATION
E650      JZ      C51          ;
E652      SUB     SI,AX         ; SI IS FROM ADDRESS
E654      MOV     AH,DH        ; GET TOTAL # ROWS
E656      SUB     AH,BL        ; COUNT TO MOVE IN SCROLL

```



```

E658 EB E62F R
E658 28 F6
E65D 28 F0
E66F FE CC
E661 75 F5
E663 58
E684 80 20
E886 EB E63B R
E669 2B FD
E668 FE C8
E66D 75 F7
E66F EB 91
E871 8A DE
E673 EB EE
E675

```

```

C48: CALL C45 ; MOVE ONE ROW
SUB S1,8P
SUB D1,8P
DEC AH
JNZ C48
C49: POP AX ; RECOVER ATTRIBUTE IN AH
AL, ' '
C50: CALL C46 ; CLEAR ONE ROW
D1,8P ; GO TO NEXT ROW
DEC BL
JNZ C50
JMP C43 ; SCROLL_END
C51: MOV BL,DH
JMP C49
SCROLL_DOWN ENDP

```

```

; MODE_ALIVE
; THIS ROUTINE READS 256 LOCATIONS IN MEMORY AS EVERY OTHER
; LOCATION IN 512 LOCATIONS. THIS IS TO INSURE THE DATA
; INTEGRITY OF MEMORY DURING MODE CHANGES.

```

```

E675
E675 50
E676 56
E677 51
E678 33 F6
E67A 89 0100
E67D AC
E67E 46
E67F E2 FC
E681 59
E682 0E
E683 58
E684 C3
E685

```

```

MODE_ALIVE PROC NEAR
; SAVE USED REGS
PUSH AX
PUSH S1
PUSH CX
XOR S1,S1
MOV CX,256
C52: LODSB
INC S1
LOOP C52
POP CX
POP S1
POP AX
MODE_ALIVE ENDP

```

```

; SET_PALETTE
; THIS ROUTINE WRITES THE PALETTE REGISTERS

```

```

; INPUT
(AL) = 0 SET PALETTE REG
(BH) = VALUE TO SET
(BL) = PALETTE REG TO SET
(AL) = 1 SET BORDER COLOR REG
(BH) = VALUE TO SET
(AL) = 2 SET ALL PALETTE REGS AND BORDER REG
NOTE: REGISTERS ARE WRITE ONLY.

```

```

E685
E685 50
E686 08 F4
E688 36: 8B 44 0C
E69C 8E C0
E69E 8B F2
E690 8A 03DA
E693 EC
E694 24 08
E698 75 FB
E698 EC
E699 24 08
E698 74 FB
E690 58
E69E 0A C0
E6A0 74 0C
E6A2 3C 02
E6A4 74 17
E6A6 3C 01
E6A8 75 28
E6AA 80 02
E6AC EB 08
E6AE 8A C3
E6B0 24 0F
E6B2 0C 10
E6B4 EE
E6B6 8A C7
E6B7 EE
E6B8 32 C0
E6BA EE
E6BB EB 18
E6BD 84 10
E6BF 8A C4
E6C1 EE
E6C2 26: 8A 04
E6C5 EE
E6C6 46
E6C7 FE C4
E6C9 80 FC 20
E6CC 72 F1
E6CE 80 02
E6D0 EE
E6D1 26: 8A 04
E6D4 EE

```

```

SET_PALETTE PROC NEAR
PUSH AX
MOV SI,SP
MOV AX,55:[SI+12] ; GET SEG FROM STACK
MOV ES,AX
MOV SI,0X ; OFFSET IN SI
MOV DX,VGA_CTL ; SET VGA CONTROL PORT
C53: IN AL,0X ; GET VGA STATUS
AND AL,08H ; IN VERTICAL RETRACE?
JNZ C53 ; YES, WAIT FOR IT TO GO AWAY
C54: IN AL,DX ; GET VGA STATUS
AND AL,08H ; IN VERTICAL RETRACE?
JZ C54 ; NO, WAIT FOR IT
POP AX
OR AL,AL ; SET PALETTE REG?
JZ C55 ; YES, GO DO IT
CMP AL,2 ; SET ALL REGS?
JE C57
CMP AL,1 ; SET BORDER COLOR REG?
JNE C59 ; NO, DON'T DO ANYTHING
MOV AL,2 ; SET BORDER COLOR REG NUMBER
C55: MOV AL,BL ; GET DESIRED REG NUMBER IN AL
AND AL,0FH ; STRIP UNUSED BITS
OR AL,10H ; MAKE INTO REAL REG NUMBER
C56: OUT DX,AL ; SELECT REG
MOV AL,BH ; GET DATA IN AL
OUT DX,AL ; SET NEW DATA
XOR AL,AL ; SET REG 0 SO DISPLAY WORKS AGAIN
OUT DX,AL
JMP SHORT C59
C57: MOV AH,10H ; AH IS REG COUNTER
MOV AL,AH ; REG ADDRESS IN AL
OUT DX,AL ; SELECT IT
MOV AL,BYTE PTR ES:[SI] ; GET DATA
OUT DX,AL ; PUT IN VGA REG
INC SI ; NEXT DATA BYTE
AH ; NEXT REG
CMP AH,20H ; LAST PALETTE REG?
JNB C58 ; NO, DO NEXT ONE
MOV AL,2 ; SET BORDER REG
OUT DX,AL ; SELECT IT
MOV AL,BYTE PTR ES:[SI] ; GET DATA
OUT DX,AL ; PUT IN VGA REG

```

```

E6D4 EE          OUT    DX,AL          ; PUT IN VGA REG
E6D5 E9 0F70 R   CS9:   JMP    VIDEO_RETURN ; ALL 00NE
E6D8             SET_PALLETTE ENDP
E6D8             MFG_UP PROC NEAR
E6D8 50          PUSH   AX
E6D9 1E          PUSH   DS
                ASSUME DS:XXDATA
E6DA BB ---- R   MOV    AX,XXDATA
E6DB BE DB      MOV    DS,AX
E6DF A0 0005 R   MOV    AL,MFG_TST ; GET MFG CHECKPOINT
E6E2 E6 10      OUT    IOH,AL      ; OUTPUT IT TO TESTER
E6E4 FE C8      DEC    AL          ; DROP IT BY 1 FOR THE NEXT TEST
E6E6 A2 0005 R   MOV    MFG_TST,AL
                ASSUME DS:AB50
E6E9 1F         POP    DS
E6EA 58         POP    AX
E6EB C3         RET
E6EC             MFG_UP ENOP
                ASSUME CS:CODE,DS:DATA
E6F2             ORG    066F2H
E6F2 E9 0B18 R   JMP    NEAR PTR BOOT_STRAP
-----
; SUBROUTINE TO SET UP CONDITIONS FOR THE TESTING OF B259 AND
; B259 INTERRUPTS. ENABLES MASKABLE EXTERNAL INTERRUPTS,
; CLEARS THE B259 INTR RECEIVED FLAG BIT, AND ENABLES THE
; DEVICE'S B259 INTR (WHICHEVER IS BEING TESTED).
; IT EXPECTS TO BE PASSED:
; (DS) = ADDRESS OF SEGMENT WHERE INTR_FLAG IS DEFINED
; (DI) = OFFSET OF THE INTERRUPT BIT MASK
; UPON RETURN:
; INTR_FLAG BIT FOR THE DEVICE = 0
; NO REGISTERS ARE ALTERED.
-----
E6F5 50          SUI    PROC NEAR
E6F5 FB          PUSH   AX
E6F6 FB          STI    ; ENABLE MASKABLE EXTERNAL
                ; INTERRUPTS
E6F7 2E: BA 25   MOV    AH,CS:[DI] ; GET INTERRUPT BIT MASK
E6FA 20 26 00B4 R AND    INTR_FLAG,AH ; CLEAR B259 INTERRUPT REC'D FLAG
                ; BIT
E6FE E4 21      IN    AL,INTA01 ; CURRENT INTERRUPTS
E700 22 C4      AND    AL,AH    ; ENABLE THIS INTERRUPT, TOO
E702 E6 21      OUT    INTA01,AL ; WRITE TO B259 (INTERRUPT
                ; CONTROLLER)
E704 58         POP    AX
E705 C3         RET
E706             SUI    ENDP
-----
; SUBROUTINE WHICH CHECKS IF A B259 INTERRUPT IS GENERATED BY THE
; B259 INTERRUPT.
; IT EXPECTS TO BE PASSED:
; (DI) = OFFSET OF INTERRUPT BIT MASK
; (OS) = ADDRESS OF SEGMENT WHERE INTR_FLAG IS DEFINED.
; IT RETURNS:
; (CF) = 1 IF NO INTERRUPT IS GENERATED
; 0 IF THE INTERRUPT OCCURRED
; (AL) = COMPLEMENT OF THE INTERRUPT MASK
; NO OTHER REGISTERS ARE ALTERED.
-----
E706             CS059 PROC NEAR
E706 51          PUSH   CX
E707 2B C9      SUB    CX,CX      ; SET PROGRAM LOOP COUNT
E709 2E: BA 05   MOV    AL,CS:[DI] ; GET INTERRUPT MASK
E70C 34 FF      XOR    AL,OFFH    ; COMPLEMENT MASK SO ONLY THE INTR
                ; TEST BIT IS ON
E70E 84 06 00B4 R AT25: TEST INTR_FLAG,AL ; B259 INTERRUPT OCCUR?
E712 75 03      JNE    AT27      ; YES - CONTINUE
E714 E2 FB      LOOP  AT25     ; WAIT SOME MORE
E716 F9        STC    ; TIME'S UP - FAILED
E717 59        POP    CX
E718 C3        RET
E719             CS059 ENDP
-----
; SUBROUTINE TO WAIT FOR ALL ENABLED B250 INTERRUPTS TO CLEAR (SO
; NO INTRs WILL BE PENDING). EACH INTERRUPT COULD TAKE UP TO
; 1 MILLISECOND TO CLEAR. THE INTERRUPT IDENTIFICATION
; REGISTER WILL BE CHECKED UNTIL THE INTERRUPT(S) IS CLEARED
; OR A TIMEOUT OCCURS.
; EXPECTS TO BE PASSED:
; (OX) = ADDRESS OF THE INTERRUPT ID REGISTER
; RETURNS:
; (AL) = CONTENTS OF THE INTR ID REGISTER
; (CF) = 1 IF INTERRUPTS ARE STILL PENDING
; 0 IF NO INTERRUPTS ARE PENDING (ALL CLEAR)
; NO OTHER REGISTERS ARE ALTERED.
-----
E719             WB250C PROC NEAR
E719 51          PUSH   CX
E71A 2B C9      SUB    CX,CX
E71C EC        IN    AL,DX ; READ INTR ID REG
E71D 3C 01      CMP    AL,1      ; INTERRUPTS STILL PENDING?
E71F 74 05      JE     AT29      ; NO - GOOD FINISH
E721 E2 F9      LOOP  AT26     ; KEEP TRYING
E723 F9        STC    ; TIME'S UP - ERROR
E724 E9 01      JMP    SHORT AT30
E726 FB        RET
E727 59        POP    CX
E728 C3        RET
E729             WB250C ENDP

```

```

-----INT 14-----
;RS232_10
THIS ROUTINE PROVIDES BYTE STREAM I/O TO THE COMMUNICATIONS
PORT ACCORDING TO THE PARAMETERS:
(AH)=0 INITIALIZE THE COMMUNICATIONS PORT
(AL) HAS PARMS FOR INITIALIZATION

-----7-----6-----5-----4-----3-----2-----1-----0-----
BAUD RATE ----- PARITY ----- STOPBIT ----- WORD LENGTH-----

000 - 110                XO - NONE                0 - 1    10 - 7 BITS
001 - 180                01 - ODD                  1 - 2    11 - 8 BITS
010 - 300                11 - EVEN
011 - 600
100 - 1200
101 - 2400
110 - 4800
111 - 4800

ON RETURN, THE RS232 INTERRUPTS ARE DISABLED AND
CONDITIONS ARE SET AS IN CALL TO COMMO
STATUS (AH=3)
(AH)=1 SEND THE CHARACTER IN (AL) OVER THE COMMO LINE
(AL) REGISTER IS PRESERVED
ON EXIT, BIT 7 OF AH IS SET IF THE ROUTINE WAS
UNABLE TO TRANSMIT THE BYTE OF DATA OVER
THE LINE. IF BIT 7 OF AH IS NOT SET, THE
REMAINDER OF AH IS SET AS IN A STATUS
REQUEST, REFLECTING THE CURRENT STATUS OF
THE LINE.
(AH)=2 RECEIVE A CHARACTER IN (AL) FROM COMMO LINE BEFORE
RETURNING TO CALLER
ON EXIT, AH HAS THE CURRENT LINE STATUS, AS SET BY
THE STATUS ROUTINE, EXCEPT THAT THE ONLY
BITS LEFT ON, ARE THE ERROR BITS
(7, 4, 3, 2, 1). IN THIS CASE, THE TIME OUT BIT
INDICATES DATA SET READY WAS NOT RECEIVED.
THUS, AH IS NON ZERO ONLY WHEN AN ERROR
OCCURRED. (NOTE: IF THE TIME-OUT BIT IS SET,
OTHER BITS IN AH MAY NOT BE RELIABLE.)
(AH)=3 RETURN THE COMMO PORT STATUS IN (AX)
AH CONTAINS THE LINE CONTROL STATUS
BIT 7 = TIME OUT
BIT 6 = TRANS SHIFT REGISTER EMPTY
BIT 5 = TRAN HOLDING REGISTER EMPTY
BIT 4 = BREAK DETECT
BIT 3 = FRAMING ERROR
BIT 2 = PARITY ERROR
BIT 1 = OVERRUN ERROR
BIT 0 = DATA READY
AL CONTAINS THE MODEM STATUS
BIT 7 = RECEIVED LINE SIGNAL DETECT
BIT 6 = RING INDICATOR
BIT 5 = DATA SET READY
BIT 4 = CLEAR TO SEND
BIT 3 = DELTA RECEIVE LINE SIGNAL DETECT
BIT 2 = TRAILING EDGE RING DETECTOR
BIT 1 = DELTA DATA SET READY
BIT 0 = DELTA CLEAR TO SEND
(DX) = PARAMETER INDICATING WHICH RS232 CARD (0,1 ALLOWED)
DATA AREA RS232_BASE CONTAINS THE BASE ADDRESS OF THE B250 ON THE
CARD. LOCATION 400H CONTAINS UP TO 4 RS232 ADDRESSES POSSIBLE
DATA AREA RS232_TIM_OUT (BYTE) CONTAINS OUTER LOOP COUNT
VALUE FOR TIMEOUT (DEFAULT=1)
;OUTPUT
AX MODIFIED ACCORDING TO PARMS OF CALL
ALL OTHERS UNCHANGED
-----
ASSUME CS:CODE,DS:DATA
ORG 0E729H
A1 LABEL WORD
E729 03F9 DW 1017 ; 110 BAUD ; TABLE OF INIT VALUE
E72B 02EA DW 746 ; 180
E72D 0175 DW 373 ; 300
E72F 008A DW 186 ; 600
E731 005D DW 23 ; 1200
E733 002F DW 47 ; 2400
E735 0017 DW 23 ; 4800
E737 0017 DW 23 ; 4800
RS232_10 PROC FAR
;----- VECTOR TO APPROPRIATE ROUTINE
STI ; INTERRUPTS BACK ON
E73A 1E PUSH 05 ; SAVE SEGMENT
E73B 52 PUSH 0X
E73C 56 PUSH S1
E73D 57 PUSH D1
E73E 51 PUSH CX
E73F 53 PUSH 8X
E740 88 F2 MOV SI,DX ; RS232 VALUE TO SI
E742 8B FA MOV DI,0X ; AND TO DI (FOR TIMEOUTS)
E744 01 E6 SHL SI,1 ; WORD OFFSET
E746 EB 138B R CALL D05 ; POINT TO BIOS DATA SEGMENT
E749 8B 94 0000 R MOV DX,RS232_BASE(SI) ; GET BASE ADDRESS
E74D 0B D2 OR DX,DX ; TEST FOR 0 BASE ADDRESS
E74F 74 13 JZ A3 ; RETURN
E751 0A E4 OR AH,AH ; TEST FOR (AH)=0
E753 74 16 JZ A4 ; COMMUN INIT
E755 FE CC DEC AH ; TEST FOR (AH)=1
E757 74 47 JZ A5 ; SEND AL
E759 FE CC DEC AH ; TEST FOR (AH)=2
E75B 74 8C JZ A12 ; RECEIVE INTO AL
E75D FE CC DEC AH ; TEST FOR (AH)=3
E75F 75 03 JNZ A3
E761 E9 E7F3 R JMP A18 ; COMMUNICATION STATUS

```

```

E764
E764 5B POP BX ; RETURN FROM RS232
E765 59 POP CX
E766 5F POP DI
E767 5E POP SI
E766 5A POP DX
E769 1F POP DS
E76A CF IRET
;----- INITIALIZE THE COMMUNICATIONS PORT
E76B 8A E0 MOV AH,AL ; SAVE INIT PARMS IN AH
E76D 83 C2 03 ADD DX,3 ; POINT TO 8250 CONTROL REGISTER
E770 80 80 MOV AL,80H
E772 EE OUT DX,AL ; SET DLAB=1
;----- DETERMINE BAUD RATE DIVISOR
E773 8A D4 MOV DL,AH ; GET PARMS TO DL
E775 81 04 MOV CL,4
E777 D2 C2 ROL DL,CL
E779 81 E2 000E ANO DX,0EH ; ISOLATE THEM
E77D 9F E729 R MOV DI,OFFSET A1 ; BASE OF TABLE
E780 03 FA ADD DI,DX ; PUT INTO INDEX REGISTER
E782 8B 94 0000 R MOV DX,RS232_BASE[SI] ; POINT TO HIGH ORDER OF DIVISOR
E786 42 INC DX
E787 2E: BA 45 01 MOV AL,CS:[DI]*1 ; GET HIGH ORDER OF DIVISOR
E788 EE OUT DX,AL ; SET HS OF DIV TO 0
E78C 4A DEC DX
E78D 2E: BA 05 MOV AL,CS:[DI] ; GET LOW ORDER OF DIVISOR
E790 EE OUT DX,AL ; SET LOW OF DIVISOR
E791 83 C2 03 ADD DX,3
E794 8A C4 MOV AL,AH ; GET PARMS BACK
E796 24 1F AND AL,01FH ; STRIP OFF THE BAUD BITS
E798 EE OUT DX,AL ; LINE CONTROL TO 8 BITS
E799 4A DEC DX
E79A 4A DEC DX
E798 80 00 MOV AL,0
E79D EE OUT DX,AL ; INTERRUPT ENABLES ALL OFF
E79E EB 53 JMP SHORT A1B ; COM STATUS
;----- SEND CHARACTER IN (AL) OVER COMMO LINE
E7A0
A5: PUSH AX ; SAVE CHAR TO SEND
E7A1 83 C2 04 ADD DX,4 ; MODEM CONTROL REGISTER
E7A4 80 03 MOV AL,3 ; DTR AND RTS
E7A8 EE OUT DX,AL ; DATA TERMINAL READY, REQUEST TO
; SEND
E7A7 42 INC DX ; MODEM STATUS REGISTER
E7A8 42 INC DX
E7A9 87 30 MOV BH,30H ; DATA SET READY & CLEAR TO SEND
E7AB EB E802 R CALL WAIT_FOR_STATUS ; ARE BOTH TRUE?
E7AE 74 0B JE A9 ; YES, READY TO TRANSMIT CHAR
E780 59 POP CX
A7: MOV AL,CL ; RELOAD DATA BYTE
E781 8A C1 AB: OR AH,80H ; INDICATE TIME OUT
E783 80 CC 80 JMP A3 ; RETURN
E786 EB AC
A9: DEC DX ; CLEAR TO SEND
E788 4A DEC DX ; LINE STATUS REGISTER
E789 87 20 MOV BH,20H ; IS TRANSMITTER READY
E7AB EB E802 R CALL WAIT_FOR_STATUS ; TEST FOR TRANSMITTER READY
E78E 75 F0 JNZ A7 ; RETURN WITH TIME OUT SET
E7C0 83 EA 05 SUB DX,5 ; DATA PORT
E7C3 59 POP CX ; RECOVER IN CX TEMPORARILY
E7C4 8A C1 MOV AL,CL ; MOVE CHAR TO AL FOR OUT, STATUS
; IN AH
E7C6 EE OUT DX,AL ; OUTPUT CHARACTER
E7C7 EB 98 JMP A3 ; RETURN
;----- RECEIVE CHARACTER FROM COMMO LINE
E7C9 83 C2 04 A12: ADD DX,4 ; MODEM CONTROL REGISTER
E7CC 80 01 MOV AL,1 ; DATA TERMINAL READY
E7CE EE OUT DX,AL
E7CF 42 INC DX ; MODEM STATUS REGISTER
E7D0 42 INC DX
E7D1 87 20 MOV BH,20H ; DATA SET READY
E7D3 EB E802 R CALL WAIT_FOR_STATUS ; TEST FOR DSR
E7D6 75 DB JNZ A8 ; RETURN WITH ERROR
E7D8 4A DEC DX ; LINE STATUS REGISTER
E7D9 EC
A16: IN AL,DX
E7DA A8 01 TEST A1 ; RECEIVE BUFFER FULL
E7DC 75 09 JNZ A17 ; TEST FOR REC. BUFF. FULL
E7DE F6 06 0071 R 80 TEST BIOS_BREAK,80H ; TEST FOR BREAK KEY
E7E3 74 F4 JZ A16 ; LOOP IF NO BREAK KEY
E7E5 EB CC JMP A8 ; SET TIME OUT ERROR
E7E7 24 1E A17: AND AL,00011110B ; TEST FOR ERROR CONDITIONS ON REC'V
; CHAR
E7E9 8A E0 MOV AH,AL
E7EB 8B 94 0000 R MOV DX,RS232_BASE[SI] ; DATA PORT
E7EF EC IN AL,DX ; GET CHARACTER FROM LINE
E7F0 EB E764 R JMP A3 ; RETURN
;----- COMMO PORT STATUS ROUTINE
E7F3 8B 94 0000 R A1B: MOV DX,RS232_BASE[SI]
E7F7 83 C2 05 ADD DX,5 ; CONTROL PORT
E7FA EC IN AL,DX ; GET LINE CONTROL STATUS
E7FB 8A E0 MOV AH,AL ; PUT IN AH FOR RETURN
E7FD 42 INC DX ; POINT TO MODEM STATUS REGISTER
E7FE EC IN AL,DX ; GET MODEM CONTROL STATUS
E7FF E9 E764 R JMP A3 ; RETURN
;-----
; WAIT FOR STATUS ROUTINE
; ENTRY: BH=STATUS BIT(S) TO LOOK FOR,
; DX=ADDR. OF STATUS REG
; EXIT: ZERO FLAG ON = STATUS FOUND
; ZERO FLAG OFF = TIMEOUT.
; AH=LAST STATUS READ

```

```

EB02
EB02 8A 9D 007C R
EB06 2B C9
EB08 EC
EB09 8A E0
EB08 22 C7
EB0D 3A C7
EB0F 74 08
EB11 E2 F5
EB13 FE C8
EB15 75 EF
EB17 0A FF
EB19
EB1A C3
EB1A
EB1A

```

```

WAIT_FOR_STATUS PROC NEAR
MOV BL,RS232_TIM_OUTD13 ;LOAD OUTER LOOP COUNT
WFS0: SUB CX,CX
WFS1: IN AL,DX ;GET STATUS
MOV AH,AL ;MOVE TO AH
AND AL,BH ;ISOLATE BITS TO TEST
CMP AL,BH ;EXACTLY = TO MASK
JE WFS_END ;RETURN WITH ZERO FLAG ON
LOOP WFS1 ;TRY AGAIN
DEC BL
JNZ WFS0
OR BH,BH ;SET ZERO FLAG OFF
WFS_END: RET
WAIT_FOR_STATUS ENDP
RS232_IO ENDP

```

```

EB1A
EB1A 80 40
EB1C E6 43
EB1E 50
EB1F 5B
EB20 E4 41
EB22 8A E0
EB24 50
EB25 5B
EB26 E4 41
EB28 86 C4
EB2A C3
EB2B
EB2E
EB2E E9 13DD R

```

```

-----
; THIS ROUTINE WILL READ TIMER1. THE VALUE READ IS RETURNED IN AX.
-----
READ_TIME PROC NEAR
MOV AL,40H ; LATCH TIMER1
OUT TIM_CTL,AL
PUSH AX ; WAIT FOR 8253 TO INIT ITSELF
POP AX
IN AL,TIMER+1 ; READ LSB
MOV AH,AL ; SAVE IT IN HIGH BYTE
PUSH AX ; WAIT FOR 8253 TO INIT ITSELF
POP AX
IN AL,TIMER+1 ; READ MSB
XCHG AL,AH ; PUT BYTES IN PROPER ORDER
RET
READ_TIME ENDP
ORG 0E92EH
JMP NEAR PTR KEYBOARD_IO

```

```

-----
; ASYNCHRONOUS COMMUNICATIONS ADAPTER POWER ON DIAGNOSTIC TEST
; DESCRIPTION:
; THIS SUBROUTINE PERFORMS A THOROUGH CHECK OUT OF AN INSB250 LSI
; CHIP.
; THE TEST INCLUDES:
; 1) INITIALIZATION OF THE CHIP TO ASSUME ITS MASTER RESET STATE.
; 2) READING REGISTERS FOR KNOWN PERMANENT ZERO BITS.
; 3) TESTING THE INSB250 INTERRUPT SYSTEM AND THAT THE 8250
; INTERRUPTS TRIGGER AN 8259 (INTERRUPT CONTROLLER) INTERRUPT.
; 4) PERFORMING THE LOOP BACK TEST:
; A) TESTING WHAT WAS WRITTEN/READ AND THAT THE TRANSMITTER
; HOLDING REG EMPTY BIT AND THE RECEIVER INTERRUPT WORK
; PROPERLY.
; B) TESTING IF CERTAIN BITS OF THE DATA SET CONTROL REGISTER
; ARE 'LOOPED BACK' TO THOSE IN THE DATA SET STATUS
; REGISTER.
; C) TESTING THAT THE TRANSMITTER IS IDLE WHEN TRANSMISSION
; TEST IS FINISHED
; THIS SUBROUTINE EXPECTS TO HAVE THE FOLLOWING PARAMETER PASSED:
; (DX)= ADDRESS OF THE INSB250 CARD TO TEST.
; NOTE: THE ASSUMPTION HAS BEEN MADE THAT THE MODEM ADAPTER IS
; ---- LOCATED AT 03F8H; THE SERIAL PRINTER AT 02F8H.
; IT RETURNS:
; (CF) = 1 IF ANY PORTION OF THE TEST FAILED
; = 0 IF TEST PASSED
; (BX) = FAILURE KEY FOR ERROR MESSAGE (ONLY VALID IF TEST FAILED)
; (BH) = 23H SERIAL PRINTER ADAPTER TEST FAILURE
; = 24H MODEM ADAPTER TEST FAILURE
; (BL) = 2 PERMANENT ZERO BITS IN INTERRUPT ENABLE REGISTER
; WERE INCORRECT
; 3 PERMANENT ZERO BITS IN INTERRUPT IDENTIFICATION
; REGISTER WERE INCORRECT
; 4 PERMANENT ZERO BITS IN DATA SET CONTROL REGISTER
; WERE INCORRECT
; 5 PERMANENT ZERO BITS IN THE LINE STATUS REGISTER
; WERE INCORRECT
; 6 RECEIVED DATA AVAILABLE INTERRUPT TEST FAILED
; (THE INTERRUPT WAS NOT GENERATED)
; 16H RECEIVED DATA AVAILABLE INTERRUPT FAILED TO CLEAR
; 7 RESERVED FOR REPORTING THE TRANSMITTER HOLDING
; REGISTER EMPTY INTERRUPT TEST FAILED
; (NOT USED AT THIS TIME BECAUSE OF THE DIFFERENCES
; BETWEEN THE 8250'S WHICH WILL BE USED)
; 17H TRANSMITTER HOLDING REG EMPTY INTR FAILED TO CLEAR
; B-B RECEIVER LINE STATUS INTERRUPT TEST FAILED
; (THE INTERRUPT WAS NOT GENERATED)
; B - OVERRUN ERROR
; 9 - PARITY ERROR
; A - FRAMING ERROR
; B - BREAK INTERRUPT ERROR
; 1B-1B RECEIVER LINE STATUS INTERRUPT FAILED TO CLEAR
; C-F MODEM STATUS INTERRUPT TEST FAILED
; (THE INTERRUPT WAS NOT GENERATED)
; C - DELTA CLEAR TO SEND ERROR
; D - DELTA DATA SET READY ERROR
; E - TRAILING EDGE RING INDICATOR ERROR
; F - DELTA RECEIVE LINE SIGNAL DETECT ERROR

```

```

;
; IC-1F MODEM STATUS INTERRUPT FAILED TO CLEAR
;
; 10H AN B250 INTERRUPT OCCURRED AS EXPECTED, BUT NO
; B259 (INTR CONTROLLER) INTERRUPT WAS GENERATED
;
; 11H DURING THE TRANSMISSION TEST, THE TRANSMITTER
; HOLDING REGISTER WAS NOT EMPTY WHEN IT SHOULD
; HAVE BEEN.
;
; 12H DURING THE TRANSMISSION TEST, THE RECEIVED DATA
; AVAILABLE INTERRUPT DIDN'T OCCUR.
;
; 13H TRANSMISSION ERROR - THE CHARACTER RECEIVED
; DURING LOOP MODE WAS NOT THE SAME AS THE ONE
; TRANSMITTED.
;
; 14H DURING TRANSMISSION TEST, THE 4 DATA SET CONTROL
; OUTPUTS WERE NOT THE SAME AS THE 4 DATA SET
; CONTROL INPUTS.
;
; 15H THE TRANSMITTER WAS NOT IDLE AFTER THE TRANS-
; MISSION TEST COMPLETED.
;
ON EXIT:
; - THE MODEM OR SERIAL PRINTER'S B259 INTERRUPT (WHICHEVER
; DEVICE WAS TESTED) IS DISABLED.
; - THE B250 IS IN THE MASTER RESET STATE.
; ONLY THE DS REGISTER IS PRESERVED - ALL OTHERS ARE ALTERED.
-----
= 0084 WRAP EQU 84H ; LOOP BACK TRANSMISSION TEST
; INTERRUPT VECTOR ADDRESS
; (IN DIAGNOSTICS)
;
; ASSUME CS:CODE,DS:DATA
;
UART PROC NEAR
;
; PUSH DS ; CURRENT ENABLED INTERRUPTS
; IN AL,INTA01 ; SAVE FOR EXIT
; PUSH AX ; DISABLE TIMER INTR DURING THIS
; OR AL,0000001B ; TEST
;
; OUT INTA01,AL ; SAVE CALLER'S FLAGS (SAVE INTR
; PUSHF ; FLAG)
;
; PUSH DX ; SAVE BASE ADDRESS OF ADAPTER CARD
; CALL D05 ; SET UP 'DATA' AS DATA SEGMENT
; ; ADDRESS
;
; -----
; INITIALIZE PORTS FOR MASTER RESET STATES AND TEST PERMANENT
; ZERO DATA BITS FOR CERTAIN PORTS.
; -----
E83E EB 0AC4 R CALL I8250
E841 73 03 JNC AT1 ; ALL OK
E843 EB E94B R JMP AT14 ; A PORT'S ZERO BITS WERE NOT ZERO!
;
; -----
; INSB250 INTERRUPT SYSTEM TEST
; ONLY THE INTERRUPT BEING TESTED WILL BE ENABLED.
; -----
; SET DI AND SI FOR CALLS TO 'SUI'
;
E848 9F 0041 R AT1: MOV DI,OFFSET IMASKS ; BASE ADDRESS OF INTERRUPT MASKS
E849 33 F6 XOR SI,SI ; MODEM INDEX
E84B 80 FE 02 CMP DH,2 ; OR SERIAL?
E84E 75 02 JNE AT2 ; NO - IT'S MODEM
E850 46 INC SI ; IT'S SERIAL PRINTER
E851 47 INC DI ; SERIAL PRINTER B259 MASK ADDRESS
;
; RECEIVED DATA AVAILABLE INTERRUPT TEST
;
E852 EB E6F5 R AT2: CALL SUI ; SET UP FOR INTERRUPTS
E855 FE C3 INC 9L ; ERROR REPORTER (INIT. IN I8250)
E857 42 INC DX ; POINT TO INTERRUPT ENABLE
; REGISTER
;
E858 80 01 MOV AL,1 ; ENABLE RECEIVED DATA AVAILABLE
; INTR
;
E85A EE OUT DX,AL
E85B 53 PUSH BX ; SAVE ERROR REPORTER
E85C 83 C2 04 ADD DX,4 ; POINT TO LINE STATUS REGISTER
E85F 84 01 MOV AH,1 ; SET RECEIVER DATA READY BIT
E861 8B 0400 MOV BX,0400H ; INTR TO CHECK, INTR IDENTIFIER
E864 89 0003 MOV CX,3 ; INTERRUPT ID REG 'INDEX'
E867 EB 0AFB R CALL ICT ; PERFORM TEST FOR INTERRUPT
E86A 5B POP BX ; RESTORE ERROR INDICATOR
E86B 3C FF CMP AL,OFFH ; INTERRUPT ERROR OCCUR?
E86D 74 36 JE AT4 ; YES
E86F EB E706 R CALL C5059 ; GENERATE B259 INTERRUPT?
E872 72 33 JC AT5 ; NO
E874 4A DEC DX
E875 4A DEC DX ; RESET INTR BY READING RECR BUFR
E876 EC IN AL,DX ; DON'T CARE ABOUT THE CONTENTS!
E877 42 INC DX
E878 42 INC DX ; INTR ID REG
E879 EB E719 R CALL W8250C ; WAIT FOR INTR TO CLEAR
E87C 73 03 JNC AT3 ; OK
E87E E9 E94B R JMP AT13 ; DIDN'T CLEAR
;
; -----
; TRANSMITTER HOLDING REGISTER EMPTY INTERRUPT TEST
; THIS TEST HAS BEEN MODIFIED BECAUSE THE DIFFERENT B250'S
; THAT MAY BE USED IN PRODUCING THIS PRODUCT DO NOT FUNCTION
; THE SAME DURING THE STANDARD TEST OF THIS INTERRUPT
; (STANDARD BEING THE SAME METHOD FOR TESTING THE OTHER
; POSSIBLE B250 INTERRUPTS). IT IS STILL VALID FOR TESTING
; IF AN B259 INTERRUPT IS GENERATED IN RESPONSE TO THE B250
; INTERRUPT AND THAT THE B250 INTERRUPT CLEARS AS IT SHOULD.
;
; IF THE TRANSMITTER HOLDING REGISTER EMPTY INTERRUPT IS NOT
; GENERATED WHEN THAT INTERRUPT IS ENABLED, IT IS NOT TREATED
; AS AN ERROR. HOWEVER, IF THE INTERRUPT IS GENERATED, IT
; MUST GENERATE AN B259 INTERRUPT AND CLEAR PROPERLY TO PASS
; THIS TEST.
; -----

```



```

E914 E8 E719 R      CALL  WB250C      ; WAIT FOR INTERRUPT TO CLEAR
E917 72 2F          JC     AT13      ; IT DIDN'T
E919 4D             DEC,   BP
E91A 74 08          JE     AT12      ; ALL FOUR BITS TESTED - GO ON
E91C 00 E4          SHL   AH,1    ; GET READY FOR NEXT BIT
E91E 83 C2 04       ADD   DX,4      ; MODEM STATUS REGISTER
E921 EB D6          JMP   AT9       ; TEST NEXT BIT
;
;-----
; POSSIBLE 8259 INTERRUPT CONTROLLER PROBLEM
;-----
E923 83 10          AT10: MOV  BL,10H    ; SET ERROR REPORTER
E925 EB 24          AT11: JMP  SHORT AT14
;
;-----
; SET 9600 BAUD RATE AND DEFINE DATA WORD AS HAVING 8
; BITS/WORD, 2 STOP BITS, AND ODD PARITY.
;-----
E927 42             AT12: INC  DX      ; LINE CONTROL REGISTER
E928 EB F085 R      CALL  S9250
;
;-----
; SET DATA SET CONTROL WORD TO BE IN LOOP MODE
;-----
E92B 83 C2 04       ADD   DX,4
E92E EC            IN    AL,DX    ; CURRENT STATE
E92F EB 00          JMP   $+2      ; I/O DELAY
E931 0C 10          OR    AL,00010000B ; SET BIT 4 OF DATA SET CONTROL REG
E933 EE            OUT   DX,AL
E934 EB 00          JMP   $+2      ; I/O DELAY
E935 42             INC  DX
E937 42             INC  DX
E938 EC            IN    AL,DX    ; MODEM STATUS REG
; CLEAR POSSIBLE MODEM STATUS
; INTERRUPT WHICH COULD BE CAUSED
; BY THE OUTPUT BITS BEING LOOPEO
; TO THE INPUT BITS
E939 EB 00          JMP   $+2      ; I/O DELAY
E93B 83 EA 06       SUB   DX,6      ; RECEIVER BUFFER
E93E EC            IN    AL,DX    ; DUMMY READ TO CLEAR DATA READY
; BIT IF IT WENT HIGH ON WRITE TO
; MCR
;-----
; PERFORM THE LOOP BACK TEST
;-----
E93F 42             INC  DX
E940 80 00          MOV  AL,0
E942 CD 84          INT  WRAP      ; SET FOR INTERNAL WRAP TEST
E944 81 00          MOV  CL,0      ; DO LOOP BACK TRANSMISSION TEST
E946 73 05          JNC  AT15      ; ASSUME NO ERRORS
E948 80 C3 10       AT13: ADD  BL,10H  ; WRAP TEST PASSED
; ERROR INDICATOR
;-----
; AN ERROR WAS ENCOUNTERED SOMEWHERE DURING THE TEST
;-----
E94B 81 01          AT14: MOV  CL,1    ; SET FAIL INDICATOR
;-----
; HOUSEKEEPING: RE-INITIALIZE THE 8250 PORTS (THE LOOP BIT
; WILL BE RESET), DISABLE THIS DEVICE INTERRUPT, SET UP
; REGISTER BH IF AN ERROR OCCURRED, AND SET OR RESET THE
; CARRY FLAG.
;-----
E94D 5A             AT15: POP  DX      ; GET BASE ADDRESS OF 8250 ADAPTER
E94E 53             PUSH BX      ; SAVE ERROR CODE
E94F EB 0AC4 R      CALL  I6250   ; RE-INITIALIZE 8250 PORTS
E952 5B             POP  BX
E953 2E BA 25       MOV  AH,CS:[DI] ; GET DEVICE INTERRUPT MASK
E956 20 26 00B4 R  AND  INTR_FLAG,AH ; CLEAR DEVICE'S INTERRUPT FLAG BIT
E95A 80 F4 FF       XOR  AH,0FH   ; FLIP BITS
E95D E4 21          IN  AL,INTA01 ; GET CURRENT INTERRUPT PORT
E95F 0A C4          OR  AL,AH     ; DISABLE THIS DEVICE INTERRUPT
E961 E6 21          OUT  INTA01,AL
E963 9D            POPF
; RE-ESTABLISH CALLER'S INTERRUPT
; FLAG
E964 0A C9          OR   CL,CL    ; ANY ERRORS?
E966 74 0C          JE   AT17    ; NO
E968 87 24          MOV  BH,24H   ; ASSUME MODEM ERROR
E96A 80 FE 02       CMP  DH,2     ; OR IS IT SERIAL?
E96D 75 02          JNE  AT16    ; IT'S MODEM
E96F 87 23          MOV  BH,23H   ; IT'S SERIAL PRINTER
E971 F9             STC
; SET CARRY FLAG TO INDICATE ERROR
E972 EB 01          JMP  SHORT AT18
E974 F8             AT17: CLC
; RESET CARRY FLAG - NO ERRORS
E975 5B             AT18: POP  AX
; RESTORE ENTRY ENABLED INTERRUPTS
E976 ES 21          OUT  INTA01,AL ; DEVICE INTRs RE-ESTABLISHED
E978 1F             POP  DS
; RESTORE REGISTER
E979 C3             RET
E97A              UART  ENDP
E977 09             ORG  0E9B7H
E987 E9 1561 R      JMP  NEAR PTR KB_INT
;-----
; NEC_OUTPUT
; THIS ROUTINE SENDS A BYTE TO THE NEC CONTROLLER
; AFTER TESTING FOR CORRECT DIRECTION AND CONTROLLER READY
; THIS ROUTINE WILL TIME OUT IF THE BYTE IS NOT ACCEPTED
; WITHIN A REASONABLE AMOUNT OF TIME, SETTING THE DISKETTE
; STATUS ON COMPLETION
; INPUT (AH) BYTE TO BE OUTPUT
; OUTPUT
; CY = 0 SUCCESS
; CY = 1 FAILURE -- DISKETTE STATUS UPDATED
; IF A FAILURE HAS OCCURRED, THE RETURN IS MADE ONE
; LEVEL HIGHER THAN THE CALLER OF NEC_OUTPUT
; THIS REMOVES THE REQUIREMENT OF TESTING AFTER EVERY
; CALL OF NEC_OUTPUT
; (AL) DESTROYED
;-----

```



```

E98A
.E98A B2
E98B 51
E98C BA 00F4
E98F 33 C9
E991 EC
E992 A8 40
E994 74 0C
E996 E2 F9
E99B
E99B 80 0E 0041 R B0
E99D 59
E99E 5A
E99F 5B
E9A0 F9
E9A1 C3
E9A2 33 C9
E9A4 EC
E9A5 AB 80
E9A7 75 04
E9A8 E2 F9
E9AB EB EB
E9AD
E9A0 BA C4
E9AF 42

E9B0 EE
E9B1 59
E9B2 5A
E9B3 C3
E9B4

NEC_OUTPUT PROC NEAR
      PUSH DX ; SAVE REGISTERS
      PUSH CX
      MOV DX,NEC_STAT ; STATUS PORT
      XOR CX,CX ; COUNT FOR TIME OUT
      IN AL,DX ; GET STATUS
      TEST AL,D10 ; TEST DIRECTION BIT
      JZ J25 ; DIRECTION OK
      LOOP J23

J24: OR DISKETTE_STATUS,TIME_OUT ; TIME_ERROR
      POP CX
      POP DX ; SET ERROR CODE AND RESTORE REGS
      POP AX ; DISCARD THE RETURN ADDRESS
      STC ; INDICATE ERROR TO CALLER
      RET

J25: XOR CX,CX ; RESET THE COUNT
      IN AL,DX ; GET THE STATUS
      TEST AL,R0M ; IS IT READY?
      JNZ J27 ; YES, GO OUTPUT
      LOOP J26 ; COUNT DOWN AND TRY AGAIN
      JMP J24 ; ERROR CONDITION

J27: MOV AL,AH ; GET BYTE TO OUTPUT
      INC DX ; DATA PORT IS 1 GREATER THAN
      ; STATUS PORT
      OUT DX,AL ; OUTPUT THE BYTE
      POP CX ; RECOVER REGISTERS
      POP DX
      RET ; CY = 0 FROM TEST INSTRUCTION

NEC_OUTPUT ENDP

```

```

-----
GET_PARM
THIS ROUTINE FETCHES THE INDEXED POINTER FROM
THE DISK_BASE BLOCK POINTED AT BY THE DATA
VARIABLE DISK_POINTER
A BYTE FROM THAT TABLE IS THEN MOVED INTO AH,
THE INDEX OF THAT BYTE BEING THE PARM IN BX
ENTRY --
BL = INDEX OF BYTE TO BE FETCHED * 2
IF THE LOW BIT OF BL IS ON, THE BYTE IS IMMEDIATELY
OUTPUT TO THE NEC CONTROLLER
EXIT --
AH = THAT BYTE FROM BLOCK
BX = DESTROYED
-----

```

```

E9B4
E9B4 1E
E9B5 56
E9B6 2B C0
E9B8 32 FF
E9BA BE D8

E9BC C5 36 0079 R
E9C0 D1 EB

E9C2 9C
E9C3 BA 20
E9C5 B3 FB 01

E9C8 75 05
E9CA 80 CC 01
E9CD EB 0C
E9CF B3 FB 0A
E9D2 75 07
E9D4 80 FC 04
E9D7 7D 02
E9D9 B4 04
E9DB 9D
E9DC 5E
E9DD 1F

E9DE 72 AA
E9E0 C3
E9E1

GET_PARM PROC NEAR
      PUSH DS ; SAVE SEGMENT
      PUSH SI ; SAVE REGISTER
      SUB AX,AX ; ZERO TO AX
      XOR BH,BH ; ZERO BH
      MOV DS,AX
      ASSUME DS:ABS0
      LDS SI,DISK_POINTER ; POINT TO BLOCK
      SHR BX,1 ; DIVIDE BX BY 2, AND SET FLAG FOR
      ; EXIT
      PUSHF ; SAVE OUTPUT BIT
      MOV AH,[SI+BX] ; GET THE BYTE
      CMP BX,1 ; IS THIS THE PARM WITH DMA
      ; INDICATOR
      JNZ J27_1
      OR AH,1 ; TURN ON NO DMA BIT
      JMP SHORT J27_2

J27_1: CMP BX,10 ; MOTOR STARTUP DELAY?
      JNE J27_2
      CMP AH,4 ; GREATER THAN OR EQUAL TO 1/2 SEC?
      JGE J27_2 ; YES, OKAY
      MOV AH,4 ; NO, FORCE 1/2 SECOND DELAY

J27_2: POPF ; GET OUTPUT BIT
      POP SI ; RESTORE REGISTER
      POP D9 ; RESTORE SEGMENT
      ASSUME DS:DATA
      JC NEC_OUTPUT ; IF FLAG SET, OUTPUT TO CONTROLLER
      RET ; RETURN TO CALLER

GET_PARM ENDP

```

```

-----
BOUND_SETUP
THIS ROUTINE SETS UP BUFFER ADDRESSING FOR READ/WRITE/VERIFY
OPERATIONS.
INPUT
ES HAS ORIGINAL BUFFER SEGMENT VALUE
BP POINTS AT BASE OF SAVED PARAMETERS ON STACK
OUTPUT
ES HAS SEGMENT WHICH WILL ALLOW 64K ACCESS. THE
COMBINATION ES:D1 AND DS:S1 POINT TO THE BUFFER. THIS
CALCULATED ADDRESS WILL ALWAYS ACCESS 64K OF MEMORY.
BX DESTROYED
-----

```

```

E9E1          BOUND_SETUP      PROC      NEAR
E9E1 51        PUSH      CX          ; SAVE REGISTERS
E9E2 88 5E 0C MOV      BX,[BP+12] ; GET OFFSET OF BUFFER FROM STACK
E9E5 53        PUSH      BX          ; SAVE OFFSET TEMPORARILY
E9E6 81 04    MOV      CL,4          ; SHIFT COUNT
E9E8 03 EB    SHR      BX,CL        ; SHIFT OFFSET FOR NEW SEGMENT
; VALUE
E9EA BC C1    MOV      CX,ES        ; PUT ES IN REGISTER SUITABLE FOR
; ADDING TO
E9EC 03 C8    ADD      CX,BX        ; GET NEW VALUE FOR ES
E9EE 8E C1    MOV      ES,CX        ; UPDATE THE ES REGISTER
E9F0 5B      POP      BX          ; RECOVER ORIGINAL OFFSET
E9F1 81 E3 00F AND     BX,0000FH    ; NEW OFFSET
E9F5 88 F3    MOV      SI,BX        ; DS:SI POINT AT BUFFER
E9F7 88 F8    MOV      DI,BX        ; ES:DI POINT AT BUFFER
E9F9 59      POP      CX
E9FA C3      RET
E9FB          BOUND_SETUP      ENDP
;-----
; SEEK
; THIS ROUTINE WILL MOVE THE HEAD ON THE NAMED DRIVE
; TO THE NAMED TRACK. IF THE DRIVE HAS NOT BEEN ACCESSED
; SINCE THE DRIVE RESET COMMAND WAS ISSUED, THE DRIVE WILL BE
; RECALIBRATED.
; INPUT
; (DL) = DRIVE TO SEEK ON
; (CH) = TRACK TO SEEK TO
; OUTPUT
; CY = 0 SUCCESS
; CY = 1 FAILURE -- DISKETTE_STATUS SET ACCORDINGLY
; (AX) DESTROYED
;-----
E9FB          SEEK      PROC      NEAR
E9FB 88      PUSH      SI          ; SAVE REGISTER
E9FC 53      PUSH      BX          ; SAVE REGISTER
E9FD 51      PUSH      CX
E9FE 8E 0074 R MOV     SI,OFFSET TRACK0 ; BASE OF CURRENT HEAD POSITIONS
EA01 80 01    MOV     AL,1             ; ESTABLISH MASK FOR RECAL
EA03 8A CA    MOV     CL,DL           ; USE DRIVE AS A SHIFT COUNT
EA05 81 E1 00FF AND    CX,0FFH         ; MASK OFF HIGH BYTE
EA09 03 F1    ADD     SI,CX           ; POINT SI AT CORRECT DRIVE
EA0B 02 C0    ROL     AL,CL          ; GET MASK FOR DRIVE
;----- SI CONTAINS OFFSET FOR CORRECT DRIVE, AL CONTAINS BIT MASK
; IN POSITION 0,1 OR 2
EA00 5B      POP      CX          ; RESTORE PARAMETER REGISTER
EA0E 8B EA66 R MOV     BX,OFFSET J32   ; SET UP ERROR RECOVERY ADDRESS
EA11 53      PUSH      BX          ; NEEDED FOR ROUTINE NEC_OUTPUT
EA12 84 06 003E R TEST   SEEK_STATUS,AL   ; TEST DRIVE FOR RECAL
EA16 75 1B    JNZ     J2B             ; NO RECAL
EA18 08 06 003E R OR     SEEK_STATUS,AL   ; TURN ON THE NO RECAL BIT IN FLAG
EA1C 80 3C 00 CMP     BYTE PTR[SI],0  ; LAST REFERENCED TRACK=0?
EA1F 74 12    JZ      J2B             ; YES IGNORE RECAL
EA21 B4 07    MOV     AH,07H         ; RECALIBRATE COMMAND
EA23 EB E8BA R CALL   NEC_OUTPUT      ;
EA26 8A E2    MOV     AH,DL          ; RECAL REQUIRED ON DRIVE IN DL
EA28 EB E98A R CALL   NEC_OUTPUT      ; OUTPUT THE DRIVE NUMBER
;----- HEAD IS MOVING TO CORRECT TRACK
EA2B E2 39    CALL   CHK_STAT_2     ; GET THE STATUS OF RECALIBRATE
EA30 C8 04 00 JC     J32_2          ; SEEK_ERROR
;----- DRIVE IS IN SYNCH WITH CONTROLLER, SEEK TO TRACK
EA33 8A 04    MOV     AL,BYTE PTR[SI] ; GET THE PCN
EA35 2A C5    SUB     AL,CH          ; GET SEEK_WAIT VALUE
EA37 74 2C    JZ     J31_1          ; ALREADY ON CORRECT TRACK
EA39 B4 0F    MOV     AH,0FH        ; SEEK COMMAND TO NEC
EA3B EB E98A R CALL   NEC_OUTPUT      ;
EA3E 8A E2    MOV     AH,DL          ; DRIVE NUMBER
EA40 E9 E8BA R CALL   NEC_OUTPUT      ;
EA43 8A E5    MOV     AH,CH          ; TRACK NUMBER
EA45 EB E98A R CALL   NEC_OUTPUT      ;
EA48 EB EA6F R CALL   CHK_STAT_2     ; GET ENDING INTERRUPT AND SENSE
; STATUS
;----- WAIT FOR HEAD SETTLE
EA49 9C      PUSHF     ; SAVE STATUS FLAGS
EA4C 51      PUSH     CX          ; SAVE REGISTER
EA4D 83 12    MOV     BL,1B         ; HEAD SETTLE PARAMETER
EA4F EB E9B4 R CALL   GET_PARM       ;
EA52          J29:          MOV     CX,650        ; HEAD0_SETTLE
EA52 89 0226 MOV     OR     AH,AH    ; 1 MS LOOP
EA55 0A E4    OR     AH,AH         ; TEST FOR TIME EXPIRED
EA57 74 06    JZ     J30           ;
EA59 E2 FE    LOOP   J30          ; DELAY FOR 1 MS
EA5B FE CC    DEC     AH           ; DECREMENT THE COUNT
EA5D EB F3    JMP     J29          ; DO IT SOME MORE
EA5F 89      POP     CK          ; RESTORE REGISTER
EA60 90      POPF
EA61 72 06    JC     J32_2        ;
EA63 88 2C    MOV     BYTE PTR[SI],CH ;
EA65 58      POP     BX          ; GET RID OF DUMMY RETURN
EA66          J31:          POP     BX          ; SEEK_ERROR
EA66 5B      POP     SI          ; RESTORE REGISTER
EA67 5E      POP     DI          ; UPDATE CORRECT
EA68 C3      RET              ; RETURN TO CALLER
EA69 C8 04 FF MOV     BYTE PTR[SI],OFFH ; UNKNOWN STATUS ABOUT SEEK
; OPERATION
EA6C 5B      POP     BX          ; GET RID OF DUMMY RETURN
EA8D EB F7    JMP     SHORT J32
EA6F          SEEK      ENDP

```

```

-----
; CHK_STAT_2
; THIS ROUTINE HANDLES THE INTERRUPT RECEIVED AFTER
; A RECALIBRATE, SEEK, OR RESET TO THE ADAPTER.
; THE INTERRUPT IS WAITED FOR, THE INTERRUPT STATUS SENSED,
; AND THE RESULT RETURNED TO THE CALLER.
;
; INPUT
; NONE
;
; OUTPUT
; CY = 0 SUCCESS
; CY = 1 FAILURE -- ERROR IS IN DISKETTE_STATUS
; (AX) DESTROYED
-----
EABF          CHK_STAT_2  PROC   NEAR
EABF          PUSH   BX           ; SAVE REGISTERS
EAF0          PUSH   SI           ;
EA71          XOR    BX,BX        ; NUMBER OF SENSE INTERRUPTS TO
;                               ; ISSUE
EA73          MOV    SI,OFFSET J33_3 ; SET UP DUMMY RETURN FROM
;                               ; NEC_OUTPUT
EA76          PUSH   SI           ; PUT ON STACK
EA77          MOV    AH,0BH       ; SENSE INTERRUPT STATUS
EA79          CALL  NEC_OUTPUT    ; ISSUE SENSE INTERRUPT STATUS
EA7C          CALL  RESULTS      ;
EA7F          JC    J35          ; NEC TIME OUT, FLAGS SET IN
;                               ; RESULTS
EAB1          MOV    AL,NEC_STATUS ; GET STATUS
EAB4          TEST  AL,SEEK_END   ; IS SEEK OR RECAL OPERATION DONE?
EAB6          JNZ  J35_1         ; JUMP IF EXECUTION OF SEEK OR
;                               ; RECAL DONE
EAB8          J33_3: DEC  BX       ; DEC LOOP COUNTER
EAB9          JNZ  J33_2         ; DO ANOTHER LOOP
EAB8          OR   DISKETTE_STATUS,TIME_OUT ; RETURN
EAB9          J34:  STC          ; CALLER
EAB9          J35:  POP  SI       ; RESTORE REGISTERS
EAB9          POP  SI
EAB3          POP  BX
EAB4          RET
;-----SEEK END HAS OCCURED, CHECK FOR NORMAL TERMINATION
EAB5          J35_1: AND  AL,0COH  ; MASK NORMAL TERMINATION BITS
EAB7          JZ   J36           ; JUMP IF NORMAL TERMINATION
EAB9          OR   DISKETTE_STATUS,BAD_SEEK
EABE          JMP  J34
CHK_STAT_2   ENDP
-----
; RESULTS
; THIS ROUTINE WILL READ ANYTHING THAT THE NEC CONTROLLER
; HAS TO SAY FOLLOWING AN INTERRUPT.
; IT IS ASSUMED THAT THE NEC DATA PORT = NEC STATUS PORT + 1.
;
; INPUT
; NONE
;
; OUTPUT
; CY = 0 SUCCESSFUL TRANSFER
; CY = 1 FAILURE -- TIME OUT IN WAITING FOR STATUS
; NEC_STATUS AREA HAS STATUS BYTE LOADED INTO IT
; (AH) DESTROYED
-----
EAA0          RESULTS PROC   NEAR
EAA0          CLD
EAA1          MOV    DI,OFFSET NEC_STATUS ; POINTER TO DATA AREA
EAA4          PUSH  CX           ; SAVE COUNTER
EAA5          PUSH  DX           ;
EAA6          PUSH  BX           ;
EAA7          MOV    BL,7        ; MAX STATUS BYTES
;----- WAIT FOR REQUEST FOR MASTER
EAA9          J38:  XOR    CX,CX   ; INPUT_LOOP
EAA9          XOR    CX,CX       ; COUNTER
EAB0          MOV    DX,NEC_STAT ; STATUS PORT
EAB0          J39:  IN     AL,DX   ; WAIT FOR MASTER
EAB0          TEST  AL,080H     ; GET STATUS
EAB1          JNZ  J40A         ; MASTER READY
EAB3          LOOP J39          ; TEST_DIR
EAB5          OR   DISKETTE_STATUS,TIME_OUT ; WAIT_MASTER
EAB8          J40:  STC          ; RESULTS_ERROR
EAB8          STC          ; SET ERROR RETURN
;----- RESULT OPERATI0N IS DONE
EAB9          J44:  POP  BX       ;
EAB9          POP  DX           ;
EAB9          POP  CX           ;
EABE          RET
;----- TEST THE DIRECTION BIT
EABF          J40A: IN  AL,DX    ; GET STATUS REG AGAIN
EAC0          TEST  AL,040H    ; TEST DIRECTION BIT
EAC2          JNZ  J42         ; OK TO READ STATUS
EAC4          J41:  OR   DISKETTE_STATUS,BAD_NEC ; NEC_FAIL
EAC4          JMP  J40         ; RESULTS_ERROR
;----- READ IN THE STATUS
EACB          J42:  INC  DX       ; INPUT_STAT
EACB          INC  DX           ; POINT AT DATA PORT
EACC          IN   AL,DX       ; GET THE DATA
EACD          MOV  EDI,AL       ; STORE THE BYTE
EACF          INC  DI           ; INCREMENT THE POINTER
EAD0          MOV  CX,10        ; LOOP TO KILL TIME FOR NEC
EAD3          J43:  LOOP J43     ;
EAD5          DEC  DX           ; POINT AT STATUS PORT
EAD6          IN   AL,DX       ; GET STATUS
EAD7          TEST AL,010H     ; TEST FOR NEC STILL BUSY
EAD9          JZ   J44         ; RESULTS DONE
EADB          DEC  BL           ; DECREMENT THE STATUS COUNTER
EADD          JNZ  J3B         ; GO BACK FOR MORE
EADF          JMP  J41         ; CHIP HAS FAILED

```



```

;CLOCK_WAIT
; THIS PROCEDURE IS CALLED WHEN THE TIME OF DAY
; IS BEING UPDATED. IT WAITS IF TIMERO IS ALMOST
; READY TO WRAP UNTIL IT IS SAFE TO READ AN ACCURATE
; TIMER1.
; INPUT
; NONE.
; OUTPUT
; NONE. AX IS DESTROYED.
-----
CLOCK_WAIT PROC NEAR
XOR AL,AL ; READ MOOE TIMERO FOR 8253
OUT TIM_CTL,AL ; OUTPUT TO THE 8253
PUSH AX
POP AX ; WAIT FOR 8253 TO INITIALIZE
; ITSELF
EB37 E4 40 IN AL,TIMERO ; READ LEAST SIGNIFICANT BYTE
EB39 86 C4 XCHG AL,AH ; SAVE IT
EB3B E4 40 IN AL,TIMERO ; READ MOST SIGNIFICANT BYTE
EB3D 86 C4 XCHG AL,AH ; REARRANGE FOR PROPER ORDER
EB3F 3D 012C CMP AX,THRESHOLD ; IS TIMERO CLOSE TO WRAPPING?
EB42 72 ED JC CLOCK_WAIT ; JUMP IF CLOCK IS WITHIN THRESHOLD
EB44 C3 RET ; OK TO READ TIMER1
EB45 ENDP
-----
;GET_DRIVE
; THIS ROUTINE WILL CALCULATE A BIT MASK FOR THE DRIVE WHICH
; IS SELECTED BY THE CURRENT INT 13 CALL. THE DRIVE SELECTED
; CORRESPONDS TO THE BIT IN THE MASK, I.E. DRIVE ZERO
; CORRESPONDS TO BIT ZERO AND A 01H IS RETURNED. THE BIT IS
; CALCULATED BY ACCESSING THE PARAMETERS PASSED TO INT 13
; WHICH WERE SAVED ON THE STACK.
; INPUT
; BYTE PTR[BP] MUST POINT TO DRIVE FOR SELECTION.
; OUPUT
; AL CONTAINS THE BIT MASK. ALL OTHER REGISTERS ARE INTACT
-----
GET_DRIVE PROC NEAR
PUSH CX ; SAVE REGISTER.
MOV CL, BYTE PTR[BP] ; GET DRIVE NUMBER
MOV AL, 1 ; INITIALIZE AL WITH VALUE FOR
; SHIFTING
EB4B D2 E0 SHL AL,CL ; SHIFT BIT POSITION BY DRIVE
; NUMBER (DRIVE IN RANGE 0-2)
EB4D 24 07 AND AL,07H ; ONLY THREE DRIVES ARE SUPPORTED.
; RANGE CHECK
EB4F 59 POP CX ; RESTORE REGISTERS
EB50 C3 RET
EB51 ENDP
-----
; THIS ROUTINE CHECKS OPTIONAL ROM MODULES (CHECKSUM
; FOR MODULES FROM C0000->D0000, CRC CHECK FOR CARTRIDGES
; (D0000->F0000)
; IF CHECK IS OK, CALLS INIT/TEST CODE IN MODULE
; MFG ERROR CODE= 25XX (XX=MSB OF SEGMENT IN ERROR)
-----
ROM_CHECK PROC NEAR
SUB SI,SI ; SET SI TO POINT TO BEGINNING
; (REL. TO DS)
EB53 2A C0 SUB AL,AL ; ZERO OUT AL
EB55 8A 87 02 MOV AH,[BX+2] ; GET LENGTH INDICATOR
EB5B D1 E0 SHL AX,1 ; FORM COUNT
EB5A 50 PUSH AX ; SAVE COUNT
EB5B 81 FA D000 CMP DX,0D000H ; SEE IF POINTER IS BELOW D000
EB5F 9C PUSHF ; SAVE RESULTS
EB60 81 04 MOV CL,4 ; ADJUST
EB62 03 E8 SHR AX,CL
EB64 03 D0 ADD DX,AX ; SET POINTER TO NEXT MODULE
EB66 9D POPF ; RECOVER FLAGS FROM POINTER RANGE
; CHECK
EB67 59 POP CX ; RECOVER COUNT IN CX REGISTER
EB68 52 PUSH DX ; SAVE POINTER
EB69 7C 07 JL ROM_1 ; DO ARITHMETIC CHECKSUM IF BELOW
; D000
EB6B E8 FE71 R CALL CRC_CHECK ; DO CRC CHECK
EB6E 74 2B JZ ROM_CHECK_1 ; PROCDED IF OK
EB70 E8 05 JMP SHORT ROM_2 ; ELSE POST ERROR
ROM_1: CALL ROS_CHECKSUM ; DO ARITHMETIC CHECKSUM
; PROCDED IF OK
ROM_2: MOV DX,1626H ; POSITION CURSOR, ROW 22, COL 3B
EB77 BA 1626 MOV AH,2
EB7A 84 02 MOV BH,7
EB7E CD 10 INT 10H
EB80 8C DA MOV DX,DS ; RECOVER DATA SEG
EB82 8A C8 MOV AL,DH
EB84 E8 18A9 R CALL XPC_BYTE ; DISPLAY MSB OF DATA SEG
EB87 BA DE MOV BL,DH ; FORM XX VALUE OF ERROR CODE
EB89 87 26 MOV BH,25H ; FORM 25 PORTION
EB8B 80 FE D0 CMP DH,0D0H ; IN CARTRIDGE SPACE?
EB8E BE 003B R HDV SI,OFFSET CART_ERR
EB91 7D 03 JGE ROM_CHECK_0
EB93 BE 003A R HDV SI,OFFSET ROM_ERR
EB96 ROM_CHECK_0:
EB96 EB 0B8C R CALL E_MSG ; GO ERROR ROUTINE
EB99 EB 16 JMP SHORT ROM_CHECK_END ; AND EXIT
ROM_CHECK_1:
EB99 88 ---- R MOV AX,XXDATA ; SET ES TO POINT TO XXDATA AREA
EB9E BE C0 MOV ES,AX
EBA0 26: C7 06 0014 R 0003 MOV ES:10_ROM_INIT,0003H ; LOAD OFFSET
EBA7 26: 8C 1E 0016 R 0003 MOV ES:10_ROM_SEG,DS ; LOAD SEGMENT
EBAC 26: FF 1E 0014 R CALL DWORD_PTR ES:10_ROM_INIT ; CALL INIT./TEST ROUTINE

```

```

EBB1
EBB1 5A
EBB2 C3
EBB3

ROM_CHECK_END:
POP DX ; RECOVER POINTER
RET ; RETURN TO CALLER

ROM_CHECK ENDP
-----
-- INT 13
; DISKETTE 1/0
; THIS INTERFACE PROVIDES ACCESS TO THE 5 1/4" DISKETTE DRIVES
; INPUT
(AH)=0 RESET DISKETTE SYSTEM
; HARD RESET TO NEC, PREPARE COMMAND, RECAL REQD ON
; ALL DRIVES
(AH)=1 READ THE STATUS OF THE SYSTEM INTO (AL)
; DISKETTE_STATUS FROM LAST OP'N IS USED
; REGISTERS FOR READ/WRITE/VERIFY/FORMAT
(DL) - DRIVE NUMBER (0-3 ALLOWED, VALUE CHECKED)
(DH) - HEAD NUMBER (0-1 ALLOWED, NOT VALUE CHECKED)
(CH) - TRACK NUMBER (0-39, NOT VALUE CHECKED)
(CL) - SECTOR NUMBER (1-8, NOT VALUE CHECKED, NOT USED FOR
; FORMAT)
(AL) - NUMBER OF SECTORS ( MAX = 8, NOT VALUE CHECKED, NOT
; USED FOR FORMAT, HOWEVER, CANNOT BE ZERO!!!)
(ES:BX) - ADDRESS OF BUFFER ( NOT REQUIRED FOR VERIFY)
;
(AH)=2 READ THE DESIRED SECTORS INTO MEMORY
(AH)=3 WRITE THE DESIRED SECTORS FROM MEMORY
(AH)=4 VERIFY THE DESIRED SECTORS
(AH)=5 FORMAT THE DESIRED TRACK
; FOR THE FORMAT OPERATION, THE BUFFER POINTER
(ES, BX) MUST POINT TO THE COLLECTION OF DESIRED
; ADDRESS FIELDS FOR THE TRACK. EACH FIELD IS
; COMPOSED OF 4 BYTES, (C,H,R,N), WHERE
; C = TRACK NUMBER, H=HEAD NUMBER, R = SECTOR NUMBER,
; N= NUMBER OF BYTES PER SECTOR (00=128, 01=256,
; 02=512, 03=1024, ). THERE MUST BE ONE ENTRY FOR
; EVERY SECTOR ON THE TRACK. THIS INFORMATION IS USED
; TO FIND THE REQUESTED SECTOR DURING READ/WRITE
; ACCESS.
; DATA VARIABLE -- DISK_POINTER
; DOUBLE WORD POINTER TO THE CURRENT SET OF DISKETTE PARAMETERS
; OUTPUT
AH = STATUS OF OPERATION
; STATUS BITS ARE DEFINED IN THE EQUATES FOR
; DISKETTE_STATUS VARIABLE IN THE DATA SEGMENT OF
; THIS MODULE
CY = 0 SUCCESSFUL OPERATION (AH=0 ON RETURN)
CY = 1 FAILED OPERATION (AH HAS ERROR REASON)
FOR READ/WRITE/VERIFY
DS, BX, DX, CH, CL PRESERVED
AL = NUMBER OF SECTORS ACTUALLY READ
**** AL MAY NOT BE CORRECT IF TIME OUT ERROR OCCURS
NOTE: IF AN ERROR IS REPORTED BY THE DISKETTE CODE, THE
; APPROPRIATE ACTION IS TO RESET THE DISKETTE, THEN
; RETRY THE OPERATION. ON READ ACCESSES, NO MOTOR
; START DELAY IS TAKEN, SO THAT THREE RETRIES ARE
; REQUIRED ON READS TO ENSURE THAT THE PROBLEM IS NOT
; DUE TO MOTOR START-UP.
-----
ASSUME CS:CODE, DS:DATA, ES:DATA
ORG OEC99H
DISKETTE_10 PROC FAR
; INTERRUPTS BACK ON
STI ; SAVE ES
PUSH ES ; ALLOCATE ONE WORD OF STORAGE FOR
PUSH AX ; TIMER1 INITIAL VALUE
; ALLOCATE ONE WORD ON STACK FOR
; USE IN PROC5 ENABLE AND DISABLE.
; WILL HOLD 8259 MASK.
EC59 80 PUSH AX ; SAVE COMMAND AND N_SECTORS
EC5E 53 PUSH BX ; SAVE ADDRESS
ECBF 51 PUSH CX
EC60 1E PUSH DS ; SAVE SEGMENT REGISTER VALUE
EC61 58 PUSH SI ; SAVE ALL REGISTERS DURING
; OPERATION
EC62 57 PUSH DI
EC63 55 PUSH BP
EC64 52 PUSH DX
EC65 8B EC MOV BP, BP ; SET UP POINTER TO HEAD PARM
EC67 EB 1388 R CALL DDS ; SET DS=DATA
EC6A EB EC90 R CALL J1 ; CALL THE REST TO ENSURE DS
; RESTORED
; GET THE MOTOR WAIT PARAMETER
EC6D 83 04 MOV BL, 4
EC6F EB E984 R CALL GET_PARM
EC72 8B 26 0040 R MOV MOTOR_COUNT, AH ; SET THE TIMER COUNT FOR THE MOTOR
EC76 8A 26 0041 R MOV AH, DISKETTE_STATUS ; GET STATUS OF OPERATION
EC7A 8B 66 0F MOV [BP+1B], AH ; RETURN STATUS IN AL
EC7D 5A POP DX ; RESTORE ALL REGISTERS
EC7E 5D POP BP
EC7F 5F POP DI
EC80 5E POP SI
EC81 1F POP DS
EC82 59 POP CX
EC83 5B POP BX ; RECOVER OFFSET
EC84 59 POP AX
EC85 83 C4 04 ADD SP, 4 ; DISCARD DUMMY SPACE FOR 8259 MASK
EC88 07 POP ES ; RECOVER SEGMENT
EC89 80 FC 01 CMP AH, 1 ; SET THE CARRY FLAG TO INDICATE
EC8C F5 CNC ; SUCCESS OR FAILURE
EC8D CA 0002 RET 2 ; THROW AWAY SAVED FLAGS

```

```

EC90
EC90
EC90 8A F0
EC92 80 26 003F R 7F
EC97 0A E4
EC98 74 27
EC98 FE CC
EC9D 74 74
EC9F C6 06 0041 R 00
ECA4 80 FA 02
ECA7 77 13
ECA9 FE CC
ECAB 74 6D
ECAD FE CC
CAF 75 03
JB1 E9 ED3D R
CB4
ECB4 FE CC
ECB6 74 82
ECB8 FE CC
ECBA 74 82
ECBC
ECBC C6 06 0041 R 01
ECC1 C3
ECC2
ECC2 8A 00F2
ECC5 FA
ECC6 A0 003F R
ECC9 24 07
ECCB EE
ECCC C6 06 003E R 00
ECD1 C6 06 0041 R 00
ECD6 0C 80
ECD8 EE
ECD9 FB
ECDA BE ECFA R
ECDD 56
ECEE B9 0010
ECE1 B4 0B
ECE3 EB E98A R
ECE6 EB EAA0 R
ECE9 A0 0042 R
CEC 3C C0
CEE 74 12
CFO E2 EF
CF2 80 0E 0041 R 20
ECF7 9E
ECF8 EB 1B
ECFA BE ECFA R
ECFD 56
ECFE E2 E1
ED00 EB F0
ED02 8E
ED03 B4 03
ED05 EB E98A R
ED08 B3 01
ED0A EB E984 R
ED0D B3 03
ED0F EB E984 R
ED 12
ED 12 C3
ED 13
ED 13
ED 13 A0 0041 R
ED 16 B8 46 0E
ED 19 C3
ED 1A
ED 1A
ED 1A
ED 1A
ED 1A B4 46
ED 1C EB 26
D 1E
D 1E
D 1E
ED 1E 80 0E 003F R 80
ED 23 B4 4D
ED 25 EB 1D
DISKETTE_10
J1 PROC
MOV DH,AL ; SAVE # SECTORS IN DH
AND MOTOR_STATUS,07FH ; INDICATE A READ OPERATION
OR AH,AH ; AH=0
JZ DISK_RESET
DEC AH ; AH=1
JZ DISK_STATUS
MOV DISKETTE_STATUS,0 ; RESET THE STATUS INDICATOR
CMP DL,2 ; TEST FOR DRIVE IN 0-2 RANGE
JA J3 ; ERROR IF ABOVE
DEC AH ; AH=2
JZ DISK_READ
DEC AH ; AH=3
JNZ J2 ; TEST_DISK_VERF
JMP DISK_WRITE
J2: ; TEST_DISK_VERF
DEC AH ; AH=4
JZ DISK_VERF
DEC AH ; AH=5
JZ DISK_FORMAT
J3: ; BAO_COMMAND
MOV DISKETTE_STATUS,BAD_CMD ; ERROR CODE, NO SECTORS
; TRANSFERRED
; UNDEFINED OPERATION
RET
J1 ENDP
;----- RESET THE DISKETTE SYSTEM
DISK_RESET PROC NEAR
MOV DK,NEC_CTL ; ADAPTER CONTROL PORT
CLI ; NO INTERRUPTS
MOV AL,MOTOR_STATUS ; FIND OUT IF MOTOR IS RUNNING
AND AL,07H ; DRIVE BITS
OUT DX,AL ; RESET THE ADAPTER
MOV SEEK_STATUS,0 ; SET RECAL REQUIRED ON ALL DRIVES
MOV DISKETTE_STATUS,0 ; SET OK STATUS FOR DISKETTE
OR AL,FDC_RESET ; TURN OFF RESET
OUT DX,AL ; TURN OFF THE RESET
STI ; REENABLE THE INTERRUPTS
MOV SI,OFFSET J4_2 ; DUMMY RETURN FOR
PUSH SI ; PUSH RETURN IF ERROR
; IN NEC_OUTPUT
MOV CX,10H ; NUMBER OF SENSE INTERRUPTS TO
; ISSUE
J4_0: MOV AH,09H ; COMMAND FOR SENSE INTERRUPT
; STATUS
CALL NEC_OUTPUT ; OUTPUT THE SENSE INTERRUPT
; STATUS
CALL RESULTS ; GET STATUS FOLLOWING COMPLETION
OF RESET
MOV AL,NEC_STATUS ; IGNORE ERROR RETURN AND DO OWN
TEST ; TEST
CMP AL,0COH ; TEST FOR DRIVE READY TRANSITION
JZ J7 ; EVERYTHING OK
LOOP J4_0 ; RETRY THE COMMAND
J4_1: OR DISKETTE_STATUS,BAD_NEC ; SET ERROR CODE
SI
POP SHORT JB
J4_2: MOV SI,OFFSET J4_2 ; NEC_OUTPUT FAILED, RETRY THE
SENSE INTERRUPT
PUSH SI ; OFFSET OF BAD RETURN IN
NEC_OUTPUT
LOOP J4_0 ; RETRY
JMP SHORT J4_1
;----- SEND SPECIFY COMMAND TO NEC
J7: POP SI ; GET RID OF DUMMY ARGUMENT
MOV AH,03H ; SPECIFY COMMAND
CALL NEC_OUTPUT ; OUTPUT THE COMMAND
MOV BL,1 ; STEP RATE TIME AND HEAD UNLOAD
CALL GET_PARM ; OUTPUT TO THE NEC CONTROLLER
MOV BL,3 ; PARM1 HEAD LOAD AND NO DMA
CALL GET_PARM ; TO THE NEC CONTROLLER
JB: ; RESET_RET
RET ; RETURN TO CALLER
DISK_RESET ENDP
;----- DISKETTE STATUS ROUTINE
DISK_STATUS PROC NEAR
MOV AL,DISKETTE_STATUS
MOV BYTE PTR[B*+143],AL ; PUT STATUS ON STACK, IT WILL
; POP IN AL
RET
DISK_STATUS ENDP
;----- DISKETTE VERIFY
DISK_VERF LABEL NEAR
;----- DISKETTE READ
DISK_READ PROC NEAR
J9: MOV AH,04EH ; DISK_READ_CNTL
; SET UP READ COMMAND FOR NEC
; CONTROLLER
JMP SHORT RW_OPN ; GO DO THE OPERATION
DISK_READ ENDP
;----- DISKETTE FORMAT
DISK_FORMAT PROC NEAR
OR MOTOR_STATUS,80H ; INDICATE A WRITE OPERATION
MOV AH,04DH ; ESTABLISH THE FORMAT COMMAND
JMP SHORT RW_OPN ; DO THE OPERATION

```

```

ED27          J10:          MOV     BL,7          ; CONTINUATION OF RW_OPN FOR FMT
ED27 83 07          CALL   GET_PARM        ; GET THE
ED29 EB E9B4 R     CALL   GET_PARM        ; BYTES/SECTOR VALUE TO NEC
ED2C 83 09          MOV     BL,9          ; GET THE
ED2E EB E9B4 R     CALL   GET_PARM        ; SECTORS/TRACK VALUE TO NEC
ED31 83 0F          MOV     BL,15         ; GET THE
ED33 EB E9B4 R     CALL   GET_PARM        ; GAP LENGTH VALUE TO NEC
ED36 8B 0011       MOV     BX,17         ; GET THE FILLER BYTE
ED39 53           PUSH    BX            ; SAVE PARAMETER INDEX ON STACK
ED3A E9 EDCD R     JMP     J16           ; TO THE CONTROLLER
ED3D

DISK_FORMAT      ENDP
;----- DISKETTE WRITE ROUTINE
ED3D          DISK_WRITE  PROC     NEAR
ED3D 8D 0E 003F R B0 OR     MOTOR_STATUS,BOH ; INDICATE A WRITE OPERATION
ED42 B4 45          MOV     AH,045H       ; NEC COMMAND TO WRITE TO DISKETTE
ED44          DISK_WRITE  ENDP
;----- ALLOW WRITE ROUTINE TO FALL INTO RW_OPN
;-----
; RW_OPN
;----- THIS ROUTINE PERFORMS THE READ/WRITE/VERIFY OPERATION
RW_OPN         PROC     NEAR
ED44 50           PUSH    AX            ; SAVE THE COMMAND
;----- TURN ON THE MOTOR AND SELECT THE DRIVE
ED45 51           PUSH    CX            ; SAVE THE T/S PARMS
ED46 FA          CLI             ; NO INTERRUPTS WHILE DETERMINING
;----- MOTOR STATUS
ED47 C6 06 0040 R FF MOV     MOTOR_COUNT,OFFH ; SET LARGE COUNT DURING OPERATION
ED4C EB EB45 R     CALL   GET_DRIVE      ; GET THE DRIVE PARAMETER FROM THE
;----- STACK
ED4F 84 06 003F R TEST   MOTOR_STATUS,AL ; TEST MOTOR FOR OPERATING
ED53 75 1F          JNZ     J14           ; IF RUNNING, SKIP THE WAIT
ED55 00 26 003F R F0 AND     MOTOR_STATUS,OF0H ; TURN OFF RUNNING DRIVE
ED5A 08 06 003F R OR     MOTOR_STATUS,AL ; TURN ON THE CURRENT MOTOR
ED5E FB          STI             ; INTERRUPTS BACK ON
ED5F 0C 80         OR     AL,FDC_RESET  ; NO RESET. TURN ON MOTOR
ED61 E6 F2         OUT     NEC_CTL,AL
;----- WAIT FOR MOTOR BOTH READ AND WRITE
ED63 83 14         MOV     BL,20         ; GET MOTOR START TIME
ED65 EB E9B4 R     CALL   GET_PARM        ;
ED68 0A E4         OR     AH,AH          ;
ED6A          J12:          JZ     J14           ; TEST WAIT TIME
ED6A 74 08         JZ     J14           ; EXIT WITH TIME EXPIRED
ED6C 2B C9         SUB     CX,CX         ; SET UP 1/8 SECOND LOOP TIME
ED6E E2 FE         LOOP    J13          ; WAIT FOR THE REQUIRED TIME
ED70 FE CC         DEC     AH            ; DECREMENT TIME VALUE
ED72 EB FC         JMP     J12          ; ARE WE DONE YET
ED74          J14:          STI             ; MOTOR_RUNNING
ED74 FB          ; INTERRUPTS BACK ON FOR BYPASS
ED75 59           POP     CX            ; WAIT
;----- DO THE SEEK OPERATION
ED76 EB E9FB R     CALL   SEEK           ; MOVE TO CORRECT TRACK
ED79 58           POP     AX            ; RECOVER COMMAND
ED7A 8A FC         MOV     BH,AH         ; SAVE COMMAND IN BH
ED7C B6 00         MOV     DH,0          ; SET NO SECTORS READ IN CASE OF
;----- ERROR
ED7E 73 03         JNC     J14_1         ; IF NO ERROR CONTINUE, JUMP AROUND
ED80          J14_1:       JMP     J17           ; JMP
ED80 E9 EED7 R     J14_1:       MOV     SI,OFFSET J17 ; CARRY SET JUMP TO MOTOR WAIT
ED83 BE EED7 R     ; DUMMY RETURN ON STACK FOR
ED86 56           PUSH    SI            ; NEC_OUTPUT
;----- SO THAT IT WILL RETURN TO MOTOR
;----- OFF LOCATION
;----- SEND OUT THE PARAMETERS TO THE CONTROLLER
ED87 EB E98A R     CALL   NEC_OUTPUT     ; OUTPUT THE OPERATION COMMAND
ED8A 8A 86 01       MOV     AH,[BP+1]     ; GET THE CURRENT HEAD NUMBER
ED8D 00 E4         SAL     AH,1          ; MOVE IT TO BIT 2
ED8F 00 E4         SAL     AH,1          ;
ED91 80 E4 04       AND     AH,4          ; ISOLATE THAT BIT
ED94 0A E2         OR     AH,DL          ; OR IN THE DRIVE NUMBER
ED96 EB E98A R     CALL   NEC_OUTPUT     ;
;----- TEST FOR FORMAT COMMAND
ED99 80 FF 4D       CMP     BH,04DH       ; IS THIS A FORMAT OPERATION?
ED9C 75 02         JNE     J15          ; NO. CONTINUE WITH R/W/V
ED9E EB 87         JMP     J10          ; IF SO, HANDLE SPECIAL
EDA0 8A E5         MOV     AH,CH         ; CYLINDER NUMBER
EDA2 EB E98A R     CALL   NEC_OUTPUT     ;
EDA5 8A 68 01       MOV     AH,[BP+1]     ; HEAD NUMBER FROM STACK
EDA8 EB E98A R     CALL   NEC_OUTPUT     ;
EDAB 8A E1         MOV     AH,CL         ; SECTOR NUMBER
EDA0 EB E98A R     CALL   NEC_OUTPUT     ;
ED80 B3 07         MOV     BL,7          ; BYTES/SECTOR PARM FROM BLOCK
ED82 EB E9B4 R     CALL   GET_PARM        ; TO THE NEC
ED85 83 08         MOV     BL,8          ; EOT PARM FROM BLOCK
ED87 EB E9B4 R     CALL   GET_PARM        ; RETURNED IN AH
ED8A 02 4E 0E       ADD     CL,[BP+14]    ; ADD CURRENT SECTOR TO NUMBER IN
;----- TRANSFER
ED8D FE C9         DEC     CL            ; CURRENT_SECTOR + N_SECTORS - 1
ED8F 8A E1         MOV     AH,CL         ; EOT PARAMETER IS THE CALCULATED
;----- ONE
EDC1 E6 E98A R     CALL   NEC_OUTPUT     ;
EDC4 B3 08         MOV     BL,11         ; GAP_LENGTH PARM FROM BLOCK
EDC6 EB E9B4 R     CALL   GET_PARM        ; TO THE NEC
EDC9 BB 00DD       MOV     BX,13         ; DTL PARM FROM BLOCK
EDCC 53           PUSH    BX            ; SAVE INDEX TO DISK PARAMETER ON
;----- STACK

```



```

E0DC FC          J16: CLD          ; FORWARD DIRECTION
E0DE B0 70      ;----- START TIMER1 WITH INITIAL VALUE OF FFFF
                MOV     AL,01110000B ; SELECT TIMER1,LSB-MSB, MODE 0,
                ; BINARY COUNTER
E0DD E6 43      OUT     TIM_CTL,AL ; INITIALIZE THE COUNTER
E0D2 50         PUSH   AX
E0D3 58         POP    AX          ; ALLOW ENOUGH TIME FOR THE 8253 TO
                ; INITIALIZE ITSELF
E0D4 B0 FF      MOV     AL,OFFH ; INITIAL COUNT VALUE FOR THE 8253
E0D6 E6 41      OUT     TIMER+1,AL ; OUTPUT LEAST SIGNIFICANT BYTE
E0D8 50         PUSH   AX
E0D9 58         POP    AX          ; WAIT
E0DA E6 41      OUT     TIMER+1,AL ; OUTPUT MOST SIGNIFICANT BYTE
                ;----- INITIALIZE CX FOR JUMP AFTER LAST PARAMETER IS PASSED TO NEC
E0DC BA 46 OF   MOV     AL,(8B*15) ; RETRIEVE COMMAND PARAMETER
E0DF A8 01      TEST    AL,01H ; IS THIS AN ODD NUMBERED FUNCTION?
E0E1 74 05      JZ     J16_1 ; JUMP IF NOT ODD NUMBERED
E0E3 B9 EE4E R  MOV     CX,OFFSET WRITE_LOOP
E0E6 EB 0C      JMP    SHORT J16_3
E0E8 3C 02      J16_1: CMP    AL,2 ; IS THIS A READ?
E0EA 75 05      JNZ    J16_2 ; JUMP IF VERIFY
E0EC B9 EE3A R  MOV     CX,OFFSET READ_LOOP
E0EF EB 03      JMP    SHORT J16_3
E0F1 B9 EE20 R  J16_2: MOV     CX,OFFSET VERIFY_LOOP
E0F4          ;----- FINISH INITIALIZATION
                J16_3:
                ;-----
                ;*****
                ; ALL INTERRUPTS ARE ABOUT TO BE DISABLED. THERE IS A POTENTIAL
                ; THAT THIS TIME PERIOD WILL BE LONG ENOUGH TO MISS TIME OF
                ; DAY INTERRUPTS. FOR THIS REASON, TIMER1 WILL BE USED TO
                ; KEEP TRACK OF THE NUMBER OF TIME OF DAY INTERRUPTS WHICH
                ; WILL BE MISSED. THIS INFORMATION IS USED AFTER THE DISKETTE
                ; OPERATION TO UPDATE THE TIME OF DAY.
                ;-----
E0F4 B0 10      MOV     AL,10H ; DISABLE NMI
E0F6 E6 A0      OUT     NMI_PORT,AL ; NO KEYBOARD INTERRUPT
E0FB EB EB31 R  CALL   CLOCK_WAIT ; WAIT IF TIMER0 IS ABOUT TO
                ; INTERRUPT
                ;-----
                ;----- ENABLE WATCHDOG TIMER
                ;-----
                ;*****
                ; GIVEN THE CURRENT SYSTEM CONFIGURATION A METHOD IS NEEDED
                ; TO PULL THE NEC OUT OF "FATAL ERROR" SITUATIONS. A TIMER
                ; ON THE ADAPTER CARD IS PROVIDED WHICH WILL PERFORM THIS
                ; FUNCTION. THE WATCHDOG TIMER ON THE ADAPTER CARD IS ENABLED
                ; AND STROBED BEFORE THE 8259 INTERRUPT 6 LINE IS ENABLED.
                ; THIS IS BECAUSE OF A GLITCH ON THE LINE LARGE ENOUGH TO
                ; TRIGGER AN INTERRUPT.
                ;-----
E0FB EB EB45 R  CALL   GET_DRIVE ; GET BIT MASK FOR DRIVE
E0FE BA 00F2   MOV     DX,NEC_CTL ; CONTROL PORT TO NEC
E0E1 0C E0      OR     AL,FDC_RESET+WD_ENABLE+WD_STROBE
E0E3 EE        OUT     DX,AL ; OUTPUT CONTROL INFO FOR
                ; WATCHDOG(WD) ENABLE
E0E4 24 A7      AND     AL,FDC_RESET+WD_ENABLE+7H
E0E6 EE        OUT     DX,AL ; OUTPUT CONTROL INFO TO STROBE
                ; WATCHDOG
E0E7 BA 00F4   MOV     DX,NEC_STAT ; PORT TO NEC STATUS
E0EA 80 20      MOV     AL,20H ; SELECT TIMER1 INPUT FROM TIMERO
                ; OUTPUT
E0EC E6 A0      OUT     NMI_PORT,AL
                ;----- READ TIMER1 NOW AND SAVE THE INITIAL VALUE
E0EE EB EB1A R  CALL   READ_TIME ; GET TIMER1 VALUE
E0E1 89 46 12   MOV     [BP+18],AX ; SAVE INITIAL VALUE FOR CLOCK
                ; UPDATE IN TEMPORARY STORAGE
E0E4 EB EAF8 R  CALL   DISABLE ; DISABLE ALL INTERRUPTS
                ;----- NEC BEGINS OPERATION WHEN NEC RECEIVES LAST PARAMETER
E0E7 58        POP    BX ; GET PARAMTER FROM STACK
E0E18 EB E9B4 R CALL   GET_PARAM ; OUTPUT LAST PARAMETER TO THE NEC
E0E1B 58        POP    AX ; CAN NOW DISCARD THAT DUMMY RETURN
                ; ADDRESS
E0E1C 06        PUSH   ES ;
E0E1D 1F        POP    DS ; INITIALIZE DS FOR WRITE
E0E1E FF E1     JMP    CX ; JUMP TO APPROPRIATE R/W/V LOOP
                ;-----
                ;*****
                ; DATA IS TRANSFERRED USING POLLING ALGORITHMS. THESE LOOPS
                ; TRANSFER A DATA BYTE AT A TIME WHILE POLLING THE NEC FOR
                ; NEXT DATA BYTE AND COMPLETION STATUS.
                ;-----
                ;-----VERIFY OPERATION
                ;-----VERIFY_LOOP:
E0E20 EC        IN     AL,DX ; READ STATUS
E0E21 A8 20     TEST    AL,BUSY_BIT ; HAS NEC ENTERED EXECUTION PHASE
                ; YET?
E0E23 74 FB     JZ     VERIFY_LOOP ; NO, CONTINUE SAMPLING
E0E25          J22_2:
E0E28 A8 80     TEST    AL,RQM ; IS DATA READY?
E0E27 75 07     JNZ    J22_4 ; JUMP IF DATA TRANSFER IS READY
E0E29 EC        IN     AL,DX ; READ STATUS PORT
E0E2A A8 20     TEST    AL,BUSY_BIT ; ARE WE DONE?
E0E2C 75 F7     JNZ    J22_2 ; JUMP IF MORE TRANSFERS
E0E2E EB 35     JMP    SHORT OP_END ; TRANSFER DONE
E0E30 42        J22_4: INC    DX ; POINT AT NEC DATA REGISTER
E0E31 EC        IN     AL,DX ; READ DATA
E0E32 4A        DEC    DX ; POINT AT NEC STATUS REGISTER
E0E33 EC        IN     AL,DX ; READ STATUS PORT
E0E34 A8 20     TEST    AL,BUSY_BIT ; ARE WE DONE?
E0E36 75 ED     JNZ    J22_2 ; CONTINUE
E0E38 EB 28     JMP    SHORT OP_END ; WE ARE DONE

```

```

;-----READ OPERATION
EE3A          READ_LOOP:
EE3A          IN      AL,DX          ; READ STATUS REGISTER
EE3B          EC      AL,BUSY_BIT   ; HAS NEC STARTED THE EXECUTION
EE3B          AB 20          TEST          ; PHASE?
EE3D          74 FB          JZ      READ_LOOP          ; HAS NOT STARTED YET
EE3F          EC      IN      AL,DX          ; READ STATUS PORT
EE40          AB 20          TEST          ; HAS NEC COMPLETED EXECUTION
EE40          ;              ; PHASE?
EE42          74 21          JZ      OP_END          ; JUMP IF EXECUTION PHASE IS OVER
EE44          AB 80          TEST          ; IS DATA READY?
EE46          74 F7          JZ      J22_5          ; READ THE DATA
EE48          42          INC      DX          ; POINT AT NEC_DATA
EE49          EC      IN      AL,DX          ; READ DATA
EE4A          AA          STOSB          ; TRANSFER DATA
EE4B          4A          DEC      DX          ; POINT AT NEC_STATUS
EE4C          EB F1          JMP     J22_5          ; CONTINUE WITH READ OPERATION

;-----WRITE AND FORMAT OPERATION
EE4E          WRITE_LOOP:
EE4E          EC      IN      AL,DX          ; READ NEC STATUS PORT
EE4F          AB 20          TEST          ; HAS THE NEC ENTERED EXECUTION
EE4F          ;              ; PHASE YET?
EE51          74 FB          JZ      WRITE_LOOP          ; NO, CONTINUE LOOPING
EE53          B9 2080          MOV     CX,BUSY_BIT*256+RGM
EE56          EC      IN      AL,DX          ; READ STATUS PORT
EE57          B4 C5          TEST          ; IS THE FEC STILL IN THE EXECUTION
EE57          ;              ; PHASE?
EE59          74 0A          JZ      OP_END          ; JUMP IF EXECUTION PHASE IS DONE.
EE5B          B4 C1          TEST          ; IS THE DATA PORT READY FOR THE
EE5B          ;              ; TRANSFER?
EE5D          74 F7          JZ      J22_7          ; JUMP TO WRITE DATA
EE5F          42          INC      DX          ; POINT AT DATA REGISTER
EE60          AC          LODSB          ; TRANSFER BYTE
EE61          EE          OUT     DX,AL          ; WRITE THE BYTE ON THE DISKETTE
EE62          4A          DEC      DX          ; POINT AT THE STATUS REGISTER
EE63          EB F1          JMP     J22_7          ; CONTINUE WITH WRITE OR FORMAT

;-----TRANSFER PROCESS IS OVER
EE65          9C          OP_END: PUSHF          ; SAVE THE CARRY BIT SET IN
EE65          ;              ; DISK_INT
EE66          EB EB45 R          CALL    GET_DRIVE          ; GET BIT MASK FOR DRIVE SELECTION
EE69          OC 80          OR     AL,FDC_RESET          ; NO RESET, KEEP DRIVE SPINNING
EE6B          BA 00F2          MOV     DX,NEC_CTL          ;
EE6E          EE          OUT     DX,AL          ; DISABLE WATCHDOG

;-----UPDATE TIME OF DAY
EE6F          E9 138B R          CALL    DDS          ; POINT DS AT BIOS DATA SEGMENT
EE72          E8 B31 R          CALL    CLOCK_WAIT          ; WAIT IF TIMER0 IS CLOSE TO
EE72          ;              ; WRAPPING
EE75          E8 EB1A R          CALL    READ_TIME          ; GET THE INITIAL VALUE OF TIMER1
EE78          8B 8E 12          MOV     BX,[BP+1B3]          ; UPDATE NUMBER OF INTERRUPTS
EE78          2B C3          SUB     AX,BX          ; MISSED
EE78          ;              ; PUT IT IN AX
EE7D          F7 DB          NEG     AX          ; SAVE IT FOR REUSE IN ISSUING USER
EE7F          50          PUSH   AX          ; TIMER INTERRUPTS
EE80          01 06 006C R          ADD     TIMER_LOW,AX          ; ADD NUMBER OF TIMER INTERRUPTS TO
EE80          ;              ; TIME
EE84          73 04          JNC    J16_4          ; JUMP IF TIMER_LOW DID NOT SPILL
EE84          ;              ; OVER TO TIMER_HI
EE86          FF 06 006E R          INC     TIMER_HIGH          ;
EE8A          83 3E 006E R 1B          CMP     TIMER_HIGH,01BH          ; TEST FOR COUNT TOTALING 24 HOURS
EE8F          75 19          JNZ    J16_5          ; JUMP IF NOT 24 HOURS
EE91          81 3E 0D6C R 0080          CMP     TIMER_LOW,0B0H          ; LOW VALUE = 24 HOUR VALUE?
EE97          7C 11          JL     J16_5          ; NOT 24 HOUR VALUE?

;-----TIMER HAS GONE 24 HOURS
EE99          C7 06 006E R 0000          MOV     TIMER_HIGH,0          ; ZERO OUT TIMER_HIGH VALUE
EE9F          B1 2E 006C R 0080          SUB     TIMER_LOW,0B0H          ; VALUE REFLECTS CORRECT TICKS PAST
EE9F          ;              ; 0B0H
EEA5          C6 06 0070 R 01          MOV     TIMER_OFL,1          ; INDICATES 24 HOUR THRESHOLD
EEAA          EB EB08 R          J16_5: CALL   ENABLE          ; ENABLE ALL INTERRUPTS
EEAD          59          POP     CX          ; CX:AX, COUNT FOR NUMBER OF USER
EEAE          E3 26          JXCZ   J16_7          ; TIME INTERRUPTS
EEAE          ;              ; IF ZERO DO NOT ISSUE ANY
EEB0          1E          PUSH   DS          ; INTERRUPTS
EEB0          ;              ; SAVE ALL REGISTERS SAVED PRIOR TO
EEB1          50          PUSH   AX          ; INT IC CALL FROM TIMERINT
EEB1          ;              ; THIS PROVIDES A COMPATIBLE
EEB2          52          PUSH   DX          ; INTERFACE TO IC
EEB3          ;              ;
EEB3          J16_6: INT     ICH          ; TRANSFER CONTROL TO USER
EEB3          ;              ; INTERRUPT
EEB5          E2 FC          LOOP   J16_6          ; DO ALL USER TIMER INTERRUPTS
EEB7          5A          POP     DX          ;
EEB8          5B          POP     AX          ;
EEB9          1F          POP     DS          ; RESTORE REGISTERS

;-----CLOCK IS UPDATED AND USER INTERRUPTS IC HAVE BEEN ISSUED.
EEBA          0A C0          ; CHECK IF KEYSTROKE OCCURED
EEBC          74 1B          OR     AL,AL          ; AL WAS SET DURING CALL TO ENABLE
EEBC          ;              ; NO KEY WAS PRESSED WHILE SYSTEM
EEBE          8B 0080          MOV     BX,0B0H          ; WAS MASKED
EEC1          B9 0048          MOV     CX,048H          ; DURATION OF TONE
EEC4          E8 E035 R          CALL   KB_NOISE          ; FREQUENCY OF TONE
EEC4          ;              ; NOTIFY USER OF MISSED KEYBOARD
EEC4          ;              ; INPUT

```

```

;-----CLEAR SHIFT STATES DONT LEAVE POSSIBLY OF DANGLING STATES
; OF MISSED BREAKS
ECC7 80 26 0017 R FO      AND      KB_FLAG,OF0H      ; CLEAR ALT,CLRL,LEFT AND RIGHT
;                               ; SHIFTS
ECC8 80 26 0018 R OF      AND      KB_FLAG_1,OFH      ; CLEAR POTENTIAL BREAK OF INS,CAPS
;                               ; NUM AND SCROLL SHIFT
EED1 80 26 0088 R 1F      AND      KB_FLAG_2,1FH      ; CLEAR FUNCTION STATES
EED6 90                    J16_7: POPF      ; GET THE FLAGS
EED7                    J17:
EED7 72 40                JC      J20
EED9  E8 EAA0 R          CALL     RESULTS      ; GET THE NEC STATUS
EEDC  72 3B              JC      J20          ; LOOK FOR ERROR
;-----CHECK THE RESULTS RETURNED BY THE CONTROLLER
EED8  FC                CLD      ; SET THE CORRECT DIRECTION
EEDF  BE 0042 R          MOV     SI,OFFSET NEC_STATUS ; POINT TO STATUS FIELD
EEE2  AC                LODS     NEC_STATUS ; GET ST0
EEE3  24 C0              AND     AL,0COH      ; TEST FOR NORMAL TERMINATION
EEE5  74 98              JZ      J22          ; OPN_OK
EEE7  3C 40              CMP     AL,040H      ; TEST FOR ABNORMAL TERMINATION
EEE9  75 2B              JNZ     J1B          ; NOT ABNORMAL, BAD NEC

```

NOTE

```

; THE CURRENT SYSTEM CONFIGURATION HAS NO OMA. IN ORDER TO
; STOP THE NEC AN EOT MUST BE PASSED TO FORCE THE NEC TO HALT
; THEREFORE, THE STATUS RETURNED BY THE NEC WILL ALWAYS SHOW
; AN EOT ERROR. IF THIS IS THE ONLY ERROR RETURNED AND THE
; NUMBER OF SECTORS TRANSFERRED EQUALS THE NUMBER SECTORS
; REQUESTED IN THIS INTERRUPT CALL THEN THE OPERATION HAS
; COMPLETED SUCCESSFULLY. IF AN EOT ERROR IS RETURNED AND THE
; REQUESTED NUMBER OF SECTORS IS NOT THE NUMBER OF SECTORS
; TRANSFERRED THEN THE ERROR IS LEGITIMATE. WHEN THE EOT
; ERROR IS INVALID THE STATUS BYTES RETURNED ARE UPDATED TO
; REFLECT THE STATUS OF THE OPERATION IF OMA HAD BEEN PRESENT

```

```

EEEB  AC                LODS     NEC_STATUS ; GET ST1
EEEC  3C 80              CMP     AL,80H      ; IS THIS THE ONLY ERROR?
EEEE  74 2A              JE      J21_1      ; NORMAL TERMINATION, NO ERROR
EEEF  D0 E0              SAL     AL,1        ; NOT EOT ERROR, BYPASS ERROR BITS
EEF0  D0 E0              SAL     AL,1
EEF2  D0 E0              SAL     AL,1        ; TEST FOR CRC ERROR
EEF4  D0 E0              SAL     AL,1
EEF6  B4 10              MOV     AH,BAD_CRC
EEF8  72 18              JC      J19          ; RW_FAIL
EEFA  D0 E0              SAL     AL,1        ; TEST FOR DMA OVERRUN
EEFC  B4 08              MOV     AH,BAD_DMA
EEFE  72 12              JC      J19          ; RW_FAIL
EFFF  D0 E0              SAL     AL,1
EF02  D0 E0              SAL     AL,1        ; TEST FOR RECORO NOT FOUND
EF04  B4 04              MOV     AH,RECORD_NOT_FND
EF06  72 0A              JC      J19          ; RW_FAIL
EF08  D0 E0              SAL     AL,1
EF0A  D0 E0              SAL     AL,1        ; TEST MISSING ADDRESS MARK
EF0C  B4 02              MOV     AH,BAD_ADDR_MARK
EF0E  72 02              JC      J19          ; RW_FAIL
;-----NEC MUST HAVE FAILED
EF10                    J1B:      ; RW-NEC-FAIL
EF10  B4 20              MOV     AH,BAD_NEG
EF12                    J19:      ; RW-FAIL
EF12  08 26 0041 R      OR      DISKETTE_STATUS,AH
EF16  E8 EAE1 R        CALL     NUM_TRANS      ; HOW MANY WERE REALLY TRANSFERRED
EF19                    J20:      ; RW_ERR
EF19  C3                RET      ; RETURN TO CALLER
;-----OPERATION WAS SUCCESSFUL
EF1A                    J21_1:
EF1A  BA 5E 0E          MOV     BL,[BP+14]      ; GET NUMBER OF SECTORS PASSED
;                               ; FROM STACK
EF1D  E8 EAE1 R        CALL     NUM_TRANS      ; HOW MANY GOT MOVED, AL CONTAINS
;                               ; NUM OF SECTORS
EF20  3A 08              CMP     BL,AL          ; NUMBER REQUESTED=NUMBER ACTUALLY
EF22  74 DC              JE      J21_2          ; TRANSFERRED?
EF22  74 DC              JE      J21_2          ; TRANSFER SUCCESSFUL
;-----OPERATION ATTEMPTED TO ACCESS DATA PAST REAL EOT. THIS IS
; A REAL ERROR
EF24  80 0E 0041 R 04  OR      DISKETTE_STATUS,RECORD_NOT_FND
EF29  C6 06 0043 R 80  MOV     ST0
EF2E  F9                STC
EF2F  C3                RET
EF30  33 C0              XOR     AX,AX          ; CLEAR AX FOR NEC_STATUS UPDATE
EF32  33 F6              XOR     SI,SI          ; INDEX TO NEC_STATUS ARRAY
EF34  8B B4 0042 R      MOV     NEC_STATUS[SI],AL ; ZERO OUT BYTE, ST0
EF38  46                INC     SI              ; POINT INDEX AT SECOND BYTE
EF39  8B B4 0042 R      MOV     NEC_STATUS[SI],AL ; ZERO OUT BUYE, ST1
EF3D  E8 03              JMP     SHORT J21_3    ; OPN_OK
EF3F  E8 EAE1 R        CALL     NUM_TRANS      ; NO ERRORS
EF42  32 E4              XOR     AH,AH
EF44  C3                RET
EF45                    RW_OPN  ENDP

```

DISK_INT

```

; THIS ROUTINE HANDLES THE DISKETTE INTERRUPT. AN INTERRUPT
; WILL OCCUR ONLY WHEN THE ONE-SHOT TIMER IS FIRED. THIS
; OCCURS IN AN ERROR SITUATION. THIS ROUTINE SETS ERRORS IN
; THE DISKETTE STATUS BYTE AND DISABLES THE ONE-SHOT TIMER.
; THEN THE RETURN ADDRESS ON THE STACK IS CHANGED TO RETURN
; TO THE OP_END LABEL.

```

INPUT

NONE.

OUTPUT

```

; NONE. DS POINTS AT BIOS DATA AREA. CARRY FLAG IS SET SO
; THAT ERROR WILL BE CAUGHT IN THE ENVIRONMENT RETURNED TO.

```

```

EF57          ORG      0EF57H
EF57          DISK_INT  PROC   FAR
EF57          PUSH    DS
EF58          PUSH    AX
EF59          PUSH    DX          ; SAVE REGISTER
EF5A          PUSH    BP          ; SAVE THE BP REGISTER
EF5B          CALL    DDS          ; SETUP DS TO POINT AT BIOS DATA
;----- CHECK IF INTERRUPT OCCURED IN INT13 OR WHETHER IT IS A
; SPURIOUS INTERRUPT
EF5E          MOV     BP,SP          ; POINT BP AT STACK
EF60          PUSH    CS          ; WAS IT IN THE BIOS AREA
EF61          POP     AX
EF62          CMP     AX,WORD PTR[BP+10]; GET INTERRUPTED SEGMENT
EF65          JNE     D13          ; NOT IN BIOS, ERROR CONDITION
EF67          MOV     AX,WORD PTR[BP+8]; GET IP ON THE STACK
EF6A          CMP     AX,OFFSET VERIFY_LOOP ; RANGE CHECK IP FOR DISK
; TRANSFER
EF6D          JL      D13          ; BELOW TRANSFER CODE
EF6E          CMP     AX,OFFSET OP_END+1 ; UPPER RANGE OF TRANSFER CODE
EF72          JGE     D13          ; ABOVE RANGE OF WATCHDOG TERRAIN
;----- VALID DISKETTE INTERRUPT CHANGE RETURN ADDRESS ON STACK TO
; PULL OUT OF LOOP
EF74          MOV     WORD PTR[BP+8],OFFSET OP_END
EF79          OR      WORD PTR[BP+12],1 ; TURN ON CARRY FLAG IN FLAGS ON
; STACK
;-----
;*****
; A WRITE PROTECTED DISKETTE WILL ALWAYS GET STUCK IN WRITE LOOP
; WAITING FOR BEGINNING OF EXECUTION PHASE. WHEN THE WATCHDOG
; FIRES AND THE STATUS IN PORT NEC_STAT = DXH (X MEANS DON'T CARE)
; STATUS FROM THE RESULT PHASE IS AVAILABLE. THE STATUS IS READ
; AND WRITE PROTECT IS CHECKED FOR.
;-----
EF7E          BA      00F4
EF81          EC
EF82          24 F0
EF84          3C D0
EF86          75 14
EF8B          EB EAA0 R
EF8B          BE 0042 R
;-----
EF8E          BA 44 01
EF91          AB 02
EF93          74 07
EF95          80 0E 0041 R 03
EF9A          EB 13
;-----TIME OUT ERROR
EF9C          90 0E 0041 R 80
EFA1          C6 06 003E R 00
;-----
EFA6          BA 00F2
EFA9          5D
;-----
EFAA          EB EB45 R
EFAE          55
EFAE          EE
EFAF          80 20
EF81          E9 20
EF83          5D
EF84          5A
EF85          5B
EF86          1F
EF87          CF
EF8B          CF
;-----
DISK_INT      ENOP
;-----
; DISK_BASE
; THIS IS THE SET OF PARAMETERS REQUIRED FOR
; DISKETTE OPERATION. THEY ARE POINTED AT BY THE
; DATA VARIABLE DISK_POINTER. TO MODIFY THE PARAMETERS,
; BUILD ANOTHER PARAMETER BLOCK AND POINT AT IT
;-----
EFC7          ORG      0EFC7H
EFC7          DISK_BASE LABEL BYTE
EFC7          DB      11001111B ; SRT=C, HD UNLOAD=OF - 1ST SPECIFY
; BYTE
EFCB          DB      3 ; HD LOAD=1, MODE=NO DMA - 2ND
; SPECIFY BYTE
EFC9          DB      MOTOR_WAIT ; WAIT AFTER OPN TIL MOTOR OFF
EFCB          DB      2 ; 512 BYTES/SECTOR
EFCB          DB      B ; EOT ( LAST SECTOR ON TRACK)
EFCB          DB      02AH ; GAP LENGTH
EFCD          DB      OFFH ; DTL
EFCB          DB      050H ; GAP LENGTH FOR FORMAT
EFCF          DB      0F6H ; FILL BYTE FOR FORMAT
EFD0          DB      25 ; HEAD SETTLE TIME (MILLISECONDS)
EFD1          DB      4 ; MOTOR START TIME (1/8 SECONDS)

```

```

----- INT 17 -----
;
; PRINTER_IO
; THIS ROUTINE PROVIDES COMMUNICATION WITH THE PRINTER
; (AH)=0 PRINT THE CHARACTER IN (AL)
; ON RETURN, AH=1 IF CHARACTER COULD NOT BE PRINTED
; (TIME OUT), OTHER BITS SET AS ON NORMAL STATUS CALL
; (AH)=1 INITIALIZE THE PRINTER PORT
; RETURNS WITH (AH) SET WITH PRINTER STATUS
; (AH)=2 READ THE PRINTER STATUS INTO (AH)
;
; 7 6 5 4 3 2-1 0
; : : : : : : :
; : : : : : : : UNUSED
; : : : : : : : 1 = I/O ERROR
; : : : : : : :
; : : : : : : : 1 = SELECTED
; : : : : : : : 1 = OUT OF PAPER
; : : : : : : : 1 = ACKNOWLEDGE
; : : : : : : : 1 = NOT BUSY
;
; (DX) = PRINTER TO BE USED (0,1,2) CORRESPONDING TO ACTUAL
; VALUES IN PRINTER_BASE AREA
; DATA AREA PRINTER_BASE CONTAINS THE BASE ADDRESS OF THE PRINTER
; CARO(S) AVAILABLE (LOCATED AT BEGINNING OF DATA SEGMENT, 40BH
; ABSOLUTE, 3 WORDS), UNLESS THERE IS ONLY A SERIAL PRINTER
; ATTACHED, IN WHICH CASE THE WORD AT 40:B WILL CONTAIN A 02FBH.
; REGISTERS AH IS MODIFIED
; ALL OTHERS UNCHANGED
-----

```

```

;
; ASSUME CS:CODE,DS:DATA
; ORG 0EFD2H
; PRINTER_IO PROC FAR
;
; INTERRUPTS BACK ON
; SAVE SEGMENT
;
; REDIRECT TO SERIAL ONLY IF:
; 1) SERIAL PRINTER IS ATTACHED, AND...
; 2) WORD AT PRINTER_BASE = 02FBH.
; POWER ONS WILL ONLY PUT A 02FBH IN THE PRINTER BASE IF THERE'S
; NO PARALLEL PRINTER ATTACHED.
;
; MOV CX,EQUIP_FLAG ; GET FLAG IN CX
; TEST CH,00100000B ; SERIAL ATTACHED?
; JZ B0 ; NO -HANDLE NORMALLY
; MOV BX,PRINTER_BASE ; SEE IF THERE'S AN RS232
; CMP BX,02FBH ; BASE IN THE PRINTER BASE.
; JNE B0
; JMP B1_A ; IF THERE IS REDIRECT
; ; ELSE... HANDLE AS PARALLEL
;
; CONTROL IS PASSED TO THIS POINT IF THERE IS A PARALLEL OR
; THERE'S NO SERIAL PRINTER ATTACHED.
;
; B0: MOV SI,DX ; GET PRINTER PARM
; MOV BL,PRINT_TIM_OUT[SI] ; LOAD TIMEOUT VALUE
; SHL 9I,1 ; WORD OFFSET INTO TABLE
; MOV DX,PRINTER_BASE[SI] ; GET BASE ADDRESS FOR PRINTER
; ; CARD
; OR DX,DX ; TEST DX FOR ZERO, INDICATING NO
; ; PRINTER
; JZ B1 ; IF NO PARALLEL, RETURN
; OR AH,AH ; TEST FOR (AH)=0
; JZ B2 ; PRINT_AL
; DEC AH ; TEST FOR (AH)=1
; JZ B8 ; INIT_PRT
; DEC AH ; TEST FOR (AH)=2
; JZ B5 ; PRINTER STATUS
; ; RETURN
;
; B1: POP BX ; RECOVER REGISTERS
; POP CX ; RECOVER REGISTERS
; POP SI
; POP DX
; POP DS
; IRET
;
; PRINT THE CHARACTER IN (AL)
; B2: PUSH AX ; SAVE VALUE TO PRINT
; OUT DX,AL ; OUTPUT CHAR TO PORT
; INC DX ; POINT TO STATUS PORT
;
; -----WAIT BUSY
;
; B3: SUB CX,CX ; INNER LOOP (64K)
; B3_1: IN AL,DX ; GET STATUS
; MOV AH,AL ; STATUS TO AH ALSO
; TEST AL,80H ; IS THE PRINTER CURRENTLY BUSY
; JNZ B4 ; OUT_STROBE
; LOOP IF NOT
; DEC BL ; DROP OUTER LOOP COUNT
; JNZ B3 ; MAKE ANOTHER PASS IF NOT ZERO
; OR AH,1 ; SET ERROR FLAG
; AND AH,OF9H ; TURN OFF THE UNUSED BITS
; JMP SHORT B7 ; RETURN WITH ERROR FLAG SET
;
; B4: MOV AL,ODH ; OUT_STROBE
; INC DX ; SET THE STROBE HIGH
; OUT DX,AL
; MOV AL,0CH ; SET THE STROBE LOW
; OUT DX,AL
; POP AX ; RECOVER THE OUTPUT CHAR

```

```

;----- PRINTER STATUS
F035 50 B5: PUSH AX ; SAVE AL REG
F036 88 94 000B R B6: MOV DX,PRINTER_BASE[SI]
F03A 42 INC DX
F03B EC IN3 AL,DX ; GET PRINTER STATUS
F03C 8A E0 MOV AH,AL
F03E 80 E4 FB AND AH,0FBH ; TURN OFF UNUSED BITS
F041 B7: ; STATUS_SET
F041 5A POP DX ; RECOVER AL REG
F042 8A C2 MOV AL,DL ; GET CHARACTER INTO AL
F044 80 F4 4B XOR AH,4BH ; FLIP A COUPLE OF BITS
F047 E8 C4 JMP B1 ; RETURN FROM ROUTINE

;----- INITIALIZE THE PRINTER PORT
F049 50 B8: PUSH AX ; SAVE AL
F04A 42 INC DX ; POINT TO OUTPUT PORT
F04B 42 INC DX
F04C 80 0B MOV AL,B ; SET INIT LINE LOW
F04E EE OUT DX,AL
F04F B8 03E9 MOV AX,1000
F052 48 B9: ; INIT_LOOP
F053 7B FD DEC AX ; LOOP FOR RESET TO TAKE
F055 80 C0 JNZ B9 ; INIT_LOOP
F055 80 C0 MOV AL,0CH ; NO INTERRUPTS, NON AUTO LF, INIT
; HIGH
F057 EE OUT DX,AL
F05B EB DC JMP B6 ; PRT_STATUS_1
F05A PRINTER_TO ENDP
F065 00 ORG 0F065H
F065 E9 000B R JMP NEAR PTR VIDEO_10

;-----
; SUBROUTINE TO SAVE ANY SCAN CODE RECEIVED ;
; BY THE NMI ROUTINE (PASSED IN AL) ;
; DURING POST IN THE KEYBOARD BUFFER ;
; CALLED THROUGH INT. 4BH ;
;-----
F068 KEY_SCAN_SAVE PROC FAR
ASSUME DS:DATA
F068 EB 13BB R CALL ODS ; POINT DS TO DATA AREA
F06B BE 001E R SI,OFFSET KB_BUFFER ; POINT TO FIRST LOC. IN BUFFER
F06E 8B 04 MOV [SI],AL ; SAVE SCAN CODE
F070 8B C4 MOV AX,SP ; CHECK FOR STACK UNDERFLOW
F072 80 E4 E0 AND AH,11100000B ; (THESE BITS WILL BE 111 IF
; UNDERFLOW HAPPEND)
F075 74 00 JZ KS_1
F077 32 C0 XOR AL,AL
F079 E6 A0 OUT 0A0H,AL ; SHUT OFF NMI
F07B BB 2000 MOV BX,2000H ; ERROR CODE 2000H
F07E BE 003E R MOV SI,OFFSET KEY_ERR ; POST MESSAGE
F081 EB 098C R CALL E_MSG ; AND HALT SYSTEM
F084 CF IRET ; RETURN TO CALLER
F085 KEY_SCAN_SAVE ENDP

;-----
; SUBROUTINE TO SET AN INSB250 CHIP'S BAUD RATE TO 9600 BPS AND
; DEFINE IT'S DATA WORD AS HAVING 8 BITS/WORD, 2 STOP BITS, AND
; 000 PARITY.
;
; EXPECTS TO BE PASSED:
; (DX) = LINE CONTROL REGISTER
;
; UPON RETURN:
; (DX) = TRANSMIT/RECEIVE BUFFER ADDRESS
;
; ALSO, ALTERS REGISTER AL. ALL OTHERS REMAIN INTACT.
;-----
F085 SB250 PROC NEAR
F085 80 80 NOV AL,80H ; SET DLAB = 1
F087 EE OUT DX,AL
F088 EB 00 JMP $+2 ; I/O DELAY
F08A B3 EA 03 SUB DX,3 ; LSB OF DIVISOR LATCH
F08B 80 C0 MOV AL,12 ; DIVISOR = 12 PRODUCES 9600 BPS
F08F EE OUT DX,AL ; SET LSB
F090 EB 00 JMP $+2 ; I/O DELAY
F092 42 INC DX ; MSB OF DIVISOR LATCH
F093 80 00 MOV AL,0 ; HIGH ORDER OF DIVISORS
F095 EE OUT DX,AL ; SET MSB
F096 EB 00 JMP $+2 ; I/O DELAY
F098 42 INC DX
F099 42 INC DX ; LINE CONTROL REGISTER
F09A 80 0F MOV AL,00001111B ; 8 BITS/WORD, 2 STOP BITS, 000
; PARITY
F09C EE OUT DX,AL
F09D EB 00 JMP $+2 ; I/O DELAY
F09F B3 EA 03 SUB DX,3 ; RECEIVER BUFFER
F0A2 EC IN AL,DX ; IN CASE WRITING TO PORT LCR
; CAUSED DATA READY TO GO HIGH!
F0A3 C3 RET
F0A4 SB250 ENDP

;----- TABLES FOR USE IN SETTING OF CRT MODE
F0A4 00 ORG 0F0A4H
F0A4 VIDEO_PARMS LABEL BYTE
;----- INIT_TABLE
F0A4 3B 2B 2C 06 1F 06 0B 3BH,2BH,2CH,06H,1FH,6,19H ; SETUP FOR 40X25
19 0B
F0AB 1C 02 07 06 07 0B 1CH,2,7,6,7
F0B0 00 00 00 00 0B 0,0,0,0

```

```

= 0010
FOB4 71 50 5A 0C 1F 06 06
      1B
FOBB 1C 02 07 06 07
FOC0 00 00 00 00
FOC4 39 28 2B 06 7F 06 06
      64
FOCB 70 02 01 26 07
FOD0 00 00 00 00
FOD4 71 50 56 0C 3F 06 06
      32
FODB 3B 02 03 26 07
FOE0 00 00 00 00

M0040 EQU $-VIDEO_PARMS
DB 71H,50H,5AH,0CH,1FH,6,19H ; SETUP FOR 80X25
DB 1CH,2,7,6,7
DB 0,0,0,0
DB 3BH,28H,2BH,06H,7FH,6,64H ; SET UP FOR GRAPHICS
DB 70H,2,1,26H,7
DB 0,0,0,0
DB 71H,50H,56H,0CH,3FH,6,32H ; SET UP FOR GRAPHICS
DB 3BH,2,3,26H,7 ; USING 32K OF MEMORY
DB 0,0,0,0 ; (MODES 9 & A)
;-----
; READ_AC_CURRENT
; THIS ROUTINE READS THE ATTRIBUTE AND CHARACTER AT THE
; CURRENT CURSOR POSITION AND RETURNS THEM TO THE CALLER
; INPUT
; (AH) = CURRENT CRT MODE
; (BH) = DISPLAY PAGE ( ALPHA MODES ONLY )
; (DS) = DATA SEGMENT
; (ES) = REGEN SEGMENT
; OUTPUT
; (AL) = CHAR READ
; (AH) = ATTRIBUTE READ
;-----
FOE4 80 FC 04
FOE7 72 03
FOE9 E9 F3F1 R
FOEC EB F0F7 R
FOEF 8B F3
FOF1 06
FOF2 1F
FOF3 AD
FOF4 E9 0F70 R
FOF7
FOF7 BA CF
FOF9 32 ED
FOFB 8B F1
FOFD D1 E6
FOFF 8B 84 0050 R
F103 33 D8
F105 E3 06
F107
F107 03 1E 004C R
F108 E2 FA
F10D
F10D EB ESC2 R
F110 03 D8
F112 C3
F113

READ_AC_CURRENT PROC NEAR
    ASSUME CS:CODE,DS:DATA,ES:DATA
    READ_AC_CURRENT PROC NEAR
    CMP AH,4 ; IS THIS GRAPHICS?
    JC C60
    JMP GRAPHICS_READ
C60:
    CALL FIND_POSITION ; READ_AC_CONTINUE
    MOV SI,BX ; ESTABLISH ADDRESSING IN SI
    PUSH ES ; * 2 FOR WORD OFFSET
    POP DS ; GET SEGMENT FOR QUICK ACCESS
    LODSW ; GET THE CHAR/ATTR
    JMP VIDEO_RETURN
    READ_AC_CURRENT ENDP
    FIND_POSITION PROC NEAR
    MOV CL,BH ; DISPLAY PAGE TO CX
    XOR CH,CH
    MOV SI,CX ; MOVE TO SI FOR INDEX
    SAL SI,1 ; * 2 FOR WORD OFFSET
    MOV AX,ESI+OFFSET CURSOR_POSN ; GET ROW/COLUMN OF
    ; THAT PAGE
    XOR BX,BX ; SET START ADDRESS TO ZERO
    JCXZ C62 ; NO_PAGE
    ADD BX,CRT_LEN ; PAGE_LOOP
    LOOP C61 ; LENGTH OF BUFFER
C62:
    CALL POSITION ; NO_PAGE
    ADD BX,AX ; DETERMINE LOCATION IN REGEN
    RET ; ADD TO START OF REGEN
    FIND_POSITION ENDP
;-----
; WRITE_AC_CURRENT
; THIS ROUTINE WRITES THE ATTRIBUTE AND CHARACTER AT
; THE CURRENT CURSOR POSITION
; INPUT
; (AH) = CURRENT CRT MODE
; (BH) = DISPLAY PAGE
; (CX) = COUNT OF CHARACTERS TO WRITE
; (AL) = CHAR TO WRITE
; (BL) = ATTRIBUTE OF CHAR TO WRITE
; (DS) = DATA SEGMENT
; (ES) = REGEN SEGMENT
; OUTPUT
; NONE
;-----
F113
F113 80 FC 04
F116 72 03
F118 E9 F3F1 R
F11B
F11B BA E3
F11D 50
F11E 51
F11F EB F0F7 R
F122 8B F8
F124 59
F125 5B
F126
F126 AB
F127 E2 FD
F129 E9 0F70 R
F12C

WRITE_AC_CURRENT PROC NEAR
    WRITE_AC_CURRENT PROC NEAR
    CMP AH,4 ; IS THIS GRAPHICS?
    JC C63
    JMP GRAPHICS_WRITE
C63:
    MOV AH,BL ; WRITE_AC_CONTINUE
    PUSH AX ; GET ATTRIBUTE TO AH
    PUSH CX ; SAVE ON STACK
    CALL FIND_POSITION ; SAVE WRITE COUNT
    MOV DI,BX ; ADDRESS TO DI REGISTER
    POP CX ; WRITE COUNT
    POP AX ; CHARACTER IN AX REG
C64:
    STQSW ; WRITE_LOOP
    LDP C64 ; PUT THE CHAR/ATTR
    JMP VIDEO_RETURN ; AS MANY TIMES AS REQUESTED
    WRITE_AC_CURRENT ENDP

```

```

;-----
; WRITE_C_CURRENT
; THIS ROUTINE WRITES THE CHARACTER AT
; THE CURRENT CURSOR POSITION, ATTRIBUTE UNCHANGED
;
; INPUT
; (AH) = CURRENT CRT MODE
; (BH) = DISPLAY PAGE
; (CX) = COUNT OF CHARACTERS TO WRITE
; (AL) = CHAR TO WRITE
; (DS) = DATA SEGMENT
; (ES) = REGEN SEGMENT
;
; OUTPUT
; NONE
;-----
F12C      WRITE_C_CURRENT PROC NEAR
F12C      CMP     AH,4           ; IS THIS GRAPHICS?
F12F      JC      C65
F131      JMP     GRAPHICS_WRITE
F134      PUSH   AX           ; SAVE ON STACK
F135      PUSH   CX           ; SAVE WRITE COUNT
F136      CALL   FIND_POSITION
F139      MOV    DI,BX        ; ADDRESS TO DI
F13B      POP    CX           ; WRITE COUNT
F13C      POP    BX           ; BL HAS CHAR TO WRITE
F13D      MOV    AL,BL        ; WRITE_LOOP
F13F      STOSB                ; RECOVER CHAR
F140      INC    DI           ; PUT THE CHAR/ATTR
F141      LOOP  C66           ; BUMP POINTER PAST ATTRIBUTE
F143      JMP     VIDEO_RETURN ; AS MANY TIMES AS REQUESTED
F146      WRITE_C_CURRENT ENDP
;-----
; READ_DOT -- WRITE DOT
; THESE ROUTINES WILL WRITE A DOT, OR READ THE
; DOT AT THE INDICATED LOCATION
;
; ENTRY --
; DX = ROW (0-199) (THE ACTUAL VALUE DEPENDS ON THE MODE)
; CX = COLUMN ( 0-839) ( THE VALUES ARE NOT RANGE CHECKED )
; AL = DOT VALUE TO WRITE (1,2 OR 4 BITS DEPENDING ON MODE,
; REQ'D FOR WRITE DOT ONLY, RIGHT JUSTIFIED)
; BIT 7 OF AL = 1 INDICATES XOR THE VALUE INTO THE LOCATION
;
; DB = DATA SEGMENT
; ES = REGEN SEGMENT
;
; EXIT
; AL = DOT VALUE READ, RIGHT JUSTIFIED, READ ONLY
;-----
ASSUME    CS:CODE,DS:DATA,ES:DATA
F146      READ_DOT PROC NEAR
F146      CMP    CRT_MODE,0AH ; 640X200 4 COLOR?
F148      JE     READ_ODD      ; YES, HANDLE SEPARATELY
F14D      CALL   C72           ; DETERMINE BYTE POSITION OF DOT
F150      MOV    AL,ES:[SI]    ; GET THE BYTE
F153      AND    AL,AH         ; MASK OFF THE OTHER BITS IN THE
; BYTE
F155      SHL   AL,CL          ; LEFT JUSTIFY THE VALUE
F157      MOV    CL,DH         ; GET NUMBER OF BITS IN RESULT
F159      ROL   AL,CL          ; RIGHT JUSTIFY THE RESULT
F15B      JMP    VIDEO_RETURN  ; RETURN FROM VIDEO IO
; IN 640X200 4 COLOR MODE, THE 2 COLOR BITS (C1,C0) ARE DIFFERENT
; THAN OTHER MODES. C0 IS IN THE EVEN BYTE, C1 IS IN THE FOLLOWING
; ODD BYTE - BOTH AT THE SAME BIT POSITION WITHIN THEIR RESPECTIVE
; BYTES.
F15E      READ_ODD:
F15E      CALL   C72           ; DETERMINE POSITION OF DOT
F161      PUSH  DX             ; SAVE INFO
F162      PUSH  CX
F163      PUSH  AX
F164      MOV    AL,ES:[SI+1] ; GET C1 COLOR BIT FROM ODD BYTE
F168      AND    AL,AH         ; MASK OFF OTHER BITS
F16A      SHL   AL,CL          ; LEFT JUSTIFY THE VALUE
F16C      MOV    CL,DH         ; GET NUMBER OF BITS IN RESULT
F16E      INC    CL
F170      ROL   AL,CL          ; RIGHT JUSTIFY THE RESULT
F172      MOV    BX,AX         ; SAVE IN BX REG
F174      POP    AX           ; RESTORE POSITION INFO
F175      POP    CX
F176      POP    DX
F177      MOV    AL,ES:[SI]    ; GET C0 COLOR BIT FROM EVEN BYTE
F17A      AND    AL,AH         ; MASK OFF OTHER BITS
F17C      SHL   AL,CL          ; LEFT JUSTIFY THE VALUE
F17E      MOV    CL,DH         ; GET NUMBER OF BITS IN RESULT
F180      ROL   AL,CL          ; RIGHT JUSTIFY THE RESULT
F182      OR    AL,BL         ; COMBINE C1 & C0
F184      JMP    VIDEO_RETURN

```



```

F187      READ_DOT      ENDP
F187      WRITE_DOT    PROC    NEAR
F187 51    PUSH    CX          ; SAVE COL
F188 52    PUSH    DX          ; SAVE ROW
F189 50    PUSH    AX          ; SAVE DOT VALUE
F18A 50    PUSH    AX          ; TWICE
F18B EB F1D9 R CALL    C72          ; DETERMINE BYTE POSITION OF THE
; DOT
F18E 02 E8 SHR     AL,CL        ; SHIFT TO SET UP THE BITS FOR
; OUTPUT
F190 22 C4 AND     AL,AH          ; STRIP OFF THE OTHER BITS
F192 26: BA 0C MOV    CL,ES:[SI]      ; GET THE CURRENT BYTE
F195 5B    POP     BX          ; RECOVER XOR FLAG
F196 F6 C3 80 TEST   BL,80H         ; IS IT ON
F199 75 36 JNZ    C70          ; YES, XOR THE 00T
F19B F6 D4 NOT    AH          ; SET THE MASK TO REMOVE THE
; INDICATED BITS
F19D 22 CC AND     CL,AH          ; OR IN THE NEW VALUE OF THOSE BITS
F19F 0A C1 OR     AL,CL          ; FINISH_DOT
F1A1      CG7:         MOV    ES:[SI],AL      ; RESTORE THE BYTE IN MEMORY
F1A4 59    POP     AX          ;
F1A5 5A    POP     DX          ; RECOVER ROW
F1A6 59    POP     CX          ; RECOVER COL
F1A7 60 3E 0049 R 0A CMP    CRT_MODE,0A0H ; S40X200 4 COLOR?
F1AC 75 20 JNE    C69          ; NO, JUMP
F1AE 50    PUSH    AX          ; SAVE DOT VALUE
F1AF 50    PUSH    AX          ; TWICE
F1B0 00 EB SHR    AL,1          ; SHIFT c1 BIT INTO c0 POSITION
F1B2 EB F1D9 R CALL    C72          ; DETERMINE BYTE POSITION OF THE
; DOT
F1B5 02 E8 SHR    AL,CL        ; SHIFT TO SET UP THE BITS FOR
; OUTPUT
F1B7 22 C4 AND     AL,AH          ; STRIP OFF THE OTHER BITS
F1B9 26: BA 4C 01 MOV    CL,ES:[SI+1]   ; GET THE CURRENT BYTE
F1BD 5B    POP     BX          ; RECOVER XOR FLAG
F1BE F6 C3 80 TEST   BL,80H         ; IS IT ON
F1C1 75 12 JNZ    C71          ; YES, XOR THE DOT
F1C3 F6 D4 NOT    AH          ; SET THE MASK TO REMOVE THE
; INDICATED BITS
F1C5 22 CC AND     CL,AH          ; OR IN THE NEW VALUE OF THOSE BITS
F1C7 0A C1 OR     AL,CL          ; FINISH_DOT
F1C9 26: 8B 44 01 MOV    ES:[SI+1],AL   ; RESTORE THE BYTE IN MEMORY
F1CD 58    POP     AX          ;
F1CE E9 0F70 R JMP    VIDEO_RETURN ; RETURN FROM VIDEO IO
F1D1      C70:         XOR    AL,CL          ; XOR_DOT
F1D1 32 C1 XOR    AL,CL          ; EXCLUSIVE OR THE DOTS
F1D3 EB CC JMP    C67          ; FINISH UP THE WRITING
F1D5      C71:         XOR    AL,CL          ; XOR_DOT
F1D5 32 C1 XOR    AL,CL          ; EXCLUSIVE OR THE DOTS
F1D7 EB FO JMP    C68          ; FINISH UP THE WRITING
F1D9      WRITE_DOT    ENDP
;-----
; THIS SUBROUTINE DETERMINES THE REGEN BYTE LOCATION OF THE
; INDICATED ROW COLUMN VALUE IN GRAPHICS MODE.
; ENTRY --
; DX = ROW VALUE (0-199)
; CX = COLUMN VALUE (0-639)
; EXIT --
; SI = OFFSET INTO REGEN BUFFER FOR BYTE OF INTEREST
; AH = MASK TO STRIP OFF THE BITS OF INTEREST
; CL = BITS TO SHIFT TO RIGHT JUSTIFY THE MASK IN AH
; DH = # BITS IN RESULT
;-----
F1D9      PROC    NEAR
F1D9 53    PUSH    BX          ; SAVE BX DURING OPERATION
F1DA 50    PUSH    AX          ; WILL SAVE AL DURING OPERATION
;-----
; DETERMINE 1ST BYTE IN INDICATED ROW BY MULTIPLYING ROW VALUE
; BY 40( LOW BIT OF ROW DETERMINES EVEN/ODD, 80 BYTES/ROW
F1DB 80 28 MOV    AL,40          ;
F1DD 52    PUSH    DX          ; SAVE ROW VALUE
F1DE 80 E2 FE AND    DL,0FEH        ; STRIP OFF ODD/EVEN BIT
F1E1 80 3E 0049 R 09 CMP    CRT_MODE,090H ; MODE USING 32K REGEN?
F1E6 72 03 JC     C73          ; NO, JUMP
F1E8 80 E2 FC AND    DL,0FCH        ; STRIP OFF LOW 2 BITS
F1EB F6 E2 MUL    DL          ; AX HAS ADDRESS OF 1ST BYTE OF
; INDICATED ROW
F1ED 5A    POP     DX          ; RECOVER IT
F1EE F6 C2 01 TEST   DL,1          ; TEST FOR EVEN/ODD
F1F1 74 03 JZ     C74          ; JUMP IF EVEN ROW
F1F3 05 2000 ADD    AX,2000H      ; OFFSET TO LOCATION OF ODD ROWS
F1F6 80 3E 0045 R 09 CMP    CRT_MODE,090H ; MODE USING 32K REGEN?
F1FB 72 08 JC     C75          ; NO, JUMP
F1FD F6 C2 02 TEST   DL,2          ; TEST FOR ROW 2 OR ROW 3
F200 74 03 JZ     C75          ; JUMP IF ROW 0 OR 1
F202 05 4000 ADD    AX,4000H     ; OFFSET TO LOCATION OF ROW 2 OR 3
F205 8B FO MOV    SI,AX          ; MOVE POINTER TO SI
F207 58    POP     AX          ; RECOVER AL VALUE
F208 8B D1 MOV    DX,CX          ; COLUMN VALUE TO DX

```

```

;----- DETERMINE GRAPHICS MODE CURRENTLY IN EFFECT
;SET UP THE REGISTERS ACCORDING TO THE MODE
;CH = MASK FOR LOW OF COLUMN ADDRESS ( 7/3/1 FOR HIGH/MED/LOW RES)
;CL = # OF ADDRESS BITS IN COLUMN VALUE ( 3/2/1 FOR H/M/L)
;BL = MASK TO SELECT BITS FROM POINTED BYTE (80H/COH/FOH FOR H/N/L)
;BH = NUMBER OF VALID BITS IN POINTED BYTE ( 1/2/4 FOR H/M/L)
F20A 8B 02C0
F20D 89 0302
F210 80 3E 0049 R 04
F215 74 21
F217 80 3E 0049 R 05
F21C 74 1A
F21E 8B 04F0
F221 89 0101
F224 80 3E 0049 R 0A
F229 74 07
F22B 80 3E 0049 R 06
F230 75 06
F232 8B 0180
F235 89 0703
;----- DETERMINE BIT OFFSET IN BYTE FROM COLUMN MASK
;----- DETERMINE BYTE OFFSET FOR THIS LOCATION IN COLUMN
F238 22 EA
;----- DETERMINE BIT OFFSET IN BYTE FROM COLUMN MASK
;----- DETERMINE BYTE OFFSET FOR THIS LOCATION IN COLUMN
F23A D3 EA
F23C 03 F2
F23E 80 3E 0049 R 0A
F243 75 02
F245 03 F2
F247 8A F7
;----- DETERMINE BIT OFFSET IN BYTE FROM COLUMN MASK
;----- DETERMINE BYTE OFFSET FOR THIS LOCATION IN COLUMN
F249 2A C9
F24B 80 C8
;----- DETERMINE BIT OFFSET IN BYTE FROM COLUMN MASK
;----- DETERMINE BYTE OFFSET FOR THIS LOCATION IN COLUMN
F24D 02 CD
F24F FE CF
F251 75 F8
;----- DETERMINE BIT OFFSET IN BYTE FROM COLUMN MASK
;----- DETERMINE BYTE OFFSET FOR THIS LOCATION IN COLUMN
F253 8A E3
F256 D2 EC
;----- DETERMINE BIT OFFSET IN BYTE FROM COLUMN MASK
;----- DETERMINE BYTE OFFSET FOR THIS LOCATION IN COLUMN
F257 58
F258 C3
F259
;----- DETERMINE BIT OFFSET IN BYTE FROM COLUMN MASK
;----- DETERMINE BYTE OFFSET FOR THIS LOCATION IN COLUMN
C72 ENDP
;----- DETERMINE BIT OFFSET IN BYTE FROM COLUMN MASK
;----- DETERMINE BYTE OFFSET FOR THIS LOCATION IN COLUMN
; SCROLL UP
; THIS ROUTINE SCROLLS UP THE INFORMATION ON THE CRT
; ENTRY --
; CH,CL = UPPER LEFT CORNER OF REGION TO SCROLL
; DH,DL = LOWER RIGHT CORNER OF REGION TO SCROLL
; BOTH OF THE ABOVE ARE IN CHARACTER POSITIONS
; BH = FILL VALUE FOR BLANKED LINES
; AL = # LINES TO SCROLL (AL=0 MEANS BLANK THE ENTIRE FIELD)
; DS = DATA SEGMENT
; ES = REGEN SEGMENT
; EXIT --
; NOTHING, THE SCREEN IS SCROLLED
;----- DETERMINE BIT OFFSET IN BYTE FROM COLUMN MASK
;----- DETERMINE BYTE OFFSET FOR THIS LOCATION IN COLUMN
F259 GRAPHICS_UP PROC NEAR
F259 8A D8 ; SAVE LINE COUNT IN BL
F25B 8B C1 ; GET UPPER LEFT POSITION INTO AX REG
;----- USE CHARACTER SUBROUTINE FOR POSITIONING
;----- ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE
F25D EB F72C R
F260 8B F8 ; SAVE RESULT AS DESTINATION
;----- DETERMINE SIZE OF WINDOW
;----- DETERMINE BIT OFFSET IN BYTE FROM COLUMN MASK
;----- DETERMINE BYTE OFFSET FOR THIS LOCATION IN COLUMN
F262 2B D1
F264 81 C2 0101
F26B 00 E6
;----- DETERMINE BIT OFFSET IN BYTE FROM COLUMN MASK
;----- DETERMINE BYTE OFFSET FOR THIS LOCATION IN COLUMN
F26A D0 E6
;----- DETERMINE BIT OFFSET IN BYTE FROM COLUMN MASK
;----- DETERMINE BYTE OFFSET FOR THIS LOCATION IN COLUMN
F26C 80 3E 0049 R 06
F271 74 1D
;----- DETERMINE BIT OFFSET IN BYTE FROM COLUMN MASK
;----- DETERMINE BYTE OFFSET FOR THIS LOCATION IN COLUMN
F273 D0 E2
F275 D1 E7
F277 80 3E 0049 R 04
F27C 74 12
F27E 80 3E 0049 R 05
F283 74 08
F285 80 3E 0049 R 0A
F28A 74 04
;----- DETERMINE BIT OFFSET IN BYTE FROM COLUMN MASK
;----- DETERMINE BYTE OFFSET FOR THIS LOCATION IN COLUMN
F28C D0 E2
;----- DETERMINE BIT OFFSET IN BYTE FROM COLUMN MASK
;----- DETERMINE BYTE OFFSET FOR THIS LOCATION IN COLUMN
F28E D1 E7

```

```

;----- DETERMINE THE SOURCE ADDRESS IN THE BUFFER
F290          C80:          ; FIND_SOURCE
F290 06          PUSH      ES          ; GET SEGMENTS BOTH POINTING TO
; REGEN
F291 1F          POP        DS
F292 2A ED       SUB        CH,CH          ; ZERO TO HIGH OF COUNT REG
F294 00 E3       SAL        BL,1          ; MULTIPLY NUMBER OF LINES BY 4
F296 00 E3       SAL        BL,1
F298 74 67       JZ         C8B          ; IF ZERO, THEN BLANK ENTIRE FIELD
F29A 8A C3       MOV        AL,BL          ; GET NUMBER OF LINES IN AL
F29C 84 50       MOV        AH,80          ; 80 BYTES/ROW
F29E F6 E4       MUL        AH          ; DETERMINE OFFSET TO SOURCE
F2A0 8B F7       MOV        SI,D1          ; SET UP SOURCE
F2A2 03 F0       ADD        SI,AX          ; ADD IN OFFSET TO 1T
F2A4 8A E6       MOV        AH,DH          ; NUMBER OF ROWS IN FIELD
F2A6 2A E3       SUB        AH,BL          ; DETERMINE NUMBER TO MOVE
;----- LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD
; FIELDS
F2A8          C81:          ; ROW_LOOP
F2A8 EB F3C7 R   CALL        C95          ; MOVE ONE ROW
F2AB 1E          PUSH      DS          ; SAVE DATA SEG
F2AC EB 138B R   CALL        DDS          ; POINT TO BIOS DATA AREA
F2AF 80 3E 0049 R 09 CMP        CRT_MODE,8    ; MODE USES 32K REGEN?
F2B4 1F          POP        DS          ; RESTORE DATA SEG
F2B5 72 15       JC         C82          ; NO, JUMP
F2B7 91 C6 2000   ADD        DI,2000H      ; ADJUST POINTERS
F2B8 81 C7 2000   ADD        DI,2000H
F2BF EB F3C7 R   CALL        C95          ; MOVE 2 MORE ROWS
F2C2 81 EE 3F80   SUB        SI,4000H-80   ; BACK UP POINTERS
F2C6 81 EF 3F80   SUB        DI,4000H-80   ;
F2CA FE CC       DEC        AH          ; ADJUST COUNT
F2CC 81 EE 1F80   SUB        SI,2000H-80   ; MOVE TO NEXT ROW
F2D0 81 EF 1F80   SUB        DI,2000H-80   ;
F2D4 FE CC       DEC        AH          ; NUMBER OF ROWS TO MOVE
F2D8 75 D0       JNZ       C81          ; CONTINUE TILL ALL MOVED
;----- FILL IN THE VACATED LINES
F2D8          C83:          ; CLEAR_ENTRY
F2D8 8A C7       MOV        AL,BH          ; ATTRIBUTE TO FILL WITH
F2DA EB F3E0 R   CALL        C96          ; CLEAR THAT ROW
F2DD 1E          PUSH      DS          ; SAVE DATA SEG
F2DE EB 138B R   CALL        DDS          ; POINT TO BIOS DATA AREA
F2E1 80 3E 0049 R 09 CMP        CRT_MODE,9    ; MODE USES 32K REGEN?
F2E6 1F          POP        DS          ; RESTORE DATA SEG
F2E7 72 0D       JC         C85          ; NO, JUMP
F2E9 81 C7 2000   ADD        DI,2000H
F2ED EB F3E0 R   CALL        C96          ; CLEAR 2 MORE ROWS
F2F0 81 EF 3F80   SUB        DI,4000H-80   ; BACK UP POINTERS
F2F4 FE CB       DEC        BL          ; ADJUST COUNT
F2F6 81 EF 1F80   SUB        DI,2000H-80   ; POINT TO NEXT LINE
F2FA FE CB       DEC        BL          ; NUMBER OF LINES TO FILL
F2FC 75 DC       JNZ       C84          ; CLEAR_LOOP
F2FE E9 0F70 R   JMP        VIDEO_RETURN  ; EVERYTHING DONE
F301          C85:          ; BLANK_FIELD
F301 8A DE       MOV        BL,DH          ; SET BLANK COUNT TO EVERYTHING IN
; FIELD
F303 EB D3       JMP        C83          ; CLEAR THE FIELD
F305          GRAPHICS_UP  ENDP
;-----
; SCROLL DOWN
; THIS ROUTINE SCROLLS DOWN THE INFORMATION ON THE CRT
; ENTRY --
; CH,CL = UPPER LEFT CORNER OF REGION TO SCROLL
; DH,DL = LOWER RIGHT CORNER OF REGION TO SCROLL
; BOTH OF THE ABOVE ARE IN CHARACTER POSITIONS
; BH = FILL VALUE FOR BLANKED LINES
; AL = # LINES TO SCROLL (AL=0 MEANS BLANK THE ENTIRE FIELD)
; DS = DATA SEGMENT
; ES = REGEN SEGMENT
; EXIT --
; NOTHING, THE SCREEN IS SCROLLED
;-----
F305          GRAPHICS_DOWN  PROC   NEAR
F305 FD          STD          ; SET DIRECTION
F306 8A D8       MOV        BL,AL          ; SAVE LINE COUNT IN BL
F308 8B C2       MOV        AX,DX          ; GET LOWER RIGHT POSITION INTO AX REG
;----- USE CHARACTER SUBROUTINE FOR POSITIONING
;----- ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE
F30A EB F72C R   CALL        GRAPH_POSN
F30D 8B F8       MOV        DI,AX          ; SAVE RESULT AS DESTINATION
; ADDRESS
;----- DETERMINE SIZE OF WINDOW
F30F 2B D1       SUB        DX,CX          ; SET DIRECTION
F311 81 C2 0101  ADD        DX,101H        ; ADJUST VALUES
F315 00 E6       SAL        DH,1          ; MULTIPLY # ROWS BY 4 SINCE 8 VERT
; DOTS/CHAR
F317 00 E6       SAL        DH,1          ; AND EVEN/ODD ROWS
;----- DETERMINE CRT MODE
F319 80 3E 0049 R 06 CMP        CRT_MODE,6    ; TEST FOR HIGH RES
F31E 74 22       JZ         C87          ; FIND_SOURCE_DOWN

```

```

;----- MEDIUM RES DOWN
F320 D0 E2 SAL DL,1 ; # COLUMNS * 2, SINCE 2 BYTES/CHAR
; (OFFSET OK)
F322 D1 E7 SAL D1,1 ; OFFSET #2 SINCE 2 BYTES/CHAR
F324 47 INC D1 ; POINT TO LAST BYTE
F325 80 3E 0049 R 04 CMP CRT_MODE,4 ; TEST FOR MEDIUM RES
F32A 74 16 JZ C87 ; FIND_SOURCE_DOWN
F32C 80 3E 0049 R 05 CMP CRT_MODE,5 ; TEST FOR MEDIUM RES
F331 74 0F JZ C87 ; FIND_SOURCE_DOWN
F333 80 3E 0049 R 0A CMP CRT_MODE,0AH ; TEST FOR MEDIUM RES
F338 74 08 JZ C87 ; FIND_SOURCE_DOWN
F33A 4F DEC D1 ;
F33B D0 E2 SAL DL,1 ; # COLUMNS * 2 AGAIN, SINCE 4
; BYTES/CHAR (OFFSET OK)
F33D D1 E7 SAL D1,1 ; OFFSET #2 AGAIN, SINCE 4
; BYTES/CHAR
F33F B3 C7 03 ADO D1,3 ; POINT TO LAST BYTE
; DETERMINE THE SOURCE ADDRESS IN THE BUFFER
F342 CB7: ; FIND_SOURCE_DOWN
F342 2A ED SUB CH,CH ; ZERO TO HIGH OF COUNT REG
F344 B8 00F0 MOV AX,240 ; OFFSET TO LAST ROW OF PIXELS IF
; 18K REGEN
F347 80 3E 0049 R 09 CMP CRT_MODE,9 ; USING 32K REGEN?
F34C 72 03 JC C8B ; NO, JUMP
F34E B8 00A0 MOV AX,160 ; OFFSET TO LAST ROW OF PIXELS IF
; 32K REGEN
F351 03 F8 CB8: ADD D1,AX ; POINT TO LAST ROW OF PIXELS
F353 D0 E3 SAL BL,1 ; MULTIPLY NUMBER OF LINES BY 4
F355 D0 E3 SAL BL,1 ;
F357 74 6A JZ C94 ; IF ZERO, THEN BLANK ENTIRE FIELD
F359 BA C3 MOV AL,BL ; GET NUMBER OF LINES IN AL
F35B B4 B0 MOV AH,B0 ; 80 BYTES/ROW
F35D F6 E4 MUL AH ; DETERMINE OFFSET TO SOURCE
F35F 88 F7 MOV SI,DI ; SET UP SOURCE
F361 28 F0 SUB SI,AX ; SUBTRACT THE OFFSET
F363 BA E6 MOV AH,DH ; NUMBER OF ROWS IN FIELD
F365 2A E3 SUB AH,BL ; DETERMINE NUMBER TO MOVE
F367 06 PUSH ES ; BOTH SEGMENTS TO REGEN
F368 1F POP DS ;
;----- LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD
; FIELDS
F369 CB9: ; ROW_LOOP_DOWN
F36B EB F3C7 R CALL C95 ; MOVE ONE ROW
F36C 1E PUSH DS ; SAVE DATA SEG
F36D EB 138B R CALL D05 ; POINT TO BIOS DATA AREA
F370 80 3E 0049 R 09 CMP CRT_MODE,9 ; MODE USES 32K REGEN?
F375 1F POP DS ; RESTORE DATA SEG
F376 72 15 JC C90 ; NO, JUMP
F378 B1 C6 2000 ADO SI,2000H ; ADJUST POINTERS
F37C 91 C7 2000 ADD DI,2000H ;
F380 EB F3C7 R CALL C95 ; MOVE 2 MORE ROWS
F383 81 EE 4050 SUB SI,4000H+80 ; BACK UP POINTERS
F387 81 EF 4050 SUB DI,4000H+80 ;
F388 FE CC DEC AH ; ADJUST COUNT
F38D 81 EE 2050 CB9: SUB SI,2000H+80 ; MOVE TO NEXT ROW
F391 81 EF 2050 SUB DI,2000H+80 ;
F395 FE CC OEC AH ; NUMBER OF ROWS TO MOVE
F397 75 D0 JNZ C89 ; CONTINUE TILL ALL MOVED
;----- FILL IN THE VACATED LINE(S)
F399 CB1: ; CLEAR_ENTRY_DOWN
F399 BA C7 MOV AL,8H ; ATTRIBUTE TO FILL WITH
F39B CB2: ; CLEAR_LOOP_DOWN
F39B EB F3E0 R CALL C96 ; CLEAR A ROW
F39E 1E PUSH DS ; SAVE DATA SEG
F39F EB 138B R CALL D05 ; POINT TO BIOS DATA AREA
F3A2 80 3E 0049 R 09 CMP CRT_MODE,9 ; MODE USES 32K REGEN?
F3A7 1F POP DS ; RESTORE DATA SEG
F3A8 72 0D JC C93 ; NO, JUMP
F3AA B1 C7 2000 ADO DI,2000H ;
F3AE EB F3E0 R CALL C96 ; CLEAR 2 MORE ROWS
F3B1 81 EF 4050 SUB DI,4000H+80 ; BACK UP POINTERS
F3B5 FE C8 DEC BL ; ADJUST COUNT
F3B7 81 EF 2050 CB3: SUB DI,2000H+80 ; POINT TO NEXT LINE
F3B8 FE C8 DEC BL ; NUMBER OF LINES TO FILL
F3BD 75 DC JNZ C92 ; CLEAR_LOOP_DOWN
F3BF FC CLD ; RESET THE DIRECTION FLAG
F3C0 E9 0F70 R JMP VIDEO_RETURN ; EVERYTHING DONE
F3C3 CB4: ; BLANK_FIELD_DOWN
F3C3 BA DE MOV BL,DH ; SET BLANK COUNT TO EVERYTHING IN
; FIELD
F3C5 EB D2 JMP C91 ; CLEAR THE FIELD
F3C7 GRAPHIC6_DOWN ENDP
;----- ROUTINE TO MOVE ONE ROW OF INFORMATION
F3C7 CB5: ;
F3C7 8A CA PROC NEAR ;
F3C9 55 MOV CL,DL ; NUMBER OF BYTES IN THE ROW
F3CA 57 PUSH SI ;
F3CB F3/ A4 REP PUSH DI ; SAVE POINTERS
F3CD 5F REP MOVSB ; MOVE THE EVEN FIELD
F3CE 5E POP DI ;
F3CF 91 C6 2000 ADD SI,2000H ;
F3D3 81 C7 2000 ADD DI,2000H ; POINT TO THE ODD FIELD
F3D7 56 PUSH SI ;
F3D8 57 PUSH DI ; SAVE THE POINTERS
F3D9 8A CA MOV CL,DL ; COUNT BACK
F3DB F3/ A4 REP MOVSB ; MOVE THE ODD FIELD
F3DD 5F POP DI ;
F3DE 5E POP SI ; POINTERS BACK
F3DF C3 RET ; RETURN TO CALLER
F3E0 CB5 ENDP

```

```

;----- CLEAR A SINGLE ROW
F3E0          PROC NEAR
F3E0 8A CA    MOV CL,0L          ; NUMBER OF BYTES IN FIELD
F3E2 57       PUSH DI          ; SAVE POINTER
F3E3 F3/ AA   REP STOSB        ; STORE THE NEW VALUE
F3E5 5F       POP DI          ; POINTER BACK
F3E6 81 C7 2000 ADD DI,2000H        ; POINT TO ODD FIELD
F3EA 57       PUSH DI
F3EB 8A CA    MOV CL,0L
F3ED F3/ AA   REP STOSB        ; FILL THE ODD FIELD
F3EF 5F       POP DI
F3F0 C3       RET             ; RETURN TO CALLER
F3F1          ENDP
;-----
; GRAPHICS WRITE
; THIS ROUTINE WRITES THE ASCII CHARACTER TO THE CURRENT
; POSITION ON THE SCREEN.
; ENTRY --
; AL = CHARACTER TO WRITE
; BL = COLOR ATTRIBUTE TO BE USED FOR FOREGROUND COLOR
; IF BIT 7 IS SET, THE CHAR IS XOR'D INTO THE REGEN BUFFER
; (0 IS USED FOR THE BACKGROUND COLOR)
; CX = NUMBER OF CHARS TO WRITE
; DS = DATA SEGMENT
; ES = REGEN SEGMENT
; EXIT --
; NOTHING IS RETURNED
;
; GRAPHICS READ
; THIS ROUTINE READS THE ASCII CHARACTER AT THE CURRENT CURSOR
; POSITION ON THE SCREEN BY MATCHING THE DOTS ON THE SCREEN TO
; THE CHARACTER GENERATOR CODE POINTS
; ENTRY --
; NONE (0 IS ASSUMED AS THE BACKGROUND COLOR)
; EXIT --
; AL = CHARACTER READ AT THAT POSITION (0 RETURNED IF NONE FOUND)
;
; FOR BOTH ROUTINES, THE IMAGES USED TO FORM CHARS ARE CONTAINED IN
; ROM. INTERRUPT 44H IS USED TO POINT TO THE TABLE FOR THE FIRST
; 128 CHARS. INTERRUPT 17H IS USED TO POINT TO THE TABLE FOR THE
; SECOND 128 CHARS.
;-----
F3F1          ASSUME CS:CODE,DS:DATA,ES:DATA
F3F1 32 E4    GRAPHICS_WRITE PROC NEAR
F3F3 50       XOR AH,AH          ; ZERO TO HIGH OF CODE POINT
;-----
; DETERMINE POSITION IN REGEN BUFFER TO PUT CODE POINTS
F3F4 EB F729 R CALL R59          ; FIND LOCATION IN REGEN BUFFER
F3F7 8B FB    MOV DI,AX          ; REGEN POINTER IN DI
;-----
; DETERMINE REGION TO GET CODE POINTS FROM
F3F9 5B       POP AX          ; RECOVER CODE POINT
F3FA 8E 0110 R MOV SI,OFFSET CSET_PTR ; ASSUME FIRST HALF
F3FD 3C 80    CMP AL,80H        ; IS IT IN FIRST HALF?
F3FF 72 05    JB RI            ; JUMP IF IT IS
F401 BE 007C R MOV SI,OFFSET EXT_PTR ; SET POINTER FOR SECOND HALF
F404 2C 80    SUB AL,80H        ; ZERO ORIGIN FOR SECOND HALF
F406          R1:
F406 1E       PUSH DS          ; EXTEND_CHAR
F407 33 02    XOR DX,DX          ; SAVE DATA POINTER
F409 8E DA    MOV DS,DX          ; ESTABLISH VECTOR ADDRESSING
;-----
; ASSUME DS:ABS0
F40B C5 34    LDS SI,0WORD PTR [SI] ; GET THE OFFSET OF THE TABLE
F40D 8C DA    MOV DX,DS          ; GET THE SEGMENT OF THE TABLE
;-----
; ASSUME DS:DATA
F40F 1F       POP DS          ; RECOVER DATA SEGMENT
F410 52       PUSH DX          ; SAVE TABLE SEGMENT ON STACK
;-----
; DETERMINE GRAPHICS MODE IN OPERATION
F411 D1 E0    SAL AX,1          ; MULTIPLY CODE POINT
F413 01 E0    SAL AX,1          ; VALUE BY 8
F415 D1 E0    SAL AX,1
F417 03 F0    ADD SI,AX          ; SI HAS OFFSET OF DESIRED CODES
F419 60 3E 0049 R 04 CMP CRT_MODE,4
F41E 74 45    JE R9            ; TEST FOR MEDIUM RESOLUTION MODE
F420 60 3E 0049 R 05 CNP CRT_MODE,5
F425 74 3E    JE R9            ; TEST FOR MEDIUM RESOLUTION MODE
F427 80 3E 0049 R 0A CMP CRT_MODE,0AH
F42C 75 03    JNE R3          ; TEST FOR MEDIUM RESOLUTION MODE
F42E E9 F404 R JMP R16          ; TEST FOR HIGH RESOLUTION MODE
F431 80 3E 0049 R 06 R3: CMP CRT_MODE,6
F436 75 53    JNE R12         ; GOTO LOW RESOLUTION IF NOT
;-----
; HIGH RESOLUTION MODE
F43B 1F       POP DS          ; RECOVER TABLE POINTER SEGMENT
F439 57       PUSH DI          ; SAVE REGEN POINTER
F43A 56       PUSH SI          ; SAVE CODE POINTER
F43B 86 04    MOV DH,4         ; NUMBER OF TIMES THROUGH LOOP
R6: LODSB      ; GET BYTE FROM CODE POINTS
F43E F6 C3 80 TEST BL,80H      ; SHOULD WE USE THE FUNCTION
F441 75 16    JNZ RB          ; TO PUT CHAR IN?
F443 AA       STOSB      ; STORE IN REGEN BUFFER
F444 AC       LODSB
F445 26 88 85 1FFF MOV ES:[DI+2000H-1],AL ; STORE IN SECOND HALF
F44A 83 C7 4F R5: MOV DI,79
F44D FE CE    DEC DH          ; DONE WITH LOOP
F44F 75 EC    JNZ R6
F451 5E       POP SI
F452 5F       POP DI          ; RECOVER REGEN POINTER
F453 47       INC DI          ; POINT TO NEXT CHAR POSITION
F454 E2 E3    LOOP R5         ; MORE CHARS TO WRITE

```

```

F456 E9 0F70 R
F459 26: 32 05
F45C AA
F45D AC
F45E 26: 32 85 1FFF
F463 EB E0

R705: JMP VIDEO_RETURN
RB: XOR AL,ES:[D1] ; EXCLUSIVE OR WITH CURRENT DATA
      STOSB ; STORE THE CODE POINT
      LODSB ; AGAIN FOR ODD FIELD
      XOR AL,ES:[D1+2000H-1]
      JMP R7 ; BACK TO MAINSTREAM
;----- MEDIUM RESOLUTION WRITE
R9: POP OS ; MED_RES_WRITE
     MOV DL,BL ; RECOVER TABLE POINTER SEGMENT
     GAL DI,1 ; SAVE HIGH COLOR BIT
     CALL R40 ; OFFSET*2 SINCE 2 BYTES/CHAR
           ; EXPAND BL TO FULL WORD OF COLOR
R10: MED_CHAR
     PUSH DI ; SAVE REGEN POINTER
     PUSH SI ; SAVE THE CODE POINTER
     MOV DH,4 ; NUMBER OF LOOPS
R11: CALL R35 ; DO FIRST 2 BYTES
     ADD DI,2000H ; NEXT SPOT IN REGEN
     CALL R35 ; DO NEXT 2 BYTES
     SUB DI,2000H-80
     DEC DH
     JNZ R11 ; KEEP GOING
     POP SI ; RECOVER CODE POINTER
     POP DI ; RECOVER REGEN POINTER
     INC DI ; POINT TO NEXT CHAR POSITION
     LOOP R10 ; MORE TO WRITE
     JMP R705
;----- LOW RESOLUTION WRITE
R12: POP DS ; LOW_RES_WRITE
     MOV DL,BL ; RECOVER TABLE POINTER SEGMENT
     GAL DI,1 ; SAVE HIGH COLOR BIT
     SAL DI,1 ; OFFSET*4 SINCE 4 BYTES/CHAR
     CALL R42 ; EXPAND BL TO FULL WORD OF COLOR
           ; MED_CHAR
R13: PUSH DI ; SAVE REGEN POINTER
     PUSH SI ; SAVE THE CODE POINTER
     MOV DH,4 ; NUMBER OF LOOPS
R14: CALL R39 ; EXPAND DOT ROW IN REGEN
     ADD DI,2000H ; POINT TO NEXT REGEN ROW
     CALL R39 ; EXPAND DOT ROW IN REGEN
     PUSH DS ; SAVE DS
     CALL DDS ; POINT TO BIOS DATA AREA
     CMP CRT_MODE,09H ; USING 32K REGEN AREA?
     POP DS ; RECOVER DS
     JNE R15 ; JUMP IF 16K REGEN
     ADD DI,2000H ; POINT TO NEXT REGEN ROW
     CALL R39 ; EXPAND DOT ROW IN REGEN
     ADD DI,2000H ; POINT TO NEXT REGEN ROW
     CALL R35 ; EXPAND DOT ROW IN REGEN
     SUB DI,4000H-80 ; ADJUST REGEN POINTER
R15: DEC DH
     SUB DI,2000H-80 ; ADJUST REGEN POINTER TO NEXT ROW
     DEC DH
     JNZ R14 ; KEEP GOING
     POP SI ; RECOVER CODE POINTER
     POP DI ; RECOVER REGEN POINTER
     ADD DI,4 ; POINT TO NEXT CHAR POSITION
     LOOP R13 ; MORE TO WRITE
     JMP R705
;----- 640X200 4 COLOR GRAPHICS WRITE
R16: POP DS ; RECOVER TABLE SEGMENT POINTER
     MOV DL,BL ; SAVE HIGH COLOR BIT
     SAL DI,1 ; OFFSET*2 SINCE 2 BYTES/CHAR
           ; EXPAND LOW 2 COLOR BITS IN BL (c1c0)
           ; INTO BX (c0c0c0c0c0c0c0c0c1c1c1c1c1c1)
     XOR AX,AX
     TEST BL,1 ; c0 COLOR BIT ON?
     JZ R17 ; NO, JUMP
     MOV AH,OFFH ; YES, SET ALL c0 BITS ON
R17: TEST BL,2 ; c1 COLOR BIT ON?
     JZ R18 ; NO, JUMP
     MOV AL,OFFH ; YES, SET ALL c1 BITS ON
R18: MOV BX,AX ; COLOR MASK IN BX
R19: PUSH DI ; SAVE REGEN POINTER
     PUSH SI ; SAVE CODE POINT POINTER
     MOV DH,2 ; SET LOOP COUNTER
R20: CALL R21 ; DO FIRST DOT ROW
     ADD DI,2000H ; ADJUST REGEN POINTER
     CALL R21 ; DO NEXT DOT ROW
     ADD DI,2000H ; ADJUST REGEN POINTER
     CALL R21 ; DO NEXT DOT ROW
     ADD DI,2000H ; ADJUST REGEN POINTER
     CALL R21 ; DO NEXT DOT ROW
     SUB DI,6000H-160 ; ADJUST REGEN POINTER TO NEXT ROW
     DEC DH
     JNZ R20 ; KEEP GOING
     POP SI ; RECOVER CODE POINT POINTER
     POP DI ; RECOVER REGEN POINTER
     INC DI ; POINT TO NEXT CHARACTER
     INC DI
     LOOP R19 ; MORE TO WRITE
     JMP VIDEO_RETURN
F465
F465 1F
F466 8A D3
F468 D1 E7
F46A EB F659 R
F46D
F46D 57
F46E 56
F46F 86 04
F471 EB F626 R
F474 81 C7 2000
F478 EB F626 R
F47B 81 EF 1F80
F47F FE CE
F48 75 EE
F483 5E
F484 5F
F485 47
F486 47
F487 E2 E4
F489 EB C8

F488
F488 1F
F48C 8A D3
F48E D1 E7
F490 D1 E7
F492 EB F68E R
F495
F495 57
F496 56
F497 86 04
F498 EB F645 R
F49C 81 C7 2000
F4A0 EB F645 R
F4A3 1E
F4A4 EB 1388 R
F4A7 80 3E 0049 R 09
F4AC 1F
F4AD 75 14
F4AF 81 C7 2000
F4B2 EB F645 R
F4B5 81 C7 2000
F4BA EB F645 R
F4BD 81 EF 3F80
F4C1 FE CE
F4C3 81 EF 1F80
F4C7 FE CE
F4C9 75 CE
F4CB 5E
F4CC 5F
F4CD 83 C7 04
F4D0 E2 C3
F4D2 EB B2

F4D4 1F
F4D5 8A D3
F4D7 D1 E7

F4D9 33 C0
F4DB F6 C3 01
F4DE 74 02
F4E0 84 FF
F4E2 F6 C3 02
F4E5 74 02
F4E7 80 FF
F4E9 8B 0B
F4EB
F4EB 57
F4EC 56
F4ED 86 02
F4EF EB F518 R
F4F2 81 C7 2000
F4F6 EB F518 R
F4F9 81 C7 2000
F4FD EB F518 R
F500 81 C7 2000
F504 EB F518 R
F507 81 EF 5F60
F508 FE CE
F509 75 E0
F50F 5E
F510 5F
F511 47
F512 47
F513 E2 D6
F515 E9 0F70 R

```

```

F518
F518 AC
F519 8A E0
F51B 23 C3
F510 F6 C2 80
F520 74 07
F522 26: 32 25
F525 26: 32 45 01
F529 26: 88 25
F52C 26: 88 45 01
F530 C3
F531
F531
F531
R21 PROC NEAR
L0D5B ; GET CODE POINT
MOV AH,AL ; COPY INTO AH
AND AX,8X ; SET COLOR
TEST DL,80H ; XOR FUNCTION?
JZ R22 ; NO, JUMP
XOR AH,ES:[DI] ; EXCLUSIVE OR WITH CURRENT DATA
XOR AL,ES:[DI+1]
R22: MOV ; STORE IN REGEN BUFFER
MOV ES:[DI+1],AL
RET
R21 ENDP
GRAPHICS_WRITE ENDP
-----
; GRAPHICS_READ
-----
GRAPHICS_READ PROC NEAR
CALL R09 ; CONVERTED TO OFFSET IN REGEN
MOV SI,AX ; SAVE IN SI
SUB SP,8 ; ALLOCATE SPACE TO SAVE THE READ
; CODE POINT
MOV BP,SP ; POINTER TO SAVE AREA
;----- DETERMINE GRAPHICS MODES
PUSH ES
MOV DH,4 ; NUMBER OF PASSES
CMP CRT_MODE,6 ; HIGH RESOLUTION
JZ R23
CMP CRT_MODE,4 ; MEDIUM RESOLUTION
JZ R28
CMP CRT_MODE,5 ; MEDIUM RESOLUTION
JZ R25
CMP CRT_MODE,0AH ; MEDIUM RESOLUTION
JZ R28
JMP SHORT R25 ; LOW RESOLUTION
;----- HIGH RESOLUTION READ
; GET VALUES FROM REGEN BUFFER AND CONVERT TO CODE POINT
R23: POP DS ; POINT TO REGEN SEGMENT
R24: MOV AL,[SI] ; GET FIRST BYTE
MOV [BP],AL ; SAVE IN STORAGE AREA
INC BP ; NEXT LOCATION
MOV AL,[SI+2000H] ; GET LOWER REGION BYTE
MOV [BP],AL ; ADJUST AND STORE
INC BP
ADD SI,80 ; POINTER INTO REGEN
DEC DH ; LDDP CONTROL
JNZ R24 ; DO IT SOME MORE
JMP SHORT R31 ; GO MATCH THE SAVED CODE POINTS
;----- LOW RESOLUTION READ
R25: POP DS ; POINT TO REGEN SEGMENT
SAL SI,1 ; OFFSET*4 SINCE 4 BYTES/CHAR
SAL SI,1
R26: CALL R55 ; GET 4 BYTES FROM REGEN INTO
; SINGLE SAVE
ADD SI,2000H ; GOTO LOWER REGION
CALL R55 ; GET 4 BYTES FROM REGEN INTO
; SINGLE SAVE
PUSH DS ; SAVE DS
CALL D05 ; POINT TO BIOS DATA AREA
CMP CRT_MODE,9 ; DO WE HAVE A 32K REGEN AREA?
POP DS
JNE R27 ; NO, JUMP
ADD SI,2000H ; GOTO LOWER REGION
CALL R55 ; GET 4 BYTES FROM REGEN INTO
; SINGLE SAVE
ADD SI,2000H ; GOTO LOWER REGION
CALL R55 ; GET 4 BYTES FROM REGEN INTO
; SINGLE SAVE
SUB SI,4000H-80 ; ADJUST POINTER
DEC DH
R27: SUB SI,2000H-80 ; ADJUST POINTER BACK TO UPPER
OEC DH
JNZ R28 ; DO IT SOME MORE
JMP SHORT R31 ; GO MATCH THE SAVED CODE POINTS
;----- MEDIUM RESOLUTION READ
R28: POP DS ; MED_RES_READ
SAL SI,1 ; POINT TO REGEN SEGMENT
CALL R50 ; OFFSET*2 SINCE 2 BYTES/CHAR
R29: CALL R50 ; GET PAIR BYTES FROM REGEN INTO
; SINGLE SAVE
ADD SI,2000H ; GO TO LOWER REGION
CALL R50 ; GET THIS PAIR INTO SAVE
PUSH DS ; SAVE DS
CALL D05 ; POINT TO BIOS DATA AREA
CMP CRT_MODE,0AH ; DO WE HAVE A 32K REGEN AREA?
POP DS
JNE R30 ; NO, JUMP
ADD SI,2000H ; GOTO LOWER REGION
CALL R50 ; GET PAIR BYTES FROM REGEN INTO
; SINGLE SAVE
ADD SI,2000H ; GOTO LOWER REGION
CALL R50 ; GET PAIR BYTES FROM REGEN INTO
; SINGLE SAVE
SUB SI,4000H-80 ; ADJUST POINTER
DEC DH
R30: SUB SI,2000H-80 ; ADJUST POINTER BACK INTO UPPER
DEC DH
JNZ R29 ; KEEP GOING UNTIL ALL 8 DONE

```

```

;----- SAVE AREA HAS CHARACTER IN IT, MATCH IT
R31:
F5E2 33 C0          MOV     AX,AX
F5E4 8E D8          MOV     DS,AX
                   ASSUME DS:ABS0
F5E6 C4 3E 0110 R  LES     D1,CSET_PTR
F5EA 83 ED 08          SUB     BP,8
                   ; GET POINTER TO FIRST HALF
                   ; ADJUST POINTER TO BEGINNING OF
                   ; SAVE AREA
F5ED 8B F5          MOV     SI,BP
F5EF FC           CLD
F5F0 32 C0          XOR     AL,AL
F5F2 16           PUSH    SS
F5F3 1F           POP     DS
F5F4 BA 00B0       MOV     DX,128
F5F7 56           PUSH    SI
F5F8 57           PUSH    DI
F5F9 89 00B0       MOV     CX,8
F5FC F3/ A6        REPE   CMPSB
F5FE 5F           POP     DI
F5FF 5E           POP     SI
F600 74 1E        JZ     R34
                   ; IF ZERO FLAG SET, THEN MATCH
                   ; OCCURRED
F602 FE C0        INC     AL
                   ; NO MATCH, MOVE ON TO NEXT
F604 83 C7 0B     ADD     DI,8
F607 4A          DEC     DX
F608 75 ED        JNZ    R33
                   ; NEXT CODE POINT
                   ; LOOP CONTROL
                   ; DO ALL OF THEM
;----- CHAR NOT MATCHED, MIGHT BE IN SECOND HALF
F60A 0A C0        OR     AL,AL
F60C 74 12        JE     R34
F60E 28 C0        SUB     AX,AX
F610 8E D8          MOV     OS,AX
                   ; ESTABLISH ADDRESSING TO VECTOR
                   ASSUME OS:ABS0
F612 C4 3E 007C R  LES     D1,EXT_PTR
F616 BC C0        MOV     AX,ES
F618 0B C7        OR     AX,DI
F61A 74 04        JZ     R34
F61C 80 B0        MOV     AL,128
F61E EB D2        JMP     R32
                   ; NO SENSE LOOKING
                   ; ORIGIN FOR SECOND HALF
                   ; GO BACK AND TRY FOR IT
                   ASSUME DS:DATA
;----- CHARACTER IS FOUND (AL=0 IF NOT FOUND)
R34: ADD     SP,8
                   ; READJUST THE STACK, THROW AWAY
                   ; WORK AREA
                   ; ALL DONE
F623 E9 0F70 R    JMP     VIDEO_RETURN
F626          GRAPHICS_READ ENDP
;-----
R35: PROC NEAR
F628 AC          LODSB
F629 EB F67E R    CALL R43
F62A 23 C3        AND     AX,BX
                   ; GET CODE POINT
                   ; DOUBLE UP ALL THE BITS
                   ; CONVERT THEM TO FOREGROUND COLOR
                   ; ( O BACK )
F62C F6 C2 B0     TEST    DL,B0H
F62F 74 07        JZ     R37
F631 26: 32 25    XOR     AH,ES:[D1]
F634 26: 32 45 01 XOR     AL,ES:[D1+1]
F638 26: 8B 25    MOV     ES:[D1],AH
F63B 26: 8B 45 01 MOV     ES:[D1+1],AL
F63F C3          RET
R35: ENDP
;-----
R38: PROC NEAR
F640 EB F6A0 R    CALL R45
F643 EB E5        JMP     R36
F645          ENDP
                   ; QUAD UP THE LOW NIBBLE
;-----
; EXPAND 1 DOT ROW OF A CHAR INTO 4 BYTES IN THE REGEN BUFFER
;-----
R39: PROC NEAR
F645 AC          LODSB
F646 50          PUSH    AX
F647 51          PUSH    CX
F648 81 04        MOV     CL,4
F64A D2 EB        SHR     AL,CL
F64C 59          POP     CX
F64D E9 F640 R    CALL R3B
F650 58          POP     AX
F651 47          INC     DI
F652 47          INC     DI
F653 EB F640 R    CALL R3B
F656 4F          DEC     DI
F657 4F          DEC     DI
F658 C3          RET
R39: ENDP
                   ; GET CODE POINT
                   ; SAVE
                   ; MOV HIGH NIBBLE TO LOW
                   ; EXPAND TO 2 BYTES & PUT IN REGEN
                   ; RECOVER CODE POINT
                   ; ADJUST REGEN POINTER
                   ; EXPAND LOW NIBBLE & PUT IN REGEN
                   ; RESTORE REGEN POINTER
;-----
; EXPAND_MED_COLOR
; THIS ROUTINE EXPANDS THE LOW 2 BITS IN BL TO
; FILL THE ENTIRE BX REGISTER
; ENTRY --
; BL = COLOR TO BE USED ( LOW 2 BITS )
; EXIT --
; BX = COLOR TO BE USED ( B REPLICATIONS OF THE 2 COLOR BITS )
;-----

```



```

F659
F659 80 E3 03
F65C 8A C3
F65E 51
F65F 89 0003
F662 00 E0
F664 00 E0
F666 0A 08
F668 E2 F8
F66A 8A F8
F66C 59
F66D C3
F66E

```

```

R40 PROC NEAR
      AND BL,3 ; ISOLATE THE COLOR BITS
      MOV AL,BL ; COPY TO AL
      PUSH CX ; SAVE REGISTER
      MOV CX,3 ; NUMBER OF TIMES TO DO THIS
R41: SAL AL,1 ; LEFT SHIFT BY 2
      OR BL,AL ; ANOTHER COLOR VERSION INTO BL
      LOOP R41 ; FILL ALL OF BL
      MOV BH,BL ; FILL UPPER PORTION
      POP CX ; REGISTER BACK
      RET ; ALL DONE
R40 ENDP

```

```

-----
; EXPAND_LOW_COLOR
; THIS ROUTINE EXPANDS THE LOW 4 BITS IN BL TO
; FILL THE ENTIRE 8X REGISTER
; ENTRY --
; BL = COLOR TO BE USED ( LOW 4 BITS )
; EXIT --
; BX = COLOR TO BE USED ( 4 REPLICATIONS OF THE 4 COLOR BITS )
-----

```

```

F66E
F66E 51
F66F 60 E3 0F
F672 8A F8
F674 81 04
F676 D2 E7
F678 0A F8

```

```

R42 PROC NEAR
      PUSH CX
      AND BL,0FH ; ISOLATE THE COLOR BITS
      MOV BH,BL ; COPY TO BH
      MOV CL,4 ; MOVE TO HIGH NIBBLE
      SHL BH,CL
      OR BH,BL ; MAKE BYTE FROM HIGH AND LOW
                        NIBBLES
      MOV BL,BH
      POP CX
      RET ; ALL DONE
R42 ENDP

```

```

-----
; EXPAND_BYTE
; THIS ROUTINE TAKES THE BYTE IN AL AND DOUBLES ALL
; OF THE BITS, TURNING THE 8 BITS INTO 16 BITS.
; THE RESULT IS LEFT IN AX
-----

```

```

F67E
F67E 52
F67F 51
F680 53
F681 28 D2
F683 89 0001
F686 88 08
F688 23 09
F68A 08 D3
F68C D1 E0
F68E D1 E1
F690 88 08
F692 23 09
F694 08 D3
F696 D1 E1

```

```

R43 PROC NEAR
      PUSH DX ; SAVE REGISTERS
      PUSH CX
      PUSH BX
      SUB DX,DX ; RESULT REGISTER
      MOV CX,1 ; MASK REGISTER
R44: MOV BX,AX ; BASE INTO TEMP
      AND BX,CX ; USE MASK TO EXTRACT A BIT
      OR DX,BX ; PUT INTO RESULT REGISTER
      SHL AX,1
      SHL CX,1 ; SHIFT BASE AND MASK BY 1
      MOV BX,AX ; BASE TO TEMP
      AND BX,CX ; EXTRACT THE SAME BIT
      OR DX,BX ; PUT INTO RESULT
      SHL CX,1 ; SHIFT ONLY MASK NOW, MOVING TO
                        NEXT BASE
      JNC R44 ; USE MASK BIT COMING OUT TO
                        TERMINATE
      MOV AX,OX ; RESULT TO PARM REGISTER
      POP BX
      POP CX ; RECOVER REGISTERS
      POP DX
      RET ; ALL DONE
R43 ENDP

```

```

-----
; EXPAND_NIBBLE
; THIS ROUTINE TAKES THE LOW NIBBLE IN AL AND QUADS ALL
; OF THE BITS, TURNING THE 4 BITS INTO 16 BITS.
; THE RESULT IS LEFT IN AX
-----

```

```

F6A0
F6A0 52
F6A1 33 D2
F6A3 AB 08
F6A5 74 03
F6A7 80 CE F0
F6AA AB 04
F6AC 74 03
F6AE 80 CE 0F
F6B1 A8 02
F6B3 74 03
F6B5 80 CA F0
F6B8 A9 01
F6BA 74 03
F6BC 80 CA 0F
F6BF 8B C2 0F
F6C1 5A
F6C2 C3
F6C3

```

```

R45 PROC NEAR
      PUSH DX ; SAVE REGISTERS
      XOR DX,DX ; RESULT REGISTER
      TEST AL,8
      JZ R46
      OR DH,OF0H
R46: TEST AL,4
      JZ R47
      OR DH,OFH
R47: TEST AL,2
      JZ R48
      OR DL,OFDH
R48: TEST AL,1
      JZ R49
      OR DL,OFH
R49: MOV AX,DX ; RESULT TO PARM REGISTER
      POP DX ; RECOVER REGISTERS
      RET ; ALL DONE
R45 ENDP

```

```

-----
; MED_READ_BYTE
; THIS ROUTINE WILL TAKE 2 BYTES FROM THE REGEN BUFFER,
; COMPARE AGAINST THE CURRENT FOREGROUND COLOR, AND PLACE
; THE CORRESPONDING ON/OFF BIT PATTERN INTO THE CURRENT
; POSITION IN THE SAVE AREA
; ENTRY --
; SI,DS = POINTER TO REGEN AREA OF INTEREST
; BX = EXPANDED FOREGROUND COLOR
; BP = POINTER TO SAVE AREA
; EXIT --
; BP IS INCREMENT AFTER SAVE
-----
R50 PROC NEAR
; GET FIRST BYTE
MOV AH,[SI]
; GET SECOND BYTE
MOV AL,[SI+1]
; SAVE DS
PUSH DS
; POINT TO BIOS DATA AREA
CALL DRS
; IN 640X200 4 COLOR MODE?
CMP CRT_MODE,0AH
; RESTORE REGEN SEG
POP DS
; NO, JUMP
JNE R52
; IN 640X200 4 COLOR MODE, ALL THE CO BITS ARE IN ONE BYTE, AND ALL
; THE CI BITS ARE IN THE NEXT BYTE... HERE WE CHANGE THEM BACK TO
; NORMAL c100 ADJACENT PAIRS.
PUSH BX
; SAVE REG
MOV CX,8
; SET LOOP COUNTER
R51: SAR AH,1
; CO BIT INTO CARRY
RCR BX,1
; AND INTO BX
SAR AL,1
; CI BIT INTO CARRY
RCR BX,1
; AND INTO BX
LOOP R51
; REPEAT
MOV AX,BX
; RESULT INTO AX
POP BX
; RESTORE BX
R52: MOV CX,0C000H
; 2 BIT MASK TO TEST THE ENTRIES
XOR DL,DL
; RESULT REGISTER
R53: TEST AX,CX
; IS THIS SECTION BACKGROUND?
JZ R54
; IF ZERO, IT IS BACKGROUND
STC
; WASN'T, SO SET CARRY
R54: RCL DL,1
; MOVE THAT BIT INTO THE RESULT
SHR CX,1
; MOVE THE MASK TO THE RIGHT BY 2
SHR CX,1
; BITS
JMC R53
; DO IT AGAIN IF MASK DIDN'T FALL
OUT
; STORE RESULT IN SAVE AREA
MOV [BP],DL
; ADJUST POINTER
INC BP
; ALL DONE
RET
R50 ENDP

```

```

F6C3
F6C3 9A 24
F6C5 8A 44 01
F6C8 1E
F6C8 EB 138B R
F6CC 80 3E 0049 R 0A
F8D1 1F
F8D2 7B 11

```

```

F6D4 83
F6D5 8B 000B
F8D9 00 FC
F8DA D1 D8
F8DC 00 FB
F8DE D1 D8
F8E0 E2 FE
F8E2 8B C3
F6E4 86
F6E5 8B C000
F6E9 32 D2
F8EA 85 C1
F8EC 74 01
F8EE F9
F8EF D0 D2
F6F1 D1 E9
F6F3 D1 E9
F6F5 73 F3
F6F7 8B 56 00
F6FA 4B
F6FB C3
F6FC

```

```

-----
; LOW_READ_BYTE
; THIS ROUTINE WILL TAKE 4 BYTES FROM THE REGEN BUFFER,
; COMPARE FOR BACKGROUND COLOR, AND PLACE
; THE CORRESPONDING ON/OFF BIT PATTERN INTO THE CURRENT
; POSITION IN THE SAVE AREA
; ENTRY --
; SI,DS = POINTER TO REGEN AREA OF INTEREST
; BP = POINTER TO SAVE AREA
; EXIT --
; BP IS INCREMENT AFTER SAVE
-----

```

```

F6FC
F6FE 8A 24
F6FE 8A 44 01
F701 32 D2
F703 E9 F714 R
F706 8A 64 02
F709 8A 44 03
F70C E9 F714 R
F70F 8B 56 00
F712 4B
F713 C3
F714
F714 89 F000
F717 85 C1
F719 74 01
F71B F9
F71C D0 D2
F71E D1 E9
F720 D1 E9
F722 D1 E9
F724 D1 E9
F726 73 EF
F728 C3
F729

```

```

R55 PROC NEAR
; GET FIRST 2 BYTES
MOV AH,[SI]
MOV AL,[SI+1]
; BUILD HIGH NIBBLE
XOR DL,DL
; GET SECOND 2 BYTES
MOV AH,[SI+2]
MOV AL,[SI+3]
; BUILD LOW NIBBLE
CALL R56
; STORE RESULT IN SAVE AREA
MOV [BP],DL
; ADJUST POINTER
INC BP
RET
R55 ENDP
R56 PROC NEAR
; 4 BIT MASK TO TEST THE ENTRIES
MOV CX,0F000H
; IS THIS SECTION BACKGROUND?
R57: TEST AX,CX
; IF ZERO, IT IS BACKGROUND
JZ R58
; WASN'T, SO SET CARRY
BTC
R58: RCL OL,1
; MOVE THAT BIT INTO RESULT
SHR CX,1
; MOVE MASK RIGH 4 BITS
SHR CX,1
SHR CX,1
SHR CX,1
JNC R57
; DO IT AGAIN IF MASK DIDN'T FALL OUT
RET
R56 ENDP

```

```

;-----
; V4_POSITION
; THIS ROUTINE TAKES THE CURSOR POSITION CONTAINED IN
; THE MEMORY LOCATION, AND CONVERTS IT INTO AN OFFSET
; INTO THE REGEN BUFFER, ASSUMING ONE BYTE/CHAR.
; FOR MEDIUM RESOLUTION GRAPHICS, THE NUMBER MUST
; BE DOUBLED.
; ENTRY -- NO REGISTERS, MEMORY LOCATION CURSOR_POSN IS USED
; EXIT--
; AX CONTAINS OFFSET INTO REGEN BUFFER
;-----

```

```

F729
F729 A1 0050 R
F72C
F72C 53
F72D 8B 0B
F72F 8A C4
F731 F5 2B 004A R
F735 80 3E 0049 R 09
F73A 73 02
F73C D1 E0
F73E D1 E0
F740 2A FF
F742 03 C3
F744 5B
F745 C3
F748

```

```

R59 PROC NEAR
MOV AX,CURSOR_POSN ; GET CURRENT CURSOR
GRAPH_POSN LABEL NEAR
PUSH BX ; SAVE REGISTER
MOV BX,AX ; SAVE A COPY OF CURRENT CURSOR
MOV AL,AH ; GET ROWS TO AL
MUL BYTE PTR CRT_COLS ; MULTIPLY BY BYTES/COLUMN
CMP CRT_MODE,9 ; MODE USING 32K REGEN?
JNC R60 ; YES, JUMP
SHL AX,1 ; MULTIPLY * 4 SINCE 4 ROWS/BYTE
R60: SHL BX,BH
SUB BX,BH ; ISOLATE COLUMN VALUE
ADD AX,BX ; DETERMINE OFFSET
POP BX ; RECOVER POINTER
RET ; ALL DONE
R59 ENDP

```

```

;-----
; LIGHT PEN
; THIS ROUTINE TESTS THE LIGHT PEN SWITCH AND THE LIGHT
; PEN TRIGGER. IF BOTH ARE SET, THE LOCATION OF THE LIGHT
; PEN IS DETERMINED. OTHERWISE, A RETURN WITH NO INFORMATION
; IS MADE.
; ON EXIT:
; (AH) = 0 IF NO LIGHT PEN INFORMATION IS AVAILABLE
; BX,CX,DX ARE DESTROYED
; (AH) = 1 IF LIGHT PEN IS AVAILABLE
; (DH,DL) = ROW,COLUMN OF CURRENT LIGHT PEN POSITION
; (CH) = RASTER POSITION
; (BX) = BEST GUESS AT PIXEL HORIZONTAL POSITION
;-----

```

```

F746
F748 03 03 05 0B 03 03
03 00 02 03 04
F751

```

```

;-----
; ASSUME CS:CODE,DS:DATA
;-----
SUBTRACT_TABLE
V1 LABEL BYTE
DB 3,3,5,5,3,3,3,0,2,3,4 ;

```

```

F751 32 E4
F753 8A 03DA
F756 EC
F757 AB 04
F759 74 03
F75B E9 F803 R

```

```

READ_LPEN PROC NEAR
;-----
; WAIT FOR LIGHT PEN TO BE DEPRESSSED
XOR AH,AH ; SET NO LIGHT PEN RETURN CODE
MOV DX,VGA_CTL ; GET ADDRESS OF VGA CONTROL REG
IN AL,DX ; GET STATUS REGISTER
TEST AL,4 ; TEST LIGHT PEN SWITCH
JZ V7B ; NOT SET, RETURN
JMP V7 ;
;-----
; NOW TEST FOR LIGHT PEN TRIGGER
V7B: TEST AL,2 ; TEST LIGHT PEN TRIGGER
JNZ V7A ; RETURN WITHOUT RESETTING TRIGGER
JMP V7 ;
;-----
; TRIGGER HAS BEEN SET, READ THE VALUE IN
V7A: MOV AH,1B ; LIGHT PEN REGISTERS ON 6845
;-----
; INPUT REGS POINTED TO BY AH, AND CONVERT TO ROW COLUMN IN DX
MOV DX,ADDR_6845 ; ADDRESS REGISTER FOR 6845
MOV AL,AH ; REGISTER TO READ
OUT DX,AL ; SET IT UP
INC DX ; DATA REGISTER
IN AL,DX ; GET THE VALUE
MOV CH,AL ; SAVE IN CX
DEC DX ; ADDRESS REGISTER
INC AH
MOV AL,AH ; SECOND DATA REGISTER
OUT DX,AL
INC DX ; POINT TO DATA REGISTER
IN AL,DX ; GET SECOND DATA VALUE
MOV AH,CH ; AX HAS INPUT VALUE
;-----
; AX HAS THE VALUE READ IN FROM THE 6845
MOV BL,CRT_MODE ; MODE VALUE TO BX
SUB BH,BH ; DETERMINE AMOUNT TO SUBTRACT
MOV BL,CS:V1(BX) ; TAKE IT AWAY
SUB AX,BX ; IN TOP OR BOTTOM BORDER?
CMP AX,4000 ; NO, OKAY
JB V15 ; YES, SET TO ZERO
XOR AX,AX
V15: MOV BX,CRT_START
SHR BX,1 ; CONVERT TO CORRECT PAGE ORIGIN
SUB AX,BX ; IF POSITIVE, DETERMINE MODE
JNS V2 ; <0 PLAYS AS 0
SUB AX,AX
;-----
; DETERMINE MODE OF OPERATION
V2: ; DETERMINE_MODE
MOV CL,3 ; SET #8 SHIFT COUNT
CMP CRT_MODE,4 ; DETERMINE IF GRAPHICS OR ALPHA
JB V4 ; ALPHA_PEN
;-----
; GRAPHICS MODE
MOV DL,40 ; DIVISOR FOR GRAPHICS
CMP CRT_MODE,9 ; USING 32K REGEN?
JB V20 ; NO, JUMP
MOV DL,80 ; YES, SET RIGHT DIVSOR
DIV DL ; DETERMINE ROW(AL) AND COLUMN(AH)
; AL RANGE 0-99, AH RANGE 0-39

```

```

F75E AB 02
F760 75 03
F762 E9 F80D R
F765 B4 10
F767 8B 16 0063 R
F76B 8A C4
F76D EE
F76E 42
F76F EC
F770 9A E8
F772 4A
F773 FE C4
F775 6A C4
F777 EE
F778 42
F779 EC
F77A 8A E5
F77C 9A 1E 0049 R
F780 2A FF
F782 2E: BA 9F F746 R
F787 2B C3
F789 30 0FA0
F78C 72 02
F78E 33 C0
F790 8B 1E 004E R
F794 D1 E0
F796 2B C3
F798 79 02
F79A 2B C0
F79C B1 03
F79E 80 3E 0049 R 04
F7A3 72 4A
F7A5 B2 28
F7A7 80 3E 0049 R 03
F7AC 72 02
F7AE B2 50
F7B0 F6 F2

```

```

;-----
; INPUT REGS POINTED TO BY AH, AND CONVERT TO ROW COLUMN IN DX
MOV DX,ADDR_6845 ; ADDRESS REGISTER FOR 6845
MOV AL,AH ; REGISTER TO READ
OUT DX,AL ; SET IT UP
INC DX ; DATA REGISTER
IN AL,DX ; GET THE VALUE
MOV CH,AL ; SAVE IN CX
DEC DX ; ADDRESS REGISTER
INC AH
MOV AL,AH ; SECOND DATA REGISTER
OUT DX,AL
INC DX ; POINT TO DATA REGISTER
IN AL,DX ; GET SECOND DATA VALUE
MOV AH,CH ; AX HAS INPUT VALUE
;-----
; AX HAS THE VALUE READ IN FROM THE 6845
MOV BL,CRT_MODE ; MODE VALUE TO BX
SUB BH,BH ; DETERMINE AMOUNT TO SUBTRACT
MOV BL,CS:V1(BX) ; TAKE IT AWAY
SUB AX,BX ; IN TOP OR BOTTOM BORDER?
CMP AX,4000 ; NO, OKAY
JB V15 ; YES, SET TO ZERO
XOR AX,AX
V15: MOV BX,CRT_START
SHR BX,1 ; CONVERT TO CORRECT PAGE ORIGIN
SUB AX,BX ; IF POSITIVE, DETERMINE MODE
JNS V2 ; <0 PLAYS AS 0
SUB AX,AX
;-----
; DETERMINE MODE OF OPERATION
V2: ; DETERMINE_MODE
MOV CL,3 ; SET #8 SHIFT COUNT
CMP CRT_MODE,4 ; DETERMINE IF GRAPHICS OR ALPHA
JB V4 ; ALPHA_PEN
;-----
; GRAPHICS MODE
MOV DL,40 ; DIVISOR FOR GRAPHICS
CMP CRT_MODE,9 ; USING 32K REGEN?
JB V20 ; NO, JUMP
MOV DL,80 ; YES, SET RIGHT DIVSOR
DIV DL ; DETERMINE ROW(AL) AND COLUMN(AH)
; AL RANGE 0-99, AH RANGE 0-39

```

```

F7B2 8A E8          ;----- DETERMINE GRAPHIC ROW POSITION
F7B4 02 E0          MOV     CH,AL          ; SAVE ROW VALUE IN CH
F7B6 80 3E 0049 R 09 ADD     CH,CH          ; #2 FOR EVEN/ODD FIELD
F7B8 72 06          CMP     CRT_MODE,9    ; USING 32K REGEN?
F780 00 EC          J8     V21            ; NO, JUMP
F78F 00 E0          SHR     AH,1          ; ADJUST ROW & COLUMN
F7C1 02 E0          SHL     AL,1          ;
F7C3 0A DC          ADD     CH,CH          ; #4 FOR 4 SCAN LINES
F7C5 8A DC          MOV     BL,AH          ; COLUMN VALUE TO BX
F7C7 2A FF          SUB     BH,BH          ; MULTIPLY BY 8 FOR MEDIUM RES
F7C9 80 3E 0049 R 08 CMP     CRT_MODE,6    ; DETERMINE MEDIUM OR HIGH RES
F7CB 72 15          J8     V3             ; MODE 4 OR 5
F7CD 77 06          JA     V23            ; MODE 8, 9, OR A
F7CF 81 04          MOV     CL,4          ; SHIFT VALUE FOR HIGH RES
F7D1 00 E4          SAL     AH,1          ; MULTIPLY VALUE TIMES 2 FOR HIGH RES
F7D3 E8 00          JMP     SHORT V3      ;
F7D5 80 3E 0049 R 09 CMP     CRT_MODE,9    ; CHECK MODE
F7D7 77 F3          JA     V22            ; MODE A
F7D9 74 04          JE     V3             ; MODE 9
F7DB 81 02          MOV     CL,2          ; MODE B SHIFT VALUE
F7DD 00 EC          SHR     AH,1          ;
F7E3 03 E3          V3:    SHL     BX,CL          ; NOT_HIGH_RES
F7E5 8A D4          SHL     BX,CL          ; MULTIPLY #16 FOR HIGH RES
;----- DETERMINE ALPHA CHAR POSITION
F7E7 8A F0          MOV     DL,AH          ; COLUMN VALUE FOR RETURN
F7E9 00 EE          MOV     DH,AL          ; ROW VALUE
F7EB 00 E8          SHR     DH,1          ; DIVIDE BY 4
F7ED 00 E2          SHR     DH,1          ; FOR VALUE IN 0-24 RANGE
F7EF 08 12          JMP     SHORT V5      ; LIGHT_PEN_RETURN_SET
;----- ALPHA MODE ON LIGHT PEN
F7F1 03 E3          V4:    MOV     AL,AH          ; ALPHA_PEN
F7F3 F8 3E 004A R 09 OI V    PTR CRT_COLS ; DETERMINE ROW, COLUMN VALUE
F7F5 8A F0          MOV     DH,AL          ; ROWS TO DH
F7F7 BA D4          MOV     DL,AH          ; COLS TO DL
F7F9 D2 E0          SAL     AL,CL          ; MULTIPLY ROWS * 8
F7FB 8A E8          MOV     CH,AL          ; GET RASTER VALUE TO RETURN REG
F7FD 8A DC          MOV     BL,AH          ; COLUMN VALUE
F7FF 32 FF          KOR     BH,BH          ; TO BX
F801 03 E3          V5:    SAL     BX,CL          ; LIGHT_PEN_RETURN_SET
F803 84 01          MOV     AH,1          ; INDICATE EVERYTHING SET
F805 03 E3          V6:    MOV     AL,AH          ; LIGHT_PEN_RETURN
F807 52            PUSH    DX             ; SAVE RETURN VALUE (IN CASE)
F809 8B 16 00B3 R 08 MOV     OX,ADDR_6B45 ; GET BASE ADDRESS
F80B 83 C2 07          ADD     OX,7           ; POINT TO RESET PARM
F80D 5E            OUT     OX,AL          ; ADDRESS, NOT DATA, IS IMPORTANT
F80F 00 EA          POP     DX             ; RECOVER VALUE
F811 03 E3          V7:    MOV     AL,AH          ; RETURN_NO_RESET
F813 0F            POP     DI             ;
F815 0F            POP     SI             ;
F817 0F            POP     DS             ; DISCARD SAVED BX,CX,DX
F819 0F            POP     OS             ;
F81B 0F            POP     OS             ;
F81D 0F            POP     DS             ;
F81F 0F            POP     ES             ;
F821 00 E9          IRET
;----- READ_LPEN
;----- TEMPORARY INTERRUPT SERVICE ROUTINE
; 1. THIS ROUTINE IS ALSO LEFT IN PLACE AFTER THE
; POWER ON DIAGNOSTICS TO SERVICE UNUSED
; INTERRUPT VECTORS. LOCATION 'INTR_FLAG' WILL
; CONTAIN EITHER: 1. LEVEL OF HARDWARE INT. THAT
; CAUSED CODE TO BE EXEC.
; 2. 'FF' FOR NON-HARDWARE INTERRUPTS THAT WERE
; EXECUTED ACCIDENTLY.
;-----
F825 01 11          PROC    NEAR
F827 00 00          ASSUME DS:DATA
F829 00 00          PUSH   DS
F82B 00 00          PUSH   AX              ; SAVE REG AX CONTENTS
F82D 00 00          CALL  DDS
F82F 00 00          MOV   AL,0BH          ; READ IN-SERVICE REG
F831 00 00          OUT  INTA00,AL        ; (FIND OUT WHAT LEVEL BEING
F833 00 00          NOP                                ; SERVICED)
F835 00 00          IN   AL,INTA00        ; GET LEVEL
F837 00 00          MOV  AH,AL            ; SAVE IT
F839 00 00          OR   AL,AH            ; DD? (NO HARDWARE ISR ACTIVE)
F83B 00 00          JNZ  HW_INT           ;
F83D 00 00          MOV  AH,OFFH          ;
F83F 00 00          JMP  SHORT SET_INTR_FLAG ; SET FLAG TO FF IF NON-HDWARE
F841 00 00          HW_INT: IN  AL,INTA01   ; GET MASK VALUE
F843 00 00          OR   AL,AH            ; MASK OFF LVL BEING SERVICED
F845 00 00          OUT  INTA01,AL
F847 00 00          MOV  AL,E01          ;
F849 00 00          OUT  INTA00,AL
F84B 00 00          SET_INTR_FLAG:
F84D 00 00          MOV  INTR_FLAG,AH    ; SET FLAG
F84F 00 00          POP  AX              ; RESTORE REG AX CONTENTS
F851 00 00          POP  DS
F853 00 00          STI                                ; INTERRUPTS BACK ON
F855 00 00          DUMMY_RETURN:
F857 00 00          IRET                    ; NEED IRET FOR VECTOR TABLE
F859 00 00          D11  ENDP

```

FB41
 FB41
 FB41 FB
 FB42 IE
 FB43 BB ---- R
 FB46 BE DB
 FB4B A1 0013 R
 FB4B 1F
 FB4C CF
 FB4D

```

----- INT 12 -----
MEMORY_SIZE_DETERMINE
INPUT
NO REGISTERS
THE MEMORY_SIZE VARIABLE IS SET DURING POWER ON DIAGNOSTICS
OUTPUT
(AH) = NUMBER OF CONTIGUOUS 1K BLOCKS OF MEMORY
-----
ASSUME CS:CODE,DS:DATA
ORG OFB41H
MEMORY_SIZE_DETERMINE PROC FAR
STI ; INTERRUPTS BACK DN
PUSH DS ; SAVE SEGMENT
MOV AX,DATA ; ESTABLISH ADDRESSING
MOV DS,AX
MOV AX,MEMORY_SIZE ; GET VALUE
POP DS ; RECOVER SEGMENT
IRET ; RETURN TO CALLER
MEMORY_SIZE_DETERMINE ENDP

```

FB4D
 FB4D
 FB4D FB
 FB4E IE
 FB4F BB ---- R
 FB52 BE DB
 FB54 A1 0010 R
 FB57 1F
 FB5B CF
 FB5B

```

----- INT 11 -----
EQUIPMENT_DETERMINATION
THIS ROUTINE ATTEMPTS TO DETERMINE WHAT OPTIONAL
DEVICES ARE ATTACHED TO THE SYSTEM.
INPUT
NO REGISTERS
THE EQUIP_FLAG VARIABLE IS SET DURING THE POWER ON
DIAGNOSTICS USING THE FOLLOWING HARDWARE ASSUMPTIONS:
PORT 62 (0->3) = LOW ORDER BYTE OF EQUIPMENT
PORT 3FA = INTERRUPT ID REGISTER OF 8250
BITS 7-3 ARE ALWAYS 0
PORT 37B = OUTPUT PORT OF PRINTER -- 8255 PORT THAT
CAN BE READ AS WELL AS WRITTEN
OUTPUT
(AH) IS SET, BIT SIGNIFICANT, TO INDICATE ATTACHED I/O
BIT 15,14 = NUMBER OF PRINTERS ATTACHED
BIT 13 = 1 = SERIAL PRINTER ATTACHEO
BIT 12 = GAME I/O ATTACHED
BIT 11,10,9 = NUMBER OF RS232 CARDS ATTACHED
BIT 8 0 = DMA CHIP PRESENT ON SYSTEM, 1 = NO DMA ON SYSTEM
BIT 7,6 = NUMBER OF DISKETTE DRIVES
00=1, 01=2, 10=3, 11=4 ONLY IF BIT 0 = 1
BIT 5,4 = INITIAL VIDEO MODE
00 - UNUSED
01 - 40X25 BW USING COLOR CARD
10 - 80X25 BW USING COLOR CARD
11 - 80X25 BW USING BW CARD
BIT 3,2 = PLANAR RAM SIZE (10=48K, 11=64K)
BIT 1 NOT USED
BIT 0 = 1 (IPL DISKETTE INSTALLED)
NO OTHER REGISTERS AFFECTED
-----
ASSUME CS:CODE,DS:DATA
ORG OFB4DH
EQUIPMENT PROC FAR
STI ; INTERRUPTS BACK ON
PUSH OS ; SAVE SEGMENT REGISTER
MOV AX,DATA ; ESTABLISH ADDRESSING
MOV DS,AX
MOV AX,EQUIP_FLAG ; GET THE CURRENT SETTINGS
POP DS ; RECOVER SEGMENT
IRET ; RETURN TO CALLER
EQUIPMENT ENDP

```

FB59
 FB59
 FB59 FB
 FB5A IE
 FB5B EB 1388 R
 FB5E 80 26 0071 R 7F
 FB63 EB FB6A R
 FB66 1F
 FB67 CA 0002
 FB6A
 FB6A

```

----- INT 15 -----
CASSETTE I/O
(AH) = 0 TURN CASSETTE MOTOR ON
(AH) = 1 TURN CASSETTE MOTOR OFF
(AH) = 2 READ 1 OR MORE 256 BYTE BLOCKS FROM CASSETTE
(ES,BX) = POINTER TO DATA BUFFER
(CX) = COUNT OF BYTES TO READ
ON EXIT
(ES,BX) = POINTER TO LAST BYTE READ + 1
(OX) = COUNT OF BYTES ACTUALLY READ
(CY) = 0 IF NO ERROR OCCURRED
= 1 IF ERROR OCCURRED
(AH) = ERROR RETURN IF (CY) = 1
= 01 IF CRC ERROR WAS DETECTED
= 02 IF DATA TRANSITIONS ARE LOST
= 04 IF NO DATA WAS FOUND
(AH) = 3 WRITE 1 OR MORE 256 BYTE BLOCKS TO CASSETTE
(ES,BX) = POINTER TO DATA BUFFER
(CX) = COUNT OF BYTES TO WRITE
ON EXIT
(EX,BX) = POINTER TO LAST BYTE WRITTEN + 1
(CX) = 0
(AH) = ANY OTHER THAN ABOVE VALUES CAUSES (CY)= 1
AND (AH)= 80 TO BE RETURNED (INVALID COMMAND).
-----
ASSUME DS:DATA, ES:NOTHING, SS:NOTHING, CS:CODE
ORG OFB59H
CASSETTE_I0 PROC FAR
STI ; INTERRUPTS BACK ON
PUSH DS ; ESTABLISH ADDRESSING TO DATA
CALL DDS
AND BIOS_BREAK, 7FH ; MAKE SURE BREAK FLAG IS OFF
CALL W1 ; CASSETTE_I0_CONT
POP DS
RET 2 ; INTERRUPT RETURN
CASSETTE_I0 ENDP
W1 PROC NEAR

```

```

-----
PURPOSE:
: TO CALL APPROPRIATE ROUTINE DEPENDING ON REG AH
: AH ROUTINE
-----
: 0 MOTOR ON
: 1 MOTOR OFF
: 2 READ CASSETTE BLCK
: 3 WRITE CASSETTE BLOCK
-----
F86A 0A E4 OR AH,AH ; TURN ON MOTOR?
F86C 74 13 JZ MOTOR_ON ; YES, DO IT
F86E FE CC DEC AH ; TURN OFF MOTOR?
F870 74 18 JZ MOTOR_OFF ; YES, DO IT
F872 FE CC DEC AH ; READ CASSETTE BLOCK?
F874 74 1A JZ READ_BLOCK ; YES, DO IT
F876 FE CC DEC AH ; WRITE CASSETTE BLOCK?
F878 75 03 JNZ W2 ; NOT_DEFINED
F87A E9 F997 R JMP WRITE_BLOCK ; YES, DO IT
F87D ; COMMAND NOT DEFINED
F87D B4 80 W2: MOV AH,0B0H ; ERROR, UNDEFINED OPERATION
F87F F9 STC ; ERROR FLAG
F880 C3 RET
F881 W1 EMDP
MOTOR_ON PROC NEAR
-----
PURPOSE:
: TO TURN ON CASSETTE MOTOR
-----
F881 E4 61 IN AL,PORT_B ; READ CASSETTE OUTPUT
F883 24 F7 AND AL,NOT_0BH ; CLEAR BIT TO TURN ON MOTOR
F885 E6 61 OUT PORT_B,AL ; WRITE IT OUT
F887 2A E4 SUB AH,AH ; CLEAR AH
F889 C3 RET
MOTOR_ON ENOP
MOTOR_OFF PROC NEAR
-----
PURPOSE:
: TO TURN CASSETTE MOTOR OFF
-----
F88A E4 61 IN AL,PORT_B ; READ CASSETTE OUTPUT
F88C 0C 08 OR AL,0BH ; SET BIT TO TURN OFF
F88E EB F5 JMP W3 ; WRITE IT, CLEAR ERROR, RETURN
F890 W3: MOTOR_OFF ENOP
READ_BLOCK PROC NEAR
-----
PURPOSE:
: TD READ 1 OR MORE 256 BYTE BLOCKS FROM CASSETTE
-----
ON ENTRY:
: ES IS SEGMENT FOR MEMORY BUFFER (FOR COMPACT CODE)
: BX POINTS TO START OF MEMORY BUFFER
: CX CONTAINS NUMBER OF BYTES TO READ
ON EXIT:
: BX POINTS 1 BYTE PAST LAST BYTE PUT IN MEM
: CX CONTAINS DECREMENTED BYTE COUNT
: DX CONTAINS NUMBER OF BYTES ACTUALLY READ
:
: CARRY FLAG IS CLEAR IF NO ERROR DETECTED
: CARRY FLAG IS SET IF CRC ERROR DETECTED
-----
F890 53 PUSH BX ; SAVE BX
F891 51 PUSH CX ; SAVE CX
F892 56 PUSH SI ; SAVE SI
F893 BE 0007 MOV SI,7 ; SET UP RETRY COUNT FOR LEADER
F896 EB FAS0 R CALL BEGIN_OP ; BEGIN BY STARTING MOTOR
F899 W4: SEARCH FOR LEADER
F899 E4 82 IN AL,PORT_C ; GET INITIAL VALUE
F89B 24 10 AND AL,010H ; MASK OFF EXTRANEUS BITS
F89D A2 006B R MOV LAST_VAL,AL ; SAVE IN LOC LAST_VAL
F8A0 BA 3F7A MOV DX,16250 ; # OF TRANSITIONS TO LOOK FOR
F8A3 W5: WAIT_FOR_EDGE
F8A3 F6 06 0071 R 80 TEST 8105_BREAK, BOH ; CHECK FOR BREAK KEY
F8A8 75 03 JNZ W6A ; JUMP IF NO BREAK KEY
; JUMP IF BREAK KEY HIT
F8AA 4A DEC DX ;
F8AB 75 03 JNZ W7 ; JUMP IF BEGINNING OF LEADER
F8AD E9 F92F R W6A: JMP W17 ; JUMP IF NO LEADER FOUND
F8B0 EB F96F R W7: CALL READ_HALF_BIT ; IGNORE FIRST EDGE
F8B3 E3 EE JCXZ W5 ; JUMP IF NO EDGE DETECTED
F8B5 BA 0378 MOV DX,0378H ; CHECK FOR HALF BITS
F8B8 89 0200 MOV CX,200H ; MUST HAVE AT LEAST THIS MANY ONE
; SIZE PULSES BEFORE CHCKNG FOR
; SYNC BIT (0)
F8BB FA CLI ; DISABLE INTERRUPTS
F8BC W8: SEARCH-LDR
F8BC F6 06 0071 R 80 TEST 8105_BREAK, BOH ; CHECK FOR BREAK KEY
F8C1 75 6C JNZ W17 ; JUMP IF BREAK KEY HIT
F8C3 51 PUSH CX ; SAVE REG CX
F8C4 EB F96F R CALL READ_HALF_BIT ; GET PULSE WIDTH
F8C7 08 C9 OR CX,CX ; CHECK FOR TRANSITION
F8C9 59 POP CX ; RESTORE ONE BIT COUNTER
F8CA 74 CD JZ W4 ; JUMP IF NO TRANSITION
F8CC 3B D3 CMP DX,BX ; CHECK PULSE WIDTH
F8CE E3 04 JCXZ W9 ; IF CX=0 THEN WE CAN LOOK
; FOR SYNC BIT (0)
F8D0 73 C7 JNC W4 ; JUMP IF ZERO BIT (NOT GOOD
; LEADER)
F8D2 E2 EB LOOP W8 ; DEC CX AND READ ANOTHER HALF ONE
; BIT
F8D4 W9: FIND-SYNC
F8D4 72 E6 JC W8 ; JUMP IF ONE BIT (STILL LEADER)

```

```

;----- A SYNCH BIT HAS BEEN FOUND.  READ SYN CHARACTER:
F8D6 E8 F96F R      CALL READ_HALF_BIT ; SKIP OTHER HALF OF SYNC BIT (0)
F8D9 E8 F941 R      CALL READ_BYTE   ; READ SYNC BYTE
F8DC 3C 16          CMP AL, 16H      ; SYNCHRONIZATION CHARACTER
F8DE 75 49          JNE W16      ; JUMP IF BAD LEADER FOUND.

;----- GOOD CRC SO READ DATA BLOCK(S)
FBE0 5E            POP SI          ; RESTORE REGS
FBE1 59            POP CX
FBE2 58            POP BX

;-----
; READ 1 OR MORE 256 BYTE BLOCKS FROM CASSETTE
; ON ENTRY:
; ES IS SEGMENT FOR MEMORY BUFFER (FOR COMPACT CODE)
; BX POINTS TO START OF MEMORY BUFFER
; CX CONTAINS NUMBER OF BYTES TO READ
; ON EXIT:
; BX POINTS 1 BYTE PAST LAST BYTE PUT IN MEM
; CX CONTAINS DECREMENTED BYTE COUNT
; DX CONTAINS NUMBER OF BYTES ACTUALLY READ
;-----
FBE3 51            PUSH CX          ; SAVE BYTE COUNT
FBE4              W10:      ; COME HERE BEFORE EACH
;                   ; 256 BYTE BLOCK IS READ
FBE4 C7 06 0069 R FFFF MOV CRC_REG,0FFFFH ; INIT CRC REG
FBEA 8A 0100        MOV DX,256      ; SET DX TO DATA BLOCK SIZE
FBE0              W11:      ; RD_BLK
FBE8 F6 06 0071 R 80 TEST BIOS_BREAK, 80H ; CHECK FOR BREAK KEY
FBF2 75 23          JNZ W13      ; JUMP IF BREAK KEY HIT
FBF4 E8 FB41 R      CALL READ_BYTE   ; READ BYTE FROM CASSETTE
FBF7 72 1E          JC W13       ; CY SET INDICATES NO DATA
;                   ; TRANSITIONS
F8F9 E3 06          JCXZ W12      ; IF WE'VE ALREADY REACHED
;                   ; END OF MEMORY BUFFER
;                   ; SKIP REST OF BLOCK
F8FB 26: 8B 07      MOV ES:[BX],AL    ; STORE DATA BYTE AT BYTE PTR
F8FE 43             INC BX          ; INC BUFFER PTR
F8FF 49             DEC CX          ; DEC BYTE COUNTER
F900              W12:      ; LOOP UNTIL DATA BLOCK HAS BEEN READ FROM CASSETTE
F900 4A             DEC DX          ; DEC BLOCK CNT
F901 7F EA          JG W11         ; RD_BLK
F903 E8 F941 R      CALL READ_BYTE   ; NOW READ TWO CRC BYTES
F906 E8 F941 R      CALL READ_BYTE
F909 2A E4          SUB AH,AH      ; CLEAR AH
F908 81 3E 0069 R 100F CMP CRC_REG,100FH ; IS THE CRC CORRECT?
F911 75 06          JNE W14        ; IF NOT EQUAL CRC IS BAD
F913 E3 06          JCXZ W15        ; IF BYTE COUNT IS ZERO
;                   ; THEN WE HAVE READ ENOUGH
;                   ; SO WE WILL EXIT
F915 E8 CD          JMP W10        ; STILL MORE, SO READ ANOTHER BLOCK
F917              W13:      ; MISSING-DATA
;                   ; NO DATA TRANSITIONS SO
;                   ; END OF MEMORY BUFFER
;                   ; SET AH=02 TO INDICATE
F917 84 01          MOV AH,01H     ; DATA TIMEOUT
F819              W14:      ; BAD-CRC
F919 FE C4          INC AH         ; EXIT EARLY ON ERROR
;                   ; SET AH=01 TO INDICATE CRC ERROR
F918              W15:      ; RD-BLK-EX
F918 5A             POP DX         ; CALCULATE COUNT OF
F91C 2B D1          SUB DX,CX      ; DATA BYTES ACTUALLY READ.
;                   ; RETURN COUNT IN REG DX
F91E 50             PUSH AX        ; SAVE AX (RET CODE)
F91F F6 C4 90      TEST AH,90H   ; CHECK FOR ERRORS
F922 75 13          JNZ W18        ; JUMP IF ERROR DETECTED
F924 EB F941 R      CALL READ_BYTE   ; READ TRAILER
F927 EB 0E          JMP SHORT W18   ; SKIP TO TURN OFF MOTOR
F929              W16:      ; BAD-LEADER
F929 4E             DEC SI        ; CHECK RETRIES
F92A 74 03          JZ W17        ; JUMP IF TOO MANY RETRIES
F92C E9 F899 R      JMP W4        ; JUMP IF NOT TOO MANY RETRIES
F92F              W17:      ; NO VALID DATA FOUND
;                   ; NO DATA FROM CASSETTE ERROR, I.E. TIMEOUT
F92F 5E             POP SI        ; RESTORE REGS
F930 59            POP CX        ; RESTORE REGS
F931 5B            POP BX
F932 2B D2          SUB DX,DX     ; ZERO NUMBER OF BYTES READ
F934 84 04          MOV AH,04H   ; TIME OUT ERROR (NO LEADER)
F936 50             PUSH AX
;                   ; NOT-OFF
F937              W18:      ; REENABLE INTERRUPTS
F937 FB             STI          ; TURN OFF MOTOR
F938 EB FB8A R      CALL MOTOR_OFF
F93B 5B            POP AX        ; RESTORE RETURN CODE
F93C 80 FC 01      CMP AH,01H   ; SET CARRY IF ERROR (AH=0)
F93F F5             CMC
F940 C3             RET
F941              READ_BLOCK ENDP

```

```

PURPOSE:
; TO READ A BYTE FROM CASSETTE
; ON EXIT
; REG AL CONTAINS READ DATA BYTE
-----
F941          READ_BYTE  PROC   NEAR
F941 53      PUSH   BX           ; SAVE REGS BX,CX
F942 51      PUSH   CX
F943 B1 08   MOV    CL,8H          ; SET BIT COUNTER FOR 8 BITS
F945 51      W19:  PUSH   CX           ; BYTE=ASH
; SAVE CX
-----
; READ DATA BIT FROM CASSETTE
-----
F946 E8 F96F R CALL  READ_HALF_BIT  ; READ ONE PULSE
F949 E3 20   JCXZ  W21           ; IF CX=0 THEN TIMEOUT
; BECAUSE OF NO DATA TRANSITIONS
; SAVE 1ST HALF BIT'S
; PULSE WIDTH (IN BX)
F94B 53      PUSH   BX           ; READ COMPLEMENTARY PULSE
; COMPUTE DATA BIT
F94C E8 F96F R CALL  READ_HALF_BIT  ; IF CX=0 THEN TIMEOUT DUE TO
F94F 5B      POP    AX           ; NO DATA TRANSITIONS
; PERIOD
F950 E3 19   JCXZ  W21           ; CHECK FOR ZERO BIT
; CARRY IS SET IF ONE BIT
F952 03 08   ADD    BX,AX           ; SAVE CARRY IN AH
F954 B1 FB 06F0 CMP  BX, 06F0H       ; RESTORE CX
F95B F5      CMC                    ; NOTE:
F959 9F      LAHF                    ; MS BIT OF BYTE IS READ FIRST.
F95A 59      POP    CX           ; REG CH IS SHIFTED LEFT WITH
; CARRY BEING INSERTED INTO LS
; BIT OF CH.
; AFTER ALL 8 BITS HAVE BEEN
; READ, THE MS BIT OF THE DATA
; BYTE WILL BE IN THE MS BIT OF
; REG CH
; ROTATE REG CH LEFT WITH CARRY TO
; LS BIT OF REG CH
F95B D0 D5   RCL    CH,1           ; RESTORE CARRY FOR CRC ROUTINE
; GENERATE CRC FOR BIT
F95D 9E      SAHF                    ; LOOP TILL ALL 8 BITS OF DATA
F95E EB FA3C R CALL  CRC_GEN         ; ASSEMBLED IN REG CH
F961 FE C9   DEC    CL           ; BYTE_ASH
; RETURN DATA BYTE IN REG AL
F963 75 E0   JNZ    W19           ; RD-BYT-EX
F965 BA C5   MOV    AL,CH         ; RESTORE REGS CX,BX
F967 F8      CLC                    ;
F968 59      W20:  POP    CX           ;
F969 58      POP    BX           ;
F96A C3      RET                    ; FINISHED
F96B 59      W21:  NO-DATA
F96C F9      POP    CX           ; RESTORE CX
F96D E8 F8   STC                    ; INDICATE ERROR
F96F          JMP    W20           ; RD_BYT_EX
-----
READ_BYTE  ENDP
-----
PURPOSE:
; TO COMPUTE TIME TILL NEXT DATA
; TRANSITION (EDGE)
; ON ENTRY:
; EDGE_CNT CONTAINS LAST EDGE COUNT
; ON EXIT:
; AX CONTAINS OLD LAST EDGE COUNT
; BX CONTAINS PULSE WIDTH (HALF BIT)
-----
F96F          READ_HALF_BIT  PROC   NEAR
F96F 89 0064 MOV    CX,100         ; SET TIME TO WAIT FOR 8IT
F972 BA 26 006B R MOV    AH,LAST_VAL   ; GET PRESENT INPUT VALUE
F976          W22:  RD-H-BIT
F976 E4 62   IN    AL,PORT_C     ; INPUT DATA BIT
F978 24 10   AND    AL,010H       ; MASK OFF EXTRANEIOUS BITS
F97A 3A C4   CMP    AL,AH         ; SAME AS BEFORE?
F97C E1 F8   LOOPE  W22          ; LOOP TILL IT CHANGES
F97E A2 006B R MOV    LAST_VAL,AL   ; UPDATE LAST_VAL WITH NEW VALUE
F981 90 40   MOV    AL,40H       ; READ TIMER'S COUNTER COMMAND
F983 E6 43   OUT    TIM_CTL,AL ; LATCH COUNTER
F985 8B 1E 0067 R MOV    BX,EDGE_CNT   ; BX GETS LAST EDGE COUNT
F889 E4 41   IN    AL,TIMER+1    ; GET LS BYTE.
F98B 8A E0   MOV    AH,AL         ; SAVE IN AH
F990 E4 41   IN    AL,TIMER+1    ; GET MS BYTE
F99F B6 C4   XCHG  AL,AH         ; XCHG AL,AH
F991 28 D8   SUB    BX,AX         ; SET BX EQUAL TO HALF BIT PERIOD
F993 A3 0067 R MOV    EDGE_CNT,AX  ; UPDATE EDGE COUNT;
F996 C3      RET                    ;
F997          READ_HALF_BIT  ENDP

```



```

-----
; PURPOSE
; WRITE 1 OR MORE 256 BYTE BLOCKS TO CASSETTE.
; THE DATA IS PADDED TO FILL OUT THE LAST 256 BYTE BLOCK.
; ON ENTRY:
; BX POINTS TO MEMORY BUFFER ADDRESS
; CX CONTAINS NUMBER OF BYTES TO WRITE
; ON EXIT:
; BX POINTS 1 BYTE PAST LAST BYTE WRITTEN TO CASSETTE
; CX IS ZERO

```

```

F997
F997 53          WRITE_BLOCK PROC NEAR
F998 51          PUSH BX
F999 E4 61      IN AL,PORT_B ; DISABLE SPEAKER
F99B 24 F0      AND AL,NOT 02H
F99D 0C 01      OR AL,01H ; ENABLE TIMER
F99F E6 61      OUT PORT_B,AL
F9A1 80 B6      MOV AL,0B6H ; SET UP TIMER - MODE 3 SQUARE WAVE
F9A3 E6 43      OUT TIM_CTL,AL
F9A5 EB FA50 R  CALL BEGIN_OP ; START MOTOR AND DELAY
F9A8 B9 04A0    MOV AX,1184 ; SET NORMAL BIT SIZE
F9AB EB FA35 R  CALL W31 ; SET TIMER
F9AE B9 0800    MOV CX,0800H ; SET CX FOR LEADER BYTE COUNT
F9B1           W23: ; WRITE LEADER
F9B1 F9          STC ; WRITE ONE BITS
F9B2 EB FA1F R  CALL WRITE_BIT
F9B5 E2 FA      LOOP W23 ; LOOP 'TIL LEADER IS WRITTEN
F9B7 FA          CLI ; DISABLE INTS.
F9B8 FB          CLC ; WRITE SYNC BIT (0)
F9B9 EB FA1F R  CALL WRITE_BIT
F9BC B9          POP CX ; RESTORE REGS CX,BX
F9BD 5B          POP BX
F9BE 80 B6      MOV AL,16H ; WRITE SYNC CHARACTER
F9C0 EB FA08 R  CALL WRITE_BYTE

```

```

-----
; PURPOSE
; WRITE 1 OR MORE 256 BYTE BLOCKS TO CASSETTE
; ON ENTRY:
; BX POINTS TO MEMORY BUFFER ADDRESS
; CONTAINS NUMBER OF BYTES TO WRITE
; ON EXIT:
; BX POINTS 1 BYTE PAST LAST BYTE WRITTEN TO CASSETTE
; CX IS ZERO

```

```

F9C3
F9C3 C7 06 0069 R FFFF
F9C9 BA 0100
F9CC
F9CC 26: BA 07
F9CF EB FA0B R
F9D2 E3 02
F9D4 43
F9D5 49
F9D6
F9D6 4A
F9D7 7F F3
F9D9
F9D9 A1 0069 R
F9DC F7 D0
F9DE 50
F9DF 86 E0
F9E1 EB FA0B R
F9E4 58
F9E5 EB FA0B R
F9E8 08 C9
F9EA 75 D7
F9EC B1
F9ED FB
F9EE B9 0020
F9F1
F9F1 F9
F9F2 EB FA1F R
F9F5 E2 FA
F9F7 59
F9F8 80 80
F9FA E6 43
F9FC B9 0001
F9FF E9 FA35 R
FA02 EB FB8A R
FA05 2B C0
FA07 C3
FA08
F9C3           WR_BLOCK:
F9C3           MOV CRC_REG,OFFFHH ; INIT CRC
F9C9           MOV DX,256 ; FOR 256 BYTES
F9CC           W24: ; WR-BLK
F9CC           MOV AL,ES:[BX] ; READ BYTE FROM MEM
F9CF           CALL WRITE_BYTE ; WRITE IT TO CASSETTE
F9D2           JCXZ W2B ; UNLESS CX=0, ADVANCE PTRS & DEC
F9D4           ; COUNT
F9D5           INC BX ; INC BUFFER POINTER
F9D6           DEC CX ; DEC BYTE COUNTER
F9D9           W25: ; SKIP-ADV
F9D9           DEC DX ; DEC BLOCK CNT
F9DA           JG W24 ; LOOP TILL 256 BYTE BLOCK
F9DB           ; IS WRITTEN TO TAPE

```

```

-----
; WRITE CRC
; WRITE 1'S COMPLEMENT OF CRC REG TO CASSETTE
; WHICH IS CHECKED FOR CORRECTNESS WHEN THE BLOCK IS READ
; REG AX IS MODIFIED

```

```

F9D9           MOV AX,CRC_REG ; WRITE THE ONE'S COMPLEMENT OF THE
F9DC           NOT AX ; TWO BYTE CRC TO TAPE
F9DE           PUSH AX ; FOR 1'S COMPLEMENT
F9DF           XCHG AH,AL ; SAVE IT
F9E1           CALL WRITE_BYTE ; WRITE MS BYTE FIRST
F9E4           POP AX ; WRITE IT
F9E5           CALL WRITE_BYTE ; GET IT BACK
F9E8           OR CX,CX ; NOW WRITE LS BYTE
F9EA           JNZ WR_BLOCK ; IS BYTE COUNT EXHAUSTED?
F9EC           PUSH CX ; JUMP IF NOT DONE YET
F9ED           STI ; SAVE REG CX
F9EE           MOV CX,32 ; RE-ENABLE INTERRUPTS
F9F1           W26: ; WRITE OUT TRAILER BITS
F9F1           STC ; TRAIL-LOOP
F9F2           CALL WRITE_BIT
F9F5           LOOP W26 ; WRITE UNTIL TRAILER WRITTEN
F9F7           POP CX ; RESTORE REG CX
F9F8           MOV AL,080H ; TURN TIMERZ OFF
F9FA           OUT TIM_CTL,AL
F9FC           MOV AX,1
F9FF           CALL W31 ; SET TIMER
FA02           CALL MOTOR_OFF ; TURN MOTOR OFF
FA05           SUB AX,AX ; NO ERRORS REPORTED ON WRITE OP
FA07           RET ; FINISHED
FA08           WRITE_BLOCK ENDP

```

```

;-----
; WRITE A BYTE TO CASSETTE.
; BYTE TO WRITE IS IN REG AL.
;-----
FA08          WRITE_BYTE  PROC  NEAR
FA08  51      PUSH  CX          ; SAVE REGS CX,AX
FA09  50      PUSH  AX
FA0A  8A E8   MOV   CH,AL      ; AL=BYTE TO WRITE.
; (MS BIT WRITTEN FIRST)
; FOR 8 DATA BITS IN BYTE.
; NOTE: TWO EDGES PER BIT
FA0C  B1 08   MOV   CL,8      ; DISASSEMBLE THE DATA BIT
FA0E          W27:          RCL  CH,1      ; ROTATE MS BIT INTO CARRY
FA0E  D0 D5   PUSHF          ; SAVE FLAGS.
FA10  9C      ; NOTE: DATA BIT IS IN CARRY
FA11  E8 FA1F R CALL  WRITE_BIT    ; WRITE DATA BIT
FA14  9D      POPF          ; RESTORE CARRY FOR CRC CALC
FA15  E9 FA3C R CALL  CRC_GEN      ; COMPUTE CRC ON DATA BIT
FA1B  FE C9   DEC   CL        ; LOOP TILL ALL 8 BITS DONE
FA1A  75 F2   JNZ  W27      ; JUMP IF NOT DONE YET
FA1C  58      POP  AX        ; RESTORE REGS AX,CX
FA1D  59      POP  CX
FA1E  C3      RET           ; WE ARE FINISHED
FA1F          WRITE_BYTE  ENDP
;-----
FA1F          WRITE_BIT  PROC  NEAR
; PURPOSE:
; TO WRITE A DATA BIT TO CASSETTE
; CARRY FLAG CONTAINS DATA BIT
; I.E. IF SET DATA BIT IS A ONE
; IF CLEAR DATA BIT IS A ZERO
; NOTE: TWO EDGES ARE WRITTEN PER BIT
; ONE BIT HAS 500 USEC BETWEEN EDGES
; FOR A 1000 USEC PERIOD (1 MILLISEC)
; ZERO BIT HAS 250 USEC BETWEEN EDGES
; FOR A 500 USEC PERIOD (.5 MILLISEC)
; CARRY FLAG IS DATA BIT
;-----
FA1F  88 04A0 MOV  AX,1184      ; ASSUME IT'S A '1'
FA22  72 03   JC   W29      ; SET AX TO NOMINAL ONE SIZE
FA24  88 0250 MOV  AX,592       ; JUMP IF ONE BIT
FA27          W25:          ; NO, SET TO NOMINAL ZERO SIZE
FA27  50      PUSH  AX        ; WRITE-BIT-AX
; WRITE BIT WITH PERIOD EQ TO VALUE
; AX
FA28  E4 62   W29:  IN   AL,PORT_C  ; INPUT TIMER_0 OUTPUT
FA2A  24 20   AND  AL,020H      ; LOOP TILL HIGH
FA2C  74 FA   JZ   W30      ; NOW WAIT TILL TIMER'S OUTPUT IS
FA2E  E4 62   W30:  IN   AL,PORT_C  ; LOW
FA30  24 20   AND  AL,020H
FA32  75 FA   JNZ  W30      ; RELOAD TIMER WITH PERIOD
; FOR NEXT DATA BIT
; RESTORE PERIOD COUNT
FA34  5B      POP  AX        ; SET TIMER
FA35  E6 42   W31:  OUT  042H,AL    ; SET LOW BYTE OF TIMER 2
FA37  8A C4   MOV  AL,AH        ; SET HIGH BYTE OF TIMER 2
FA39  E6 42   OUT  042H,AL
FA3B  C3      RET
FA3C          WRITE_BIT  ENDP
;-----
FA3C          CRC_GEN    PROC  NEAR
; UPDATE CRC REGISTER WITH NEXT DATA BIT
; CRC IS USED TO DETECT READ ERRORS
; ASSUMES DATA BIT IS IN CARRY
; REG AX IS MODIFIED
; FLAGS ARE MODIFIED
;-----
FA3C  A1 0069 R MOV  AX,CRC_REG
; THE FOLLOWING INSTUCTIONS
; WILL SET THE OVERFLOW FLAG
; IF CARRY AND MS BIT OF CRC
; ARE UNEQUAL
FA3F  D1 D8   RCR  AX,1
FA41  D1 D0   RCL  AX,1
FA43  F8      CLC
FA44  71 04   JNO  W32      ; CLEAR CARRY
; SKIP IF NO OVERFLOW
; IF DATA BIT XORED WITH
; CRC REG BIT IS IS ONE
; THEN XOR CRC REG WITH
; 0B10H
FA46  35 0810 XOR  AX,0810H
FA49  F9      STC
FA4A  D1 D0   W32:  RCL  AX,1
; SET CARRY
; ROTATE CARRY (DATA BIT)
; INTO CRC_REG
FA4C  A3 0069 R MOV  CRC_REG,AX
FA4F  C3      RET           ; UPDATE CRC_REG
FA50          CRC_GEN    ENDP
;-----

```

```

FA50
FA50 EB F8B1 R
FA53 B3 42

FA55 B9 0700
FA5B E2 FE
FA5A FE CB
FA5C 75 F7
FA5E C3
FA5F

FA5F 33 D2
FA61 32 E4

FA63 80 0D
FA65 CD 17
FA67 32 E4
FA69 8D 0A
FA6B CD 17
FA6D C3
FA6E

```

```

-----
BEGIN_OP      PROC      NEAR      ; START TAPE AND DELAY
              CALL      MOTOR_ON    ; TURN ON MOTOR
              MOV      BL,42H       ; DELAY FOR TAPE DRIVE
              ; TO GET UP TO SPEED (1/2 SEC)
              ; INNER LOOP= APPROX. 10 MILLISEC

W33:  MOV      CX,700H
W34:  LOOP     W34
      DEC      BL
      JNZ     W33
      RET

BEGIN_OP      ENDP
;----- CARRIAGE RETURN, LINE FEED SUBROUTINE
CRLF  PROC      NEAR
      XOR      DX,DX                ; PRINTER 0
      XOR      AH,AH                ; WILL NOW SEND INITIAL LF,CR TO
      ; PRINTER
      MOV      AL,0DH                ; CR
      INT      17H                  ; SEND THE LINE FEED
      XOR      AH,AH                ; NOW FOR THE CR
      MOV      AL,0AH                ; LF
      INT      17H                  ; SEND THE CARRIAGE RETURN
      RET
CRLF  ENDP
-----

```

```

-----
; CHARACTER GENERATOR GRAPHICS FOR 320X200 AND 640X200
; GRAPHICS FOR CHARACTERS 00H THRU 7FH
-----

```

FA6E	ORG	DFASEH	
FA6E	CRT_CHAR_GEN	LABEL	BYTE
00 00	00	000H,000H	000H,000H,000H,000H,000H,000H ; D_00
FA76 7E 81 A5 81 8D 99	01	07EH,0B1H,0A5H,0B1H,0BDH,099H,0B1H,07EH	; D_01
81 7E			
FA7E 7E FF DB FF C3 E7	02	07EH,0FFH,0DBH,0FFH,0C3H,0E7H,0FFH,07EH	; D_02
FF 7E			
FA86 6C FE FE FE 7C 38	03	06CH,0FEH,0FEH,0FEH,07CH,03BH,010H,000H	; D_03
10 00			
FA8E 10 38 7C FE 7C 38	04	010H,03BH,07CH,0FEH,07CH,03BH,010H,000H	; D_04
10 00			
FA96 38 7C 38 FE C3 7C	05	03BH,07CH,03BH,0FEH,0FEH,07CH,03BH,07CH	; D_05
38 7C			
FA9E 10 10 38 7C FE 7C	06	010H,010H,03BH,07CH,0FEH,07CH,03BH,07CH	; D_06
38 7C			
FAA0 00 00 19 3C 3C 18	07	000H,000H,01BH,03CH,03CH,01BH,000H,000H	; D_07
00 00			
FAAE FF FF E7 C3 C3 E7	08	0FFH,0FFH,0E7H,0C3H,0C3H,0E7H,0FFH,0FFH	; D_08
FF FF			
FA86 00 3C 66 42 42 66	09	000H,03CH,066H,042H,042H,066H,03CH,000H	; D_09
3C 00			
FA8E FF C3 99 8D 8D 99	0A	0FFH,0C3H,099H,08DH,08DH,099H,0C3H,0FFH	; D_0A
C3 FF			
FA6C 0F 07 0F 7D CC CC	0B	00FH,007H,00FH,07DH,0CCH,0CCH,0CCH,07BH	; D_0B
CC 78			
FACE 3C 66 66 66 3C 18	0C	03CH,066H,066H,066H,03CH,01BH,07EH,01BH	; D_0C
7E 18			
FAD6 3F 33 3F 30 30 70	0D	03FH,033H,03FH,030H,030H,070H,0F0H,0E0H	; D_0D
F0 E0			
FADE 7F 63 7F 63 63 67	0E	07FH,063H,07FH,063H,063H,067H,0E6H,0C0H	; D_0E
E6 C0			
FAE6 99 5A 3C E7 E7 3C	0F	099H,05AH,03CH,0E7H,0E7H,03CH,05AH,099H	; D_0F
5A 99			
FAEE 80 E0 F8 FE F8 E0	10	080H,0E0H,0F8H,0FEH,0F8H,0EDH,080H,000H	; D_10
80 00			
FAF6 02 0E 3E FE 3E 0E	11	002H,00EH,03EH,0FEH,03EH,00EH,002H,000H	; D_11
02 00			
FAFE 18 3C 7E 18 18 7E	12	01BH,03CH,07EH,01BH,01BH,07EH,03CH,01BH	; D_12
3C 18			
F806 66 66 86 66 86 00	13	066H,066H,066H,066H,066H,000H,066H,000H	; D_13
66 00			
F80E 7F D8 D8 7B 1B 1B	14	07FH,0DBH,0DBH,07BH,01BH,01BH,01BH,000H	; D_14
1B 00			
F816 3E 63 8B 8C 8C 3B	15	03EH,063H,03BH,08CH,06CH,03BH,0CCH,07BH	; D_15
CC 78			
FB1E 00 00 00 00 7E 7E	16	000H,000H,000H,000H,07EH,07EH,07EH,000H	; D_16
7E 00			
FB26 1B 3C 7E 1B 7E 3C	17	01BH,03CH,07EH,01BH,07EH,03CH,01BH,0FFH	; D_17
18 FF			
FB2E 1B 3C 7E 1B 1B 18	18	01BH,03CH,07EH,01BH,01BH,01BH,01BH,000H	; D_18
18 00			
F836 18 18 18 19 7E 3C	19	01BH,01BH,01BH,01BH,07EH,03CH,01BH,000H	; D_19
18 00			
FB3E 00 1B 0C FE 0C 1B	1A	000H,01BH,00CH,0FEH,00CH,01BH,000H,000H	; D_1A
00 00			
FB46 00 30 60 FE 60 30	1B	000H,030H,060H,0FEH,060H,030H,000H,000H	; D_1B
00 00			
FB4E 00 00 C0 C0 C0 FE	1C	000H,000H,0C0H,0C0H,0C0H,0FEH,000H,000H	; D_1C
00 00			
F856 00 24 86 FF 66 24	1D	000H,024H,066H,0FFH,066H,024H,000H,000H	; D_1D
00 00			
F85E 00 18 3C 7E FF FF	1E	000H,01BH,03CH,07EH,0FFH,0FFH,000H,000H	; D_1E
00 00			
F866 00 FF FF 7E 3C 18	1F	000H,0FFH,0FFH,07EH,03CH,01BH,000H,000H	; D_1F
00 00			

FB6E	00 00 00 00 00 00	DB	000H,000H,000H,000H,000H,000H,000H,000H ; SP D_20
	00 00		
FB78	30 78 78 30 30 00	DB	030H,078H,078H,030H,030H,000H,030H,000H ; ! D_21
	30 00		
FB7E	6C 6C 6C 00 00 00	DB	06CH,06CH,06CH,000H,000H,000H,000H,000H ; " D_22
	00 00		
FB86	6C 6C FE 6C FE 6C	DB	06CH,06CH,0FEH,06CH,0FEH,06CH,06CH,000H ; # D_23
	6C 00		
FB8E	30 7C 0C 78 0C F8	DB	030H,07CH,0C0H,078H,00CH,0FBH,030H,000H ; \$ D_24
	30 00		
FB96	00 C8 CC 1B 30 66	DB	000H,0C6H,0CCH,018H,030H,068H,0CBH,000H ;
	C6 00		
			PER CENT D_25
FB9E	38 6C 38 76 DC CC	DB	038H,06CH,038H,076H,00CH,0CCH,076H,000H ; & D_26
	76 00		
FBA6	60 60 C0 00 00 00	DB	060H,060H,0C0H,000H,000H,000H,000H,000H ; ' D_27
	00 00		
FBAE	18 30 60 60 60 30	DB	018H,030H,060H,060H,060H,030H,018H,000H ; (D_28
	18 00		
FBB6	60 30 1B 1B 1B 30	DB	060H,030H,018H,018H,018H,030H,060H,00DH ;) D_29
	60 00		
FBBE	00 66 3C FF 3C 66	DB	000H,066H,03CH,0FFH,03CH,066H,000H,000H ; # D_2A
	00 00		
FBC6	00 30 FC 30 30 30	DB	000H,030H,030H,0FCH,030H,030H,000H,000H ; + D_2B
	00 00		
FBC E	00 00 00 00 30 30	DB	000H,000H,000H,000H,00DH,030H,030H,060H ; , D_2C
	30 60		
FBD6	00 00 00 FC 00 00	DB	000H,000H,000H,0FCH,000H,000H,000H,000H ; - D_2D
	00 00		
FBD E	00 00 00 00 00 30	DB	000H,000H,000H,000H,000H,030H,030H,000H ; . D_2E
	30 00		
FBE6	06 0C 1B 30 60 C0	DB	006H,00CH,018H,030H,060H,0C0H,08DH,000H ; / D_2F
	80 00		
			0 D_30
FBE E	7C C6 CE DE F6 E6	DB	07CH,0C6H,0CEH,0DEH,0F6H,0E6H,07CH,000H ; 0
	7C 00		
FBF6	30 70 30 30 30 30	DB	030H,070H,030H,030H,030H,030H,0FCH,000H ; 1 D_31
	FC 00		
FBF E	78 CC 0C 3B 60 CC	DB	078H,0CCH,00CH,03BH,060H,0CCH,0FCH,000H ; 2 D_32
	FC 00		
FC06	78 CC 0C 3B 0C CC	DB	078H,0CCH,00CH,03BH,00CH,0CCH,078H,000H ; 3 D_33
	78 00		
FC0 E	1E 2C 6C CC FE 0C	DB	01CH,03CH,06CH,0CCH,0FEH,00CH,01EH,000H ; 4 D_34
	1E 00		
FC16	FC C0 F8 0C 0C CC	DB	0FCH,0C0H,0FBH,00CH,00CH,0CCH,078H,000H ; 5 D_35
	78 00		
FC1 E	38 60 C0 F8 CC CC	DB	038H,060H,0C0H,0FBH,0CCH,0CCH,078H,000H ; 6 D_36
	78 00		
FC26	FC CC 0C 1B 30 30	DB	0FCH,0CCH,00CH,018H,030H,030H,030H,000H ; 7 D_37
	30 00		
FC2 E	78 CC CC 78 CC CC	DB	078H,0CCH,0CCH,078H,0CCH,0CCH,078H,000H ; 8 D_38
	78 00		
FC36	78 CC CC 7C 0C 1B	DB	078H,0CCH,0CCH,07CH,00CH,018H,070H,000H ; 9 D_39
	00 30		
FC3 E	00 30 00 00 00 30	DB	000H,030H,030H,000H,000H,030H,030H,000H ; : D_3A
	30 00		
FC46	00 30 30 00 00 30	DB	000H,030H,030H,000H,000H,030H,030H,060H ; ; D_3B
	30 60		
FC4 E	18 30 60 C0 60 30	DB	018H,030H,060H,0C0H,060H,030H,018H,000H ; < D_3C
	18 00		
FC56	00 00 FC 00 00 FC	DB	000H,000H,0FCH,000H,000H,0FCH,000H,000H ; = D_3D
	00 00		
FC5 E	60 30 1B 0C 1B 30	DB	060H,030H,018H,00CH,018H,030H,060H,000H ; > D_3E
	60 00		
FC66	7B CC 0C 1B 30 00	DB	078H,0CCH,00CH,018H,030H,000H,030H,000H ; ? D_3F
	30 00		
			@ D_40
FC6 E	7C C6 DE DE DE C0	DB	07CH,0C6H,0DEH,0DEH,0DEH,0C0H,078H,000H ;
	78 00		
FC76	30 78 CC CC FC CC	DB	030H,078H,0CCH,0CCH,0FCH,0CCH,0CCH,000H ; A D_41
	CC 00		
FC7 E	FC 66 66 7C 66 66	DB	0FCH,066H,066H,07CH,066H,066H,0FCH,000H ; B D_42
	FC 00		
FC86	3C 66 C0 C0 C0 66	DB	03CH,066H,0C0H,0C0H,0C0H,066H,03CH,000H ; C D_43
	3C 00		
FC8 E	F8 6C 66 66 66 6C	DB	0FBH,06CH,066H,066H,066H,06CH,0FBH,000H ; D D_44
	F8 00		
FC96	FE 62 68 78 68 62	DB	0FEH,062H,068H,078H,068H,062H,0FEH,000H ; E D_45
	FE 00		
FC9 E	FE 62 68 78 68 60	DB	0FEH,062H,068H,078H,068H,060H,0F0H,000H ; F D_46
	F0 00		
FCA6	3C 66 C0 C0 CE 66	DB	03CH,066H,0C0H,0C0H,0CEH,066H,03EH,000H ; G D_47
	3E 00		
FCA E	CC CC CC FC CC CC	DB	0CCH,0CCH,0CCH,0FCH,0CCH,0CCH,0CCH,000H ; H D_48
	CC 00		
FCB6	78 30 30 30 30 30	DB	078H,030H,030H,030H,030H,078H,000H ; I D_4B
	78 00		
FCB E	1E 0C 0C 0C CC CC	DB	01EH,00CH,00CH,00CH,0CCH,0CCH,078H,000H ; J D_4A
	78 00		
FCC6	E6 66 6C 78 6C 66	DB	0E6H,066H,06CH,078H,06CH,066H,0E6H,000H ; K D_48
	E6 00		
FCC E	F0 60 60 60 62 66	DB	0F0H,060H,060H,060H,062H,068H,0FEH,000H ; L D_4C
	FE 00		
FCD6	C6 EE FE FE D6 C6	DB	0C6H,0EEH,0FEH,0FEH,0D6H,0C6H,0C6H,000H ; M D_4D
	C6 00		
FCD E	C6 E6 F6 DE CE C6	DB	0C6H,0E6H,0F6H,0DEH,0CEH,0C6H,0C6H,000H ; N D_4E
	C6 00		
FCE6	38 C6 C6 C6 C6 6C	DB	038H,06CH,0CBH,0C6H,0C6H,06CH,038H,000H ; O D_4F
	38 00		

FCEE	FC 66 66 7C 60 60	DB	0FCH, 066H, 066H, 07CH, 060H, 060H, 0F0H, 000H ;	P D_50
FO 00				
FCF6	78 CC CC CC DC 78	DB	07BH, 0CCH, 0CCH, 0CCH, 0DCH, 07BH, 01CH, 000H ;	Q D_51
IC 00				
FCFE	FC 66 66 7C 6C 66	DB	0FCH, 066H, 066H, 07CH, 06CH, 066H, 0E6H, 000H ;	R D_52
E6 00				
FD06	78 CC E0 70 1C CC	DB	07BH, 0CCH, 0E0H, 070H, 01CH, 0CCH, 07BH, 000H ;	S D_53
78 00				
FD0E	FC B4 30 30 30 30	DB	0FCH, 084H, 030H, 030H, 030H, 030H, 07BH, 000H ;	T D_54
78 00				
FD16	CC CC CC CC CC CC	DB	0CCH, 0CCH, 0CCH, 0CCH, 0CCH, 0CCH, 0FCH, 000H ;	U D_55
FC 00				
FD1E	CC CC CC CC CC 78	DB	0CCH, 0CCH, 0CCH, 0CCH, 0CCH, 07BH, 030H, 000H ;	V D_56
30 00				
FD26	C6 C6 C6 D6 FE EE	DB	0C6H, 0C6H, 0C6H, 0D6H, 0FEH, 0EEH, 0C6H, 000H ;	W D_57
C6 00				
FD2E	C6 C6 C6 38 38 6C	DB	0C6H, 0C6H, 06CH, 03BH, 03BH, 06CH, 0C6H, 000H ;	X D_58
C6 00				
FD36	CC CC CC 78 30 30	DB	0CCH, 0CCH, 0CCH, 07BH, 030H, 030H, 07BH, 000H ;	Y D_59
78 00				
FD3E	FE C6 8C 18 32 66	DB	0FEH, 0C6H, 08CH, 01BH, 032H, 066H, 0FEH, 000H ;	Z D_5A
FE 00				
FD46	78 80 60 60 80 60	DB	07BH, 060H, 060H, 060H, 060H, 060H, 07BH, 000H ;	I D_5B
78 00				
FD4E	C0 60 30 18 0C 06	DB	0C0H, 060H, 030H, 01BH, 00CH, 066H, 002H, 000H ;	
02 00				
FD56	78 18 18 18 18 18	DB	07BH, 01BH, 01BH, 01BH, 01BH, 01BH, 07BH, 000H ;	J D_5D
78 00				
FD5E	10 38 6C C6 00 00	DB	010H, 03BH, 06CH, 0C6H, 000H, 000H, 000H, 000H ;	
00 00				
FD66	00 00 00 00 00 00	DB	000H, 000H, 00DH, 000H, 000H, 000H, 000H, 0FFH ;	CIRCUMFLEX D_5E _ D_5F
00 FF				
FD6E	30 30 18 00 00 00	DB	030H, 030H, 01BH, 000H, 000H, 000H, 000H, 000H ;	' D_60
00 00				
FD76	00 00 78 0C 7C CC	DB	000H, 000H, 07BH, 0CCH, 07CH, 0CCH, 076H, 000H ;	
76 00				
FD7E	E0 60 60 7C 66 66	DB	0E0H, 060H, 060H, 07CH, 066H, 066H, 0DCH, 000H ;	LC B D_62
DC 00				
FD86	00 00 78 CC C0 CC	DB	000H, 000H, 07BH, 0CCH, 0C0H, 0CCH, 076H, 000H ;	LC C 0_63
78 00				
FD8E	1C 0C 0C 7C CC CC	DB	01CH, 0DCH, 00CH, 07CH, 0CCH, 0CCH, 076H, 000H ;	LC D D_64
76 00				
FD96	00 00 78 CC FC C0	DB	000H, 000H, 07BH, 0CCH, 0FCH, 0C0H, 07BH, 000H ;	LC E 0_65
78 00				
FD9E	38 6C 60 F0 60 60	DB	03BH, 06CH, 060H, 0F0H, 060H, 060H, 0F0H, 000H ;	LC F D_66
F0 00				
FDA6	00 00 76 CC CC 7C	DB	000H, 000H, 076H, 0CCH, 0CCH, 07CH, 00CH, 0FBH ;	LC G 0_67
0C F8				
FDAE	E0 60 6C 76 66 66	DB	0E0H, 080H, 06CH, 076H, 066H, 066H, 0E6H, 000H ;	LC H 0_68
E6 00				
FD86	30 00 70 30 30 30	DB	030H, 000H, 070H, 030H, 030H, 030H, 07BH, 000H ;	LC I 0_69
78 00				
FD8E	0C 00 0C 0C 0C CC	DB	00CH, 000H, 00CH, 00CH, 00CH, 0CCH, 0CCH, 07BH ;	LC J D_6A
CC 78				
FOC6	E0 60 66 6C 78 6C	DB	0E0H, 060H, 066H, 06CH, 07BH, 06CH, 0E6H, 000H ;	LC K D_6B
E6 00				
FDCE	70 30 30 30 30 30	DB	070H, 030H, 030H, 030H, 030H, 030H, 07BH, 000H ;	LC L D_6C
78 00				
FD06	00 00 CC FE FE 06	DB	000H, 000H, 0CCH, 0FEH, 0FEH, 006H, 0C6H, 000H ;	LC M D_6D
C6 00				
FD0E	00 00 FB CC CC CC	DB	00DH, 00DH, 0FBH, 0CCH, 0CCH, 0CCH, 0CCH, 000H ;	LC N D_6E
CC 00				
FDEB	00 00 78 CC CC CC	DB	000H, 000H, 07BH, 0CCH, 0CCH, 0CCH, 07BH, 000H ;	LC O D_6F
78 00				
FDEE	00 00 0C 66 66 7C	DB	000H, 000H, 00CH, 066H, 08BH, 07CH, 060H, 0F0H ;	LC P D_70
60 F0				
FD06	00 00 76 CC CC 7C	DB	000H, 000H, 076H, 0CCH, 0CCH, 07CH, 00CH, 01EH ;	LC Q 0_71
0C 1E				
FD0E	00 00 0C 76 66 60	DB	000H, 000H, 00CH, 076H, 066H, 060H, 0F0H, 000H ;	LC R 0_72
F0 00				
FE06	00 00 7C C0 78 0C	DB	000H, 000H, 07CH, 0C0H, 07BH, 00CH, 0FBH, 000H ;	LC S 0_73
F8 00				
FE0E	10 30 7C 30 30 34	DB	010H, 030H, 07CH, 030H, 030H, 034H, 01BH, 000H ;	LC T D_74
18 00				
FE16	00 00 CC CC CC CC	DB	000H, 000H, 0CCH, 0CCH, 0CCH, 0CCH, 076H, 000H ;	LC U 0_75
78 00				
FE1E	00 00 CC CC CC 78	DB	000H, 000H, 0CCH, 0CCH, 0CCH, 07BH, 030H, 000H ;	LC V D_76
30 00				
FE26	00 00 C6 06 FE FE	DB	000H, 000H, 0C6H, 0D6H, 0FEH, 0FEH, 06CH, 000H ;	LC W 0_77
6C 00				
FE2E	00 00 C6 6C 38 6C	DB	000H, 000H, 0C6H, 06CH, 03BH, 06CH, 0C6H, 000H ;	LC X 0_78
C6 00				
FE36	00 00 CC CC CC 7C	DB	000H, 000H, 0CCH, 0CCH, 0CCH, 07CH, 00CH, 0FBH ;	LC Y 0_79
0C F8				
FE3E	00 00 FC 9B 30 64	DB	000H, 000H, 0FCH, 09BH, 030H, 064H, 0FCH, 000H ;	LC Z D_7A
FC 00				
FE46	1C 30 30 E0 30 30	DB	01CH, 030H, 030H, 0E0H, 030H, 030H, 01CH, 000H ;	I D_7B
1C 00				
FE4E	18 18 18 00 18 18	DB	01BH, 01BH, 01BH, 000H, 01BH, 01BH, 01BH, 000H ;	: D_7C
18 00				
FE56	E0 30 30 1C 30 30	DB	0E0H, 030H, 030H, 01CH, 030H, 030H, 0E0H, 000H ;	: D_7D
E0 00				
FE5E	76 DC 00 00 00 00	DB	076H, 0DCH, 000H, 000H, 000H, 000H, 000H, 000H ;	~ D_7E
00 00				
FE66	00 10 38 6C C6 C6	DB	000H, 010H, 03BH, 06CH, 0C6H, 0C6H, 0FEH, 000H ;	
FE 00				

; DELTA D_7F

FE6E
FE6E E9 1393 R

ORG OFE6EH
JMP NEAR PTR TIME_OF_DAY

```

-----
CRC CHECK/GENERATION ROUTINE
ROUTINE TO CHECK A ROM MODULE USING THE POLYNOMIAL:
X16 + X12 + X5 + 1
CALLING PARAMETERS:
DS = DATA SEGMENT OF ROM SPACE TO BE CHECKED
SI = INDEX OFFSET INTO DS POINTING TO 1ST BYTE
CX = LENGTH OF SPACE TO BE CHECKED (INCLUDING CRC BYTES)
ON EXIT:
ZERO FLAG = SET = CRC CHECKED OK
AH = 00
AL = 7F
BX = 0000
CL = 04
DX = 0000 IF CRC CHECKED OK, ELSE, ACCUMULATED CRC
SI = (SI(ENTRY)+BX(ENTRY))
NOTE: ROUTINE WILL RETURN IMMEDIATELY IF "RESET_FLAG
IS EQUAL TO "1234H" (WARM START)
-----

```

FE71

```

CRC_CHECK PROC NEAR
ASSUME DS:NOTHING
MOV BX,CX ; SAVE COUNT
MOV DX,OFFFHH ; INIT. ENCODE REGISTER
CLD ; SET DIR FLAG TO INCREMENT
XOR AH,AH ; INIT. WORK REG HIGH
MOV CL,4 ; SET ROTATE COUNT
CRC_1: LODSB ; GET A BYTE
XOR DH,AL ; FORM AJ + CJ + 1
MOV AL,DH
ROL AX,CL ; SHIFT WORK REG BACK 4
XOR DX,AX ; ADD INTO RESULT REG
ROL AX,1 ; SHIFT WORK REG BACK 1
XCHG DH,DL ; SWAP PARTIAL SUM INTO RESULT REG
XOR OX,AX ; ADD WORK REG INTO RESULTS
ROR AX,CL ; SHIFT WORK REG OVER 4
AND AL,11100000H ; CLEAR OFF (EFGH)
XOR DX,AX ; ADD (ABCD) INTO RESULTS
ROR AX,1 ; SHIFT WORK REG ON OVER (AH=0 FOR
NEXT PASS)
XOR DH,AL ; ADD (ABCD INTO RESULTS LOW)
DEC BX ; DECREMENT COUNT
JNZ CRC_1 ; LOOP TILL COUNT = 0000
OR DX,DX ; DX S/B = 0000 IF O.K.
RET ; RETURN TO CALLER
CRC_CHECK ENDP
-----

```

FE9A

```

FE9A 32 C0
FE9C EE
FE9D FE C3
FE9F 42
FEA0 EC
FEA1 C3
FEA2

RR1 PROC NEAR
XOR AL,AL
OUT DX,AL ; DISABLE ALL INTERRUPTS
INC BL ; BUMP ERROR REPORTER
RR2: INC DX ; INCR PORT ADDR
RR3: IN AL,DX ; READ REGISTER
RET
RR1 ENDP
-----

```

```

; THIS ROUTINE HANDLES THE TIMER INTERRUPT FROM
; CHANNEL 0 OF THE 8253 TIMER. INPUT FREQUENCY IS 1.19318 MHZ
; AND THE DIVISOR IS 65536, RESULTING IN APPROX. 18.2 INTERRUPTS
; EVERY SECOND.
; THE INTERRUPT HANDLER MAINTAINS A COUNT OF INTERRUPTS SINCE POWER
; ON TIME, WHICH MAY BE USED TO ESTABLISH TIME OF DAY.
; INTERRUPTS MISSED WHILE INTS. WERE DISABLED ARE TAKEN CARE OF
; BY THE USE OF TIMER 1 AS A OVERFLOW COUNTER
; THE INTERRUPT HANDLER ALSO DECREMENTS THE MOTOR CONTROL COUNT
; OF THE DISKETTE, AND WHEN IT EXPIRES, WILL TURN OFF THE DISKETTE
; MOTOR, AND RESET THE MOTOR RUNNING FLAGS
; THE INTERRUPT HANDLER WILL ALSO INVOKE A USER ROUTINE THROUGH
; INTERRUPT ICH AT EVERY TIME TICK. THE USER MUST CODE A ROUTINE
; AND PLACE THE CORRECT ADDRESS IN THE VECTOR TABLE.
-----

```

FEA5

```

ORG OFE6B8
ASSUME DS:DATA
TIMER_INT PROC FAR
STI ; INTERRUPTS BACK ON
PUSH DS
PUSH AX
PUSH DX ; SAVE MACHINE STATE
CALL DDB
INC TIMER_LOW ; INCREMENT TIME
MOV T4, T4 ; TEST_DAY
JNZ T4 ; INCREMENT HIGH WORD OF TIME
INC TIMER_HIGH
MOV T4, T4 ; TEST_DAY
CMP TIMER_HIGH,01B8H ; TEST FOR COUNT EQUALLING 24 HOURS
JZ T5 ; TEST FOR COUNT EQUALLING 24 HOURS
CMP TIMER_LOW,0B0H ; DISKETTE_CTL
JNZ T5 ; DISKETTE_CTL
-----

```

```

;----- TIMER HAS GONE 24 HOURS
FEC5 28 C0 SUB AX,AX
FEC7 A3 006E R MOV TIMER_HIGH,AX
FECA A3 006C R MOV TIMER_LOW,AX
FECD C6 06 0070 R 01 MOV TIMER_OF,1
;----- TEST FOR DISKETTE TIME OUT
FED2 T5: ; LOOP TILL ALL OVERFLOWS TAKEN
; CARE OF
FED2 FE 0E 0040 R DEC MOTOR_COUNT
FED6 75 09 JNZ T6 ; RETURN IF COUNT NOT OUT
FED8 80 26 003F R F0 AND MOTOR_STATUS,OF0H ; TURN OFF MOTOR RUNNING BITS
FEDD 80 80 MOV AL,FC_RESET ; TURN OFF MOTOR, DO NOT RESET FOC
FEDF E6 F2 OUT NEC_CTL,AL ; TURN OFF THE MOTOR
FEF1 CD 1C T6: INT ICH ; TRANSFER CONTROL TO A USER
; ROUTINE
FEF3 80 20 MOV AL,E01
FEF5 E6 20 OUT 020H,AL ; END OF INTERRUPT TO B259
FEF7 5A POP DX
FEF8 8B POP AX
FEF9 1F POP DS ; RESET MACHINE STATE
FEFA CF IRET ; RETURN FROM INTERRUPT
FEFB TIMER_INT ENDP
;----- ARITHMETIC CHECKSUM ROUTINE
; ENTRY:
; DS = DATA SEGMENT OF ROM SPACE TO BE CHECKED
; SI = INDEX OFFSET INTO DS POINTING TO 1ST BYTE
; CX = LENGTH OF SPACE TO BE CHECKED
; EXIT: ZERO FLAG OFF=ERROR, ON= SPACE CHECKED OK
;-----
FEFB ROS_CHECKSUM PROC NEAR
FEFB RC_0: ADD AL,DS:[SI]
FEFD 46 INC SI
FEFE E2 FB LOOP RC_0
FEFF 0A C0 OR AL,AL
FEF2 C3 RET
FEF3 ROS_CHECKSUM ENDP
;-----
; THESE ARE THE VECTORS WHICH ARE MOVED INTO
; THE 8086 INTERRUPT AREA DURING POWER ON.
; ONLY THE OFFSETS ARE DISPLAYED HERE, CODE
; SEGMENT WILL BE ADDED FOR ALL OF THEM, EXCEPT
; WHERE NOTED.
;-----
FEF3 ASSUME CS:CODE
FEF3 ORG OFEF3H
FEF3 VECTOR_TABLE LABEL WORD ; VECTOR TABLE FOR MOVE TO INTERRUPTS
FEF5 FEAB R DW OFFSET TIMER_INT ; INTERRUPT 8
FEF6 1561 R DW OFFSET KB_INT ; INTERRUPT 9
FEF7 F815 R DW OFFSET D11 ; INTERRUPT A
FEF8 F815 R DW OFFSET D11 ; INTERRUPT B
FEF9 F815 R DW OFFSET D11 ; INTERRUPT C
FEFA F815 R DW OFFSET D11 ; INTERRUPT D
FEFB F815 R DW OFFSET DISK_INT ; INTERRUPT E
FEFC F815 R DW OFFSET D11 ; INTERRUPT F
FEFD 0008 R DW OFFSET VIDEO_10 ; INTERRUPT 10H
FEFE F840 R DW OFFSET EQUIPMENT ; INTERRUPT 11H
FEFF F841 R DW OFFSET MEMORY_SIZE_DETERMINE ; INTERRUPT 12H
FF00 EC89 R DW OFFSET DISKETTE_10 ; INTERRUPT 13H
FF01 E739 R DW OFFSET RS232_10 ; INTERRUPT 14H
FF02 FB89 R DW CASSETTE_10 ; INTERRUPT 15H
FF03 1300 R DW OFFSET KEYBOARD_10 ; INTERRUPT 16H
FF04 EF02 R DW OFFSET PRINTER_10 ; INTERRUPT 17H
FF05 0000 DW 00000H ; INTERRUPT 18H
FF06 0F00H DW 0F00H ; MUST BE INSERTED INTO TABLE LATER
FF07 0B18 R DW OFFSET BOOT_STRAP ; INTERRUPT 19H
FF08 1393 R DW TIME_OF_DAY ; INTERRUPT 1AH -- TIME OF DAY
FF09 FB3C R DW DUMMY_RETURN ; INTERRUPT 1BH -- KEYBD BREAK ADDR
FF0A F83C R DW DUMMY_RETURN ; INTERRUPT 1C -- TIMER BREAK ADDR
FF0B F0A4 R DW VIDEO_PARMS ; INTERRUPT 1D -- VIDEO PARAMETERS
FF0C EFC7 R DW OFFSET DISK_BASE ; INTERRUPT 1E -- DISK FARMS
FF0D E05E R DW CRT_CHARH ; INTERRUPT 1F -- VIDEO EXT
FF0E P_MSG PROC NEAR
FF0F 2E: 8A 04 G12A: MOV AL,CS:[SI] ; PUT CHAR IN AL
FF10 46 INC SI ; POINT TO NEXT CHAR
FF11 50 PUSH AX ; SAVE PRINT CHAR
FF12 E8 10BA R CALL PRY_HEX ; CALL VIDEO_10
FF13 58 POP AX ; RECOVER PRINT CHAR
FF14 3C 00 CMP AL,13 ; WAS IT CARRAGE RETURN?
FF15 75 F3 JNE G12A ; NO,KEEP PRINTING STRING
FF16 C3 RET
FF17 P_MSG ENDP
FF18 ROUTINE TO SOUND BEEPER
FF19 BEEP PROC NEAR
FF1A 80 86 MOV AL,10110110B ; SEL TIM 2,LS9,MSB,BINARY
FF1B E6 43 OUT TIMER*3,AL ; WRITE THE TIMER MODE REG
FF1C 8B 0533 MOV AX,833H ; DIVISOR FOR 1000 HZ
FF1D E6 42 OUT TIMER*2,AL ; WRITE TIMER 2 CNT - LSB
FF1E BA C4 MOV AL,AH
FF1F E6 42 OUT TIMER*2,AL ; WRITE TIMER 2 CNT - MSB
FF20 E4 61 IN AL,PORT_B ; GET CURRENT SETTING OF PORT
FF21 8A 0A MOV AH,AL ; SAVE THAT SETTING
FF22 0C 03 OR AL,03 ; TURN SPEAKER ON
FF23 E6 61 OUT PORT_B,AL
FF24 28 C9 SUB CX,CX ; SET CNT TO WAIT 500 MS
FF25 E2 FE G7: LOOP G7 ; DELAY BEFORE TURNING OFF
FF26 FE CB DEC BL ; DELAY CNT EXPIRED?
FF27 75 FA CJZ G7 ; NO - CONTINUE BEEPING SPK
FF28 BA C4 MOV AL,AH ; RECOVER VALUE OF PORT
FF29 E6 61 OUT PORT_B,AL
FF2A C3 RET ; RETURN TO CALLER
FF2B BEEP ENDP

```

 DUMMY RETURN FOR ADDRESS COMPATIBILITY

FF53
 FF53 CF

```

    ORG   OFF53H
    IRET

-- INT 5
    THIS LOGIC WILL BE INVOKED BY INTERRUPT 05H TO PRINT
    THE SCREEN. THE CURSOR POSITION AT THE TIME THIS ROUTINE
    IS INVOKED WILL BE SAVED AND RESTORED UPON COMPLETION. THE
    ROUTINE IS INTENDED TO RUN WITH INTERRUPTS ENABLED.
    IF A SUBSEQUENT 'PRINT SCREEN KEY IS DEPRESSED DURING THE
    TIME THIS ROUTINE IS PRINTING IT WILL BE IGNORED.
    ADDRESS 50:0 CONTAINS THE STATUS OF THE PRINT SCREEN:

    50:0  =0      EITHER PRINT SCREEN HAS NOT BEEN CALLED
                OR UPON RETURN FROM A CALL THIS INDICATES
                A SUCCESSFUL OPERATION.

                =1      PRINT SCREEN IS IN PROGRESS

                =OFFH  ERROR ENCOUNTERED DURING PRINTING.
    
```

FF54
 FF54
 FF54 FB
 FF55 1E

```

    ASSUME CS:CODE,DS:XXDATA
    PRINT_SCREEN ORG OFF54H
    STI         PROC     FAR
    PUSH        DS      ; MUST RUN WITH INTERRUPTS ENABLED
                    ; MUST USE 50:0 FOR DATA AREA
                    ; STORAGE
    
```

FF56 80
 FF57 53
 FF58 51

```

    PUSH        AX
    PUSH        BX
    PUSH        CX
    
```

FF59 52

```

    PUSH        DX
    MOV         AX,XXDATA ; WILL HOLD CURRENT CURSOR POSITION
                    ; HEX 50
    
```

FF5A 8B ---- R
 FF5D 8E DB
 FF5F 80 3E 0000 R 01
 FF64 74 5F
 FF66 C6 06 0000 R 01
 FF6B 84 0F

```

    MOV         DS,AX
    CMP         STATUS_BYTE,1 ; SEE IF PRINT ALREADY IN PROGRESS
    JZ          EXIT          ; JUMP IF PRINT ALREADY IN PROGRESS
    MOV         STATUS_BYTE,1 ; INDICATE PRINT NOW IN PROGRESS
    MOV         AH,15        ; WILL REQUEST THE CURRENT SCREEN
                    ; MOOE
    
```

FF60 CD 10

```

    INT         10H
                    ; [AL]=MODE
                    ; [AH]=NUMBER COLUMNS/LINE
                    ; [BH]=VISUAL PAGE
    
```

 AT THIS POINT WE KNOW THE COLUMNS/LINE ARE IN
 [AX] AND THE PAGE IF APPLICABLE IS IN [BH]. THE STACK
 HAS DS,AX,BX,CX,DX PUSHED. [AL] HAS VIDEO MODE

FF6F 8A CC
 FF71 85 19
 FF73 EB FA5F R
 FF76 51
 FF77 84 03
 FF79 CD 10
 FF78 58
 FF7C 52
 FF7D 33 D2

```

    MOV         CL,AH        ; WILL MAKE USE OF [CX] REGISTER TO
    MOV         CH,25        ; CONTROL ROW & COLUMNS
    CALL        CRLF        ; CARRIAGE RETURN LINE FEED ROUTINE
    PUSH        CX          ; SAVE SCREEN BOUNDS
    MOV         AH,3        ; WILL NOW READ THE CURSOR.
    INT         10H        ; AND PRESERVE THE POSITION
    POP         CX          ; RECALL SCREEN BOUNDS
    PUSH        DX          ; RECALL [BH]=VISUAL PAGE
    XOR         DX,DX       ; WILL SET CURSOR POSITION TO [0,0]
    
```

 THE LOOP FROM PRI10 TO THE INSTRUCTION PRIOR TO PRI20
 IS THE LOOP TO READ EACH CURSOR POSITION FROM THE SCREEN
 AND PRINT

FF7F 84 02
 FF81 CD 10
 FF83 84 08
 FF85 CD 10
 FF87 0A 0C
 FF89 76 02
 FF8B 80 20
 FF8D 52 D2
 FF8E 33 D2
 FF90 32 E4
 FF92 CD 17
 FF94 5A
 FF95 F6 C4 29
 FF98 75 21
 FF9A FE C2
 FF9C 3A CA
 FF9E 75 0F
 FFA0 32 D2
 FFA2 8A E2
 FFA4 52
 FFA5 EB FA5F R
 FFA8 5A
 FFA9 FE C6
 FFAB 3A EE
 FFAD 75 D0
 FFAF 5A
 FFB0 84 02
 FFB2 CD 10
 FFB4 C6 06 0000 R 00
 FFB9 EB 0A
 FFB8 5A
 FFB C 84 02
 FFB E 84 02
 FFC0 C6 06 0000 R FF
 FFC5 5A
 FFC6 58
 FFC7 5B
 FFC8 5B
 FFC9 1F
 FFCA CF
 FFCB

```

    MOV         AH,2        ; TO INDICATE CURSOR SET REQUEST
    INT         10H        ; NEW CURSOR POSITION ESTABLISHED
    MOV         AH,B       ; TO INDICATE READ CHARACTER
    INT         10H        ; CHARACTER NOW IN [AL]
    OR          AL,AL      ; SEE IF VALID CHAR
    JNZ        PRI15      ; JUMP IF VALID CHAR
    MOV         AL,' '     ; MAKE A BLANK
    SAVE        CURSOR POSITION
    XOR         DX,DX     ; INDICATE PRINTER 1
    MOV         AH,AH     ; TO INDICATE PRINT CHAR IN [AL]
    INT         17H      ; PRINT THE CHARACTER
    POP         DX        ; RECALL CURSOR POSITION
    TEST        AH,029H   ; TEST FOR PRINTER ERROR
    JNZ        ERR10     ; JUMP IF ERROR DETECTED
    INC         DL        ; ADVANCE TO NEXT COLUMN
    CMP         CL,0L     ; SEE IF AT END OF LINE
    JNZ        PRI10     ; IF NOT PROCEED
    XOR         DL,DL     ; BACK TO COLUMN 0
    MOV         AH,DL     ; [AH]=0
    PUSH        DX        ; SAVE NEW CURSOR POSITION
    CALL        CRLF      ; LINE FEED CARRIAGE RETURN
    POP         DX        ; RECALL CURSOR POSITION
    INC         DH        ; ADVANCE TO NEXT LINE
    CMP         CH,DH     ; FINISHED?
    JNZ        PRI10     ; IF NOT CONTINUE
    POP         DX        ; RECALL CURSOR POSITION
    MOV         AH,2     ; TO INDICATE CURSOR SET REQUEST
    INT         10H      ; CURSOR POSITION RESTORED
    MOV         STATUS_BYTE,0 ; INDICATE FINISHED.
    JMP        SHORT EXIT ; EXIT THE ROUTINE
ERR10:  POP         DX        ; GET CURSOR POSITION
        MOV         AH,2     ; TO REQUEST CURSOR SET
        INT         10H      ; CURSOR POSITION RESTORED
        MOV         STATUS_BYTE,OFFH ; INDICATE ERROR
EXIT:   POP         CX        ; RESTORE ALL THE REGISTERS USED
        POP         CX
        POP         BX
        POP         AX
        POP         DS
    IRET
PRINT_SCREEN ENDP
    
```



```

; EASE OF USE REVECTOR ROUTINE - CALLED THROUGH
; INT 1BH WHEN CASSETTE BASIC IS INVOKED (NO DISKETTE
; NO CARTRIDGES)
; KEYBOARD VECTOR IS RESET TO POINT TO "NEW_INT_9"
; BASIC VECTOR IS SET TO POINT TO F600:0

```

```

FFCB
FFCB 2B C0
FFCD 8E DB
FFCF C7 06 0024 R 1937 R
FFD5 A3 0060 R
FFD8 C7 06 0062 R F600
FFDE CD 1B
FFE0

```

```

BAS_ENT PROC FAR
ASSUME DS:ABS0
SUB AX,AX
MOV DS,AX ; SET ADDRESSING
MOV WORD PTR INT_PTR+4,OFFSET NEW_INT_9
MOV BASIC_PTR,AX ; SET INT 1B=F600:0
MOV BASIC_PTR+2,OF600H
INT 1BH ; GO TO BASIC
BAS_ENT ENDP

```

```

; INITIALIZE TIMER SUBROUTINE - ASSUMES BOTH THE LSB AND MSB
; OF THE TIMER WILL BE USED.
; CALLING PARAMETERS:
; (AH) = TIMER #
; (AL) = BIT PATTERN OF INITIALIZATION WORD
; (BX) = INITIAL COUNT
; (BH) = MSB COUNT
; (BL) = LSB COUNT
; ALTERS REGISTERS DX AND AL.

```

```

FFE0
FFE0 E6 43
FFE2 8A 0040
FFEB 02 D4
FFE7 8A C3
FFE9 EE
FFEA 52
FFEB 5A
FFEC 8A C7
FFEE EE
FFEF C3
FFFO

```

```

INIT_TIMER PROC NEAR
OUT TIM_CTL,AL ; OUTPUT INITIAL CONTROL WORD.
MOV DX,TIMER ; BASE PORT ADDR FOR TIMERS
ADD OL,AH ; ADD IN THE TIMER #
MOV AL,BL ; LOAD LSB
OUT DX,AL
PUSH DX ; PAUSE
POP DX
MOV AL,BH ; LOAD MSB
OUT DX,AL
RET
INIT_TIMER ENDP

```

```

; POWER ON RESET VECTOR

```

```

FFFO

```

```

ORG OFFFOH

```

```

; POWER ON RESET

```

```

FFF0 EA
FFF1 0043 R
FFF3 F000
FFF5 30 36 2F 30 31 2F
38 33
FFFD FF
FFFE FD
FFFF

```

```

DB OEAH ; JUMP FAR
OW OFFSET RESET
DW OF000H
DB '06/01/83' ; RELEASE MARKER
DB OFFH ; FILLER
DB OFDH ; SYSTEM IDENTIFIER
DB OFFH ; CHECKSUM
CODE ENDS
END

```

Notes:

Appendix B. LOGIC DIAGRAMS

Contents

System Board	B-3
Program Cartridge	B-20
Power Supply Board	B-23
64KB Memory and Display Expansion	B-25
Color Display	B-29
Diskette Drive Adapter	B-30
Internal Modem	B-36
Parallel Printer Attachment	B-37
Infra-Red Receiver Board	B-42
Graphics Printer	B-43
Compact Printer	B-47

Notes:



APPENDIX C: CHARACTERS, KEYSTROKES, AND COLOR

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

Value		As Characters			Color/Graphics Text Attributes	
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground
18	24	↑	Ctrl X		Blue	Dark Grey
19	25	↓	Ctrl Y		Blue	Light Blue
1A	26	→	Ctrl Z		Blue	Light Green
1B	27	←	Ctrl [, Esc, Shift Esc, Ctrl Esc		Blue	Light Cyan
1C	28	⌞	Ctrl \		Blue	Light Red
1D	29	↔	Ctrl]		Blue	Light Magenta
1E	30	▲	Ctrl 6		Blue	Yellow
1F	31	▼	Ctrl —		Blue	White
20	32	Blank Space	Space Bar, Shift, Space, Ctrl Space, Alt Space		Green	Black
21	33	!	!	Shift	Green	Blue
22	34	”	”	Shift	Green	Green
23	35	#	#	Shift	Green	Cyan
24	36	\$	\$	Shift	Green	Red
25	37	%	%	Shift	Green	Magenta
26	38	&	&	Shift	Green	Brown
27	39	.	.		Green	Light Grey
28	40	((Shift	Green	Dark Grey
29	41))	Shift	Green	Light Blue
2A	42	*	*	Note 1	Green	Light Green
2B	43	+	+	Shift	Green	Light Cyan
2C	44	,	,		Green	Light Red
2D	45	—	—		Green	Light Magenta
2E	46	.	.	Note 2	Green	Yellow
2F	47	/	/		Green	White
30	48	0	0	Note 3	Cyan	Black
31	49	1	1	Note 3	Cyan	Blue
32	50	2	2	Note 3	Cyan	Green
33	51	3	3	Note 3	Cyan	Cyan

Value		As Characters			Color/Graphics Text Attributes	
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground
34	52	4	4	Note 3	Cyan	Red
35	53	5	5	Note 3	Cyan	Magenta
36	54	6	6	Note 3	Cyan	Brown
37	55	7	7	Note 3	Cyan	Light Grey
38	56	8	8	Note 3	Cyan	Dark Grey
39	57	9	9	Note 3	Cyan	Light Blue
3A	58	:	:	Shift	Cyan	Light Green
3B	59	;	;		Cyan	Light Cyan
3C	60	<	<	Shift	Cyan	Light Red
3D	61	=	=		Cyan	Light Magenta
3E	62	>	>	Shift	Cyan	Yellow
3F	63	?	?	Shift	Cyan	White
40	64	@	@	Shift	Red	Black
41	65	A	A	Note 4	Red	Blue
42	66	B	B	Note 4	Red	Green
43	67	C	C	Note 4	Red	Cyan
44	68	D	D	Note 4	Red	Red
45	69	E	E	Note 4	Red	Magenta
46	70	F	F	Note 4	Red	Brown
47	71	G	G	Note 4	Red	Light Grey
48	72	H	H	Note 4	Red	Dark Grey
49	73	I	I	Note 4	Red	Light Blue
4A	74	J	J	Note 4	Red	Light Green
4B	75	K	K	Note 4	Red	Light Cyan
4C	76	L	L	Note 4	Red	Light Red
4D	77	M	M	Note 4	Red	Light Magenta
4E	78	N	N	Note 4	Red	Yellow
4F	79	O	O	Note 4	Red	White
50	80	P	P	Note 4	Magenta	Black
51	81	Q	Q	Note 4	Magenta	Blue
52	82	R	R	Note 4	Magenta	Green
53	83	S	S	Note 4	Magenta	Cyan
54	84	T	T	Note 4	Magenta	Red

Value		As Characters			Color/Graphics Text Attributes	
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground
55	85	U	U	Note 4	Magenta	Magenta
56	86	V	V	Note 4	Magenta	Brown
57	87	W	W	Note 4	Magenta	Light Grey
58	88	X	X	Note 4	Magenta	Dark Grey
59	89	Y	Y	Note 4	Magenta	Light Blue
5A	90	Z	Z	Note 4	Magenta	Light Green
5B	91	[[Magenta	Light Cyan
5C	92	\	\		Magenta	Light Red
5D	93]]		Magenta	Light Magenta
5E	94	^	^	Shift	Magenta	Yellow
5F	95	_	_	Shift	Magenta	White
60	96	`	`		Yellow	Black
61	97	a	a	Note 5	Yellow	Blue
62	98	b	b	Note 5	Yellow	Green
63	99	c	c	Note 5	Yellow	Cyan
64	100	d	d	Note 5	Yellow	Red
65	101	e	e	Note 5	Yellow	Magenta
66	102	f	f	Note 5	Yellow	Brown
67	103	g	g	Note 5	Yellow	Light Grey
68	104	h	h	Note 5	Yellow	Dark Grey
69	105	i	i	Note 5	Yellow	Light Blue
6A	106	j	j	Note 5	Yellow	Light Green
6B	107	k	k	Note 5	Yellow	Light Cyan
6C	108	l	l	Note 5	Yellow	Light Red
6D	109	m	m	Note 5	Yellow	Light Magenta
6E	110	n	n	Note 5	Yellow	Yellow
6F	111	o	o	Note 5	Yellow	White
70	112	p	p	Note 5	White	Black
71	113	q	q	Note 5	White	Blue
72	114	r	r	Note 5	White	Green
73	115	s	s	Note 5	White	Cyan
74	116	f	f	Note 5	White	Red
75	117	u	u	Note 5	White	Magenta
76	118	v	v	Note 5	White	Brown

C-4 Characters, Keystrokes, and Color

Value		As Characters			Color/Graphics Text Attributes	
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground
77	119	w	w	Note 5	White	Light Grey
78	120	x	x	Note 5	White	Dark Grey
79	121	y	y	Note 5	White	Light Blue
7A	122	z	z	Note 5	White	Light Green
7B	123	{	{	Shift	White	Light Cyan
7C	124			Shift	White	Light Red
7D	125	}	}	Shift	White	Light Magenta
7E	126	~	~	Shift	White	Yellow
7F	127	Δ	Ctrl ←		White	White
* * * * 80 to FF Hex are Flashing if Blink is Enabled * * * *						
80	128	Ç	Alt 128	Note 6	Black	Black
81	129	ü	Alt 129	Note 6	Black	Blue
82	130	é	Alt 130	Note 6	Black	Green
83	131	â	Alt 131	Note 6	Black	Cyan
84	132	ä	Alt 132	Note 6	Black	Red
85	133	à	Alt 133	Note 6	Black	Magenta
86	134	â	Alt 134	Note 6	Black	Brown
87	135	ç	Alt 135	Note 6	Black	Light Grey
88	136	ê	Alt 136	Note 6	Black	Dark Grey
89	137	ë	Alt 137	Note 6	Black	Light Blue
8A	138	è	Alt 138	Note 6	Black	Light Green
8B	139	ï	Alt 139	Note 6	Black	Light Cyan
8C	140	î	Alt 140	Note 6	Black	Light Red
8D	141	ì	Alt 141	Note 6	Black	Light Magenta
8E	142	Ä	Alt 142	Note 6	Black	Yellow
8F	143	Å	Alt 143	Note 6	Black	White
90	144	É	Alt 144	Note 6	Blue	Black
91	145	æ	Alt 145	Note 6	Blue	Blue
92	146	Æ	Alt 146	Note 6	Blue	Green
93	147	ô	Alt 147	Note 6	Blue	Cyan
94	148	ö	Alt 148	Note 6	Blue	Red
95	149	ò	Alt 149	Note 6	Blue	Magenta

Value		As Characters			Color/Graphics Text Attributes	
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground
96	150	ú	Alt 150	Note 6	Blue	Brown
97	151	ù	Alt 151	Note 6	Blue	Light Grey
98	152	ÿ	Alt 152	Note 6	Blue	Dark Grey
99	153	ó	Alt 153	Note 6	Blue	Light Blue
9A	154	ü	Alt 154	Note 6	Blue	Light Green
9B	155	¢	Alt 155	Note 6	Blue	Light Cyan
9C	156	£	Alt 156	Note 6	Blue	Light Red
9D	157	¥	Alt 157	Note 6	Blue	Light Magenta
9E	158	Pt	Alt 158	Note 6	Blue	Yellow
9F	159	∫	Alt 159	Note 6	Blue	White
A0	160	á	Alt 160	Note 6	Green	Black
A1	161	í	Alt 161	Note 6	Green	Blue
A2	162	ó	Alt 162	Note 6	Green	Green
A3	163	ú	Alt 163	Note 6	Green	Cyan
A4	164	ñ	Alt 164	Note 6	Green	Red
A5	165	Ñ	Alt 165	Note 6	Green	Magenta
A6	166	à	Alt 166	Note 6	Green	Brown
A7	167	ò	Alt 167	Note 6	Green	Light Grey
A8	168	¿	Alt 168	Note 6	Green	Dark Grey
A9	169	┌	Alt 169	Note 6	Green	Light Blue
AA	170	└	Alt 170	Note 6	Green	Light Green
AB	171	½	Alt 171	Note 6	Green	Light Cyan
AC	172	¼	Alt 172	Note 6	Green	Light Red
AD	173	ì	Alt 173	Note 6	Green	Light Magenta
AE	174	<<	Alt 174	Note 6	Green	Yellow
AF	175	>>	Alt 175	Note 6	Green	White
B0	176	⋮	Alt 176	Note 6	Cyan	Black
B1	177	⋮	Alt 177	Note 6	Cyan	Blue
B2	178	⋮	Alt 178	Note 6	Cyan	Green
B3	179	▬	Alt 179	Note 6	Cyan	Cyan
B4	180	▬	Alt 180	Note 6	Cyan	Red
B5	181	▬	Alt 181	Note 6	Cyan	Magenta
B6	182	▬	Alt 182	Note 6	Cyan	Brown

C-6 Characters, Keystrokes, and Color

Value		As Characters			Color/Graphics Text Attributes	
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground
B7	183		Alt 183	Note 6	Cyan	Light Grey
B8	184		Alt 184	Note 6	Cyan	Dark Grey
B9	185		Alt 185	Note 6	Cyan	Light Blue
BA	186		Alt 186	Note 6	Cyan	Light Green
BB	187		Alt 187	Note 6	Cyan	Light Cyan
BC	188		Alt 188	Note 6	Cyan	Light Red
BD	189		Alt 189	Note 6	Cyan	Light Magenta
BE	190		Alt 190	Note 6	Cyan	Yellow
BF	191		Alt 191	Note 6	Cyan	White
C0	192		Alt 192	Note 6	Red	Black
C1	193		Alt 193	Note 6	Red	Blue
C2	194		Alt 194	Note 6	Red	Green
C3	195		Alt 195	Note 6	Red	Cyan
C4	196		Alt 196	Note 6	Red	Red
C5	197		Alt 197	Note 6	Red	Magenta
C6	198		Alt 198	Note 6	Red	Brown
C7	199		Alt 199	Note 6	Red	Light Grey
C8	200		Alt 200	Note 6	Red	Dark Grey
C9	201		Alt 201	Note 6	Red	Light Blue
CA	202		Alt 202	Note 6	Red	Light Green
CB	203		Alt 203	Note 6	Red	Light Cyan
CC	204		Alt 204	Note 6	Red	Light Red
CD	205		Alt 205	Note 6	Red	Light Magenta
CE	206		Alt 206	Note 6	Red	Yellow
CF	207		Alt 207	Note 6	Red	White
D0	208		Alt 208	Note 6	Magenta	Black
D1	209		Alt 209	Note 6	Magenta	Blue
D2	210		Alt 210	Note 6	Magenta	Green
D3	211		Alt 211	Note 6	Magenta	Cyan
D4	212		Alt 212	Note 6	Magenta	Red
D5	213		Alt 213	Note 6	Magenta	Magenta
D6	214		Alt 214	Note 6	Magenta	Brown
D7	215		Alt 215	Note 6	Magenta	Light Grey

Value		As Characters			Color/Graphics Text Attributes	
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground
D8	216		Alt 216	Note 6	Magenta	Dark Grey
D9	217		Alt 217	Note 6	Magenta	Light Blue
DA	218		Alt 218	Note 6	Magenta	Light Green
DB	219		Alt 219	Note 6	Magenta	Light Cyan
DC	220		Alt 220	Note 6	Magenta	Light Red
DD	221		Alt 221	Note 6	Magenta	Light Magenta
DE	222		Alt 222	Note 6	Magenta	Yellow
DF	223		Alt 223	Note 6	Magenta	White
E0	224	α	Alt 224	Note 6	Yellow	Black
E1	225	β	Alt 225	Note 6	Yellow	Blue
E2	226	Γ	Alt 226	Note 6	Yellow	Green
E3	227	π	Alt 227	Note 6	Yellow	Cyan
E4	228	Σ	Alt 228	Note 6	Yellow	Red
E5	229	σ	Alt 229	Note 6	Yellow	Magenta
E6	230	μ	Alt 230	Note 6	Yellow	Brown
E7	231	τ	Alt 231	Note 6	Yellow	Light Grey
E8	232	Φ	Alt 232	Note 6	Yellow	Dark Grey
E9	233	θ	Alt 233	Note 6	Yellow	Light Blue
EA	234	Ω	Alt 234	Note 6	Yellow	Light Green
EB	235	δ	Alt 235	Note 6	Yellow	Light Cyan
EC	236	∞	Alt 236	Note 6	Yellow	Light Red
ED	237	ϕ	Alt 237	Note 6	Yellow	Light Magenta
EE	238	ϵ	Alt 238	Note 6	Yellow	Yellow
EF	239	\cap	Alt 239	Note 6	Yellow	White
F0	240	\equiv	Alt 240	Note 6	White	Black
F1	241	\pm	Alt 241	Note 6	White	Blue
F2	242	\geq	Alt 242	Note 6	White	Green
F3	243	\leq	Alt 243	Note 6	White	Cyan
F4	244	\int	Alt 244	Note 6	White	Red
F5	245	\int	Alt 245	Note 6	White	Magenta
F6	246	\div	Alt 246	Note 6	White	Brown
F7	247	\approx	Alt 247	Note 6	White	Light Grey
F8	248	\circ	Alt 248	Note 6	White	Dark Grey



Value		As Characters			Color/Graphics Text Attributes	
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground
F9	249	●	Alt 249	Note 6	White	Light Blue
FA	250	•	Alt 250	Note 6	White	Light Green
FB	251	√	Alt 251	Note 6	White	Light Cyan
FC	252	η	Alt 252	Note 6	White	Light Red
FD	253	2	Alt 253	Note 6	White	Light Magenta
FE	254	■	Alt 254	Note 6	White	Yellow
FF	255	BLANK	Alt 255	Note 6	White	White



NOTE 1 On the 62-key keyboard the Asterisk (*) can be keyed using two methods:

1) in the shift mode hit the  key or 2) hold Alt key and press the

 key.

On the 83-key keyboard the Asterisk (*) can be keyed using two methods:

1) hit the  key or 2) in the 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) on the 83-key keyboard 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 On the 62-key keyboard set Num Lock state using Alt/Fn/N then 3 digits after the Alt key must be typed from the numeric keys on the top row of the typematic portion of the keyboard. 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.)

On the 83-key keyboard the 3 digits after the Alt key must be typed from the numeric key pad (keys 71—73, 75—77, 79—82).

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	[k	{
12	C			,	<	L	\	l	!
13	D			—	=	M]	m	}
14	E			.	>	N	^	n	~
15	F			/	?	O	_	o	Δ

Appendix C

Character Set (80-FF) Quick Reference

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

Appendix D. UNIT SPECIFICATIONS

System Unit

Size:

Length	354 mm (13.9 in.)
Depth	290 mm (11.4 in.)
Height	97 mm (3.8 in.)

Weight:

3.71 Kg (8lb 4oz)	With Diskette Drive
2.61 Kg (5lb 8oz)	Without Diskette Drive

Transformer:

Electrical:

Input	110 Vac 60 Hz
Output to System	Pin 1 - 17 Vac, Pin 2 - GND, Pin 3 - 17 Vac

Power Cords:

Input Length	1.86 meters (6.14 feet)
Type	18 AWG
Output Length	1.22 meters (4.02 feet)
Type	18 AWG

Environment:

Air Temperature

System ON 15.6 to 32.2 degrees C (60 to 90 degrees F)

System Off 10 to 43 degrees C (50 to 110 degrees F)

Humidity

System On 8% to 80%

System Off 8% to 80%

Noise Level 45 dB

Cordless Keyboard

Size:

Length 341.5 mm (13.45 in.)

Depth 168 mm (6.61 in.)

Height 26 mm (1.02 in.)

Weight:

With Batteries 616 grams (22 ounces)

Without Batteries 700 grams (25 ounces)

Optional Cable:

6 feet, flat

Diskette Drive

Size:

Height	41.6 mm (1.6 in.)
Depth	146 mm (5.8 in.)
Width	208 mm (8.3 in.)

Weight:

1.1 kilograms (2.2 pounds)

Diskette Drive

Power:

Supply

Voltage	+5 Vdc Input	+12 Vdc Input
Nominal	+5 Vdc	+12 Vdc

Ripple

	+5 Vdc Input	+12 Vdc Input
0 to 50 kHz	100 mV	100 mV

Tolerance

	+5 Vdc Input	+12 Vdc Input
Including Ripple	+/- 5%	+/- 5%

Standby Current

	+5 Vdc Input	+12 Vdc Input
Nominal	600 mA	400 mA
Worst Case	700 mA	500 mA

Operating Current

	+5 Vdc Input	+12 Vdc Input
Nominal	600 mA	900 mA
Worst Case	700 mA	2400 mA

Mechanical and Electrical

Media	Industry-compatible 5 1/4 inch diskette
Media Life (Head Loaded)	3,000,000 revolutions/track
Media Life (Insertions)	30,000
Tracks Density	48 tracks/inch
Number of Tracks	40
Motor Start Time	500 ms
Instantaneous Speed Variation	+/- 3.0%
Rotational Speed	300 rpm +/- 1.5% (long term)
Nominal Transfer Rate (MFMM)	250,000 pulses/second
MTBF (25% Operating)	8,000 POH
Read Bit Shift	+/- 800 ns maximum
Seek Time	6 ms track-to-track maximum
Head Life	20,000 hours (normal use)
Head Load Time	Not Applicable
Head Settling Time	21 ms maximum (from last step pulse)
Error Rate	

Soft Error	1 per 1,000,000,000 bits maximum (recoverable within 10 retries)
Hard Error	1 per 1,000,000,000,000 bits maximum (nonrecoverable within 10 retries)
Access Error	1 per 3,000,000 seeks maximum

Temperature (Exclusive of media)

Operating	50 to 122 degrees F (10 to 44 degrees C)
Non-operating	-40 to 140 degrees F (-40 to 60 degrees C)

Relative Humidity (Exclusive of media)

Operating	20 to 80% (noncondensing)
Non-operating	5 to 95% (noncondensing)

Operating Altitude	7,000 feet above sea level
Operating Vibration	5 to 500 Hz 11G

Color Display

Size:

Height	297 mm (11.7 in.)
Depth	407 mm (15.6 in.)
Width	392 mm (15.4 in.)

Weight:

11.8 kilograms (26 pounds)

Heat Output:

240 BTU/hour

Power Cables:

Length 1.83 meters (6 feet)

Size 22 AWG

Graphics Printer**Size:**

Height 110 mm (4.3 in.)

Depth 370 mm (14.5 in.)

Width 400 mm (15.7 in.)

Weight:

5.9 kilograms (12.9 pounds)

Heat Output:

341 BTU/hour

Power Cable:

Length 1.83 meters (6 feet)
Size 18 AWG

Signal Cable:

Length 1.83 meters (6 feet)
Size 22 AWG

Electrical:

Minimum 104 Vac
Nominal 120 Vac
Maximum 127 Vac

Internal Modem

Power:

Parameter	+ 5 Vdc Voltage	+ 12 Vdc Voltage
Tolerance	+/- 5%	+/- 10%
Ripple	50 mV, P-P	50 mV, P-P
Maximum Current	300 mA	50 mA
Current Nominal	150 mA	25 mA

Interface

RS232C

Compact Printer

Size:

Height 88.9 mm (3.5 in)
Depth 221 mm (8.7 in)
Width 312.4 mm (12.3 in)

Weight:

2.99 kg (6.6 lb)

Heat Output:

54.6 Btu/hr

Power Cable:

Length 1.89 m (6 ft)
Size 28 AWG

Signal Cable:

Length 1.89 m (6 ft)
Size 3 by 18 AWG

Electrical:

Voltage 110 Vac 60 Hz

Glossary

μs Microsecond.

adapter. An auxiliary system 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 microprocessor throughout the rest of the system.

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

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

American Standard Code for Information

Interchange. (ASCII) The standard code, using a coded character set consisting of 7-bit coded characters (8 bits

including parity check), used for information interchange among data processing systems, data communication systems and associated equipment. The ASCII set consists of control characters and graphic characters.

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.

APA. All points addressable.

ASCII. American Standard Code for Information Interchange.

assembler. A computer program used to assemble. Synonymous with assembly program.

asynchronous communications. A communication mode in which each single byte of data is synchronized, usually by the addition of start/stop bits.

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

basic input/output system (BIOS). Provides the device level control of the major I/O devices in a computer system, which provides an operational interface to the system and relieves the programmer from concern over 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-half dot cycle per second in Morse code, one bit per second in a train of binary signals, 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 numerical application.

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 two.

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.

BIOS. Basic input/output system.

bit. In binary notation, either of the characters 0 or 1.

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

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; that is, 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.

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 binary character operated upon as a unit and usually shorter than a computer word. (2) The representation of a character.

CAS. Column address strobe.

cathode ray tube (CRT). A vacuum tube display in which a beam of electrons can be controlled to form alphanumeric characters or symbols on a luminescent screen, for example by use of a dot matrix.

cathode ray tube display (CRT display). (1) A device that presents data in visual form by means of controlled electron

beams. (2) The data display produced by the device as in (1).

CCITT. Comite Consultatif International Telegrafique et Telephonique.

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

channel. A path along which signals can be sent; for example, data channel or I/O channel.

characters per second (cps). A standard unit of measurement for printer output.

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) Loosely, one or more computer programs, or part of a computer program. (4) To represent data or a

computer program in a symbolic form that can be accepted by a data processor.

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

Comite Consultatif International. Telegrafique et Telephonique (CCITT) Consultative Committee on International Telegraphy and Telephone.

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

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. (1) The Boolean operation whose result has the Boolean value 1 if, and only if, each operand has the Boolean value 1. (2) Synonymous with AND operation.

contiguous. (1) Touching or joining at the edge or boundary. (2) Adjacent.

CPS. Characters per second.

CPU. Central processing unit.

CRC. Cyclic redundancy check.

CRT display. Cathode ray tube display.

CTS. Clear to send.
Associated with modem control.

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 humans or automatic means. (2) Any representations, such as characters or analog quantities, to which meaning is, or might be assigned.

decibel (dB). (1) A unit that expresses the ratio of two power levels on a logarithmic scale. (2) A unit for measuring relative power. The number of decibels is ten times the logarithm (base 10) of the ratio of the measured power levels; if the measured levels are voltages (across the same or equal resistance), the number of decibels is 20 times the log of the ratio.

decoupling capacitor. A capacitor that provides a

low-impedance path to ground to prevent common coupling between states of a circuit.

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 from 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 standardization committee.

DIP. Dual in-line package.

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

disk. Loosely, a magnetic disk unit.

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

DMA. Direct memory access.

DSR. Data set ready. Associated with modem control.

DTR. Data terminal ready. Associated with modem control.

dual in-line package (DIP). A widely used container for an integrated circuit. DIPs are pins usually in two parallel rows. These pins are spaced 1/10 inch apart and come in different configurations ranging from 14-pin to 40-pin configurations.

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.

EIA/CCITT. Electronic Industries Association/Consultative Committee on International Telegraphy and Telephone.

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

end-of-transmission character (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.

EOT. end-of-transmission character.

EPROM. Erasable programmable read-only memory

erasable programmable read-only memory (EPROM) A storage device whose contents can be erased by ultraviolet means and new contents stored by electrical means. EPROM information is not destroyed when power is removed.

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

ETX. End-of-text character.

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

flexible disk. Synonym for diskette.

firmware. Memory chips with integrated programs already incorporated on the chip.

gate. (1) A device or circuit that has no output until it is triggered into operation by one or more enable signals, or until an input signal exceeds a predetermined threshold amplitude. (2) A signal that triggers the passage of other signals through a circuit.

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

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

hex. Abbreviation for hexadecimal.

hexadecimal (Hex). Pertaining to a selection, choice, or condition that has 16 possible values or states. These values or states usually contain 10 digits and 6 letters, A through F/ Hexadecimal digits are equivalent to a power of 16.

high-order position. The leftmost position in a string of characters.

Hz. Hertz.

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

k. An abbreviation for the prefix kilo; that is, 1,000 decimal notation.

K. When referring to storage capacity, 2 to the tenth power; 1,024 in decimal notation.

KB (Kilobyte). 1,024 bytes.

k byte. 1,024 bytes.

kHz. A unit of frequency equal to 1,000 hertz.

kilo(k). One thousand.

latch. (1) A feedback loop in symmetrical digital circuits used to maintain a state. (2) A simple logic-circuit storage element comprising two gates as a unit.

LED. Light-emitting diode.

light-emitting diode (LED). A semi-conductor chip that gives off visible or infrared light when activated.

low-order position. The rightmost position in a string of characters.

m. (1) Milli; one thousand or thousandth part. (2) Meter.

M (Mega). 1,000,000 in decimal notation. When referring to storage capacity, 2 to the twentieth power; 1,048,576 in decimal notation.

mA. Milliampere.

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

main storage. A storage device in which the access time is effectively independent of the location of the data.

MB. Megabyte, 1,048,576 bytes.

mega (M). 10 to the sixth power, 1,000,000 in decimal notation. When referring to storage capacity, 2 to the twentieth power. 1,048,576 in decimal notation.

megabyte (MB). 1,048,576 bytes.

megahertz (MHz). A unit of measure of frequency. One megahertz equals 1,000,000 hertz.

MFM. Modified frequency modulation.

MHz. Megahertz.

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

microsecond. (μ s) One-millionth of a second.

milli(m). One thousand or one thousandth.

milliampere(mA). One thousandth of an ampere.

millisecond(ms). One thousandth of a 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 terminal equipment) to analog signals which 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.

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 specific departure from the norm. (2) A television type display, such as the IBM Monochrome Display. (3) Software or hardware that observes, supervises, controls, or verifies the operations of a system.

ms. Millisecond; one thousandth of a second.

multiplexer. A device capable of distributing the events of an interleaved sequence to the respective activities.

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.

nanosecond. (ns) One-billionth of a second.

nonconjunction. (1) The dyadic Boolean operation the result of which has the Boolean value 0 if, and only if, each operand has the Boolean value 1.

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

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.

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 inverted.

ns. Nanosecond; one-billionth of a second.

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. (1) 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.

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.

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.

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.

picture element (PEL). (1) The smallest displayable unit on a display. (2) Synonymous with pixel, PEL.

pinout. A diagram of functioning pins on a pinboard.

pixel. Picture element.

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.

printed circuit board. A piece of material, usually fiberglass, that contains a layer of conductive material, usually metal. Miniature electronic components on the fiberglass transmit electronic signals through the board by way of the metal layers.

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). Non-erasable programmable memory. PROM information is not destroyed when power is removed.

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.

PROM. Programmable read-only memory.

propagation delay. The time necessary for a signal to travel from one point on a circuit to another.

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 permissible values of the character in any digit place range from zero to one less than the radix of the digit place.

RAS. Row address strobe.

RGBI. Red-green-blue-intensity.

read-only memory (ROM). A storage device whose contents cannot be modified, except by a particular user, or when operating under particular conditions; for example, a storage device in which writing is prevented by a lockout.

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

red-green-blue-intensity (RGBI). The description of a direct-drive

color monitor which accepts red, green, blue, and intensity signal inputs.

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) On a calculator, a storage device in which specific data is stored.

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

ROM. Read-only memory.

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

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

RS-232C. The standards set by the EIA for communications between computers and external equipment.

RTS. Request to send. Associated with modem control.

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

scan line. The use of a cathode beam to test the cathode ray tube of a display used with a personal computer.

schematic. The description, usually in diagram form, of the logical and physical structure of an entire data base according to a conceptual model.

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.

sink. A device or circuit into which current drains.

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

source. The origin of a signal or electrical energy.

source circuit. (1) Generator circuit. (2) Control with sink.

SS. Start-stop transmission.

start bit. Synonym for start signal.

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

start signal. (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. Synonymous with start bit.

start-stop (SS) transmission. (1) A synchronous transmission such that a group of signals representing a character is preceded by a start signal and followed by a stop signal. (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 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.

stop bit. Synonym for stop signal.

stop signal. (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. Synonymous with stop bit.

strobe. (1) An instrument used to determine the exact speed of circular or cyclic movement. (2) A flashing signal displaying an exact event.

STX. Start-of-text character.

synchronous transmission. Data transmission in which the sending and receiving devices are operating continuously at the same frequency and are maintained, by means of correction, in a desired phase relationship.

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, respectively.

track. 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, tape, or disk, that is accessible to a given reading head position.

transistor-transistor logic (TTL). A circuit in which the multiple-diode cluster of the diode-transistor logic circuit has been replaced by a multiple-emitter transistor.

TTL. Transistor-transistor logic.

TX Data. Transmit data. Associated with modem control. External connections of the RS-232C asynchronous communications adapter interface.

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

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.

Notes:



Bibliography

Intel Corporation. *The 8086 Family User's Manual* This manual introduces the 8086 family of microcomputing components and serves as a reference in system design and implementation.

Intel Corporation. *8086/8087/8088 Macro Assembly Reference Manual for 8088/8085 Based Development System* This manual describes the 8086/8087/8088 Macro Assembly Language, and is intended for use by 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. *INS 8250 Asynchronous Communications Element*. This book documents Physical and operating characteristics of the INS 8250.

Notes:



Index

A

- +A0 3-7, 3-72
- +A0 thru A3 3-20
- A0 thru A07, memory signal 3-7
- A0 thru A19, I/O signal 2-23
- A0 thru A14, program cartridge signal 2-114
- A1 3-72
- A2 3-72
- A9 3-21, 3-72
- ACKNLG, graphics printer signal 3-113
- adapter
 - See diskette drive adapter
- adapter ROM module addresses, valid 5-18
- adapter cable
 - for serial devices 3-89
 - connector specifications 3-90
 - signal cable 3-89
 - for cassette
 - connector specifications 3-91
 - for IBM color display
 - connector specifications 3-93
- addresses
 - FDC (data register) 3-17
 - FDC (status register) 3-17
 - parallel printer attachments 5-13
 - ROM modules, valid 5-18
 - RS232-C attachments 5-13
- advanced BASIC, system ROM 4-13
- ALE, I/O signal 2-24
- ANSWER, modem command 3-44
- ASCII, extended 5-21
- ATR CD IN 3-9
- ATR LATCH 3-7

- attachable joystick
 - block diagram 3-78
 - connector specifications 3-79
 - electrical centering control 3-77
 - free floating mode 3-77
 - spring return mode 3-77
 - x-axis 3-77
 - y-axis 3-77
- AUDIO IN, I/O signal 2-28
- AUTO FEED XT 3-114
- available options 1-3

B

- BASE 1 ROM IN CARTRIDGE, program cartridge signal 2-115
- BASE 2 ROM IN CARTRIDGE, program cartridge signal 2-115
- BASIC, cartridge 4-13
- BASIC screen editor special functions 5-41
- BASIC workspace variables 5-16
- BAUDCLK 3-73
- beeper
 - block diagram 2-85
 - input sources 2-85
- BEL, graphics printer control code 3-117
- BIOS
 - example, interrupt usage 5-5
 - interrupt hex 10 4-17
 - interrupt hex 14 4-18
 - interrupt hex 16 4-15
 - interrupt hex 1D 4-17
 - memory map 5-17
 - power-on initialization stack-area memory location 5-13
 - vectors list, interrupt 5-7
 - vectors with special meanings
 - interrupt hex 1B - keyboard break address 5-8
 - interrupt hex 1C - timer tick 5-8
 - interrupt hex 1D - video parameters 5-9
 - interrupt hex 1E - diskette parameters 5-9
 - interrupt hex 1F - graphics character pointers (2nd 128) 5-9
 - interrupt hex 44 - graphics character pointers (1st 128) 5-9

- interrupt hex 48 - cordless keyboard translation 5-10
- interrupt hex 49 - non-keyboard scan-code translation-table address 5-10
- BIOS cassette logic
 - cassette read 5-49
 - cassette write 5-48
 - tape block components 5-50
 - tape block format 5-50
 - timing chart 5-49
 - data record architecture 5-50
 - error detection 5-51
 - software algorithms - interrupt hex 15 5-47
 - cassette status in AH 5-48
 - request types in register AH 5-47
- block diagrams
 - attachable joystick 3-78
 - beeper 2-85
 - cassette motor control 2-41
 - compact printer 3-134
 - diskette drive adapter 3-14
 - infra-red receiver 2-98
 - keyboard interface 2-106
 - memory and display expansion 3-6
 - modem 3-36
 - parallel printer attachment 3-97
 - read hardware 2-40
 - serial port (RS232) 2-127
 - sound subsystem 2-89
 - system 1-6
 - system board 2-9
 - video color/graphics subsystem 2-46
 - write hardware 2-40
- bootstrap stack-area memory location 5-13
- break, cordless keyboard 5-34
- BREAK, modem command 3-44
- buffer, cordless keyboard 5-36
- bus cycle time 2-13
- BUSY, graphics printer signal 3-113

C

- cable, adapter
 - See adapter cable
- cable, keyboard
 - See keyboard cord
- cable, power
 - See power cable
- cable, signal
 - See signal cable
- CABLE CONNECT 2-101, 3-87
- CAN, compact printer control code 3-141
- CAN, graphics printer control code 3-118
- CARD INSTALL 3-73
- CARD SLCTD, I/O signal 2-28
- cartridge BASIC 4-13
- cartridge, program
 - See program cartridge
- CARTRIDGE RESET TAB, program cartridge signal 2-116
- +CAS 3-8
- cassette BASIC, system ROM 4-12
- cassette interface
 - block diagram, cassette motor control 2-41
 - block diagram, read hardware 2-40
 - block diagram, write hardware 2-40
 - connector specifications 2-41
 - motor control 2-41
 - output to audio subsystem 2-39
- cassette logic, BIOS
 - See BIOS cassette logic
- character codes, cordless keyboard 5-27
- character set, compact printer 3-148, 3-149
- character set 1 3-128
 - description 3-109
- character set 2 3-130
 - description 3-109
- CLK, I/O signal 2-23
- clock crystal frequency, system 2-6
- color burst signal frequency 2-6
- color display
 - connector specifications 3-83
 - electrical requirements 3-81
 - horizontal drive frequency 3-82
 - operating characteristics 3-82

screen characteristics 3-82

video bandwidth 3-82

color/graphics

all points addressable graphics (APA) mode

high-resolution 2-color 2-58

high-resolution 4-color 2-59

low-resolution 16-color 2-56

medium-resolution 4-color 2-57

medium-resolution 16-color 2-58

modes available 2-56

screen border colors 2-45

storage organization memory map 2-61

alphanumeric (A/N) mode

attribute byte definition 2-55

attributes 2-43

display character format 2-54

block diagram 2-46

character size and description 2-43

characters available 2-44

composite connector specifications 2-83

CRT page register 2-47

CRT/processor page register

CRT page 0 thru 2 2-79

processor page 0 thru 2 2-79

video adr mode 0 and 1 2-80

direct drive connector specifications 2-82

four-color mode palette 2-50

light pen connector specifications 2-75

memory map 2-48

mode selection summary 2-81

sequence for changing modes 2-81

page register 2-47

programming considerations

6845 CRT controller 2-75

register table 2-76

RF connector specifications 2-83

ROM character generator 2-44, 2-49

sixteen-color mode palette 2-52

storage organization

accessing the RAM 2-47

RAM address 2-47

summary of available colors 2-53

two-color mode palette 2-50

video bandwidth 2-49

- video gate array
 - address register 2-74
 - border color register bit functions 2-66
 - mode control 1 register bit functions 2-64
 - mode control 2 register bit functions 2-66
 - attribute byte definition 2-67
 - mode selection summary 2-81
 - palette mask register bit functions 2-65
 - palette registers 2-71
 - format 2-71
 - register addresses 2-63
 - reset register bit functions 2-69
 - sequence for changing modes 2-81
 - status register bit functions 2-73
 - vertical retrace interrupt 2-82
- video I/O devices and addresses
 - 6845 CRT 2-45, 2-47, 2-75
 - register table 2-76
- command character, modem 3-40
- commonly used functions, cordless keyboard 5-38
- compact printer
 - block diagram 3-134
 - character set 3-148, 3-149
 - connector specifications 3-150
 - control codes 3-140 thru 3-141 thru 3-147
 - description 3-133
 - print mode combinations, allowable 3-140
 - serial interface
 - description 3-139
 - timing diagram 3-139
 - signal cable 3-133
 - specifications, general 3-135 thru 3-138
- compatibility to Personal Computers
 - black and white monochrome display 4-18
 - color graphics capability differences 4-15
 - color modes available only on PCjr 4-16
 - comparison, PCjr to Personal Computers hardware 4-10
 - non-DMA operation considerations 4-19
 - screen buffer differences 4-12
 - software determination of the computer 4-19
 - timing dependencies 4-5
 - unequal configurations 4-7
 - user available read/write memory 4-12
 - video hardware address difference 4-16

- complex sound generator
 - See SN76496N
 - See sound subsystem
- connector for television
 - channel selector switch 3-85
 - computer/television selector switch 3-85
 - connector specifications 3-86
 - signal cable 3-85
- connector locations, system board 2-10
- connector specifications
 - adapter cable for cassette 3-91
 - adapter cable for color display 3-93
 - adapter cable for serial devices 3-89
 - attachable joystick 3-79
 - audio 2-87
 - cassette (system board) 2-41
 - color display 3-83
 - compact printer 3-150
 - composite video (system board) 2-83
 - connector for television 3-85
 - direct drive (system board) 2-82
 - diskette drive 3-29
 - diskette drive adapter 3-25
 - games interface (system board) 2-123
 - graphics printer 3-115
 - infra-red receiver (system board) 2-100
 - I/O expansion 2-22
 - keyboard cord 3-88
 - light pen (system board) 2-75
 - memory and display expansion 3-10
 - modem 3-75
 - parallel printer attachment 3-104
 - power board 2-136
 - program cartridge 2-117
 - RF modulator (system board) 2-83
 - system board 2-10
- control codes, compact printer 3-140 thru 3-147
- control codes, graphics printer 3-116 thru 3-122
- control latch, parallel printer attachment 3-101
- controller, floppy disk (FDC) 3-16
- cordless keyboard
 - BASIC screen editor special functions 5-41
 - battery power 2-102
 - buffer 5-36

- CABLE CONNECT signal 2-101
- character codes 5-27
- commonly used functions 5-38
- data format 2-104
- data path 2-102
- disabling the infra-red circuits 2-101, 2-103
- DOS special functions 5-42
- extended codes 5-30
- function map, 83-key keyboard to cordless keyboard 5-25
- interface block diagram 2-106
- layout and keybutton numbering 5-22
- parity bit 2-104
- phantom-key detection 2-103
- scan-codes, matrix 5-23
- shift keys combinations, allowable
- shift keys priorities 5-33
- shift states description 5-31
 - alt key 5-32
 - caps lock 5-33
 - ctrl key 5-32
 - shift key 5-32
- special handling description
 - break 5-34
 - enable/disable keyboard click 5-36
 - function lock 5-35
 - functions 1 thru 10 5-35
 - other characteristics 5-36
 - pause 5-34
 - phantom-key scan-code (hex 55) 5-36
 - print screen 5-34
 - run diagnostics 5-36
 - screen adjustment 5-35
 - scroll lock 5-35
 - system reset 5-34
 - typematic suppression 5-36
- start bit 2-104
- stop bit 2-104
- transmission timing 2-105
- transmitter, infra-red 2-103
- 80C48, keyboard microprocessor 2-103
- COUNT, modem command 4-45
- CPU DLY 3-8
- CPU LATCH 3-9
- CR, compact printer control code 3-141

- CR, graphics printer control code 3-117
- CS2 thru -CS7, program cartridge signal 2-115
- CTS, modem 3-71

D

- D0 thru D7 3-7, 3-73
- D0 thru D7, program cartridge signal 2-114
- DACK 0, I/O signal 2-27
- DACK 0 3-21
- DATA 1 thru DATA 8, graphics printer signal 3-113
- data latch, parallel printer attachment 3-99
- DC2, compact printer control code 3-141
- DC2, graphics printer control code 3-118
- DC4, compact printer control code 3-141
- DC4, graphics printer control code 3-118
- DIAL, modem command 3-46
- differences, PCjr to Personal Computer
 - addressing of internal modem 4-18
 - addressing of serial port 4-18
 - black and white monochrome display 4-18
 - color graphics capability differences 4-15
 - color modes available only on PCjr 4-16
 - compatibility of hardware 4-10
 - compatibility of configurations 4-7
 - non-DMA operation considerations 4-19
 - screen buffer 4-12
 - software determination of the computer 4-19
 - timing dependencies 4-5
 - user available read/write memory 4-12
 - video hardware address difference 4-17
- DIRECTION 3-23
- DISABLE CAS0 3-8
- DISABLE EDATA 3-7
- diskette 3-31
- DISKETTE CARD INSTALLED 3-20
- DISKETTE CS 3-21
- diskette drive differences, PCjr to Personal Computer 4-13
- diskette drive
 - connector specifications 3-29
 - electrical requirements 3-28
 - load lever 3-27

- media cooling fan 3-28
- sensors 3-28
- diskette drive adapter
 - additional comments 3-19
 - block diagram 3-14
 - connector specifications 3-25
 - digital output register (DOR) 3-15
 - diskette drive constants 3-19
 - diskette drive interface signals 3-22
 - diskette format 3-18
 - electrical requirements 3-24
 - floppy disk controller (FDC) 3-16
 - I/O channel interface signals 3-19
 - location 2-10
 - signal cable 3-13
 - watchdog timer (WDT) 3-16
- 0 thru D7, I/O signal 2-24
- DOS special functions 5-42
 - DRIVE SELECT 3-22
 - DRQ 0 3-21
 - DRQ 0, I/O signal 2-27
 - DSR modem 3-72
 - DTMF (dial-tone modulated-frequency) 3-35
 - DTR, modem 3-71

E

- electrical centering control, joystick 3-77
- electrical requirements
 - color display 3-82
 - compact printer 3-137
 - diskette drive 3-28
 - diskette drive adapter 3-24
 - graphics printer 3-109
- enable/disable keyboard click 5-36
- ESC control codes, compact printer 3-141 thru 3-146
- ESC control codes, graphics printer 3-118 thru 3-127
- extended ASCII 5-21
- extended codes, cordless keyboard 5-31

F

- FF, compact printer control code 3-146
- FF, graphics printer control code 3-117
- FDC (floppy disk controller)
 - See floppy disk controller
- floppy disk 3-31
- floppy disk controller (FDC) 3-16
 - commands 3-18
 - functions not supported 3-18
 - I/O addresses 3-17
 - power-up parameters settings 3-17
- floppy disk drive (FDD) 3-16
- FORMAT, modem command 3-47
- function lock, cordless keyboard 5-35
- functions 1 thru 10, cordless keyboard 5-35

G

- games interface
 - block diagram 2-120
 - connector specifications 2-123
 - digital input format 2-121
 - joystick input data 2-119
 - paddle input data 2-122
 - pushbutton inputs 2-122
 - resistance range 2-121
 - resistive input format 2-121
 - resistive to digital input equation 2-119
- GATE 3-9
- graphics
 - See color/graphics
- graphics printer
 - character set 1 3-128
 - description 3-109
 - character set 2 3-130
 - description 3-109
 - control codes 3-116 thru 3-127
 - DIP switch settings 3-111
 - electrical requirements 3-109
 - environmental conditions 3-109

- interface timing diagram 3-113
- signal cable 3-107
- signal pin assignments 3-115
- signals, interface 3-113 thru 3-114
- specifications 3-107

H

- HANGUP, modem command 3-48
- hardware differences, PCjr to Personal Computer 4-10
- HEAD SELECT 1 3-23
- HLDA, I/O signal 2-27
- horizontal drive frequency, color display 3-82
- HRQ, I/O signal 2-26
- HT, compact printer control code 3-146
- HT, graphics printer control code 3-117

I

- IBM Connector for Television
 - See connector for television
- IBM PC Compact Printer
 - See compact printer
- IBM PCjr Adapter Cable for Cassette
 - See adapter cable for cassette
- IBM PCjr Adapter Cable for IBM Color Display
 - See adapter cable for IBM color display
- IBM PCjr Adapter Cable for Serial Devices
 - See adapter cable for serial devices
- IBM PCjr Attachable Joystick
 - See attachable joystick
- IBM PCjr Diskette Drive
 - See diskette drive
- IBM PCjr Diskette Drive Adapter
 - See diskette drive adapter
- IBM PCjr Internal Modem
 - See internal modem
- IBM PCjr Parallel Printer Attachment
 - See parallel printer attachment

IBM PCjr 64KB Memory and Display Expansion

See memory and display expansion

IBM Personal Computer Graphics Printer

See graphics printer.

-INDEX 3-24

infra-red link

- block diagram, receiver 2-98
- connector specifications 2-100
- functional description 2-97
- programming considerations
 - parity errors 2-99
 - phase errors 2-99
- receiver 2-97
- transmitter 2-103
- test frequency 2-98

INITIALIZE, modem command 3-48

IO/-M, I/O signal 2-26

I/O channel

- expansion connector specifications 2-22
- I/O read/write cycle times 2-21
- map 2-29
- memory read/write cycle times 2-21
- port A0 input description 2-36
- port A0 output description 2-35
- signals 2-23 thru 2-28
 - diskette drive adapter 3-19
 - memory and display expansion 3-7
 - modem 3-70, 3-73

integrated circuits

- See 6845 CRT controller
- See 80C48
- See 8088
- See INS8250A
- See MCM6665AL15
- See MK38000
- See TMM23256P
- See TMS4164-15
- See 8253-5 programmable timer/counter
- See 8255A
- See 8259A
- See 8284A clock chip
- See SN76496N

- internal modem
 - address, memory location of 5-13
 - asynchronous communications element 3-68
 - block diagram 3-36
 - command
 - arguments 3-41
 - command character 3-40
 - delimiter 3-41
 - format 3-40
 - format guidelines 3-40
 - commands 3-44 thru 3-58
 - connector specifications 3-75
 - dialing 3-60
 - status 3-60
 - default state 3-63
 - editing/changing commands 3-59
 - location 2-10
 - loss of carrier 3-60
 - modes of operation 3-68
 - opposite commands 3-60
 - programming examples 3-63
 - signals 3-70
 - smart 103 modem 3-37
 - telephone company interface 3-74
 - transmitter/receiver data format 3-70
 - 8250A 3-68
 - description of registers 3-69
- IOR 3-73
- INS8250A 3-68
- INS8250A -OUT 1, modem 3-71
- INT, graphics printer signal 3-114
- interrupt controller, programmable
 - See 8259A
- interrupt setup example 2-16
- interrupt usage example 5-5
- interrupt vector list 5-7
- interrupts, hardware
 - IRQ3 2-129
 - +IRQ4 3-70
 - +IRQ6 3-20
 - +IRQ7 3-99
 - priority 2-15
 - used by system board 2-6
 - used by I/O channel 2-6

interrupts reserved for BIOS, DOS, and BASIC 5-14

-IOR, I/O signal 2-25

-IOW 3-73

-IOW, I/O signal 2-25

IRQ1, I/O signal 2-25

IRQ2, I/O signal 2-25

IRQ7, I/O signal 2-25

J

joystick, attachable

See attachable joystick

K

keyboard

See cordless keyboard

keyboard click, enable/disable 5-36

keyboard cord

-CABLE CONNECT 2-101, 3-81

connector specifications 3-88

keyboard microprocessor

See 80C48

L

-LCG 3-9

LF, compact printer control code 3-146

LF, graphics printer control code 3-117

light pen 2-74

line spacing, graphics printer 3-108

load lever, diskette drive 3-27

location

DIP switches, graphics printer

diskette drive adapter 2-10

internal modem 2-10

memory and display expansion 2-10, 3-5

LONG RESPONSE, modem command 3-49

M

maps

- See BIOS, memory map
- See cordless keyboard, function map
- See memory maps
- See scan-code map
- See system memory map

matrix scan-codes, cordless keyboard 5-23

MCM6665AL15 2-17, 3-5

MD0 thru MD7 3-7

media cooling fan 3-28

MEM A0 thru A7 3-7

memory and display expansion

- block diagram 3-6
- configuration
 - requirements 3-5
- connector specifications 3-10
- EVEN memory space 3-5
- location 2-10
- modules used, type 3-5
- ODD memory space 3-5
- signals 3-7

memory maps

- BIOS 5-17
- BIOS, BASIC, and DOS reserved interrupts 5-14
- graphics storage 2-61
- memory address map 2-20
- reserved memory locations 5-15
- system, memory allocated for 2-20
- video color/graphics subsystem 2-46

memory, 64K RAM

- See memory and display expansion
- See RAM

memory refresh 2-17

memory requirements 4-12

memory, user available 4-12

-MEMR, I/O signal 2-25

-MEMW, I/O signal 2-26

microprocessor, keyboard

- See 80C48

microprocessor, system

- See 8088

minimum mode, 8088 2-6

MK38000 2-19
-MODEM CS/DISKETTE CS 3-72
MODEM, modem command 3-50
modem
 See internal modem
+MODEM INTR 3-73
modified frequency modulation (MFM) 3-13
modules
 See integrated circuits
motor control, cassette 2-39
-MOTOR ENABLE 3-22

N

NUL, compact printer control code 3-147
NEC fPDP765 3-13
NEW, modem command 3-50
NMI (Non-Maskable Interrupt) 2-7
Noise Generator 2-93
Non-Keyboard Scan-code Architecture 5-42
 scan-code map 5-45
 translate table format 5-44
 translate table default values 5-44

O

ORIGINATE, modem command 3-50
-OUT 2, modem 3-71
options, available 1-3

P

PA0 thru PA7 2-31
pause, cordless keyboard 5-34
parallel printer attachment
 address, memory location of 5-13
 block diagram 3-97

- connector specifications 3-104
- control latch
 - reading from 3-101
 - writing to 3-101
- data latch
 - format 3-100
 - reading from 3-99
 - writing to 3-99
 - printer control 3-101
- +IRQ7 logic diagram 3-99
- printer status signals descriptions 3-101
- PB0 thru PB7 2-31 thru 2-32
- PC0 thru PC7 2-33 thru 2-34
- PE, graphics printer signal 3-114
- phantom-key detection 2-103
- phantom-key scan-code (hex 55), cordless keyboard 5-36
- PICKUP, modem command 3-51
- port A0 input description 2-36
- port A0 output description 2-35
- power-on initialization stack-area memory location 5-13
- power cable
 - color display 3-81
- power supply
 - connector specifications 2-138
 - power available 2-135
 - power board
 - over-voltage over-current protection 2-137
 - Vdc outputs 2-136
 - transformer 2-134
 - over-voltage over-current protection 2-137
 - Vac input 2-135
 - Vac output 2-135
- pulse dialing 3-35
- print method, graphics printer 3-107
- print modes, graphics printer 3-116
- print screen, cordless keyboard 5-34
- print sizes, graphics printer 3-108
- print speed, graphics printer 3-107
- printer
 - See compact printer
 - See graphics printer
- printer status 3-101

- program cartridge
 - connector specification 2-117
 - description 2-107
 - momentary reset land 2-116
 - ROM chip select table 2-114
 - ROM locations, cartridge 2-118
 - ROM mapping 2-107
 - signals 2-114
 - slot description 2-107
 - storage conventions
 - initial program loadable 2-108
 - DOS conventions 2-110
 - cartridge BASIC 2-111
 - type ROM modules used 2-107
- programmable interrupt controller
 - See 8259A
- programmable timer/counter, 8253-5 2-6

Q

- QUERY, modem command 3-52

R

RAM, 64K

- address space mapped to 2-17
- EVEN memory 2-17
- memory refresh 2-17
- ODD memory 2-17
- parity 2-17
- read/write cycle times 2-21
- speed 2-17
- type modules used 2-17
- 6845 CRT controller 2-17
- +RAS 3-7
- READ DATA 3-24
- READY, I/O signal 2-24
- reserved interrupts, BIOS, DOS , and BASIC 5-14
- reserved memory locations 5-15

- RESET, I/O signal 2-23
- RESET 3-20
- RESET, modem 3-72
- RETRY, modem command 3-53
- RI, modem 3-71
- RLSD, modem 3-72
- ROM subsystem
 - address space mapped to 2-19
 - memory map 2-20
 - read/write cycle times 2-21
 - type modules used 2-19
- ROM module code accessed by system 5-18
- ROM module addresses, valid 5-19
- RTS, modem 3-71
- run diagnostics, cordless keyboard 5-36

S

- scan-code map 5-45
- scan-codes
 - cordless keyboard matrix 5-23
 - default non-keyboard 5-43
- screen adjustment, cordless keyboard 5-35
- scroll lock, cordless keyboard 5-35
- serial port
 - address, memory location of 5-13
 - block diagram 2-127
 - connector specifications 2-134
 - control signals 2-129
 - diskette operations conflict 2-125
 - interface 2-129
 - interrupt IRQ3 2-129
 - modes of operation 2-128
 - I/O decodes 2-128
 - output signals 2-131
 - ring indicate 2-130
 - use of the divisor-latch access-bit 2-128
 - voltage interchange levels 2-130
- 8250A
 - accessible registers 2-131
 - features 2-125
 - initialization program, sample 2-134

- programmable baud rate generator 2-132
 - baud rate at 1.7895 MHz 2-132
 - maximum operating frequency 2-132
 - output frequency equation 2-132
- SI, compact printer control code 3-147
- SI, graphics printer control code 3-118
- signal cable
 - adapter cable for cassette 3-91
 - adapter cable for serial devices 3-89
 - diskette drive 3-13
 - color display 3-81
 - connector for television 3-85
 - graphics printer 3-107
- SIN, modem 3-71
- SLCT, graphics printer signal 3-114
- smart 103 modem
 - See internal modem.
- SO, compact printer control code 3-147
- SO, graphics printer control code 3-112
- sound subsystem
 - block diagram 2-89
 - complex sound generator (SN76496N) 2-88
 - audio tone generator features 2-89
 - audio tone generator register address field 2-91
 - audio tone generator frequency 2-91
 - frequency generation 2-91
 - audio tone generator attenuator 2-92
 - audio tone generator noise generator 2-93
 - audio tone generator noise feedback control 2-93
 - control registers 2-90
 - interface 2-89
 - connector specifications 2-87
 - mpx (analog multiplexer) 2-87, 2-94
 - data transfer 2-95
 - output buffer amperage 2-95
 - signal description 2-87
 - signal destinations 2-87
 - sound sources 2-88
 - use of an external speaker 2-87
- SOUT, modem 3-71
- special-functions, cordless keyboard specific 4-14
- special functions, cordless keyboard
 - BASIC screen editor 5-41
- special functions, DOS 5-42

- SPEED, modem command 3-54
- STEP 3-22
- STROBE, graphics printer signal 3-113
- system-accessible ROM-modules 5-18
- system block diagram 1-6
- system board
 - block diagram 2-9
 - clock crystal frequency 2-6
 - connectors specifications and locations 2-10
 - interrupts used by system board 2-6
 - major components list 2-8
 - RAM, 64K 2-17
 - size 2-5
 - subsystems list 2-6
 - 8253-5 programmable timer/counter 2-6
- system memory map 5-17
- system microprocessor
 - See 8088
- system reset 5-34

T

- timers
 - watchdog timer (WDT) 3-16
- timing dependencies, compatibility 4-5
- timing diagrams
 - parallel printer interface 3-113
- timing, keyboard transmission 2-105
- timing using I/O devices 4-5
- timing using program execution speed 4-5
- TMM23256P 2-19
- TMS4164-15 2-17, 3-5
- TRACK 0 3-24
- track 00 sensor 3-28
- translate table format, non-keyboard scan-code 5-43
- transmitter, infra-red 2-103
- TRANSPARENT, modem command 3-55
- typematic suppression 5-36

U

- usage of BIOS 5-5
- usage of keyboard 5-21

V

- vectors list, interrupt 5-7
- vertical refresh, color display 3-82
- video bandwidth, color display 3-82
- video color/graphics subsystem
 - See color/graphics
- video gate array
 - register addresses 2-63
- VIDEO MEMR 3-8
- VOICE, modem command 3-56
- VT, compact printer control code 3-147

W

- WAIT, modem command 3-57
- WE 3-9
- work space variables, BASIC 5-16
- WRITE DATA 3-23
- WRITE ENABLE 3-23
- write precompensation 3-13
- WRITE PROTECT 3-24
- write protect sensor 3-28

X

- x-coordinate 2-121, 3-77
- XMIT, modem command 3-57
- +XRESET, modem 3-72

Y

y-coordinate 2-121, 3-77

Z

ZTEST, modem command 3-58

Numerals

64KB memory and display expansion

See memory and display expansion

6845 CRT 2-45, 2-47, 2-75

register table 2-76

80C48 2-103

8088

addressable range 2-6

clock frequency 2-6, 2-13

clock cycle time 2-13

minimum mode 2-6

NMI interrupt 2-15

operating frequency 2-13

8253-5 programmable timer/counter 2-6, 2-85

cassette data to cassette control 2-39

8255A-5 2-85

audio input 2-85

bit assignments 2-31

cassette data from cassette control 2-39

cassette motor control 2-39

8259A (programmable interrupt controller)

characteristics as set up 2-16

hex types of interrupts issued 2-16

interrupt assignments 2-15

I/O addresses 2-16

priority of interrupts 2-15

setup example 2-16

8284A clock chip 2-13

SN76496N 2-88

C

C

C



C

C

C



Reader's Comment Form

TECHNICAL REFERENCE

6322963

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

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





International Business Machines Corporation

P.O. Box 1328-W
Boac Raton, Florida 33432

6322963

Printed in United States of America