Technical Reference
Update to the IBM PCjr Technical Reference

This update contains information that is pertinent to the IBM PC Compact Printer.

Insert the pages contained in this package into your IBM PCjr Technical Reference.

The following pages replace existing pages in your Technical Reference.

- Table of Contents (vii, viii, ix, and x)
- 3-3 and 3-4
- A1 and A2
- B1 and B2
- D-7 and D-8
- Index-1 through Index-24

Add the following pages to your Technical Reference.

- Tab Index xi, xii, xiii, xiv
- 3-133 through 3-150
- B-47
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.
The IBM PCjr Technical Reference manual describes the hardware design and provides interface information for the IBM PCjr. 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 PCjr computer.

You should be familiar with the use of the IBM PCjr, 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
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 ..............................
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 PC\textit{jr} Parallel Printer Attachment

  - The IBM PC\textit{jr} 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 PC\textit{jr} Joystick

  - The IBM PC\textit{jr} 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 PC\textit{jr} system.

• IBM PC\textit{jr} Keyboard Cord

  - The IBM PC\textit{jr} Keyboard Cord option is used to connect the IBM PC\textit{jr} Cordless Keyboard to the system board.

1–4 Introduction
- IBM PCjr Adapter Cable for Serial Devices
  - This option is an adapter cable that allows connection of serial devices to the IBM PCjr system board.

- IBM PCjr Adapter Cable for Cassette
  - This option is an adapter cable that allows a cassette recorder to be connected to the IBM PCjr.

- IBM PCjr Adapter Cable for Color Display
  - This adapter cable allows the IBM Color Display to be connected to the IBM PCjr.

The following is a block diagram of the IBM PCjr system.
System Block Diagram (Sheet 1 of 2)
These components are contained on the 64KB memory and display expansion card. They are included here for completeness.

System Block Diagram (Sheet 2 of 2)
Notes:

1-8 Introduction
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
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:
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 Connector Specifications (Part 1 of 3)
Front View

Left Cartridge Slot

Right Cartridge Slot

A1

B1

A18

B18

I/O Expansion Connector

A1

B1

A30

B30

Right Side View

System Board Connector Specifications (Part 2 of 3)
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 μs.
PC
t Hardware Interrupts

Nine hardware levels of interrupts are available for the PC
t 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:

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

Hardware Interrupts
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.

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

Example Set Up

2-16 Interrupt Controller
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.
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:
<table>
<thead>
<tr>
<th>Cartridge Chip Selects</th>
<th>Memory Map</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFFFF</td>
<td>BIOS/Diagnostic/Cassette Basic Program Area</td>
</tr>
<tr>
<td></td>
<td>Standard Application Cartridge</td>
</tr>
<tr>
<td></td>
<td>Standard Application Cartridge</td>
</tr>
<tr>
<td></td>
<td>Reserved For Future Cartridge</td>
</tr>
<tr>
<td></td>
<td>Reserved For Future Cartridge</td>
</tr>
<tr>
<td></td>
<td>Reserved for I/O ROM</td>
</tr>
<tr>
<td></td>
<td>Video RAM</td>
</tr>
<tr>
<td></td>
<td>Reserved Future Video</td>
</tr>
<tr>
<td></td>
<td>Reserved Future User RAM</td>
</tr>
<tr>
<td></td>
<td>Expansion RAM</td>
</tr>
<tr>
<td></td>
<td>Base RAM</td>
</tr>
<tr>
<td>F0000</td>
<td></td>
</tr>
<tr>
<td>E8000</td>
<td></td>
</tr>
<tr>
<td>E0000</td>
<td></td>
</tr>
<tr>
<td>D8000</td>
<td></td>
</tr>
<tr>
<td>D0000</td>
<td></td>
</tr>
<tr>
<td>C0000</td>
<td></td>
</tr>
<tr>
<td>B8000</td>
<td></td>
</tr>
<tr>
<td>A0000</td>
<td></td>
</tr>
<tr>
<td>20000</td>
<td></td>
</tr>
<tr>
<td>10000</td>
<td></td>
</tr>
<tr>
<td>00000</td>
<td></td>
</tr>
</tbody>
</table>
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-μs/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.

<table>
<thead>
<tr>
<th>Signal</th>
<th>I/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK</td>
<td>O</td>
<td>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.</td>
</tr>
<tr>
<td>RESET</td>
<td>O</td>
<td>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 μs.</td>
</tr>
<tr>
<td>A0–A19</td>
<td>I/O</td>
<td>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</td>
</tr>
</tbody>
</table>
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**

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
-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.

<table>
<thead>
<tr>
<th>DRQ 0</th>
<th>0</th>
<th>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.</th>
</tr>
</thead>
<tbody>
<tr>
<td>-DACK 0</td>
<td>1</td>
<td>This line should come from an external DMA and should indicate that a byte is being transferred from memory to the FDC.</td>
</tr>
<tr>
<td>HLDA</td>
<td>O</td>
<td>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.</td>
</tr>
</tbody>
</table>
-CARD SLCTD
This line should be pulled down 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
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

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

### 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=SELC CLK1 INPUT D4= +Disable HRQ

---

I/O Channel 2-29
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

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA0 thru PA7 Lines</td>
<td>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.</td>
</tr>
<tr>
<td>PB0 (+Timer 2 Gate)</td>
<td>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.</td>
</tr>
<tr>
<td>PB1 (+Speaker Data)</td>
<td>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.</td>
</tr>
<tr>
<td>PB2 (+Alpha -Graphics)</td>
<td>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.</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>PB6</th>
<th>PB5</th>
<th>Sound Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>8253-5 Timer 2</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Cassette Audio Input</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>I/O Channel Audio In</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>76496</td>
</tr>
</tbody>
</table>

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**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7</td>
<td><strong>(Enable NMI)</strong> When this bit is a 1, the NMI is 'enabled'. When it is a 0, it is 'disabled'.</td>
</tr>
<tr>
<td>D6</td>
<td><strong>(IR test ENA)</strong> 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.</td>
</tr>
<tr>
<td>D5</td>
<td><strong>(Selc Clk1 input)</strong> 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.</td>
</tr>
<tr>
<td>D4</td>
<td><strong>(+Disable HRQ)</strong> 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.</td>
</tr>
</tbody>
</table>
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.
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.
2-40  Cassette Interface

Cassette-Interface Read-Hardware Block Diagram

Cassette-Interface Write-Hardware Block Diagram
Cassette-Motor Control Block Diagram

Cassette Connector Specifications
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 1FFFFF 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.
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 ELelement) 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:

<table>
<thead>
<tr>
<th>Palette Address Bit</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>Palette Register 0</td>
</tr>
<tr>
<td>1</td>
<td>Palette Register 1</td>
</tr>
</tbody>
</table>

Palette Logic (1 of 3)
In four-color modes, the palette is defined by using two bits (PA1 and PA0), with the following logic:

<table>
<thead>
<tr>
<th>Palette Address Bits</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA1</td>
<td>PA0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Palette Address Bits</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA3  PA2  PA1  PA0</td>
<td></td>
</tr>
<tr>
<td>0      0      0      0</td>
<td>Palette Register 0</td>
</tr>
<tr>
<td>0      0      0      1</td>
<td>Palette Register 1</td>
</tr>
<tr>
<td>0      0      1      0</td>
<td>Palette Register 2</td>
</tr>
<tr>
<td>0      0      1      1</td>
<td>Palette Register 3</td>
</tr>
<tr>
<td>0      1      0      0</td>
<td>Palette Register 4</td>
</tr>
<tr>
<td>0      1      0      1</td>
<td>Palette Register 5</td>
</tr>
<tr>
<td>0      1      1      0</td>
<td>Palette Register 6</td>
</tr>
<tr>
<td>0      1      1      1</td>
<td>Palette Register 7</td>
</tr>
<tr>
<td>1      0      0      0</td>
<td>Palette Register 8</td>
</tr>
<tr>
<td>1      0      0      1</td>
<td>Palette Register 9</td>
</tr>
<tr>
<td>1      0      1      0</td>
<td>Palette Register 10</td>
</tr>
<tr>
<td>1      0      1      1</td>
<td>Palette Register 11</td>
</tr>
<tr>
<td>1      1      0      0</td>
<td>Palette Register 12</td>
</tr>
<tr>
<td>1      1      0      1</td>
<td>Palette Register 13</td>
</tr>
<tr>
<td>1      1      1      0</td>
<td>Palette Register 14</td>
</tr>
<tr>
<td>1      1      1      1</td>
<td>Palette Register 15</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>I</th>
<th>R</th>
<th>G</th>
<th>B</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Black</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Blue</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Green</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Cyan</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Red</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Magenta</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Brown</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Light Gray</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Dark Gray</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Light Blue</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Light Green</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Light Cyan</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Pink</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Light Magenta</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Yellow</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>White</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Display Character Code Byte</th>
<th>Attribute Byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>7  6  5  4  3  2  1  0</td>
<td>7  6  5  4  3  2  1  0</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Attribute Function</th>
<th>Attribute Byte Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>7</td>
</tr>
<tr>
<td>Fore-Ground Blink</td>
<td>PA2</td>
</tr>
<tr>
<td>Normal</td>
<td>B</td>
</tr>
<tr>
<td>Reverse Video</td>
<td>B</td>
</tr>
<tr>
<td>Nondisplay (Off)</td>
<td>B</td>
</tr>
<tr>
<td>Nondisplay (On)</td>
<td>B</td>
</tr>
</tbody>
</table>

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:
<table>
<thead>
<tr>
<th>Graphics Mode</th>
<th>Horiz. (PELs)</th>
<th>Vert. (Rows)</th>
<th>Number of Colors Available (Includes Background Color)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low-Resolution 16-Color</td>
<td>160</td>
<td>200</td>
<td>16 (Includes b-and-w)</td>
</tr>
<tr>
<td>Medium-Resolution 4-Color</td>
<td>320</td>
<td>200</td>
<td>4 Colors of 16 Available</td>
</tr>
<tr>
<td>Medium-Resolution 16-Color</td>
<td>320</td>
<td>200</td>
<td>16 (Includes b-and-w)</td>
</tr>
<tr>
<td>High-Resolution 2-Color</td>
<td>640</td>
<td>200</td>
<td>2 Colors of 16 Available</td>
</tr>
<tr>
<td>High-Resolution 4-Color</td>
<td>640</td>
<td>200</td>
<td>4 Colors of 16 Available</td>
</tr>
</tbody>
</table>

**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:
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:

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.

```
<table>
<thead>
<tr>
<th>PA7</th>
<th>PA6</th>
<th>PA5</th>
<th>PA4</th>
<th>PA3</th>
<th>PA2</th>
<th>PA1</th>
<th>PA0</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA3</td>
<td>PA2</td>
<td>PA1</td>
<td>PA0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>First Display PEL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Second Display PEL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

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:

```
7 6 5 4 3 2 1 0
PA0 PA0 PA0 PA0 PA0 PA0 PA0 PA0
```

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.
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.
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.

<table>
<thead>
<tr>
<th>Hex Address</th>
<th>Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Mode Control 1</td>
</tr>
<tr>
<td>01</td>
<td>Palette Mask</td>
</tr>
<tr>
<td>02</td>
<td>Border Color</td>
</tr>
<tr>
<td>03</td>
<td>Mode Control 2</td>
</tr>
<tr>
<td>04</td>
<td>Reset</td>
</tr>
<tr>
<td>10-1F</td>
<td>Palette Registers</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>+HIBW/−LOBW</td>
</tr>
<tr>
<td>1</td>
<td>+Graphics/−Alpha</td>
</tr>
<tr>
<td>2</td>
<td>+B/W</td>
</tr>
<tr>
<td>3</td>
<td>+Video Enable</td>
</tr>
<tr>
<td>4</td>
<td>+16 Color Graphics</td>
</tr>
</tbody>
</table>

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 chrominace 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.

<table>
<thead>
<tr>
<th>Bit 0</th>
<th>Palette Mask 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 1</td>
<td>Palette Mask 1</td>
</tr>
<tr>
<td>Bit 2</td>
<td>Palette Mask 2</td>
</tr>
<tr>
<td>Bit 3</td>
<td>Palette Mask 3</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Bit Number</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>+ B (Blue) Border Color Select</td>
</tr>
<tr>
<td>1</td>
<td>+ G (Green) Border Color Select</td>
</tr>
<tr>
<td>2</td>
<td>+ R (Red) Border Color Select</td>
</tr>
<tr>
<td>3</td>
<td>+ I (Intensity) Border Color Select</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Bit Number</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Reserved = 0</td>
</tr>
<tr>
<td>1</td>
<td>+ Enable Blink</td>
</tr>
<tr>
<td>2</td>
<td>- Reserved = 0</td>
</tr>
<tr>
<td>3</td>
<td>+ 2-Color Graphics</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>PA2</td>
<td>PA1</td>
<td>PA0</td>
<td>PA3</td>
<td>PA2</td>
<td>PA1</td>
<td>PA0</td>
</tr>
</tbody>
</table>

Foreground Color
Background Color
Blinking

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:

```
    7 6 5 4 3 2 1 0
PA3 PA2 PA1 PA0 | PA3 PA2 PA1 PA0
```

- **Foreground Color**
- **Background Color**

**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:

<table>
<thead>
<tr>
<th>Bit 0</th>
<th>Bit 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>+Asynchronous Reset</td>
<td>+Synchronous Reset</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Bit Number</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>+ Blue</td>
</tr>
<tr>
<td>1</td>
<td>+ Green</td>
</tr>
<tr>
<td>2</td>
<td>+ Red</td>
</tr>
<tr>
<td>3</td>
<td>+ Intensity</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>+Display Enable</td>
</tr>
<tr>
<td>1</td>
<td>+Light Pen Trigger Set</td>
</tr>
<tr>
<td>2</td>
<td>–Light Pen Switch Made</td>
</tr>
<tr>
<td>3</td>
<td>+Vertical Retrace</td>
</tr>
<tr>
<td>4</td>
<td>+Video Dots</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Address Register Bit 1</th>
<th>Address Register Bit 0</th>
<th>Video Dot Information Selected</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Blue</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Green</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Red</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Intensity</td>
</tr>
</tbody>
</table>

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.
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:

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

**Note:** All register values are given in hexadecimal.
<table>
<thead>
<tr>
<th>Hex Addr.</th>
<th>Register #</th>
<th>Type</th>
<th>Units</th>
<th>I/O</th>
<th>Alphanumeric 40x25</th>
<th>Alphanumeric 80x25</th>
<th>Low/High Band Width Graphics</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>R6</td>
<td>Vertical Displayed</td>
<td>Char. Row</td>
<td>Write Only</td>
<td>19</td>
<td>19</td>
<td>64/32</td>
</tr>
<tr>
<td>7</td>
<td>R7</td>
<td>Vertical Sync Position</td>
<td>Char. Row</td>
<td>Write Only</td>
<td>1C</td>
<td>1C</td>
<td>70/38</td>
</tr>
<tr>
<td>8</td>
<td>R8</td>
<td>Interlace Mode</td>
<td>—</td>
<td>Write Only</td>
<td>02</td>
<td>02</td>
<td>02/02</td>
</tr>
<tr>
<td>9</td>
<td>R9</td>
<td>Maximum Scan Line Address</td>
<td>Scan Line</td>
<td>Write Only</td>
<td>07</td>
<td>07</td>
<td>01/03</td>
</tr>
<tr>
<td>A</td>
<td>R10</td>
<td>Cursor Start</td>
<td>Scan Line</td>
<td>Write Only</td>
<td>06</td>
<td>06</td>
<td>26/26</td>
</tr>
<tr>
<td>B</td>
<td>R11</td>
<td>Cursor End</td>
<td>Scan Line</td>
<td>Write Only</td>
<td>07</td>
<td>07</td>
<td>07/07</td>
</tr>
</tbody>
</table>

**Note:** All register values are given in hexademical.

6845 Register Table (Part 2 of 3)
<table>
<thead>
<tr>
<th>Hex Addr.</th>
<th>Register</th>
<th>Type</th>
<th>Units</th>
<th>I/O</th>
<th>Alphanumeric</th>
<th>Low/High Band Width Graphics</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>R12</td>
<td>Start Addr. (H)</td>
<td>—</td>
<td>Write Only</td>
<td>00</td>
<td>00/00</td>
</tr>
<tr>
<td>D</td>
<td>R13</td>
<td>Start Addr. (L)</td>
<td>—</td>
<td>Write Only</td>
<td>00</td>
<td>00/00</td>
</tr>
<tr>
<td>E</td>
<td>R14</td>
<td>Cursor Addr. (H)</td>
<td>—</td>
<td>Read/Write</td>
<td>00</td>
<td>00/00</td>
</tr>
<tr>
<td>F</td>
<td>R15</td>
<td>Cursor Addr. (L)</td>
<td>—</td>
<td>Read/Write</td>
<td>00</td>
<td>00/00</td>
</tr>
<tr>
<td>10</td>
<td>R16</td>
<td>Light Pen (H)</td>
<td>—</td>
<td>Read Only</td>
<td>NA</td>
<td>NA/NA</td>
</tr>
<tr>
<td>11</td>
<td>R17</td>
<td>Light Pen (L)</td>
<td>—</td>
<td>Read Only</td>
<td>NA</td>
<td>NA/NA</td>
</tr>
</tbody>
</table>

**Note:** All register values are given in hexadecimal.
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.

<table>
<thead>
<tr>
<th>Bit Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>CRT Page 0</td>
</tr>
<tr>
<td>1</td>
<td>CRT Page 1</td>
</tr>
<tr>
<td>2</td>
<td>CRT Page 2</td>
</tr>
<tr>
<td>3</td>
<td>Processor Page 1</td>
</tr>
<tr>
<td>4</td>
<td>Processor Page 2</td>
</tr>
<tr>
<td>5</td>
<td>Processor Page 3</td>
</tr>
<tr>
<td>6</td>
<td>Video Address Mode 0</td>
</tr>
<tr>
<td>7</td>
<td>Video Address Mode 1</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Video Address Mode</th>
<th>Resulting Modes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (Bit 7)</td>
<td>0 (Bit 6)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

CRT/Processor Page Register (Part 2 of 2)

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

<table>
<thead>
<tr>
<th>Hex Address</th>
<th>A9 A8 A7 A6 A5 A4 A3 A2 A1 A0</th>
<th>Function of Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>3DA</td>
<td>1 1 1 1 0 1 1 0 1 0</td>
<td>Gate Array Address and Status Register</td>
</tr>
<tr>
<td>3DB</td>
<td>1 1 1 1 0 1 1 0 1 1</td>
<td>Clear Light</td>
</tr>
<tr>
<td>3DC</td>
<td>1 1 1 1 0 1 1 1 0 0</td>
<td>Preset Light</td>
</tr>
<tr>
<td>3D0,3D4</td>
<td>1 1 1 1 0 1 0 x x 0</td>
<td>6845 Index Register</td>
</tr>
<tr>
<td>3D1,3D5</td>
<td>1 1 1 1 0 1 0 x x 1</td>
<td>6845 Data Register</td>
</tr>
<tr>
<td>3DF</td>
<td>1 1 1 1 0 1 1 1 1 1</td>
<td>CRT, Processor Page Register</td>
</tr>
</tbody>
</table>

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:

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

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.

![At Standard TTL Levels Diagram](image_url)

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

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.
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 outport bit PB0.

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

Beeper Block Diagram
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.
<table>
<thead>
<tr>
<th>Source</th>
<th>Port Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complex Sound Generator (TI 76496)</td>
<td>PB6 1</td>
</tr>
<tr>
<td></td>
<td>PB5 1</td>
</tr>
<tr>
<td>Programmable Timer (8253)</td>
<td>PB6 0</td>
</tr>
<tr>
<td></td>
<td>PB5 0</td>
</tr>
<tr>
<td>Cassette Audio</td>
<td>PB6 0</td>
</tr>
<tr>
<td></td>
<td>PB5 1</td>
</tr>
<tr>
<td>I/O Channel Audio</td>
<td>PB6 1</td>
</tr>
<tr>
<td></td>
<td>PB5 0</td>
</tr>
</tbody>
</table>

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.
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

<table>
<thead>
<tr>
<th>MSB R0</th>
<th>R1</th>
<th>LSB R2</th>
<th>Destination Control Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Tone 1 Frequency</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Tone 1 Attenuation</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Tone 2 Frequency</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Tone 2 Attenuation</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Tone 3 Frequency</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Tone 3 Attenuation</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Noise Control</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Noise Attenuation</td>
</tr>
</tbody>
</table>

Frequency (Double or Single Byte Transfer)

<table>
<thead>
<tr>
<th>1</th>
<th>Reg. Addr.</th>
<th>Low Data</th>
<th>High Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R0 R1 R2 F6 F7 F8 F9</td>
<td>X F0 F1 F2 F3 F4 F5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bit First Byte Bit</td>
<td>Bit Second Byte Bit</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>MSB 7</td>
<td>0 7</td>
<td></td>
</tr>
</tbody>
</table>

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 $N = \text{ref clock in Hz (3.579 MHz)}$

$$n = 10\text{-bit binary-number}$$

### Attenuator

<table>
<thead>
<tr>
<th>1</th>
<th>Reg. Addr.</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R0 R1 R2</td>
<td>A0 A1 A2 A3</td>
</tr>
<tr>
<td>Bit 0</td>
<td>Second</td>
<td>Bit 7</td>
</tr>
<tr>
<td>MSB</td>
<td>Byte</td>
<td>LSB</td>
</tr>
</tbody>
</table>

**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.
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.
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.

<table>
<thead>
<tr>
<th>Bits</th>
<th>Shift Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>NF0</td>
<td>NF1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

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 μs) 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.
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 $\mu$s 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 $\mu$s. 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**

<table>
<thead>
<tr>
<th>Errors</th>
<th>Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase Errors</td>
<td>The 1st half of the bit-cell sample is not equal to the inverse of the 2nd half of the bit-cell sample.</td>
</tr>
<tr>
<td>Parity Errors</td>
<td>The received encoded word did not maintain odd parity.</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Pin</th>
<th>Signal</th>
<th>Input/Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>A01</td>
<td>+12 Volts</td>
<td>Input</td>
</tr>
<tr>
<td>A02</td>
<td>Ground</td>
<td>Input</td>
</tr>
<tr>
<td>A03</td>
<td>Ground-Shield</td>
<td>Input</td>
</tr>
<tr>
<td>A04</td>
<td>I.R. TEST FREQ.</td>
<td>Input</td>
</tr>
<tr>
<td>B01</td>
<td>GROUND</td>
<td>Input</td>
</tr>
<tr>
<td>B02</td>
<td>+5 Volts</td>
<td>Input</td>
</tr>
<tr>
<td>B03</td>
<td>-I.R. KBD DATA</td>
<td>Output</td>
</tr>
<tr>
<td>B04</td>
<td>GROUND</td>
<td>Input</td>
</tr>
</tbody>
</table>

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.

![Keyboard Layout Diagram]

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 biphase serial-encoding and each is a simplex link.

The 80C48 microprocessor does the biphase serial encoding with a bit cell of 440 μs. A biphase 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 biphase 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.
<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>First Bit</td>
<td>Start Bit</td>
</tr>
<tr>
<td>Second Bit</td>
<td>Data Bit 0 (Least Significant Bit)</td>
</tr>
<tr>
<td>Third Bit</td>
<td>Data Bit 1</td>
</tr>
<tr>
<td>Fourth Bit</td>
<td>Data Bit 2</td>
</tr>
<tr>
<td>Fifth Bit</td>
<td>Data Bit 3</td>
</tr>
<tr>
<td>Sixth Bit</td>
<td>Data Bit 4</td>
</tr>
<tr>
<td>Seventh Bit</td>
<td>Data Bit 5</td>
</tr>
<tr>
<td>Eighth Bit</td>
<td>Data Bit 6</td>
</tr>
<tr>
<td>Ninth Bit</td>
<td>Data Bit 7 (Most Significant Bit)</td>
</tr>
<tr>
<td>Tenth Bit</td>
<td>Parity Bit</td>
</tr>
<tr>
<td>Eleventh Bit</td>
<td>Stop Bit</td>
</tr>
</tbody>
</table>

**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.
Eleven Stop Bit Cells

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

Cable Data

Infra-Red Data

Keyboard Transmission Timing

Cable

Infra-Red

|| BI-Phase '1'
---|---
Bit Cell
220 μs
440 μs

|| BI-Phase '0'
---|---
Bit Cell
220 μs
440 μs

40 kHz @ 50% Duty Cycle

62.5 μs
440 μs

Cordless Keyboard 2-105
Keyboard Interface Logic

- CLB Data
- I.R. Data
- CBL Con
- NMI Mask

AND

AND

NOR

Inverter

+5

D Flip Flop

CLR

NMI

- IOR From Port A0 Hex

Keyboard Data

8255 PC6

8255 PC0

2-106 Cordless Keyboard
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:

<table>
<thead>
<tr>
<th>Location</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>055H</td>
</tr>
<tr>
<td>1</td>
<td>0AAH</td>
</tr>
<tr>
<td>2</td>
<td>Length</td>
</tr>
<tr>
<td>3,4,5</td>
<td>Jump to Initialize Code</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>Last 2 Addresses</td>
<td>CRC Bytes</td>
</tr>
</tbody>
</table>

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 \((\text{length}/512)\). 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:
<table>
<thead>
<tr>
<th>Location</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>055H</td>
</tr>
<tr>
<td>1</td>
<td>0AAH</td>
</tr>
<tr>
<td>2</td>
<td>Length</td>
</tr>
<tr>
<td>3-5</td>
<td>Jump to Initialize</td>
</tr>
<tr>
<td>6</td>
<td>Command Name Length (Offset Y- Offset Z)</td>
</tr>
<tr>
<td>Z</td>
<td>First Character in Command Name</td>
</tr>
<tr>
<td>Y</td>
<td>Last Character in Command Name</td>
</tr>
<tr>
<td>W</td>
<td>Word Pointing to Routine that is Jumped to if “Name” is Typed</td>
</tr>
<tr>
<td>X</td>
<td>Next Command Name Length or “00” if No More Command Names</td>
</tr>
<tr>
<td>Last 2 Addresses</td>
<td>CRC Bytes</td>
</tr>
</tbody>
</table>

**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 \((\text{length}/512)\). 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:
<table>
<thead>
<tr>
<th>Location</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>055H</td>
</tr>
<tr>
<td>1</td>
<td>0AAH</td>
</tr>
<tr>
<td>2</td>
<td>Length</td>
</tr>
<tr>
<td>3</td>
<td>0CBH</td>
</tr>
<tr>
<td>4</td>
<td>0AAH</td>
</tr>
<tr>
<td>5</td>
<td>055H</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0FFH if unprotected Basic program or 0FEH if protected Basic program</td>
</tr>
<tr>
<td>8</td>
<td>Start of interpretable Basic code</td>
</tr>
<tr>
<td>n</td>
<td>0FFH Padding to next 2048 byte boundary</td>
</tr>
<tr>
<td>Last 2 Addresses</td>
<td>CRC Bytes</td>
</tr>
</tbody>
</table>

**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 (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 0FF.

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.

<table>
<thead>
<tr>
<th>ROM Chip Select</th>
<th>Hex Address Space</th>
<th>Typical Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS0</td>
<td>X</td>
<td>Not Used</td>
</tr>
<tr>
<td>CS1</td>
<td>X</td>
<td>Not Used</td>
</tr>
<tr>
<td>CS2</td>
<td>D0000-D7FFF</td>
<td>Optional Cartridge ROM #2</td>
</tr>
<tr>
<td>CS3</td>
<td>D8000-DFFFFF</td>
<td>Optional Cartridge ROM #1</td>
</tr>
<tr>
<td>CS4</td>
<td>E0000-E7FFF</td>
<td>Standard Cartridge ROM #2</td>
</tr>
<tr>
<td>CS5</td>
<td>E8000-EFFFFF</td>
<td>Standard Cartridge ROM #1</td>
</tr>
<tr>
<td>CS6</td>
<td>F0000-F7FFF</td>
<td>System Board ROM #2</td>
</tr>
<tr>
<td>CS7</td>
<td>F8000-FFFFF</td>
<td>System Board ROM #1</td>
</tr>
</tbody>
</table>

ROM Chip Select Table

<table>
<thead>
<tr>
<th>Signal</th>
<th>I/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A0 - A14</td>
<td>0</td>
<td>Processor Address lines A0 - A14</td>
</tr>
<tr>
<td>D0 - D7</td>
<td>I</td>
<td>Processor Data lines</td>
</tr>
</tbody>
</table>
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 I
- ROM IN
- CARTRIDGE

This line when pulled 'low' instructs the system board to de-gate the ROM module from hex F8000 - FFFFFF on the system board. This ROM module can then be replaced by a ROM module on the cartridge by using -CS7.

- BASE 2 ROM I
- IN
- CARTRIDGE

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 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.
Top of Cartridge

Momentary Reset Land

<table>
<thead>
<tr>
<th>GND</th>
<th>B01</th>
<th>A01</th>
<th>GND</th>
</tr>
</thead>
<tbody>
<tr>
<td>-CS7</td>
<td></td>
<td></td>
<td>CARTRIDGE RESET TAB</td>
</tr>
<tr>
<td>-CS3</td>
<td></td>
<td></td>
<td>-CS5</td>
</tr>
<tr>
<td>A14</td>
<td></td>
<td></td>
<td>-BASE 1 ROM IN CARTRIDGE</td>
</tr>
<tr>
<td>A12</td>
<td>B05</td>
<td>A05</td>
<td>A13</td>
</tr>
<tr>
<td>A7</td>
<td></td>
<td></td>
<td>A8</td>
</tr>
<tr>
<td>A6</td>
<td></td>
<td></td>
<td>A9</td>
</tr>
<tr>
<td>A5</td>
<td></td>
<td></td>
<td>A11</td>
</tr>
<tr>
<td>A4</td>
<td></td>
<td></td>
<td>-BASE 2 ROM IN CARTRIDGE</td>
</tr>
<tr>
<td>A3</td>
<td>B10</td>
<td>A10</td>
<td>A10</td>
</tr>
<tr>
<td>A2</td>
<td></td>
<td></td>
<td>D7</td>
</tr>
<tr>
<td>A1</td>
<td></td>
<td></td>
<td>D6</td>
</tr>
<tr>
<td>A0</td>
<td></td>
<td></td>
<td>D5</td>
</tr>
<tr>
<td>D0</td>
<td></td>
<td></td>
<td>D4</td>
</tr>
<tr>
<td>D1</td>
<td>B15</td>
<td>A15</td>
<td>D3</td>
</tr>
<tr>
<td>D2</td>
<td></td>
<td></td>
<td>-CS2</td>
</tr>
<tr>
<td>-CS6</td>
<td></td>
<td></td>
<td>-CS4</td>
</tr>
<tr>
<td>+5V</td>
<td>B18</td>
<td>A18</td>
<td>+5V</td>
</tr>
</tbody>
</table>

Connector Specification
Cartridge ROM Locations
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:

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

Where \( r \) is the resistance in ohms.
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.
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:
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.

<table>
<thead>
<tr>
<th>Buttons</th>
<th>Coordinates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paddle D</td>
<td>Paddle C</td>
</tr>
<tr>
<td>Bit 7</td>
<td>Bit 6</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Pin Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keyplug</td>
<td>A01</td>
</tr>
<tr>
<td>LOGIC GND</td>
<td>A02</td>
</tr>
<tr>
<td>Y-AXIS RESISTANCE</td>
<td>A03</td>
</tr>
<tr>
<td>+5V</td>
<td>A04</td>
</tr>
<tr>
<td>SHIELD GND</td>
<td>B01</td>
</tr>
<tr>
<td>X-AXIS RESISTANCE</td>
<td>B02</td>
</tr>
<tr>
<td>SWITCH</td>
<td>B03</td>
</tr>
<tr>
<td>SWITCH</td>
<td>B04</td>
</tr>
</tbody>
</table>

Connector Specification
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.

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

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

<table>
<thead>
<tr>
<th>Interchange Voltage</th>
<th>Binary State</th>
<th>Signal Condition</th>
<th>Interface Control Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive Voltage =</td>
<td>Binary (0)</td>
<td>= Spacing</td>
<td>= On</td>
</tr>
<tr>
<td>Negative Voltage =</td>
<td>Binary (1)</td>
<td>= Marking</td>
<td>= Off</td>
</tr>
</tbody>
</table>

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 $16 \times$ the baud rate \[\text{divisor number} = \frac{\text{frequency input}}{\text{baud rate} \times 16}\]. 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.
<table>
<thead>
<tr>
<th>Desired Baud Rate</th>
<th>Divisor Used to Generate 16x Clock (Decimal)</th>
<th>Divisor Used to Generate 16x Clock (Hex)</th>
<th>Percent Error Per Bit Difference Between Desired and Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>2237</td>
<td>8BD</td>
<td>.006</td>
</tr>
<tr>
<td>75</td>
<td>1491</td>
<td>5D3</td>
<td>.017</td>
</tr>
<tr>
<td>110</td>
<td>1017</td>
<td>1A1</td>
<td>.023</td>
</tr>
<tr>
<td>134.5</td>
<td>832</td>
<td>167</td>
<td>.054</td>
</tr>
<tr>
<td>150</td>
<td>746</td>
<td>12C</td>
<td>.050</td>
</tr>
<tr>
<td>300</td>
<td>373</td>
<td>175</td>
<td>.050</td>
</tr>
<tr>
<td>600</td>
<td>186</td>
<td>BA</td>
<td>.218</td>
</tr>
<tr>
<td>1200</td>
<td>93</td>
<td>5D</td>
<td>.218</td>
</tr>
<tr>
<td>1800</td>
<td>62</td>
<td>3E</td>
<td>.218</td>
</tr>
<tr>
<td>2000</td>
<td>56</td>
<td>38</td>
<td>.140</td>
</tr>
<tr>
<td>2400</td>
<td>47</td>
<td>2F</td>
<td>.855</td>
</tr>
<tr>
<td>3600</td>
<td>31</td>
<td>1F</td>
<td>.218</td>
</tr>
<tr>
<td>4800</td>
<td>23</td>
<td>17</td>
<td>1.291</td>
</tr>
</tbody>
</table>

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

Assembly Language Sample Program

![](image)

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

<table>
<thead>
<tr>
<th>Voltage (Vac)</th>
<th>Frequency</th>
<th>Current (Amps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nominal</td>
<td>Minimum</td>
<td>Maximum</td>
</tr>
<tr>
<td>120</td>
<td>104</td>
<td>127</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Voltage ac

D.C Outputs

<table>
<thead>
<tr>
<th>Vdc Voltage</th>
<th>Current (Amps)</th>
<th>Regulation Tolerance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nominal</td>
<td>Minimum</td>
<td>Maximum</td>
</tr>
<tr>
<td>+5</td>
<td>*1.5</td>
<td>3.6</td>
</tr>
<tr>
<td>+12</td>
<td>.04</td>
<td>1.2</td>
</tr>
<tr>
<td>-6</td>
<td>0.0</td>
<td>.025</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Voltage (Nominal)</th>
<th>Type Protection</th>
<th>Rating (Amps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>120 Vac</td>
<td>Non-resettable Fuse Thermal/Over-Current</td>
<td>5A Slo Blow</td>
</tr>
</tbody>
</table>

Output (Power Board)

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

<table>
<thead>
<tr>
<th>Protection Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Over-Voltage</td>
</tr>
<tr>
<td>+5 Vdc</td>
</tr>
<tr>
<td>12 Vdc</td>
</tr>
</tbody>
</table>

* 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.
Connector Specifications

1. Power Board (Component Side)

Input From Transformer
Grounding Pin
Connector

Supply to Diskette Drive
Fan Plug

Connector Specifications

1. Power Board
2. System Board
3. Transformer

Connector Specifications

1. +12 Vdc
2. GND
3. GND
4. +5 Vdc
5. +5 Vdc
6. +5 Vdc
7. +5 Vdc
8. GND
9. GND
10. +12 Vdc

Connector Specifications

1. 17 Vac
2. GND
3. 17 Vac

Connector Specifications

1. +12 Vdc
2. GND
3. GND
4. +5 Vdc

Connector Specifications

1. Power Board
2. Power Board
3. Power Board
4. Power Board
Fan Connector Specifications
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
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 PC<sub>jr</sub> 64KB Memory and Display Expansion.

![Memory Expansion Block Diagram](image)

Memory Expansion Block Diagram
<table>
<thead>
<tr>
<th>Signal</th>
<th>I/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+RAS</td>
<td>I</td>
<td>+Row Address Strobe. This line is inverted and then becomes the -RAS for the RAM modules.</td>
</tr>
<tr>
<td>+A0</td>
<td>I</td>
<td>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).</td>
</tr>
<tr>
<td>-DISABLE EDATA</td>
<td>O</td>
<td>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.</td>
</tr>
<tr>
<td>ATR LATCH</td>
<td>I</td>
<td>This signal indicates that the expansion RAM card should 'latch' up data from the expansion RAM into the attribute latch.</td>
</tr>
<tr>
<td>MD0 thru MD7</td>
<td>O</td>
<td>These data lines contain CRT information from the attribute latch and go to the Video Gate Array.</td>
</tr>
<tr>
<td>D0 thru D7</td>
<td>I/O</td>
<td>These data lines are from the microprocessor and are bidirectional.</td>
</tr>
<tr>
<td>MEM A0 thru A7</td>
<td>I</td>
<td>These are the multiplexed address lines for the dynamic-RAM modules. These lines are multiplexed between row address and column.</td>
</tr>
</tbody>
</table>
address, and also between microprocessor and CRT addresses.

**VIDEO MEMR** I

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.
<table>
<thead>
<tr>
<th>Signal</th>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-LCG</td>
<td>O</td>
<td>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.</td>
</tr>
<tr>
<td>GATE</td>
<td>I</td>
<td>This line is 'wrapped' and becomes the -LCG output.</td>
</tr>
<tr>
<td>-WE</td>
<td>I</td>
<td>This line instructs the memory that the cycle is a microprocessor 'write' cycle.</td>
</tr>
<tr>
<td>CPU LATCH</td>
<td>I</td>
<td>This line instructs the expansion RAM card to 'latch' the data from the expansion RAM into the microprocessor latch.</td>
</tr>
<tr>
<td>-ATR CD IN</td>
<td>O</td>
<td>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.</td>
</tr>
</tbody>
</table>
The following is the connector specifications for the IBM PCjr 64KB Memory and Display Expansion.

64KB Memory and Display Expansion
<table>
<thead>
<tr>
<th>Connector Pin</th>
<th>Signal Name</th>
<th>Signal Name</th>
<th>Connector Pin</th>
</tr>
</thead>
<tbody>
<tr>
<td>A01</td>
<td>+RAS</td>
<td>VIDEO MEMR</td>
<td>B01</td>
</tr>
<tr>
<td>A02</td>
<td>A0</td>
<td>CPU DLY</td>
<td>B02</td>
</tr>
<tr>
<td>A03</td>
<td>–DISABLE</td>
<td>–DISABLE</td>
<td>B03</td>
</tr>
<tr>
<td></td>
<td>EDATA</td>
<td>CAS 0</td>
<td></td>
</tr>
<tr>
<td>A04</td>
<td>ATR LATCH</td>
<td>+CAS</td>
<td>B04</td>
</tr>
<tr>
<td>A05</td>
<td>MD4</td>
<td>–LCG</td>
<td>B05</td>
</tr>
<tr>
<td>A06</td>
<td>MD5</td>
<td>GATE</td>
<td>B06</td>
</tr>
<tr>
<td>A07</td>
<td>MD6</td>
<td>Ground</td>
<td>B07</td>
</tr>
<tr>
<td>A08</td>
<td>MD7</td>
<td>Ground</td>
<td>B08</td>
</tr>
<tr>
<td>A09</td>
<td>MD0</td>
<td>Ground</td>
<td>B09</td>
</tr>
<tr>
<td>A10</td>
<td>MD1</td>
<td>–WE</td>
<td>B10</td>
</tr>
<tr>
<td>A11</td>
<td>MD2</td>
<td>CPU LATCH</td>
<td>B11</td>
</tr>
<tr>
<td>A12</td>
<td>MD3</td>
<td>–ATR CD IN</td>
<td>B12</td>
</tr>
<tr>
<td>A13</td>
<td>GND</td>
<td>GND</td>
<td>B13</td>
</tr>
<tr>
<td>A14</td>
<td>VCC</td>
<td>VCC</td>
<td>B14</td>
</tr>
<tr>
<td>A15</td>
<td>D7</td>
<td>D6</td>
<td>B15</td>
</tr>
<tr>
<td>A16</td>
<td>D5</td>
<td>D4</td>
<td>B16</td>
</tr>
<tr>
<td>A17</td>
<td>D3</td>
<td>D2</td>
<td>B17</td>
</tr>
<tr>
<td>A18</td>
<td>D1</td>
<td>D0</td>
<td>B18</td>
</tr>
<tr>
<td>A19</td>
<td>MEM A6</td>
<td>MEM A7</td>
<td>B19</td>
</tr>
<tr>
<td>A20</td>
<td>MEM A4</td>
<td>MEM A5</td>
<td>B20</td>
</tr>
<tr>
<td>A21</td>
<td>MEM A2</td>
<td>MEM A3</td>
<td>B21</td>
</tr>
<tr>
<td>A22</td>
<td>MEM A0</td>
<td>MEM A1</td>
<td>B22</td>
</tr>
</tbody>
</table>

Connector Specifications
Notes:

3-12 64KB Memory Expansion
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 μPD765 or compatible controller, so the μPD765 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.
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 μPD765 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:

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Drive Enable</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Watch Dog Timer Enable</td>
<td>Watch Dog Timer Trigger</td>
<td>FDC Reset</td>
</tr>
</tbody>
</table>

Note: All bits are cleared with channel reset.

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.

<table>
<thead>
<tr>
<th>FDC Register</th>
<th>I/O Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Register</td>
<td>hex F5</td>
</tr>
<tr>
<td>Main Status Register</td>
<td>hex F4</td>
</tr>
</tbody>
</table>

**Programming Summary**

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

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

**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

<table>
<thead>
<tr>
<th>2 Heads (1 per side)</th>
</tr>
</thead>
<tbody>
<tr>
<td>40 Cylinders (Tracks)/Side</td>
</tr>
<tr>
<td>9 Sectors/Track</td>
</tr>
<tr>
<td>512 Bytes/Sector</td>
</tr>
<tr>
<td>Modified Frequency Modulation (MFM)</td>
</tr>
</tbody>
</table>

**Diskette Format**
### Constant Value

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Head Load</td>
<td>Not Applicable</td>
</tr>
<tr>
<td>Head Settle</td>
<td>21 Milliseconds</td>
</tr>
<tr>
<td>Motor Start</td>
<td>500 Milliseconds</td>
</tr>
</tbody>
</table>

**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)
These eight lines form a bus through which all commands, status, and data are transferred. Bit 0 is the low-order bit.

+A0 thru 3 (Adapter Input, Load: 1 74LS)

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.

-LOW (Adapter Input, Load: 1 74LS)

The content of lines D0-7 is stored in the register addressed by lines A0-3 at the trailing edge of this signal.

-IOR (Adapter Input, Load: 1 74LS)

The content of the register addressed by lines A0-3 is 'gated' onto lines D0-7 when this line is 'active.'

-RESET (Adapter Input, Load: 1 74LS)

A down level 'aborts' any operation in process and 'clears' the digital output register (DOR).

+IRQ6 (Adapter Output, Driver: 74LS 3-state)

This line is made 'active' when the WatchDog timer times out.

-DISKETTE CARD INSTALLED (Adapter Output, Driver: Gnd.)

This line is pulled 'up' on the System Board and is wired to input port bit PC2 on port hex 62 of the
Diskette Drive Adapter 3-21

-**Diskette CS**

8255. This line is used by the program to determine if the diskette drive adapter is installed.

(Adapter Input, Load: 174LS)

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.

**A9**

(Adapter Input, Load: 174LS)

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.

**DRQ 0**

(Adapter Output, Driver: NEC µpd 765)

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.

**DACK 0**

(Adapter input, Load: NEC µpd 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:

<table>
<thead>
<tr>
<th>Signal Type</th>
<th>Voltage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Most Positive Up Level</td>
<td>+ 5.5 Vdc</td>
</tr>
<tr>
<td>Least Positive Up Level</td>
<td>+ 2.4 Vdc</td>
</tr>
<tr>
<td>Most Positive Down Level</td>
<td>+ 0.4 Vdc</td>
</tr>
<tr>
<td>Least Positive Down Level</td>
<td>- 0.5 Vdc</td>
</tr>
</tbody>
</table>

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 μs 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.
<table>
<thead>
<tr>
<th>Pin Number</th>
<th>At Standard TTL Levels</th>
<th>Diskette Drive Adapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ground - Odd Numbers — 1 Through 33</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>(See Note)</td>
<td></td>
</tr>
<tr>
<td>2, 4, 6</td>
<td>Unused</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>-INDEX</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Unused</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>-DRIVE SELECT</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Unused</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>-MOTOR ENABLE</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>-DIRECTION (Stepper Motor)</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>-STEP PULSE</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>-WRITE DATA</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>-WRITE ENABLE</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>-TRACK 0</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>-WRITE PROTECT</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>-READ DATA</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>-SELECT HEAD 1</td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>Unused</td>
<td></td>
</tr>
</tbody>
</table>

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

**Connector Specifications (Part 1 of 2)**
<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Diskette Drive Adapter</th>
<th>Signal Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7</td>
<td></td>
<td>+5V</td>
</tr>
<tr>
<td>D6</td>
<td></td>
<td>-DISKETTE CS</td>
</tr>
<tr>
<td>D5</td>
<td></td>
<td>A9</td>
</tr>
<tr>
<td>GND</td>
<td></td>
<td>A3</td>
</tr>
<tr>
<td>D4</td>
<td></td>
<td>A2</td>
</tr>
<tr>
<td>D3</td>
<td></td>
<td>A1</td>
</tr>
<tr>
<td>D2</td>
<td></td>
<td>A0</td>
</tr>
<tr>
<td>GND</td>
<td></td>
<td>D1</td>
</tr>
<tr>
<td>D0</td>
<td></td>
<td>D0</td>
</tr>
<tr>
<td>IRQ6</td>
<td></td>
<td>-IOR</td>
</tr>
<tr>
<td>+DRQ0</td>
<td></td>
<td>-IOW</td>
</tr>
<tr>
<td>-DACK0</td>
<td></td>
<td>GND</td>
</tr>
<tr>
<td>-5V</td>
<td></td>
<td>-CARD INSTL</td>
</tr>
<tr>
<td>N/C</td>
<td></td>
<td>-RESET</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+5V</td>
</tr>
</tbody>
</table>

System Board

**Note:** All levels are TTL compatible.

**Connector Specifications (Part 2 of 2)**
IBM PC<sub>jr</sub> 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:

<table>
<thead>
<tr>
<th>Specification</th>
<th>+5 Vdc Input</th>
<th>+12 Vdc Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nominal Supply</td>
<td>+5 Vdc</td>
<td>+12 Vdc</td>
</tr>
<tr>
<td>Ripple (0 to 50 kHz)</td>
<td>100 mV</td>
<td>100 mV</td>
</tr>
<tr>
<td>Tolerance (Including Ripple)</td>
<td>±5%</td>
<td>±5%</td>
</tr>
<tr>
<td>Standby Current (Nominal)</td>
<td>600 mA</td>
<td>400 mA</td>
</tr>
<tr>
<td>Standby Current (Worst Case)</td>
<td>700 mA</td>
<td>500 mA</td>
</tr>
<tr>
<td>Operating Current (Nominal)</td>
<td>600 mA</td>
<td>900 mA</td>
</tr>
<tr>
<td>Operating Current (Worst Case)</td>
<td>700 mA</td>
<td>2400 mA</td>
</tr>
</tbody>
</table>

**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)

Diskette Drive
1 GND
2 +12 Vdc
3 GND
4 +5 Vdc

Power Board
1
2 GND
3 +12 Vdc
4

Connector Specifications (Part 2 of 2)

Diskette Drive
1 GND
2 +12 Vdc
3

Power Board
1
2 GND
3

System Options
Notes:

3-30 Diskette Drive
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.
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 ± 50 ms  
| DTMF Dialing:        | Tone Duration: 85 ms ± 10 ms  
|                      | Intertone Duration: 80 ms ± 10 ms

Dialer Parameters (Part 1 of 2)
Tone Pair Frequencies:

<table>
<thead>
<tr>
<th>ASCII Digit Code</th>
<th>Frequency (Hz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>941 1336</td>
</tr>
<tr>
<td>1</td>
<td>697 1209</td>
</tr>
<tr>
<td>2</td>
<td>697 1336</td>
</tr>
<tr>
<td>3</td>
<td>697 1477</td>
</tr>
<tr>
<td>4</td>
<td>770 1209</td>
</tr>
<tr>
<td>5</td>
<td>770 1336</td>
</tr>
<tr>
<td>6</td>
<td>770 1477</td>
</tr>
<tr>
<td>7</td>
<td>852 1209</td>
</tr>
<tr>
<td>8</td>
<td>852 1336</td>
</tr>
<tr>
<td>9</td>
<td>852 1477</td>
</tr>
<tr>
<td>*</td>
<td>941 1209</td>
</tr>
<tr>
<td>#</td>
<td>941 1477</td>
</tr>
</tbody>
</table>

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**
<table>
<thead>
<tr>
<th>Mode</th>
<th>Originating End</th>
<th>Answering End</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transmit</td>
<td>1070 Space 1270 Mark</td>
<td>2025 Space 2225 Mark</td>
</tr>
<tr>
<td>Receive</td>
<td>2025 Hz Space 2225 Hz Mark</td>
<td>1070 Hz Space 1270 Hz Mark</td>
</tr>
</tbody>
</table>

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:
      \[ [\text{cc}] [\text{cc}] \]
      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 x 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:

<table>
<thead>
<tr>
<th>n</th>
<th>Parity</th>
<th>Data Length</th>
<th>Stop Bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Mark</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>Space</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Odd</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>Even</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>None</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>5-7</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>( o )</th>
<th>Mode</th>
<th>Responses</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Verbose</td>
<td>&quot;BUSY&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;CONNECTED&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;NO ANSWER&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;NO DIAL TONE&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;OK&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;RING&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;UNSUCCESSFUL&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;?&quot; (Question Mark)</td>
</tr>
<tr>
<td>1</td>
<td>Terse (Hex code)</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td></td>
<td>31</td>
</tr>
<tr>
<td></td>
<td></td>
<td>32</td>
</tr>
<tr>
<td></td>
<td></td>
<td>33</td>
</tr>
<tr>
<td></td>
<td></td>
<td>34</td>
</tr>
<tr>
<td></td>
<td></td>
<td>35</td>
</tr>
<tr>
<td></td>
<td></td>
<td>36</td>
</tr>
<tr>
<td></td>
<td></td>
<td>37</td>
</tr>
</tbody>
</table>

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)
MODEM

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.

NEW p

Purpose: Changes the command character to an ASCII character.

Default: Ctrl N (ASCII hex 0E)

ORIGINATE

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:

<table>
<thead>
<tr>
<th>Responses</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>H0 or H1</td>
<td>Hook status: H0 = on-hook, H1 = off-hook.</td>
</tr>
<tr>
<td>S0 to SF</td>
<td>Current ringcount setting in hex.</td>
</tr>
<tr>
<td>B</td>
<td>Line busy.</td>
</tr>
<tr>
<td>D</td>
<td>Line dead: no dial-tone found or no ring/no busy timeout after dialing.</td>
</tr>
<tr>
<td>L</td>
<td>Successful dial and handshake.</td>
</tr>
<tr>
<td>N</td>
<td>Dial not recorded: dial tone present after dialing.</td>
</tr>
<tr>
<td>X</td>
<td>No answer: ringcount plus 3 exceeded.</td>
</tr>
</tbody>
</table>

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: \texttt{SPEED o}

\texttt{S o}

Where \texttt{o} is one of the following:

\begin{tabular}{ll}
\texttt{0} & 110 \\
\texttt{1} & 300 \\
\texttt{2} & Reserved \\
\end{tabular}

\textbf{Note}: Do not combine this command with other commands except the \texttt{FORMAT} command on a single command line.

The \texttt{SPEED} command must be issued before the 8250A baud rate is changed.

\textbf{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: \texttt{TRANSPARENT \textit{n...n}}

\texttt{T \textit{n...n}}

Where \textit{n...n} is the number of bytes to transmit in the range of hex 0 to hex FFFF.

\textbf{Purpose:} Places the modem in the transparent mode for the next \textit{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 \texttt{TRANSPARENT} command, the command is ignored because it has no default.

The transparent mode is terminated when:

1. \textit{n...n} characters have been transmitted.

2. Loss of carrier timeout.

3. \texttt{INS8250A OUT 1} pin goes 'active.' (The \texttt{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:** \( \text{ZTEST } o \)

\[ Z o \]

Where \( o \) is one of the following:

- **0**  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.................
RING ..................................CONNECTED 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

[c]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

[c]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.”
### 8250A Register Description

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

*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.
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:

<table>
<thead>
<tr>
<th>D0</th>
<th>D1</th>
<th>D2</th>
<th>D3</th>
<th>D4</th>
<th>D5</th>
<th>D6</th>
<th>D7</th>
</tr>
</thead>
</table>

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:

<table>
<thead>
<tr>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
</table>

3-70 Internal Modem
The 'inactive' state enables entry into the transparent mode using the UNLISTEN command. The 'active' state 'disables' the transparent mode.

No connection.

Serial output from the 8250A.

No connection.

-Data Terminal Ready

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.

Serial input to the 8250A.

The ring indicator pulses with an incoming ring voltage.
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.
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 PCjr 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.
Internal Modem Connectors

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Pin Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNUSED</td>
<td>1</td>
</tr>
<tr>
<td>UNUSED</td>
<td>2</td>
</tr>
<tr>
<td>Telephone RING</td>
<td>3</td>
</tr>
<tr>
<td>TIP</td>
<td>4</td>
</tr>
<tr>
<td>UNUSED</td>
<td>5</td>
</tr>
<tr>
<td>UNUSED</td>
<td>6</td>
</tr>
</tbody>
</table>

Connector Specifications (Part 1 of 2)
<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Internal Modem</th>
<th>Signal Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>+5 Vdc</td>
<td>B01 A01</td>
<td>D7</td>
</tr>
<tr>
<td>-MODEM CS/DISKETTE CS</td>
<td></td>
<td>D6</td>
</tr>
<tr>
<td>A9</td>
<td></td>
<td>D5</td>
</tr>
<tr>
<td>-RESET</td>
<td></td>
<td>GND</td>
</tr>
<tr>
<td>GND</td>
<td>B05 A05</td>
<td>D4</td>
</tr>
<tr>
<td>A2</td>
<td></td>
<td>D3</td>
</tr>
<tr>
<td>A1</td>
<td></td>
<td>D2</td>
</tr>
<tr>
<td>A0</td>
<td></td>
<td>GND</td>
</tr>
<tr>
<td>GND</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-IOR</td>
<td>B10 A10</td>
<td>MODEM INTERRUPT</td>
</tr>
<tr>
<td>-IOW</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GND</td>
<td></td>
<td>MODEM INTERRUPT</td>
</tr>
<tr>
<td>-CARD INSTALL</td>
<td></td>
<td>BAUDCLK</td>
</tr>
<tr>
<td>+XRESET</td>
<td></td>
<td>+5 Vdc</td>
</tr>
<tr>
<td>+5 Vdc</td>
<td>B15 A15</td>
<td>+12 Vdc</td>
</tr>
</tbody>
</table>

All levels are LSTLL compatible.

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 Connector

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Pin Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keyplug</td>
<td>A01</td>
</tr>
<tr>
<td>LOGIC GND</td>
<td>A02</td>
</tr>
<tr>
<td>Y-AXIS RESISTANCE</td>
<td>A03</td>
</tr>
<tr>
<td>+5 V</td>
<td>A04</td>
</tr>
<tr>
<td>SHIELD GND</td>
<td>B01</td>
</tr>
<tr>
<td>X-AXIS RESISTANCE</td>
<td>B02</td>
</tr>
<tr>
<td>SWITCH</td>
<td>B03</td>
</tr>
<tr>
<td>SWITCH</td>
<td>B04</td>
</tr>
</tbody>
</table>

**Connector Specifications**
Notes:

3–80 Attachable Joystick
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 PC jr with the IBM PC jr Adapter Cable for the IBM Color Display which provides a direct-drive connection from the IBM PC jr.

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 Display Connector

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Pin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ground</td>
<td>1</td>
</tr>
<tr>
<td>Ground</td>
<td>2</td>
</tr>
<tr>
<td>Red</td>
<td>3</td>
</tr>
<tr>
<td>Green</td>
<td>4</td>
</tr>
<tr>
<td>Blue</td>
<td>5</td>
</tr>
<tr>
<td>Intensity</td>
<td>6</td>
</tr>
<tr>
<td>Not Used</td>
<td>7</td>
</tr>
<tr>
<td>Horizontal Drive</td>
<td>8</td>
</tr>
<tr>
<td>Vertical Drive</td>
<td>9</td>
</tr>
</tbody>
</table>

Connector Specifications
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

System Unit Connector

+12 Volts
No Pin
Video

Logic GND
Audio
Shield GND

Connector Specifications

3-86 Connector for Television
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

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Pin Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>A01 Keyplug -CBL KEYBD DATA</td>
<td>A01</td>
</tr>
<tr>
<td>A02</td>
<td>A02</td>
</tr>
<tr>
<td>A03</td>
<td>A03</td>
</tr>
<tr>
<td>B01</td>
<td>B01</td>
</tr>
<tr>
<td>B02</td>
<td>B02</td>
</tr>
<tr>
<td>B03</td>
<td>B03</td>
</tr>
</tbody>
</table>

Connector Specifications

2 -CBL KEYBD DATA A01
5 +5 V A02
3 -CABLE CONNECT B01
6 LOGIC GND B02
1 A3 B3
2 A1 B1

3-88 Keyboard Cord
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.
## Connector Specifications

<table>
<thead>
<tr>
<th>System Connector</th>
<th>Cable</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>Not Used</td>
</tr>
<tr>
<td>A2</td>
<td>DATA TERMINAL READY</td>
</tr>
<tr>
<td>A3</td>
<td>REQUEST TO SEND</td>
</tr>
<tr>
<td>A4</td>
<td>TRANSMIT DATA</td>
</tr>
<tr>
<td>A5</td>
<td>CARRIER DETECT</td>
</tr>
<tr>
<td>A6</td>
<td>DATA SET READY</td>
</tr>
<tr>
<td>A7</td>
<td>CLEAR TO SEND</td>
</tr>
<tr>
<td>A8</td>
<td>RECEIVE DATA</td>
</tr>
<tr>
<td>B1</td>
<td>SHIELD GND</td>
</tr>
<tr>
<td>B2</td>
<td>SIGNAL GND</td>
</tr>
<tr>
<td>B3 - B8</td>
<td>Not Used</td>
</tr>
</tbody>
</table>

- **25-Pin D-Shell Connector**

3-90 Serial Devices Cable
IBM PC<sub>jr</sub> Adapter Cable for Cassette

This option is an adapter cable that allows connection of a cassette recorder to the IBM PC<sub>jr</sub> cassette connector.

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

- Belden Style-51 miniture phone-plug (Auxiliary)
- Belden Style-51 miniture phone-plug (Earphone)
- Belden Style-56 subminiture phone-plug (Remote)

The following figures show the connector specifications for the Adapter Cable for Cassette.
### Connector Specifications (System End) (Part 1 of 2)

<table>
<thead>
<tr>
<th>Connector</th>
<th>Signal/Connection</th>
<th>System Connector Pin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aux. (Red)</td>
<td>Signal</td>
<td>B2</td>
</tr>
<tr>
<td></td>
<td>Gnd</td>
<td>A1</td>
</tr>
<tr>
<td>Ear (Black)</td>
<td>Signal</td>
<td>A2</td>
</tr>
<tr>
<td></td>
<td>Gnd</td>
<td>A1</td>
</tr>
<tr>
<td>Remote (Gray)</td>
<td>Signal</td>
<td>A4</td>
</tr>
<tr>
<td></td>
<td>Gnd</td>
<td>B3</td>
</tr>
</tbody>
</table>

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

3-92 Cassette Adapter Cable
IBM PC<sub>jr</sub> Adapter Cable for the IBM Color Display

This adapter cable allows the IBM Color Display to be connected to the IBM PC<sub>jr</sub>.

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

![Diagram of IBM PC<sub>jr</sub> adapter cable and connector specifications](image)

**Color Direct-Drive 9-Pin D-Shell Connector**

**Adapter Cable for IBM Color Display Connectors**
## Connector Specifications

<table>
<thead>
<tr>
<th>System Connector</th>
<th>Cable</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>Not Used</td>
</tr>
<tr>
<td>A2</td>
<td>Not Used</td>
</tr>
<tr>
<td>A3</td>
<td>Not Used</td>
</tr>
<tr>
<td>A4</td>
<td>Red</td>
</tr>
<tr>
<td>A5</td>
<td>Green</td>
</tr>
<tr>
<td>A6</td>
<td>Blue</td>
</tr>
<tr>
<td>A7</td>
<td>Intensity</td>
</tr>
<tr>
<td>A8</td>
<td>Not Used</td>
</tr>
<tr>
<td>A9</td>
<td>Not Used</td>
</tr>
<tr>
<td>B1</td>
<td>Vertical</td>
</tr>
<tr>
<td>B2</td>
<td>Not Used</td>
</tr>
<tr>
<td>B3</td>
<td>Horizontal</td>
</tr>
<tr>
<td>B4</td>
<td>Not Used</td>
</tr>
<tr>
<td>B5</td>
<td>Ground</td>
</tr>
<tr>
<td>B6</td>
<td>Not Used</td>
</tr>
<tr>
<td>B7</td>
<td>Not Used</td>
</tr>
<tr>
<td>B8</td>
<td>Not Used</td>
</tr>
<tr>
<td>B9</td>
<td>Ground</td>
</tr>
</tbody>
</table>

### 9-Pin Color Display Connector

- Pin 1: Ground
- Pin 2: Ground
- Pin 3: Not Used
- Pin 4: Not Used
- Pin 5: Intensity
- Pin 6: Blue
- Pin 7: Green
- Pin 8: Horizontal
- Pin 9: Vertical

3-94 Color Display Connector
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.
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.

<table>
<thead>
<tr>
<th>Addresses (hex)</th>
<th>Operation</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>378</td>
<td>'Read'</td>
<td>Read Data Latch</td>
</tr>
<tr>
<td>379</td>
<td>'Read'</td>
<td>Read Status</td>
</tr>
<tr>
<td>37A</td>
<td>'Read'</td>
<td>Read Control Latch</td>
</tr>
<tr>
<td>37B</td>
<td>'Read'</td>
<td>Unused</td>
</tr>
<tr>
<td>37B</td>
<td>'Write'</td>
<td>Write Data Latch</td>
</tr>
<tr>
<td>379</td>
<td>'Write'</td>
<td>Unused</td>
</tr>
<tr>
<td>37A</td>
<td>'Write'</td>
<td>Write Control Latch</td>
</tr>
<tr>
<td>37B</td>
<td>'Write'</td>
<td>Unused</td>
</tr>
</tbody>
</table>

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.

![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.
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.
<table>
<thead>
<tr>
<th>Bit</th>
<th>Signal Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSB 7</td>
<td>-BUSY</td>
<td>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.</td>
</tr>
<tr>
<td>6</td>
<td>-ACK</td>
<td>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.</td>
</tr>
<tr>
<td>5</td>
<td>-PE</td>
<td>A low level indicates that the printer has detected an end of form.</td>
</tr>
<tr>
<td>4</td>
<td>+SLCT</td>
<td>A high level indicates that the printer is selected.</td>
</tr>
<tr>
<td>3</td>
<td>-ERROR</td>
<td>A low level indicates that the printer has encountered an error condition.</td>
</tr>
<tr>
<td>2 Through 0 LSB</td>
<td></td>
<td>Unused.</td>
</tr>
</tbody>
</table>

**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:
The following figure describes the printer control signals.

Parallel Interface Timing Diagram
<table>
<thead>
<tr>
<th>Bit</th>
<th>Signal Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSB 7 Through 5</td>
<td></td>
<td>Unused.</td>
</tr>
<tr>
<td>4</td>
<td>+INTERRUPT ENABLE</td>
<td>A high level in this bit position will allow an interrupt to occur when –ACK goes high.</td>
</tr>
<tr>
<td>3</td>
<td>SLCT IN</td>
<td>A low level in this bit position selects the printer.</td>
</tr>
<tr>
<td>2</td>
<td>INIT</td>
<td>A low level will initialize the printer (50 microseconds minimum).</td>
</tr>
<tr>
<td>1</td>
<td>AUTO FD XT</td>
<td>A low level will cause the printer to line feed anytime a line is printed.</td>
</tr>
<tr>
<td>LSB 0</td>
<td>STROBE</td>
<td>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.</td>
</tr>
</tbody>
</table>

**Printer Control Signal**

The following are the connector specifications for the IBM PCjr Parallel Printer Attachment.
## 25-Pin "D"-Shell Connector

<table>
<thead>
<tr>
<th>Pin</th>
<th>Signal</th>
<th>$I_{OL\ Max}$</th>
<th>$I_{OH\ Max}$</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-STROBE</td>
<td>14 ma</td>
<td>-.6 ma</td>
<td>Attachment Card</td>
</tr>
<tr>
<td>2</td>
<td>DATA BIT 0</td>
<td>24 ma</td>
<td>-2.6 ma</td>
<td>Attachment Card</td>
</tr>
<tr>
<td></td>
<td>Through 9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DATA BIT 7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>-ACK</td>
<td>74LS Input</td>
<td>74LS Input</td>
<td>Printer</td>
</tr>
<tr>
<td>11</td>
<td>BUSY</td>
<td>74LS Input</td>
<td>74LS Input</td>
<td>Printer</td>
</tr>
<tr>
<td>12</td>
<td>PE</td>
<td>74LS Input</td>
<td>74LS Input</td>
<td>Printer</td>
</tr>
<tr>
<td>13</td>
<td>SLCT</td>
<td>74LS Input</td>
<td>74LS Input</td>
<td>Printer</td>
</tr>
<tr>
<td>14</td>
<td>-AUTO</td>
<td>14 ma</td>
<td>.6 ma</td>
<td>Attachment Card</td>
</tr>
<tr>
<td></td>
<td>FD XT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>-ERROR</td>
<td>74LS Input</td>
<td>74LS Input</td>
<td>Printer</td>
</tr>
<tr>
<td>16</td>
<td>-INIT PRINTER</td>
<td>14 ma</td>
<td>.6 ma</td>
<td>Printer</td>
</tr>
<tr>
<td>17</td>
<td>-SELECT INPUT</td>
<td>14 ma</td>
<td>.6 ma</td>
<td>Attachment Card</td>
</tr>
<tr>
<td>18</td>
<td>GND</td>
<td>N/A</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Through 25</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

**3-104 Parallel Printer Attachment**
### Parallel Printer Attachment to I/O Expansion Connector

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Signal Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>B1</td>
</tr>
<tr>
<td>D2</td>
<td>A1</td>
</tr>
<tr>
<td>D4</td>
<td>D0</td>
</tr>
<tr>
<td>GND</td>
<td>+12 Vdc</td>
</tr>
<tr>
<td>D7</td>
<td>D3</td>
</tr>
<tr>
<td>A0</td>
<td>D5</td>
</tr>
<tr>
<td>A2</td>
<td>D6</td>
</tr>
<tr>
<td>GND</td>
<td>+5 Vdc</td>
</tr>
<tr>
<td>A5</td>
<td>A1</td>
</tr>
<tr>
<td>A6</td>
<td>A3</td>
</tr>
<tr>
<td>A8</td>
<td>A4</td>
</tr>
<tr>
<td>-DACK0</td>
<td>A5</td>
</tr>
<tr>
<td>A11</td>
<td>A6</td>
</tr>
<tr>
<td>A12</td>
<td>A10</td>
</tr>
<tr>
<td>GND</td>
<td>DRQ0</td>
</tr>
<tr>
<td>A15</td>
<td>A13</td>
</tr>
<tr>
<td>A17</td>
<td>A14</td>
</tr>
<tr>
<td>GND</td>
<td>A16</td>
</tr>
<tr>
<td>A19</td>
<td>A17</td>
</tr>
<tr>
<td>GND</td>
<td>A18</td>
</tr>
<tr>
<td>B20</td>
<td>-IOR</td>
</tr>
<tr>
<td>-MEMR</td>
<td>-IOW</td>
</tr>
<tr>
<td>-MEMW</td>
<td>GND</td>
</tr>
<tr>
<td>ALE</td>
<td>HDLA</td>
</tr>
<tr>
<td>GND</td>
<td>CLK</td>
</tr>
<tr>
<td>A10</td>
<td>-CARD SLCTD</td>
</tr>
<tr>
<td>B25</td>
<td>RESET</td>
</tr>
<tr>
<td>READY</td>
<td>+5 Vdc</td>
</tr>
<tr>
<td>-CARD SLCTD</td>
<td>-HRQ</td>
</tr>
<tr>
<td>GND</td>
<td>IRQ1</td>
</tr>
<tr>
<td>IRQ7</td>
<td>IRQ2</td>
</tr>
<tr>
<td>AUDIO IN</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Connector Specifications (Part 2 of 2)
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 descendents 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:

<table>
<thead>
<tr>
<th>Height</th>
<th>107 mm (4.2 inches)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Width</td>
<td>374 mm (14.7 inches)</td>
</tr>
<tr>
<td>Depth</td>
<td>305 mm (12 inches)</td>
</tr>
<tr>
<td>Weight</td>
<td>5.5 kg (12 pounds)</td>
</tr>
</tbody>
</table>

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
### Functions and Conditions of DIP Switch 1

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

### Functions and Conditions of DIP Switch 2

<table>
<thead>
<tr>
<th>Switch Number</th>
<th>Function</th>
<th>On</th>
<th>Off</th>
<th>Factory Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-1</td>
<td>Form Length</td>
<td>12 Inches</td>
<td>11 Inches</td>
<td>Off</td>
</tr>
<tr>
<td>2-2</td>
<td>Line Spacing</td>
<td>1/8 Inch</td>
<td>1/6 Inch</td>
<td>Off</td>
</tr>
<tr>
<td>2-3</td>
<td>Auto Feed XT Signal</td>
<td>Fixed Internally</td>
<td>Not Fixed Internally</td>
<td>Off</td>
</tr>
<tr>
<td>2-4</td>
<td>1 Inch Skip Over Perforation</td>
<td>Valid</td>
<td>Invalid</td>
<td>Off</td>
</tr>
</tbody>
</table>
Parallel Interface Description

Specifications

Data Transfer Rate 1000 cycles-per-second (cps)-(maximum)

Synchronization By externally-supplied STROBE pulses

Signal Exchange -ACKNLG or BUSY signals

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

**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

<table>
<thead>
<tr>
<th>Printer Modes</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Compressed</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Emphasized</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Double Strike</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Subscript</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Superscript</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Double Width</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Underline</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

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 “Printed Character Sets.”

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

<table>
<thead>
<tr>
<th>Printer code</th>
<th>Printer Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUL</td>
<td>Null:</td>
</tr>
<tr>
<td></td>
<td>Used with ESC B and ESC D as a list terminator. NUL is also used with other printer.</td>
</tr>
</tbody>
</table>
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);

FF
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);

ESC 0
Escape Zero (1/8-Inch Line Feeding)
Changes paper feeding to 1/8-inch.
Example:
LPRINT CHR$(27);CHR$(48);

ESC 1
Escape One (7/72-Inch Line
Feeding)
Changes paper feeding to 7/72-inch.
Example:
LPRINT CHR$(27);CHR$(49);

ESC 2
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);

ESC 3
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);

ESC 6
Escape Six (Select Character Set 2)
Selects Character Set 2. (See “Printer
Character set 2”)
Example:
LPRINT CHR$(27);CHR$(54);

ESC 7
Escape Seven (Select Character Set 1)
Selects character set 1. (See "Printer Character Set 1")
Character set 1 is selected when the printer is powered on or reset.
Example:
LPRINT CHR$(27);CHR$(55);

ESC 8
Escape Eight (Ignore Paper End)
Allows the printer to print to the end of the paper. The printer ignores the Paper End switch.
Example:
LPRINT CHR$(27);CHR$(56);

ESC 9
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:
LPRINT CHR$(27);CHR$(57);

ESC <
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:
LPRINT CHR$(27);CHR$(60);

ESC A
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:
LPRINT CHR$(27);CHR$(69);

ESC F  Escape F (Emphasized Off)
Stops printing in the Emphasized-print mode.
Example:
LPRINT CHR$(27);CHR$(70);

ESC G  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:
LPRINT CHR$(27);CHR$(71);

ESC H  Escape H (Double Strike Off)
Stops printing in the Double-Strike mode.
Example:
LPRINT CHR$(27);CHR$(72);

ESC J  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:
LPRINT CHR$(27);CHR$(74);CHR$(50);

ESC K  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.

```
MSB          LSB
2^7  2^6  2^5  2^4  2^3  2^2  2^1  2^0
```

```
MSB          LSB
2^15 2^14 2^13 2^12 2^11 2^10 2^9 2^8
```

Data sent to the printer.

```
Text (20 characters)  ESC  K  n=360  Bit-image data  Next data
```
In text mode, 20 characters in text mode correspond to 120 bit-image positions (20 x 6 = 120). The printable portion left in Bit-Image mode is 360 dot positions (480 - 120 = 360).

Data sent to the printer.

<table>
<thead>
<tr>
<th>Data A</th>
<th>ESC K</th>
<th>n₁</th>
<th>n₂</th>
<th>Data B</th>
<th>Data C</th>
<th>ESC K</th>
<th>n₁</th>
<th>n₂</th>
<th>Data D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text data</td>
<td>Length of data</td>
<td>Bit-image data</td>
<td>Text data</td>
<td>Length of data</td>
<td>Bit-image data</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

480 bit-image dot positions

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 'ESCKNI 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

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

**ESC L**

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: \texttt{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. \texttt{ESC Z} can print only every third dot position.
### Printer Character Set 1 (Part 1 of 2)

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>NUL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
<td>29</td>
</tr>
<tr>
<td>LF</td>
<td>FF</td>
<td>CR</td>
<td>SO</td>
<td>SI</td>
<td>DC2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>31</td>
<td>32</td>
<td>33</td>
<td>34</td>
<td>35</td>
<td>36</td>
<td>37</td>
<td>38</td>
<td>39</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>41</td>
<td>42</td>
<td>43</td>
<td>44</td>
<td>45</td>
<td>46</td>
<td>47</td>
<td>48</td>
<td>49</td>
</tr>
<tr>
<td>SP</td>
<td>&quot;</td>
<td>'</td>
<td>#</td>
<td>$</td>
<td>%</td>
<td>&amp;</td>
<td>'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>51</td>
<td>52</td>
<td>53</td>
<td>54</td>
<td>55</td>
<td>56</td>
<td>57</td>
<td>58</td>
<td>59</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>61</td>
<td>62</td>
<td>63</td>
<td>64</td>
<td>65</td>
<td>66</td>
<td>67</td>
<td>68</td>
<td>69</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>:</td>
<td>;</td>
</tr>
<tr>
<td>70</td>
<td>71</td>
<td>72</td>
<td>73</td>
<td>74</td>
<td>75</td>
<td>76</td>
<td>77</td>
<td>78</td>
<td>79</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>81</td>
<td>82</td>
<td>83</td>
<td>84</td>
<td>85</td>
<td>86</td>
<td>87</td>
<td>88</td>
<td>89</td>
</tr>
<tr>
<td>F</td>
<td>G</td>
<td>H</td>
<td>I</td>
<td>J</td>
<td>K</td>
<td>L</td>
<td>M</td>
<td>N</td>
<td>O</td>
</tr>
<tr>
<td>90</td>
<td>91</td>
<td>92</td>
<td>93</td>
<td>94</td>
<td>95</td>
<td>96</td>
<td>97</td>
<td>98</td>
<td>99</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>101</td>
<td>102</td>
<td>103</td>
<td>104</td>
<td>105</td>
<td>106</td>
<td>107</td>
<td>108</td>
<td>109</td>
</tr>
<tr>
<td>Z</td>
<td>[</td>
<td>\</td>
<td>]</td>
<td>^</td>
<td>_</td>
<td>`</td>
<td>a</td>
<td>b</td>
<td>c</td>
</tr>
<tr>
<td>110</td>
<td>111</td>
<td>112</td>
<td>113</td>
<td>114</td>
<td>115</td>
<td>116</td>
<td>117</td>
<td>118</td>
<td>119</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>120</td>
<td>121</td>
<td>122</td>
<td>123</td>
<td>124</td>
<td>125</td>
<td>126</td>
<td>127</td>
<td>128</td>
<td>129</td>
</tr>
<tr>
<td>d</td>
<td>e</td>
<td>f</td>
<td>g</td>
<td>h</td>
<td>i</td>
<td>j</td>
<td>k</td>
<td>l</td>
<td>m</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>130</td>
<td>131</td>
<td>132</td>
<td>133</td>
<td>134</td>
<td>135</td>
<td>136</td>
<td>137</td>
<td>138</td>
<td>139</td>
</tr>
<tr>
<td>n</td>
<td>o</td>
<td>p</td>
<td>q</td>
<td>r</td>
<td>s</td>
<td>t</td>
<td>u</td>
<td>v</td>
<td>w</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>140</td>
<td>141</td>
<td>142</td>
<td>143</td>
<td>144</td>
<td>145</td>
<td>146</td>
<td>147</td>
<td>148</td>
<td>149</td>
</tr>
<tr>
<td>x</td>
<td>y</td>
<td>z</td>
<td>{</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>~</td>
</tr>
</tbody>
</table>

3-128 Printers
<p>| Printer Character Set 2 (Part 1 of 2) | 3-130 Printers |</p>
<table>
<thead>
<tr>
<th>130</th>
<th>131</th>
<th>132</th>
<th>133</th>
<th>134</th>
<th>135</th>
<th>136</th>
<th>137</th>
<th>138</th>
<th>139</th>
</tr>
</thead>
<tbody>
<tr>
<td>é</td>
<td>â</td>
<td>à</td>
<td>å</td>
<td>âç</td>
<td>ê</td>
<td>ë</td>
<td>ê</td>
<td>ë</td>
<td>ëi</td>
</tr>
<tr>
<td>140</td>
<td>141</td>
<td>142</td>
<td>143</td>
<td>144</td>
<td>145</td>
<td>146</td>
<td>147</td>
<td>148</td>
<td>149</td>
</tr>
<tr>
<td>ì</td>
<td>ì Ä</td>
<td>ì Ê</td>
<td>ì æ</td>
<td>ì Æ</td>
<td>ô</td>
<td>ô</td>
<td>ô</td>
<td>ô</td>
<td>ô</td>
</tr>
<tr>
<td>150</td>
<td>151</td>
<td>152</td>
<td>153</td>
<td>154</td>
<td>155</td>
<td>156</td>
<td>157</td>
<td>158</td>
<td>159</td>
</tr>
<tr>
<td>û</td>
<td>û Û</td>
<td>û Œ</td>
<td>û £</td>
<td>û ¥</td>
<td>û F</td>
<td>û F</td>
<td>û F</td>
<td>û F</td>
<td>û F</td>
</tr>
<tr>
<td>160</td>
<td>161</td>
<td>162</td>
<td>163</td>
<td>164</td>
<td>165</td>
<td>166</td>
<td>167</td>
<td>168</td>
<td>169</td>
</tr>
<tr>
<td>á</td>
<td>í</td>
<td>ó</td>
<td>ù</td>
<td>ñ</td>
<td>Ñ</td>
<td>ña</td>
<td>õ</td>
<td>ç</td>
<td>—</td>
</tr>
<tr>
<td>170</td>
<td>171</td>
<td>172</td>
<td>173</td>
<td>174</td>
<td>175</td>
<td>176</td>
<td>177</td>
<td>178</td>
<td>179</td>
</tr>
<tr>
<td>—</td>
<td>— 1/2</td>
<td>1/4</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>180</td>
<td>181</td>
<td>182</td>
<td>183</td>
<td>184</td>
<td>185</td>
<td>186</td>
<td>187</td>
<td>188</td>
<td>189</td>
</tr>
<tr>
<td>190</td>
<td>191</td>
<td>192</td>
<td>193</td>
<td>194</td>
<td>195</td>
<td>196</td>
<td>197</td>
<td>198</td>
<td>199</td>
</tr>
<tr>
<td>200</td>
<td>201</td>
<td>202</td>
<td>203</td>
<td>204</td>
<td>205</td>
<td>206</td>
<td>207</td>
<td>208</td>
<td>209</td>
</tr>
<tr>
<td>210</td>
<td>211</td>
<td>212</td>
<td>213</td>
<td>214</td>
<td>215</td>
<td>216</td>
<td>217</td>
<td>218</td>
<td>219</td>
</tr>
<tr>
<td>220</td>
<td>221</td>
<td>222</td>
<td>223</td>
<td>224</td>
<td>225</td>
<td>226</td>
<td>227</td>
<td>228</td>
<td>229</td>
</tr>
<tr>
<td>230</td>
<td>231</td>
<td>232</td>
<td>233</td>
<td>234</td>
<td>235</td>
<td>236</td>
<td>237</td>
<td>238</td>
<td>239</td>
</tr>
<tr>
<td>µ</td>
<td>τ</td>
<td>φ</td>
<td>Θ</td>
<td>Ω</td>
<td>δ</td>
<td>ω</td>
<td>ω</td>
<td>σ</td>
<td></td>
</tr>
<tr>
<td>240</td>
<td>241</td>
<td>242</td>
<td>243</td>
<td>244</td>
<td>245</td>
<td>246</td>
<td>247</td>
<td>248</td>
<td>249</td>
</tr>
<tr>
<td>≡</td>
<td>±</td>
<td>≥</td>
<td>≤</td>
<td>∫</td>
<td>∴</td>
<td>÷</td>
<td>≈</td>
<td></td>
<td></td>
</tr>
<tr>
<td>250</td>
<td>251</td>
<td>252</td>
<td>253</td>
<td>254</td>
<td>255</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>—</td>
<td>√</td>
<td>∩</td>
<td>2</td>
<td>SP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Printer Character Set 2 (Part 2 of 2)
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
<table>
<thead>
<tr>
<th>Print Modes</th>
<th>Characters per Inch</th>
<th>Maximum Characters per Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard</td>
<td>10</td>
<td>80</td>
</tr>
<tr>
<td>Double Width</td>
<td>5</td>
<td>40</td>
</tr>
<tr>
<td>Compressed</td>
<td>17.5</td>
<td>136</td>
</tr>
<tr>
<td>Compressed/Double Width</td>
<td>8.75</td>
<td>68</td>
</tr>
</tbody>
</table>

- **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
<table>
<thead>
<tr>
<th>Environmental Conditions</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Temperature:</strong></td>
<td>$5^\circ C$ ($+41^\circ F$) to $40^\circ C$ ($104^\circ F$)</td>
</tr>
<tr>
<td><strong>Humidity:</strong></td>
<td>10 to 80% non-condensing</td>
</tr>
<tr>
<td><strong>Power Requirement</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Voltage:</strong></td>
<td>110 Vac 60 Hz</td>
</tr>
<tr>
<td><strong>Current:</strong></td>
<td>245 mA</td>
</tr>
<tr>
<td><strong>Power Consumption:</strong></td>
<td>36 watts</td>
</tr>
<tr>
<td><strong>Heat Output:</strong></td>
<td>$57.6$ kJ ($54.6$ BTU)/hr (maximum)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Physical Characteristics</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Height:</strong></td>
<td>88.9 mm (3.5 in)</td>
</tr>
<tr>
<td><strong>Width:</strong></td>
<td>312.4 mm (12.3 in)</td>
</tr>
<tr>
<td><strong>Depth:</strong></td>
<td>221 mm (8.7 in)</td>
</tr>
<tr>
<td><strong>Weight:</strong></td>
<td>2.99 kg (6.6 lb)</td>
</tr>
<tr>
<td><strong>Power Cable Length:</strong></td>
<td>1.98 m (6.5 ft)</td>
</tr>
<tr>
<td><strong>Size:</strong></td>
<td>28 AWG</td>
</tr>
<tr>
<td><strong>Printer Cable Length:</strong></td>
<td>1.83 m (6 ft)</td>
</tr>
<tr>
<td><strong>Size:</strong></td>
<td>3 by 18 AWG</td>
</tr>
</tbody>
</table>
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](image-url)

Printers 3-139
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.

<table>
<thead>
<tr>
<th>Modes</th>
<th>XXX</th>
<th>XXX</th>
<th>XXX</th>
<th>XXX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard</td>
<td>XXX</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Compressed</td>
<td>XXX</td>
<td>XXX</td>
<td>XXX</td>
<td>XXX</td>
</tr>
<tr>
<td>Double-Width</td>
<td>XXX</td>
<td>XXX</td>
<td>XXX</td>
<td>XXX</td>
</tr>
<tr>
<td>Underline</td>
<td>XXX</td>
<td>XXX</td>
<td></td>
<td>XXX</td>
</tr>
</tbody>
</table>

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.
<table>
<thead>
<tr>
<th>CODE</th>
<th>PRINTER FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAN</td>
<td><strong>Cancel</strong></td>
</tr>
<tr>
<td></td>
<td>Clears the printer buffer. Control codes, except SO, remain in effect. Reinitializes the printer to the power on defaults.</td>
</tr>
<tr>
<td></td>
<td><strong>LPRINT CHR$(24);</strong></td>
</tr>
<tr>
<td>CR</td>
<td><strong>Carriage Return</strong></td>
</tr>
<tr>
<td></td>
<td>Ends the line the printer is on and prints any data remaining in the printer buffer. The logical character position is moved to the left margin. (No Line Feed operation takes place.) Note: IBM Personal Computer BASIC adds a Line Feed unless 128 is added.</td>
</tr>
<tr>
<td></td>
<td><strong>LPRINT CHR$(13);</strong></td>
</tr>
<tr>
<td>DC2</td>
<td><strong>Device Control 2 (Compressed Off)</strong></td>
</tr>
<tr>
<td></td>
<td>Stops printing in the Compressed mode.</td>
</tr>
<tr>
<td></td>
<td><strong>LPRINT CHR$(18);</strong></td>
</tr>
<tr>
<td>DC4</td>
<td><strong>Device Control 4 (Double Width Off)</strong></td>
</tr>
<tr>
<td></td>
<td>Stops printing in the Double Width mode.</td>
</tr>
<tr>
<td></td>
<td><strong>LPRINT CHR$(20);</strong></td>
</tr>
<tr>
<td>ESC</td>
<td><strong>Escape</strong></td>
</tr>
<tr>
<td></td>
<td>Informs the printer that the following data is a printer command. (See the following ESC commands.)</td>
</tr>
<tr>
<td></td>
<td><strong>LPRINT CHR$(27);</strong></td>
</tr>
</tbody>
</table>
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$(55);
ESCD  Escape D  (Set Horizontal Tab Stops)
Sets the horizontal tab stop positions. The following example shows the horizontal tab stop positions set at printer column positions of 10, 20 and 40. The horizontal tab stops are followed by CHR$(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.

<table>
<thead>
<tr>
<th>Dot</th>
<th>Bit Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top</td>
<td>O - - - 8</td>
</tr>
<tr>
<td></td>
<td>O - - - 7</td>
</tr>
<tr>
<td></td>
<td>O - - - 6</td>
</tr>
<tr>
<td></td>
<td>O - - - 5</td>
</tr>
<tr>
<td></td>
<td>O - - - 4</td>
</tr>
<tr>
<td></td>
<td>O - - - 3</td>
</tr>
<tr>
<td></td>
<td>O - - - 2</td>
</tr>
<tr>
<td>Bottom</td>
<td>O - - - 1</td>
</tr>
</tbody>
</table>

LPRINT CHR$(27);CHR$(75);nl;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: \texttt{ESC \(-;n;\)}; \texttt{ESC -} followed by 1, prints all of the following data with an underline. \texttt{ESC -} followed by 0 (zero), cancels the Underline print mode.
\texttt{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 \texttt{ESC <}. This occurs for one line only.
\texttt{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. \texttt{ESC C} can be used to change the page length. Always separate multiple Form Feed commands with spaces.
\texttt{LPRINT CHR$(12);}

**HT** Horizontal Tab
Tabs to the next horizontal tab stop. Tab stops are set with \texttt{ESC D}. (Tab stops are automatically set at every 8 columns when the printer power switch is set to the On position.)
\texttt{LPRINT CHR$(9);}

**LF** Line Feed
Advances the paper one line. Line spacing is 4.23 mm (1/6 inch) unless reset by \texttt{ESC 0}, \texttt{ESC 1}, \texttt{ESC 2}.
\texttt{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).
<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
</tr>
<tr>
<td>LF</td>
<td>VT</td>
<td>FF</td>
<td>CR</td>
<td>SO</td>
<td>SI</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>DC4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>31</td>
<td>32</td>
<td>33</td>
<td>34</td>
<td>35</td>
<td>36</td>
<td>37</td>
<td>38</td>
<td>39</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>41</td>
<td>42</td>
<td>43</td>
<td>44</td>
<td>45</td>
<td>46</td>
<td>47</td>
<td>48</td>
<td>49</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>51</td>
<td>52</td>
<td>53</td>
<td>54</td>
<td>55</td>
<td>56</td>
<td>57</td>
<td>58</td>
<td>59</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>61</td>
<td>62</td>
<td>63</td>
<td>64</td>
<td>65</td>
<td>66</td>
<td>67</td>
<td>68</td>
<td>69</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>71</td>
<td>72</td>
<td>73</td>
<td>74</td>
<td>75</td>
<td>76</td>
<td>77</td>
<td>78</td>
<td>79</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>81</td>
<td>82</td>
<td>83</td>
<td>84</td>
<td>85</td>
<td>86</td>
<td>87</td>
<td>88</td>
<td>89</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>90</td>
<td>91</td>
<td>92</td>
<td>93</td>
<td>94</td>
<td>95</td>
<td>96</td>
<td>97</td>
<td>98</td>
<td>99</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Z</td>
<td>[</td>
<td>\</td>
<td>]</td>
<td>^</td>
<td>_</td>
<td>`</td>
<td>a</td>
<td>b</td>
<td>c</td>
</tr>
<tr>
<td>100</td>
<td>101</td>
<td>102</td>
<td>103</td>
<td>104</td>
<td>105</td>
<td>106</td>
<td>107</td>
<td>108</td>
<td>109</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>e</td>
<td>f</td>
<td>g</td>
<td>h</td>
<td>i</td>
<td>j</td>
<td>k</td>
<td>l</td>
<td>m</td>
</tr>
<tr>
<td>110</td>
<td>111</td>
<td>112</td>
<td>113</td>
<td>114</td>
<td>115</td>
<td>116</td>
<td>117</td>
<td>118</td>
<td>119</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n</td>
<td>o</td>
<td>p</td>
<td>q</td>
<td>r</td>
<td>s</td>
<td>t</td>
<td>u</td>
<td>v</td>
<td>w</td>
</tr>
<tr>
<td>120</td>
<td>121</td>
<td>122</td>
<td>123</td>
<td>124</td>
<td>125</td>
<td>126</td>
<td>127</td>
<td>128</td>
<td>129</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>y</td>
<td>z</td>
<td>{</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Character Set (Part 1 of 2)
<table>
<thead>
<tr>
<th>130</th>
<th>131</th>
<th>132</th>
<th>133</th>
<th>134</th>
<th>135</th>
<th>136</th>
<th>137</th>
<th>138</th>
<th>139</th>
</tr>
</thead>
<tbody>
<tr>
<td>é</td>
<td>â</td>
<td>â</td>
<td>å</td>
<td>â</td>
<td>ç</td>
<td>ê</td>
<td>ë</td>
<td>è</td>
<td>ì</td>
</tr>
<tr>
<td>140</td>
<td>141</td>
<td>142</td>
<td>143</td>
<td>144</td>
<td>145</td>
<td>146</td>
<td>147</td>
<td>148</td>
<td>149</td>
</tr>
<tr>
<td>î</td>
<td>ï</td>
<td>À</td>
<td>Â</td>
<td>À</td>
<td>E</td>
<td>ã</td>
<td>Ò</td>
<td>õ</td>
<td>ò</td>
</tr>
<tr>
<td>150</td>
<td>151</td>
<td>152</td>
<td>153</td>
<td>154</td>
<td>155</td>
<td>156</td>
<td>157</td>
<td>158</td>
<td>159</td>
</tr>
<tr>
<td>û</td>
<td>ù</td>
<td>ÿ</td>
<td>Ò</td>
<td>Ò</td>
<td>Ö</td>
<td>Ò</td>
<td>¥</td>
<td>R</td>
<td>f</td>
</tr>
<tr>
<td>160</td>
<td>161</td>
<td>162</td>
<td>163</td>
<td>164</td>
<td>165</td>
<td>166</td>
<td>167</td>
<td>168</td>
<td>169</td>
</tr>
<tr>
<td>á</td>
<td>í</td>
<td>ó</td>
<td>ü</td>
<td>Ú</td>
<td>Ñ</td>
<td>ñ</td>
<td>a</td>
<td>o</td>
<td>c</td>
</tr>
<tr>
<td>170</td>
<td>171</td>
<td>172</td>
<td>173</td>
<td>174</td>
<td>175</td>
<td>176</td>
<td>177</td>
<td>178</td>
<td>179</td>
</tr>
<tr>
<td>1/2</td>
<td>1/4</td>
<td>i</td>
<td>l</td>
<td>l</td>
<td>l</td>
<td>l</td>
<td>l</td>
<td>l</td>
<td>l</td>
</tr>
<tr>
<td>180</td>
<td>181</td>
<td>182</td>
<td>183</td>
<td>184</td>
<td>185</td>
<td>186</td>
<td>187</td>
<td>188</td>
<td>189</td>
</tr>
<tr>
<td>190</td>
<td>191</td>
<td>192</td>
<td>193</td>
<td>194</td>
<td>195</td>
<td>196</td>
<td>197</td>
<td>198</td>
<td>199</td>
</tr>
<tr>
<td>200</td>
<td>201</td>
<td>202</td>
<td>203</td>
<td>204</td>
<td>205</td>
<td>206</td>
<td>207</td>
<td>208</td>
<td>209</td>
</tr>
<tr>
<td>210</td>
<td>211</td>
<td>212</td>
<td>213</td>
<td>214</td>
<td>215</td>
<td>216</td>
<td>217</td>
<td>218</td>
<td>219</td>
</tr>
<tr>
<td>220</td>
<td>221</td>
<td>222</td>
<td>223</td>
<td>224</td>
<td>225</td>
<td>226</td>
<td>227</td>
<td>228</td>
<td>229</td>
</tr>
<tr>
<td>α</td>
<td>β</td>
<td>Γ</td>
<td>Π</td>
<td>Σ</td>
<td>σ</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>230</td>
<td>231</td>
<td>232</td>
<td>233</td>
<td>234</td>
<td>235</td>
<td>236</td>
<td>237</td>
<td>238</td>
<td>239</td>
</tr>
<tr>
<td>μ</td>
<td>τ</td>
<td>φ</td>
<td>Θ</td>
<td>Ω</td>
<td>δ</td>
<td>ϵ</td>
<td>Ξ</td>
<td>Υ</td>
<td>Ω</td>
</tr>
<tr>
<td>240</td>
<td>241</td>
<td>242</td>
<td>243</td>
<td>244</td>
<td>245</td>
<td>246</td>
<td>247</td>
<td>248</td>
<td>249</td>
</tr>
<tr>
<td>≡</td>
<td>±</td>
<td>&gt;</td>
<td>&lt;</td>
<td>∫</td>
<td>÷</td>
<td>~</td>
<td>°</td>
<td>-</td>
<td>SP</td>
</tr>
<tr>
<td>250</td>
<td>251</td>
<td>252</td>
<td>253</td>
<td>254</td>
<td>255</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
I
II!
!!
!!:ll<»'

A0AATTT=;=:;::==:=d
BOa
16 Pin Connector

Signal Name - Description
Not Used

Data Terminal Ready  A02
Request To Send      A03
Transmit Data
Carrier Detect       A05
Data Set Ready       A06
Clear To Send
Not Used            A07
Not Used            A08
Not Used            B01
Not Used            B02
Not Used            B03
Ground              B04
Not Used            B05
Not Used            B06
Ground              B07
Not Used            B08

Compact Printer
Serial Port
(RS-232-C)

Data Terminal Ready Looped in Cable to Data Set Ready
Request to Send Looped in Cable to Carrier Detect

Connector Specifications

3-150  Printers
SECTION 4. COMPATIBILITY WITH THE IBM PERSONAL COMPUTER FAMILY

Contents

Compatibility Overview ........................................... 4-3
Timming 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.
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:

4-8 Unequal Configurations
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:
<table>
<thead>
<tr>
<th>Device</th>
<th>PC</th>
<th>PCXT</th>
<th>PCjr</th>
<th>PCjr Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum User Memory</td>
<td>640KB</td>
<td>640KB</td>
<td>128KB</td>
<td>Shares user RAM with Video Buffer</td>
</tr>
<tr>
<td>Cordless Keyboard</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Scan codes compatible and full 83 key capability</td>
</tr>
<tr>
<td>83 Key Keyboard</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Compatible, but Hardware interface differences</td>
</tr>
<tr>
<td>Diskette Drive</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Compatible, but different address and no DMA support</td>
</tr>
<tr>
<td>Hard Disk File</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Parallel Printer</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Compatible</td>
</tr>
<tr>
<td>RS 232 Serial Port</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Compatible, hex 2F8 address, Interrupt Level 3, Baud-Rate-Frequency divisor difference</td>
</tr>
<tr>
<td>Game Control</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Compatible interface with potential timing differences</td>
</tr>
<tr>
<td>Cassette</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Compatible</td>
</tr>
<tr>
<td>Internal Modem</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Compatible to PC Serial Port hex 3F8 address, Interrupt Level 4, frequency divisor difference</td>
</tr>
<tr>
<td>IBM Monochrome Display</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Color Graphics and Display</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Compatible, with some register differences and enhancements</td>
</tr>
<tr>
<td>Light Pen</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Compatible</td>
</tr>
</tbody>
</table>

PCjr and Personal Computers Comparison (Part 1 of 2)

4-10 Hardware Differences
**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 PCjr in relationship to that of the IBM Personal Computers. The IBM PCjr 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 PCjr 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 PCjr 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 PCjr. The IBM PCjr 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 PCjr 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

4-12 Hardware Differences
approximately 10K bytes and 14K bytes respectively from user memory. In the IBM PCjr, 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 PCjr without its expansion memory (128K byte capability). This is because of the IBM PCjr 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 PCjr.

**Diskette Capacity/Operation**

Since the IBM PCjr maximum stand-alone configuration is one diskette drive with a maximum capacity of 360K bytes diskette storage, an IBM PCjr 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 PCjr and the IBM Personal Computers both utilize the NEC μPD765 floppy diskette controller, but with different hardware addresses, and the IBM PCjr does not operate through direct memory access (DMA). Since the IBM PCjr 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.

<table>
<thead>
<tr>
<th>PCjr Special Functions</th>
<th>Required Key Combinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shift screen to the left</td>
<td>Alt + Ctrl + cursor left</td>
</tr>
<tr>
<td>Shift screen to the right</td>
<td>Alt + Ctrl + cursor right</td>
</tr>
<tr>
<td>Audio Feedback (System clicks when a key is pressed.)</td>
<td>Alt + Ctrl + Caps Lock</td>
</tr>
<tr>
<td>Customer Diagnostics</td>
<td>Alt + Ctrl + Ins</td>
</tr>
</tbody>
</table>

**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:
<table>
<thead>
<tr>
<th>Modes</th>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alphanumeric:</td>
<td></td>
</tr>
<tr>
<td>40x25 BW</td>
<td>None</td>
</tr>
<tr>
<td>40x25 Color</td>
<td>None</td>
</tr>
<tr>
<td>80x25 Color</td>
<td>Note</td>
</tr>
<tr>
<td>80x25 BW</td>
<td>None</td>
</tr>
<tr>
<td>Graphics:</td>
<td></td>
</tr>
<tr>
<td>320x200 4 Color</td>
<td>None</td>
</tr>
<tr>
<td>320x200 BW</td>
<td>None</td>
</tr>
<tr>
<td>640x200 BW</td>
<td>None</td>
</tr>
</tbody>
</table>

**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.

<table>
<thead>
<tr>
<th>Modes</th>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphics:</td>
<td></td>
</tr>
<tr>
<td>320x200 16 Color</td>
<td>Note</td>
</tr>
<tr>
<td>640x200 4 Color</td>
<td>Note</td>
</tr>
<tr>
<td>160x200 16 Color</td>
<td>None</td>
</tr>
</tbody>
</table>

**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 4-16 Hardware Differences
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 PCjr 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 PCjr. 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 PCjr Internal Modem

The IBM PCjr 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 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. Other hardware differences on the PCjr 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

\[
\begin{align*}
\text{hex FF} & = \text{the IBM Personal Computer} \\
\text{hex FE} & = \text{the IBM Personal Computer XT} \\
\text{hex FD} & = \text{the IBM PCjr}
\end{align*}
\]

Once determined, dual paths would handle any differences.
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
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:

5-4 ROM BIOS
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.
<table>
<thead>
<tr>
<th>Address (Hex)</th>
<th>Interrupt Number</th>
<th>Name</th>
<th>BIOS Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-3</td>
<td>0</td>
<td>Divide by Zero</td>
<td>D_EOI</td>
</tr>
<tr>
<td>4-7</td>
<td>1</td>
<td>Single Step</td>
<td>D_EOI</td>
</tr>
<tr>
<td>8-B</td>
<td>2</td>
<td>Keyboard NMI</td>
<td>KBDNMI</td>
</tr>
<tr>
<td>C-F</td>
<td>3</td>
<td>Breakpoint</td>
<td>D_EOI</td>
</tr>
<tr>
<td>10-13</td>
<td>4</td>
<td>Overflow</td>
<td>D_EOI</td>
</tr>
<tr>
<td>14-17</td>
<td>5</td>
<td>Print Screen</td>
<td>PRINT_SCREEN</td>
</tr>
<tr>
<td>18-1B</td>
<td>6</td>
<td>Reserved</td>
<td>D_EOI</td>
</tr>
<tr>
<td>1D-1F</td>
<td>7</td>
<td>Reserved</td>
<td>D_EOI</td>
</tr>
<tr>
<td>20-23</td>
<td>8</td>
<td>Time of Day</td>
<td>TIMER_INT</td>
</tr>
<tr>
<td>24-27</td>
<td>9</td>
<td>Keyboard</td>
<td>KB_INT</td>
</tr>
<tr>
<td>28-2B</td>
<td>A</td>
<td>Reserved</td>
<td>D_EOI</td>
</tr>
<tr>
<td>2C-2F</td>
<td>B</td>
<td>Communications</td>
<td>D_EOI</td>
</tr>
<tr>
<td>30-33</td>
<td>C</td>
<td>Communications</td>
<td>D_EOI</td>
</tr>
<tr>
<td>34-37</td>
<td>D</td>
<td>Vertical retrace</td>
<td>D_EOI</td>
</tr>
<tr>
<td>38-3B</td>
<td>E</td>
<td>Diskette Error</td>
<td>DISK_INT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Handler</td>
<td></td>
</tr>
<tr>
<td>3C-3F</td>
<td>F</td>
<td>Printer</td>
<td>D_EOI</td>
</tr>
<tr>
<td>40-43</td>
<td>10</td>
<td>Video</td>
<td>VIDEO_IO</td>
</tr>
<tr>
<td>44-47</td>
<td>11</td>
<td>Equipment Check</td>
<td>EQUIPMENT</td>
</tr>
<tr>
<td>48-4B</td>
<td>12</td>
<td>Memory</td>
<td>MEMORY_SIZE_</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DETERMINE</td>
</tr>
<tr>
<td>4C-4F</td>
<td>13</td>
<td>Diskette</td>
<td>DISKETTE_IO</td>
</tr>
<tr>
<td>50-53</td>
<td>14</td>
<td>Communications</td>
<td>RS232_IO</td>
</tr>
<tr>
<td>54-57</td>
<td>15</td>
<td>Cassette</td>
<td>CASSETTE_IO</td>
</tr>
<tr>
<td>58-5B</td>
<td>16</td>
<td>Keyboard</td>
<td>KEYBOARD_IO</td>
</tr>
<tr>
<td>5C-5F</td>
<td>17</td>
<td>Printer</td>
<td>PRINTER_IO</td>
</tr>
<tr>
<td>60-63</td>
<td>18</td>
<td>Resident BASIC</td>
<td>F600:0000</td>
</tr>
<tr>
<td>64-67</td>
<td>19</td>
<td>Bootstrap</td>
<td>BOOT.Usuario</td>
</tr>
<tr>
<td>68-6B</td>
<td>1A</td>
<td>Time of Day</td>
<td>TIME_OF_DAY</td>
</tr>
<tr>
<td>6C-6F</td>
<td>1B</td>
<td>Keyboard Break</td>
<td>DUMMY_RETURN</td>
</tr>
<tr>
<td>70-73</td>
<td>1C</td>
<td>Timer Tick</td>
<td>DUMMY_RETURN</td>
</tr>
<tr>
<td>74-77</td>
<td>1D</td>
<td>Video</td>
<td>VIDEO_PARMS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Initialization</td>
<td></td>
</tr>
<tr>
<td>78-7B</td>
<td>1E</td>
<td>Diskette</td>
<td>DISK_BASE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Parameters</td>
<td></td>
</tr>
<tr>
<td>7C-7F</td>
<td>1F</td>
<td>Video Graphics</td>
<td>CRT_CHARH</td>
</tr>
</tbody>
</table>
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.
<table>
<thead>
<tr>
<th>Address (Hex)</th>
<th>Interrupt (Hex)</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>80-83</td>
<td>20</td>
<td>DOS Program Terminate</td>
</tr>
<tr>
<td>84-87</td>
<td>21</td>
<td>DOS Function Call</td>
</tr>
<tr>
<td>88-8B</td>
<td>22</td>
<td>DOS Terminate Address</td>
</tr>
<tr>
<td>8C-8F</td>
<td>23</td>
<td>DOS Ctrl Break Exit Address</td>
</tr>
<tr>
<td>90-93</td>
<td>24</td>
<td>DOS Fatal Error Vector</td>
</tr>
<tr>
<td>94-97</td>
<td>25</td>
<td>DOS Absolute Disk Read</td>
</tr>
<tr>
<td>98-9B</td>
<td>26</td>
<td>DOS Absolute Disk Write</td>
</tr>
<tr>
<td>9C-9F</td>
<td>27</td>
<td>DOS Terminate, Fix in Storage</td>
</tr>
<tr>
<td>A0-FF</td>
<td>28-3F</td>
<td>Reserved for DOS</td>
</tr>
<tr>
<td>100-115</td>
<td>40-43</td>
<td>Reserved for BIOS</td>
</tr>
<tr>
<td>116-119</td>
<td>44</td>
<td>First 128 Graphics Characters</td>
</tr>
<tr>
<td>120-131</td>
<td>45-47</td>
<td>Reserves for BIOS</td>
</tr>
<tr>
<td>132-135</td>
<td>48</td>
<td>Cordless-Keyboard Translation</td>
</tr>
<tr>
<td>136-139</td>
<td>49</td>
<td>Non-keyboard Scan-code</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Translation Table</td>
</tr>
<tr>
<td>140-17F</td>
<td>50-5F</td>
<td>Reserved for BIOS</td>
</tr>
<tr>
<td>100-17F</td>
<td>40-5F</td>
<td>Reserved for BIOS</td>
</tr>
<tr>
<td>180-19F</td>
<td>60-67</td>
<td>Reserved for User Software</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Interrupts</td>
</tr>
<tr>
<td>1A0-1FF</td>
<td>68-7F</td>
<td>Reserved</td>
</tr>
<tr>
<td>200-217</td>
<td>80-85</td>
<td>Reserved for Basic</td>
</tr>
<tr>
<td>218-3C3</td>
<td>86-F0</td>
<td>Used by Basic Interpreter while BASIC is running</td>
</tr>
<tr>
<td>3C4-3FF</td>
<td>F1-FF</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

**BIOS, BASIC, and DOS Reserved Interrupts**

The following is a list of reserved memory locations.
<table>
<thead>
<tr>
<th>Address (Hex)</th>
<th>Mode</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>400-48F</td>
<td>ROM BIOS</td>
<td>See BIOS Listing</td>
</tr>
<tr>
<td>490-4EF</td>
<td></td>
<td>Reserved for System Usage</td>
</tr>
<tr>
<td>500-5FF</td>
<td></td>
<td>Communication Area for any application</td>
</tr>
<tr>
<td>500</td>
<td>DOS</td>
<td>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,</td>
</tr>
<tr>
<td>504</td>
<td>DOS</td>
<td>Single Drive Mode Status Byte</td>
</tr>
<tr>
<td>510-511</td>
<td>BASIC</td>
<td>BASIC's segment Address Store</td>
</tr>
<tr>
<td>512-515</td>
<td>BASIC</td>
<td>Clock Interrupt Vector Segment: Offset Store</td>
</tr>
<tr>
<td>516-519</td>
<td>BASIC</td>
<td>Break key Interrupt Vector Segment: Offset Store</td>
</tr>
<tr>
<td>51A-51D</td>
<td>BASIC</td>
<td>Disk Error Interrupt Vector Segment: Offset Store</td>
</tr>
</tbody>
</table>

Reserved Memory Locations

The following is a list of the BASIC workspace variables.
If you do DEF SEG (Default workspace segment):

<table>
<thead>
<tr>
<th>If you do DEF SEG (Default workspace segment):</th>
<th>Offset (Hex)</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line number of current line being executed</td>
<td>2E</td>
<td>2</td>
</tr>
<tr>
<td>Line number of last error</td>
<td>347</td>
<td>2</td>
</tr>
<tr>
<td>Offset into segment of start of program text</td>
<td>30</td>
<td>2</td>
</tr>
<tr>
<td>Offset into segment of start of variables</td>
<td>358</td>
<td>2</td>
</tr>
<tr>
<td>(end of program text 1-1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Keyboard buffer contents</td>
<td>6A</td>
<td>1</td>
</tr>
<tr>
<td>if 0-no characters in buffer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>if 1-characters in buffer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Character color in graphics mode</td>
<td>4E</td>
<td>1</td>
</tr>
<tr>
<td>Set to 1, 2, or 3 to get text in colors</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 to 3.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Do not set to 0.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Default = 3)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Example

100 Print Peek (&H2E) + 256*Peek (&H2F)

) L H

(100 hex 64 hex 00

BASIC Workspace Variables

The following shows the mapping of the BIOS memory
## BIOS System Map

<table>
<thead>
<tr>
<th>Starting Address in Hex</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000</td>
<td>BIOS Interrupt Vectors</td>
</tr>
<tr>
<td>00400</td>
<td>BIOS Data Area</td>
</tr>
<tr>
<td>00500</td>
<td>User Read/Write Memory</td>
</tr>
<tr>
<td>A0000</td>
<td>Reserved for Future Video</td>
</tr>
<tr>
<td>B8000</td>
<td>Reserved for Video</td>
</tr>
<tr>
<td>C0000</td>
<td>Reserved for Future I/O ROM</td>
</tr>
<tr>
<td>D0000</td>
<td>Reserved for Cartridges</td>
</tr>
<tr>
<td>E0000</td>
<td>Reserved for Cartridges</td>
</tr>
<tr>
<td>F0000</td>
<td>BIOS/Diagnostics/Cassette and BASIC Program Area</td>
</tr>
</tbody>
</table>
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:

5-20 Other Memory Usage
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.

<table>
<thead>
<tr>
<th>Key Position</th>
<th>Keyboard Characters</th>
<th>Make Code (Hex)</th>
<th>Break Code (Hex)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ESC</td>
<td>1</td>
<td>81</td>
</tr>
<tr>
<td>2</td>
<td>1/!</td>
<td>2</td>
<td>82</td>
</tr>
<tr>
<td>3</td>
<td>2/∂</td>
<td>3</td>
<td>83</td>
</tr>
<tr>
<td>4</td>
<td>3/#</td>
<td>4</td>
<td>84</td>
</tr>
<tr>
<td>5</td>
<td>4/$</td>
<td>5</td>
<td>85</td>
</tr>
<tr>
<td>6</td>
<td>5/%</td>
<td>6</td>
<td>86</td>
</tr>
<tr>
<td>7</td>
<td>6/∟</td>
<td>7</td>
<td>87</td>
</tr>
<tr>
<td>8</td>
<td>7/&amp;</td>
<td>8</td>
<td>88</td>
</tr>
<tr>
<td>9</td>
<td>8/*</td>
<td>9</td>
<td>89</td>
</tr>
<tr>
<td>10</td>
<td>9/(</td>
<td>A</td>
<td>8A</td>
</tr>
<tr>
<td>11</td>
<td>0/()</td>
<td>B</td>
<td>8B</td>
</tr>
<tr>
<td>12</td>
<td>-/—</td>
<td>C</td>
<td>8C</td>
</tr>
<tr>
<td>13</td>
<td>=/+</td>
<td>D</td>
<td>8D</td>
</tr>
<tr>
<td>14</td>
<td>BS&lt;___</td>
<td>E</td>
<td>8E</td>
</tr>
<tr>
<td>15</td>
<td>FN</td>
<td>54</td>
<td>D4</td>
</tr>
<tr>
<td>16</td>
<td>TAB</td>
<td>F</td>
<td>8F</td>
</tr>
<tr>
<td>17</td>
<td>q/Q</td>
<td>10</td>
<td>90</td>
</tr>
<tr>
<td>18</td>
<td>w/W</td>
<td>11</td>
<td>91</td>
</tr>
<tr>
<td>19</td>
<td>e/E</td>
<td>12</td>
<td>92</td>
</tr>
<tr>
<td>20</td>
<td>r/R</td>
<td>13</td>
<td>93</td>
</tr>
<tr>
<td>21</td>
<td>t/T</td>
<td>14</td>
<td>94</td>
</tr>
<tr>
<td>22</td>
<td>y/Y</td>
<td>15</td>
<td>95</td>
</tr>
<tr>
<td>23</td>
<td>u/U</td>
<td>16</td>
<td>96</td>
</tr>
<tr>
<td>24</td>
<td>i/I</td>
<td>17</td>
<td>97</td>
</tr>
<tr>
<td>25</td>
<td>o/O</td>
<td>18</td>
<td>98</td>
</tr>
<tr>
<td>26</td>
<td>p/P</td>
<td>19</td>
<td>99</td>
</tr>
<tr>
<td>27</td>
<td>[/{</td>
<td>1A</td>
<td>9A</td>
</tr>
<tr>
<td>28</td>
<td>}]}</td>
<td>1B</td>
<td>9B</td>
</tr>
<tr>
<td>29</td>
<td>ENTER</td>
<td>1C</td>
<td>9C</td>
</tr>
<tr>
<td>30</td>
<td>CTRL</td>
<td>1D</td>
<td>9D</td>
</tr>
<tr>
<td>31</td>
<td>a/A</td>
<td>1E</td>
<td>9E</td>
</tr>
</tbody>
</table>

Cordless Keyboard Maxtrix Scan Codes (Part 1 of 2)
<table>
<thead>
<tr>
<th>Key Position</th>
<th>Keyboard Characters</th>
<th>Make Code (Hex)</th>
<th>Break Code (Hex)</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>s/S</td>
<td>1F</td>
<td>9F</td>
</tr>
<tr>
<td>33</td>
<td>d/D</td>
<td>20</td>
<td>A0</td>
</tr>
<tr>
<td>34</td>
<td>f/F</td>
<td>21</td>
<td>A1</td>
</tr>
<tr>
<td>35</td>
<td>g/G</td>
<td>22</td>
<td>A2</td>
</tr>
<tr>
<td>36</td>
<td>h/H</td>
<td>23</td>
<td>A3</td>
</tr>
<tr>
<td>37</td>
<td>j/J</td>
<td>24</td>
<td>A4</td>
</tr>
<tr>
<td>38</td>
<td>k/K</td>
<td>25</td>
<td>A5</td>
</tr>
<tr>
<td>39</td>
<td>l/L</td>
<td>26</td>
<td>A6</td>
</tr>
<tr>
<td>40</td>
<td>;/:</td>
<td>27</td>
<td>A7</td>
</tr>
<tr>
<td>41</td>
<td>&quot;/&quot;</td>
<td>28</td>
<td>A8</td>
</tr>
<tr>
<td>42</td>
<td>CUR.UP</td>
<td>48</td>
<td>C8</td>
</tr>
<tr>
<td>43</td>
<td>L.F.SHIFT</td>
<td>2A</td>
<td>AA</td>
</tr>
<tr>
<td>44</td>
<td>z/Z</td>
<td>2C</td>
<td>AC</td>
</tr>
<tr>
<td>45</td>
<td>x/X</td>
<td>2D</td>
<td>AD</td>
</tr>
<tr>
<td>46</td>
<td>c/C</td>
<td>2E</td>
<td>AE</td>
</tr>
<tr>
<td>47</td>
<td>v/V</td>
<td>2F</td>
<td>AF</td>
</tr>
<tr>
<td>48</td>
<td>b/B</td>
<td>30</td>
<td>B0</td>
</tr>
<tr>
<td>49</td>
<td>n/N</td>
<td>31</td>
<td>B1</td>
</tr>
<tr>
<td>50</td>
<td>m/M</td>
<td>32</td>
<td>B2</td>
</tr>
<tr>
<td>51</td>
<td>,/&lt;</td>
<td>33</td>
<td>B3</td>
</tr>
<tr>
<td>52</td>
<td>./&gt;</td>
<td>34</td>
<td>B4</td>
</tr>
<tr>
<td>53</td>
<td>//?</td>
<td>35</td>
<td>B5</td>
</tr>
<tr>
<td>54</td>
<td>RT.SHIFT</td>
<td>36</td>
<td>B6</td>
</tr>
<tr>
<td>55</td>
<td>CUR.LF.</td>
<td>4B</td>
<td>CB</td>
</tr>
<tr>
<td>56</td>
<td>CUR.RT.</td>
<td>4D</td>
<td>CD</td>
</tr>
<tr>
<td>57</td>
<td>ALT.</td>
<td>38</td>
<td>B8</td>
</tr>
<tr>
<td>58</td>
<td>SP.BAR</td>
<td>39</td>
<td>B9</td>
</tr>
<tr>
<td>59</td>
<td>CAPS LOCK</td>
<td>3A</td>
<td>BA</td>
</tr>
<tr>
<td>60</td>
<td>INSERT</td>
<td>52</td>
<td>D2</td>
</tr>
<tr>
<td>61</td>
<td>DELETE</td>
<td>53</td>
<td>D3</td>
</tr>
<tr>
<td>62</td>
<td>CUR.DWN.</td>
<td>50</td>
<td>D0</td>
</tr>
<tr>
<td>Phantom-Key Scan Code</td>
<td></td>
<td>55</td>
<td></td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>IBM Personal Computers 83-key Keyboard Function</th>
<th>IBM PCjr Cordless Keyboard Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1-F10</td>
<td>Function key + 1-0 (F1-F10)</td>
</tr>
<tr>
<td>Ctrl Break</td>
<td>Function key + B (Break)</td>
</tr>
<tr>
<td>Ctrl PrtSc (Echo Print)</td>
<td>Function key + E (Echo)</td>
</tr>
<tr>
<td>Shift PrtSc (Print Screen)</td>
<td>Function key + P (PrtSc)</td>
</tr>
<tr>
<td>Ctrl NumLock (Pause)</td>
<td>Function key + Q (Pause)</td>
</tr>
<tr>
<td>Scroll Lock</td>
<td>Function key + S (ScLock)</td>
</tr>
<tr>
<td>Numeric keypad region:</td>
<td>Alt + Function key + N (1</td>
</tr>
<tr>
<td>Num Lock (Number keypad 1 through 10 becomes key</td>
<td>through 0 becomes numeric-key</td>
</tr>
<tr>
<td>scan codes.)</td>
<td>scan-codes)</td>
</tr>
<tr>
<td>PgUp key</td>
<td>Function key + cursor left</td>
</tr>
<tr>
<td></td>
<td>(PgUp)</td>
</tr>
<tr>
<td>PgDn key</td>
<td>Function key + cursor right</td>
</tr>
<tr>
<td></td>
<td>(PgDn)</td>
</tr>
<tr>
<td>Home key</td>
<td>Function key + cursor up</td>
</tr>
<tr>
<td></td>
<td>(Home)</td>
</tr>
<tr>
<td>End key</td>
<td>Function key + cursor down</td>
</tr>
<tr>
<td></td>
<td>(End)</td>
</tr>
<tr>
<td>Numeric keypad – sign</td>
<td>Function key plus the – sign</td>
</tr>
<tr>
<td>Numeric keypad + sign</td>
<td>Function key + = sign</td>
</tr>
<tr>
<td>\ key</td>
<td>Alt + /</td>
</tr>
<tr>
<td>’ key</td>
<td>Alt + ’</td>
</tr>
<tr>
<td>! key</td>
<td>Alt + [</td>
</tr>
<tr>
<td>~ key</td>
<td>Alt + ]</td>
</tr>
<tr>
<td>* with PrtSc</td>
<td>Alt +.</td>
</tr>
<tr>
<td>Numeric keypad .</td>
<td>Shift + Del</td>
</tr>
<tr>
<td>All 256 extended codes:</td>
<td>NumLock then Alt + numeric</td>
</tr>
<tr>
<td>Alt + numeric value</td>
<td>value (1 through 0)</td>
</tr>
<tr>
<td>from numeric keypad</td>
<td></td>
</tr>
</tbody>
</table>

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.
<table>
<thead>
<tr>
<th>Key Number</th>
<th>Base Case</th>
<th>Upper Case</th>
<th>Ctrl</th>
<th>Alt</th>
<th>Fn</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Esc</td>
<td>Esc</td>
<td>Esc</td>
<td>−1</td>
<td>**</td>
</tr>
<tr>
<td>2</td>
<td>!</td>
<td>−1</td>
<td><em>,</em>*</td>
<td>(F1)</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>Nul (000)</td>
<td><em>,</em>*</td>
<td>(F2)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>#</td>
<td>−1</td>
<td><em>,</em>*</td>
<td>(F3)</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>$</td>
<td>−1</td>
<td><em>,</em>*</td>
<td>(F4)</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>%</td>
<td>−1</td>
<td><em>,</em>*</td>
<td>(F5)</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>△</td>
<td>RSO (030)</td>
<td><em>,</em>*</td>
<td>(F6)</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>&amp;</td>
<td>−1</td>
<td><em>,</em>*</td>
<td>(F7)</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>*</td>
<td>−1</td>
<td><em>,</em>*</td>
<td>(F8)</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>(</td>
<td>−1</td>
<td><em>,</em>*</td>
<td>(F9)</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>)</td>
<td>−1</td>
<td><em>,</em>*</td>
<td>(F10)</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>US (031)</td>
<td>*</td>
<td>***</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>=</td>
<td>*</td>
<td>***</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Backspace (008)</td>
<td>DEL (127)</td>
<td>−1</td>
<td>−1</td>
<td></td>
</tr>
<tr>
<td>15 Fn</td>
<td>−1</td>
<td>−1</td>
<td>−1</td>
<td>−1</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td>&lt;—</td>
<td>Q</td>
<td>*</td>
<td>**</td>
</tr>
<tr>
<td>17</td>
<td>q</td>
<td>Q</td>
<td>*</td>
<td>***</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>w</td>
<td>W</td>
<td>*</td>
<td>−1</td>
<td>(Echo)</td>
</tr>
<tr>
<td>19</td>
<td>e</td>
<td>E</td>
<td>*</td>
<td>***</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>r</td>
<td>R</td>
<td>*</td>
<td>−1</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>t</td>
<td>T</td>
<td>*</td>
<td>−1</td>
<td></td>
</tr>
</tbody>
</table>

* - 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)
<table>
<thead>
<tr>
<th>Key Number</th>
<th>Base Case</th>
<th>Upper Case</th>
<th>Ctrl</th>
<th>Alt</th>
<th>Fn</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>y</td>
<td>Y</td>
<td>EM (025)</td>
<td>*</td>
<td>-1</td>
</tr>
<tr>
<td>23</td>
<td>u</td>
<td>U</td>
<td>NAK (021)</td>
<td>*</td>
<td>-1</td>
</tr>
<tr>
<td>24</td>
<td>i</td>
<td>I</td>
<td>HT (009)</td>
<td>*</td>
<td>-1</td>
</tr>
<tr>
<td>25</td>
<td>o</td>
<td>O</td>
<td>SI (015)</td>
<td>*</td>
<td>-1</td>
</tr>
<tr>
<td>26</td>
<td>p</td>
<td>P</td>
<td>DLE (016)</td>
<td>*</td>
<td>**</td>
</tr>
<tr>
<td>27</td>
<td>[</td>
<td>{</td>
<td>Esc (027)</td>
<td>(</td>
<td>)</td>
</tr>
<tr>
<td>28</td>
<td>]</td>
<td>}</td>
<td>GS (029)</td>
<td>(~)</td>
<td>-1</td>
</tr>
<tr>
<td>29</td>
<td>CR</td>
<td>CR</td>
<td>LF (010)</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>30 Ctrl</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>31</td>
<td>a</td>
<td>A</td>
<td>SOH (001)</td>
<td>*</td>
<td>-1</td>
</tr>
<tr>
<td>32</td>
<td>s</td>
<td>S</td>
<td>DC3 (019)</td>
<td>*</td>
<td>**</td>
</tr>
<tr>
<td>33</td>
<td>d</td>
<td>D</td>
<td>EOT (004)</td>
<td>*</td>
<td>-1</td>
</tr>
<tr>
<td>34</td>
<td>f</td>
<td>F</td>
<td>ACK (006)</td>
<td>*</td>
<td>-1</td>
</tr>
<tr>
<td>35</td>
<td>g</td>
<td>G</td>
<td>BELL (007)</td>
<td>*</td>
<td>-1</td>
</tr>
<tr>
<td>36</td>
<td>h</td>
<td>H</td>
<td>BS (008)</td>
<td>*</td>
<td>-1</td>
</tr>
<tr>
<td>37</td>
<td>j</td>
<td>J</td>
<td>LF (010)</td>
<td>*</td>
<td>-1</td>
</tr>
<tr>
<td>38</td>
<td>k</td>
<td>K</td>
<td>VT (011)</td>
<td>*</td>
<td>-1</td>
</tr>
<tr>
<td>39</td>
<td>l</td>
<td>L</td>
<td>FF (012)</td>
<td>*</td>
<td>-1</td>
</tr>
<tr>
<td>40</td>
<td>;</td>
<td>:</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>41</td>
<td>'</td>
<td>&quot;</td>
<td>-1</td>
<td>(‘)</td>
<td>-1</td>
</tr>
</tbody>
</table>

* - 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)
<table>
<thead>
<tr>
<th>Key Number</th>
<th>Base Case</th>
<th>Upper Case</th>
<th>Ctrl</th>
<th>Alt</th>
<th>Fn</th>
<th>Alt + Ctrl</th>
</tr>
</thead>
<tbody>
<tr>
<td>42</td>
<td>Cur.Up*</td>
<td>8 ****</td>
<td>–1</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>43 Left</td>
<td>–1</td>
<td>–1</td>
<td>–1</td>
<td>–1</td>
<td>–1</td>
<td></td>
</tr>
<tr>
<td>Shift</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>z</td>
<td>Z</td>
<td>SUB (026)</td>
<td>*</td>
<td>–1</td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>x</td>
<td>X</td>
<td>CAN (024)</td>
<td>*</td>
<td>–1</td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>c</td>
<td>C</td>
<td>EXT (003)</td>
<td>*</td>
<td>–1</td>
<td></td>
</tr>
<tr>
<td>47</td>
<td>v</td>
<td>V</td>
<td>SYN (022)</td>
<td>*</td>
<td>–1</td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>b</td>
<td>B</td>
<td>STX (002)</td>
<td>*</td>
<td><strong>,</strong>,**</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(Break)</td>
</tr>
<tr>
<td>49</td>
<td>n</td>
<td>N</td>
<td>SO (014)</td>
<td>****</td>
<td>****</td>
<td>****</td>
</tr>
<tr>
<td>50</td>
<td>m</td>
<td>M</td>
<td>CR (013)</td>
<td>*</td>
<td>–1</td>
<td></td>
</tr>
<tr>
<td>51</td>
<td>&lt;</td>
<td>&gt;</td>
<td>–1</td>
<td>–1</td>
<td>–1</td>
<td></td>
</tr>
<tr>
<td>52</td>
<td>&gt;</td>
<td>?</td>
<td>–1</td>
<td>(<em>)</em></td>
<td>–1</td>
<td></td>
</tr>
<tr>
<td>53</td>
<td>/</td>
<td>\</td>
<td>–1</td>
<td></td>
<td>–1</td>
<td></td>
</tr>
<tr>
<td>54 Right</td>
<td>–1</td>
<td>–1</td>
<td>–1</td>
<td></td>
<td>–1</td>
<td></td>
</tr>
<tr>
<td>Shift</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>55</td>
<td>Cur.L *</td>
<td>4 ****</td>
<td>*</td>
<td><strong>,</strong>,**</td>
<td>**</td>
<td>(PgUp)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Reverse Word</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>56</td>
<td>Cur.R *</td>
<td>6 ****</td>
<td>*</td>
<td><strong>,</strong>,**</td>
<td>**</td>
<td>(PgDn)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Advance Word</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* - 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)
<table>
<thead>
<tr>
<th>Key Number</th>
<th>Base Case</th>
<th>Upper Case</th>
<th>Ctrl</th>
<th>Alt</th>
<th>Fn</th>
<th>Alt + Ctrl</th>
</tr>
</thead>
<tbody>
<tr>
<td>57 Alt</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>58 Space</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>59 Caps</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lock</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60 Ins.</td>
<td>**</td>
<td>**</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>61 Del. *</td>
<td>**</td>
<td>**</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>62 Cur.Dn *</td>
<td>**</td>
<td>**</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* - 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.
<table>
<thead>
<tr>
<th>Second Code</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Null Character</td>
</tr>
<tr>
<td>15</td>
<td>Alt Q, W, E, R, T, Y, U, I, O, P</td>
</tr>
<tr>
<td>16 through 25</td>
<td>Alt A, S, D, F, G, H, J, K, L</td>
</tr>
<tr>
<td>30 through 38</td>
<td>Alt Z, X, C, V, B, N, M</td>
</tr>
<tr>
<td>44 through 50</td>
<td>Fn + 1, 2, 3, 4, 5, 6, 7, 8, 9, 0 (Functions 1 through 10)</td>
</tr>
<tr>
<td>59 through 68</td>
<td>Home</td>
</tr>
<tr>
<td>71</td>
<td>Up Arrow</td>
</tr>
<tr>
<td>72</td>
<td>Page Up</td>
</tr>
<tr>
<td>73</td>
<td>(Cursor Left)</td>
</tr>
<tr>
<td>75</td>
<td>(Cursor Right)</td>
</tr>
<tr>
<td>77</td>
<td>End</td>
</tr>
<tr>
<td>79</td>
<td>Down Arrow</td>
</tr>
<tr>
<td>80</td>
<td>Page Down</td>
</tr>
<tr>
<td>81</td>
<td>Ins (Insert)</td>
</tr>
<tr>
<td>82</td>
<td>Del (Delete)</td>
</tr>
<tr>
<td>83</td>
<td>F11 through F20 (Upper Case F1 through F10)</td>
</tr>
<tr>
<td>84 through 93</td>
<td>F21 through F30 (Ctrl F1 through F10)</td>
</tr>
<tr>
<td>94 through 103</td>
<td>F31 through F40 (Alt F1 through F10)</td>
</tr>
<tr>
<td>104 through 113</td>
<td>Fn/E or Ctrl/Fn/P (Start/Stop Echo to Printer)</td>
</tr>
<tr>
<td>114</td>
<td>Ctrl (Reverse Word)</td>
</tr>
<tr>
<td>115</td>
<td>Ctrl (Advance Word)</td>
</tr>
<tr>
<td>116</td>
<td>Ctrl [Erase End of Line (EOL)]</td>
</tr>
<tr>
<td>117</td>
<td>Ctrl/PgDn [Erase to End of Screen (EOS)]</td>
</tr>
<tr>
<td>118</td>
<td>Ctrl/Home [Clear Screen and Home]</td>
</tr>
<tr>
<td>119</td>
<td>Alt/1, 2, 3, 4, 5, 6, 7, 8, 9, 0, -, = (Keys 2 through 13)</td>
</tr>
<tr>
<td>120 through 131</td>
<td>Ctrl/PgUp (Top 25 Lines of Text and Home Cur.)</td>
</tr>
<tr>
<td>132</td>
<td>Reserved</td>
</tr>
<tr>
<td>133 through 149</td>
<td>Reserved for Non-Keyboard Scan Codes</td>
</tr>
<tr>
<td>150 through 190</td>
<td>Reserved for Non-Keyboard Scan Codes</td>
</tr>
</tbody>
</table>

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

Keyboard Encoding 5-33
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 alphanumerical 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.
<table>
<thead>
<tr>
<th>Function</th>
<th>Key Combinations</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Reset</td>
<td>Alt + Ctrl + Del</td>
<td>Unconditional system reset</td>
</tr>
<tr>
<td>Break</td>
<td>Fn + B</td>
<td>Breaks program execution</td>
</tr>
<tr>
<td>Pause</td>
<td>Fn + Q</td>
<td>Resumable pause in program execution</td>
</tr>
<tr>
<td>Print Screen</td>
<td>Fn + P</td>
<td></td>
</tr>
<tr>
<td>Function Lock</td>
<td>Fn and Shift then Esc (Held) concurrently</td>
<td>Locks the number keys as Function keys (F1-F10) and B, Q, P, E, S, and the cursor control keys to their function states</td>
</tr>
<tr>
<td>Screen Adjustment</td>
<td>Alt + Ctrl + cursor right or cursor left</td>
<td>Allows the user to adjust the display's image left or right</td>
</tr>
<tr>
<td>Keyboard Click</td>
<td>Alt + Ctrl + CapsLock</td>
<td>Enables or disables the keyboard audio feedback click</td>
</tr>
<tr>
<td>Run Diagnostics</td>
<td>Alt + Ctrl + Ins</td>
<td>Initiates system ROM diagnostics</td>
</tr>
<tr>
<td>Keyboard Adventure Game</td>
<td>Esc</td>
<td>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.</td>
</tr>
<tr>
<td>Cassette Autoload</td>
<td>Ctrl + Esc</td>
<td>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.</td>
</tr>
</tbody>
</table>

Cordless Keyboard Special Handling

Keyboard Encoding 5–37
### Keyboard Usage

“Keyboard Usage” is a set of guidelines of key-usage when performing commonly-used functions.

<table>
<thead>
<tr>
<th>Function</th>
<th>Keys</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Home Cursor</td>
<td>Fn Home</td>
<td>Editors; word processors</td>
</tr>
<tr>
<td>Return to outermost menu</td>
<td>Fn Home</td>
<td>Menu driven applications</td>
</tr>
<tr>
<td>Move cursor up</td>
<td>Up Arrow</td>
<td>Full screen editor, word processor</td>
</tr>
<tr>
<td>Page up, scroll backwards 25 lines</td>
<td>Fn PgUp</td>
<td>Editors; word processors</td>
</tr>
<tr>
<td>Move cursor left</td>
<td></td>
<td>Text, command entry</td>
</tr>
<tr>
<td>Move cursor right</td>
<td></td>
<td>Text, command entry</td>
</tr>
<tr>
<td>Scroll to end of text place cursor at end of line</td>
<td>Fn End</td>
<td>Editors; word processors</td>
</tr>
<tr>
<td>Move cursor down</td>
<td>Down Arrow</td>
<td>Full screen editor, word processor</td>
</tr>
<tr>
<td>Page down, scroll forwards 25 lines and home</td>
<td>Fn PgDn</td>
<td>Editors; word processors</td>
</tr>
<tr>
<td>Start/Stop insert text at cursor, shift text right in buffer</td>
<td>Ins</td>
<td>Text, command entry</td>
</tr>
</tbody>
</table>

**Keyboard - Commonly Used Functions (Part 1 of 3)**

5-38 Keyboard Encoding
<table>
<thead>
<tr>
<th>Function</th>
<th>Keys</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delete character at cursor</td>
<td>Del</td>
<td>Text, command entry</td>
</tr>
<tr>
<td>Destructive backspace</td>
<td>← Key 14</td>
<td>Text, command entry</td>
</tr>
<tr>
<td>Tab forward</td>
<td>→</td>
<td>Text entry</td>
</tr>
<tr>
<td>Tab reverse</td>
<td>←</td>
<td>Text entry</td>
</tr>
<tr>
<td>Clear screen and home</td>
<td>Ctrl Fn Home</td>
<td>Text entry</td>
</tr>
<tr>
<td>Scroll up</td>
<td>Up Arrow</td>
<td>In scroll lock mode</td>
</tr>
<tr>
<td>Scroll down</td>
<td>Down Arrow</td>
<td>In scroll lock mode</td>
</tr>
<tr>
<td>Scroll left</td>
<td>←</td>
<td>In scroll lock mode</td>
</tr>
<tr>
<td>Scroll right</td>
<td>←</td>
<td>In scroll lock mode</td>
</tr>
<tr>
<td>Delete from cursor to EOL (end of line)</td>
<td>Ctrl Fn End</td>
<td>Text, command entry</td>
</tr>
<tr>
<td>Exit/Escape</td>
<td>Esc</td>
<td>Editor, 1 level of menu and so on</td>
</tr>
<tr>
<td>Start/Stop Echo screen to printer</td>
<td>Fn PrtSc</td>
<td>Any time</td>
</tr>
<tr>
<td>Delete from cursor to EOS (end of screen)</td>
<td>Ctrl Fn PgDn</td>
<td>Text, command entry</td>
</tr>
<tr>
<td>Advance word</td>
<td>Ctrl →</td>
<td>Text entry</td>
</tr>
<tr>
<td>Reverse word</td>
<td>Ctrl ←</td>
<td>Text entry</td>
</tr>
<tr>
<td>Window Right</td>
<td>Ctrl →</td>
<td>When text is too wide to fit the screen</td>
</tr>
</tbody>
</table>

*Keyboard - Commonly Used Functions (Part 2 of 3)*
<table>
<thead>
<tr>
<th>Function</th>
<th>Keys</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Window Left</td>
<td>Ctrl ←</td>
<td>When text is too wide to fit the screen</td>
</tr>
<tr>
<td>Enter insert mode</td>
<td>Ins</td>
<td>Line Editor</td>
</tr>
<tr>
<td>Exit insert mode</td>
<td>Ins</td>
<td>Line Editor</td>
</tr>
<tr>
<td>Cancel current line</td>
<td>Esc</td>
<td>Command entry, text entry</td>
</tr>
<tr>
<td>Suspend system (Pause)</td>
<td>Ctrl Fn Pause</td>
<td>Stop list, stop program, and so on. Resumes on any key.</td>
</tr>
<tr>
<td>Break interrupt</td>
<td>Fn Break</td>
<td>Interrupt current process</td>
</tr>
<tr>
<td>System reset</td>
<td>Alt Ctrl Del</td>
<td>Reboot</td>
</tr>
<tr>
<td>Top of document and home cursor</td>
<td>Ctrl Fn PgUp</td>
<td>Editors, word processors</td>
</tr>
<tr>
<td>Standard function keys</td>
<td>Shift Fn/Fn through Fn/Fn/F10</td>
<td>Primary function keys</td>
</tr>
<tr>
<td>Secondary function keys</td>
<td>Shift F1-F10 Ctrl F1-F10 Alt F1-F10</td>
<td>Extra function keys if 10 are not sufficient.</td>
</tr>
<tr>
<td>Extra function keys</td>
<td>Alt keys 2 through 13 (1 through 9, 0) (-, =)</td>
<td>Line Editor</td>
</tr>
<tr>
<td>Extra function keys</td>
<td>Alt A through Z</td>
<td>Used when function starts with the same letter as one of the alpha keys.</td>
</tr>
</tbody>
</table>

**Keyboard - Commonly Used Functions (Part 3 of 3)**

5-40 Keyboard Encoding
<table>
<thead>
<tr>
<th>Function</th>
<th>Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Carriage return</td>
<td>⏯ (Enter)</td>
</tr>
<tr>
<td>Line feed</td>
<td>Ctrl ⏯ (Enter)</td>
</tr>
<tr>
<td>Bell</td>
<td>Ctrl G</td>
</tr>
<tr>
<td>Home</td>
<td>Fn Home</td>
</tr>
<tr>
<td>Cursor up</td>
<td>Up Arrow</td>
</tr>
<tr>
<td>Cursor down</td>
<td>Down Arrow</td>
</tr>
<tr>
<td>Cursor left</td>
<td></td>
</tr>
<tr>
<td>Cursor right</td>
<td></td>
</tr>
<tr>
<td>Advance one word</td>
<td>Ctrl ⏯</td>
</tr>
<tr>
<td>Reverse one word</td>
<td>Ctrl ⏯</td>
</tr>
<tr>
<td>Insert</td>
<td>Ins</td>
</tr>
<tr>
<td>Delete</td>
<td>Del</td>
</tr>
<tr>
<td>Clear screen</td>
<td>Ctrl Fn Home</td>
</tr>
<tr>
<td>Freeze output</td>
<td>Fn Pause</td>
</tr>
<tr>
<td>Tab advance</td>
<td></td>
</tr>
<tr>
<td>Stop Execution (break)</td>
<td>Fn Break</td>
</tr>
<tr>
<td>Delete current line</td>
<td>Esc</td>
</tr>
<tr>
<td>Delete to end of line</td>
<td>Ctrl Fn End</td>
</tr>
<tr>
<td>Position cursor to end of line</td>
<td>Fn End</td>
</tr>
</tbody>
</table>

**BASIC Screen Editor Special Functions**
<table>
<thead>
<tr>
<th>Function</th>
<th>Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Suspend</td>
<td>Fn Pause</td>
</tr>
<tr>
<td>Echo to printer</td>
<td>Fn Echo</td>
</tr>
<tr>
<td>Stop echo to printer</td>
<td>Fn Echo</td>
</tr>
<tr>
<td>Exit current function (break)</td>
<td>Fn Break</td>
</tr>
<tr>
<td>Backspace</td>
<td>← Key 14</td>
</tr>
<tr>
<td>Line feed</td>
<td>Ctrl ⌘ (Enter)</td>
</tr>
<tr>
<td>Cancel line</td>
<td>Esc</td>
</tr>
<tr>
<td>Copy character</td>
<td>Fn F1 or →</td>
</tr>
<tr>
<td>Copy until match</td>
<td>Fn F2</td>
</tr>
<tr>
<td>Copy remaining</td>
<td>Fn F3</td>
</tr>
<tr>
<td>Skip character</td>
<td>Del</td>
</tr>
<tr>
<td>Skip until match</td>
<td>Fn F4</td>
</tr>
<tr>
<td>Enter insert mode</td>
<td>Ins</td>
</tr>
<tr>
<td>Exit insert mode</td>
<td>Ins</td>
</tr>
<tr>
<td>Make new line the template</td>
<td>Fn F5</td>
</tr>
<tr>
<td>String separator in REPLACE</td>
<td>Fn F6</td>
</tr>
<tr>
<td>End of file in keyboard input</td>
<td>Fn F6</td>
</tr>
</tbody>
</table>

**DOS Special Functions**

### Non-Keyboard Scan-code Architecture

The architecture of the IBM PCjr 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 outside 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**

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Length - The number of non-keyboard scan-codes that are mapped within the table (from 1 to n).</td>
</tr>
<tr>
<td>1 to n</td>
<td>Word with low-order byte representing the scan-code-mapped values relative to the input values in the range of hex 56 thru 7E.</td>
</tr>
</tbody>
</table>
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.
### Input Scan Code

<table>
<thead>
<tr>
<th>Input Scan Code</th>
<th>Mapped Value</th>
<th>Keyboard Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>86</td>
<td>72</td>
<td>(cursor up)</td>
</tr>
<tr>
<td>87</td>
<td>73</td>
<td>PgUp</td>
</tr>
<tr>
<td>88</td>
<td>77</td>
<td>(cursor right)</td>
</tr>
<tr>
<td>89</td>
<td>81</td>
<td>PgDn</td>
</tr>
<tr>
<td>90</td>
<td>80</td>
<td>(cursor down)</td>
</tr>
<tr>
<td>91</td>
<td>79</td>
<td>End</td>
</tr>
<tr>
<td>92</td>
<td>75</td>
<td>(cursor left)</td>
</tr>
<tr>
<td>93</td>
<td>71</td>
<td>Home</td>
</tr>
<tr>
<td>94</td>
<td>57</td>
<td>Space</td>
</tr>
<tr>
<td>95</td>
<td>28</td>
<td>Enter</td>
</tr>
<tr>
<td>96</td>
<td>17</td>
<td>W</td>
</tr>
<tr>
<td>97</td>
<td>18</td>
<td>E</td>
</tr>
<tr>
<td>98</td>
<td>31</td>
<td>S</td>
</tr>
<tr>
<td>99</td>
<td>45</td>
<td>X</td>
</tr>
<tr>
<td>100</td>
<td>44</td>
<td>Z</td>
</tr>
<tr>
<td>101</td>
<td>43</td>
<td>\</td>
</tr>
<tr>
<td>102</td>
<td>30</td>
<td>A</td>
</tr>
<tr>
<td>103</td>
<td>16</td>
<td>Q</td>
</tr>
<tr>
<td>104</td>
<td>15</td>
<td>Tab</td>
</tr>
<tr>
<td>105</td>
<td>1</td>
<td>Esc</td>
</tr>
</tbody>
</table>

### Translate Table Default Values

<table>
<thead>
<tr>
<th>Scan Codes (Hex)</th>
<th>Type of Scan Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - 55</td>
<td>Normal Keyboard Scan Code (Make)</td>
</tr>
<tr>
<td>56 - 7E</td>
<td>Non-Keyboard Scan Code (Make)</td>
</tr>
<tr>
<td>81 - D5</td>
<td>Normal Keyboard Scan Code (Break)</td>
</tr>
<tr>
<td>D6 - FE</td>
<td>Non-Keyboard Scan Code (Break)</td>
</tr>
<tr>
<td>FF</td>
<td>Keyboard Buffer Full</td>
</tr>
</tbody>
</table>

**Scan-Code Map**
Notes:
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:

<table>
<thead>
<tr>
<th>Request Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>AH = 0</td>
<td>Turn Cassette Motor On</td>
</tr>
<tr>
<td>AH = 1</td>
<td>Turn Cassette Motor Off</td>
</tr>
<tr>
<td>AH = 2</td>
<td>Read Tape Block</td>
</tr>
<tr>
<td></td>
<td>Read CX bytes into memory starting at Address DS:BX</td>
</tr>
<tr>
<td></td>
<td>Return actual number of bytes read in DX</td>
</tr>
<tr>
<td></td>
<td>Return Cassette Status in AH</td>
</tr>
<tr>
<td>AH = 3</td>
<td>Write Tape Block</td>
</tr>
<tr>
<td></td>
<td>Write CX bytes onto cassette starting at Address DS:BX</td>
</tr>
<tr>
<td></td>
<td>Return Cassette Status in AH</td>
</tr>
</tbody>
</table>

AH Request Types
<table>
<thead>
<tr>
<th>Cassette Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AH = 00</td>
<td>No Errors</td>
</tr>
<tr>
<td>AH = 01</td>
<td>Cyclic Redundancy Check (CRC) Error in Read Block</td>
</tr>
<tr>
<td>AH = 02</td>
<td>No Data Transitions</td>
</tr>
<tr>
<td>AH = 04</td>
<td>No Leader</td>
</tr>
<tr>
<td>AH = 80</td>
<td>Invalid Command</td>
</tr>
</tbody>
</table>

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
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:

<table>
<thead>
<tr>
<th>Leader</th>
<th>Sync Bit</th>
<th>Sync Byte</th>
<th>Data Block</th>
<th>CRC</th>
<th>Data Block</th>
<th>CRC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motor On</td>
<td>Motor Off</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Cassette Write-Block Format**
<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leader</td>
<td>256 Bytes (of All 1’s)</td>
</tr>
<tr>
<td>Sync Bit</td>
<td>One 0 bit</td>
</tr>
<tr>
<td>Sync Byte</td>
<td>ASCII Character hex 16</td>
</tr>
<tr>
<td>Data Blocks</td>
<td>256 Bytes in Length</td>
</tr>
<tr>
<td>CRC</td>
<td>2 Bytes for each Data Block</td>
</tr>
</tbody>
</table>

**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.
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
EQUATES

<table>
<thead>
<tr>
<th>DEC</th>
<th>EQU</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>PORT_A</td>
<td>EQU</td>
<td>60H</td>
<td>0255 PORT A ADDR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PORT_B</td>
<td>EQU</td>
<td>61H</td>
<td>0255 PORT B ADDR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PORT_C</td>
<td>EQU</td>
<td>62H</td>
<td>0255 PORT C ADDR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CMD_PORT</td>
<td>EQU</td>
<td>63H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MODE_8255</td>
<td>EQU</td>
<td>10001001B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INT80</td>
<td>EQU</td>
<td>20H</td>
<td>0255 PORT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INT81</td>
<td>EQU</td>
<td>21H</td>
<td>0255 PORT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EO1</td>
<td>EQU</td>
<td>20H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIMER</td>
<td>EQU</td>
<td>40H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIM_CTL</td>
<td>EQU</td>
<td>43H</td>
<td>0253 TIMER CONTROL PORT ADDR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIMER0</td>
<td>EQU</td>
<td>40H</td>
<td>0253 TIMER/CNTER 0 PORT ADDR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KB_CTL</td>
<td>EQU</td>
<td>61H</td>
<td>CONTROL BITS FOR KEYBOARD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VGA_CTL</td>
<td>EQU</td>
<td>3AH</td>
<td>VIDEO GATE ARRAY CONTROL PORT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NM1_PORT</td>
<td>EQU</td>
<td>0A0H</td>
<td>NMI CONTROL PORT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PORT_B0</td>
<td>EQU</td>
<td>0B0H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PAGREG</td>
<td>EQU</td>
<td>0C3FH</td>
<td>CRT/CPU PAGE REGISTER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KBPORT</td>
<td>EQU</td>
<td>0E0H</td>
<td>KEYBOARD PORT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DIAG_TABLE_PTR</td>
<td>EQU</td>
<td>4000H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MINI</td>
<td>EQU</td>
<td>2000H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

DISKETTE EQUATES

<table>
<thead>
<tr>
<th>DEC</th>
<th>EQU</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>NEC_CTL</td>
<td>EQU</td>
<td>0F2H</td>
<td>CONTROL PORT FOR THE DISKETTE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FDC_RESET</td>
<td>EQU</td>
<td>08H</td>
<td>RESETS THE NEC (FLOPPY DISK CONTROLLER). 0 RESETS, 1 RELEASES THE RESET</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WD_ENABLE</td>
<td>EQU</td>
<td>20H</td>
<td>ENABLES WATCH DOG TIMER IN NEC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WD_STROBE</td>
<td>EQU</td>
<td>40H</td>
<td>STROBES WATCHDOG TIMER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DRIVE_ENABLE</td>
<td>EQU</td>
<td>01H</td>
<td>SELECTS AND ENABLES DRIVE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NEC_STAT</td>
<td>EQU</td>
<td>0F4H</td>
<td>STATUS REGISTER FOR THE NEC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BUST_BIT</td>
<td>EQU</td>
<td>20H</td>
<td>BIT = 0 AT END OF EXECUTION PHASE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DIO</td>
<td>EQU</td>
<td>40H</td>
<td>INDICATES DIRECTION OF TRANSFER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ROM</td>
<td>EQU</td>
<td>80H</td>
<td>REQUEST FOR MASTER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MINI_DATA</td>
<td>EQU</td>
<td>0F8H</td>
<td>DATA PORT FOR THE NEC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

BOSS INTERRUPT LOCATIONS

<table>
<thead>
<tr>
<th>DEC</th>
<th>EQU</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>AB50</td>
<td>SEGMENT AT 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0008</td>
<td>ORG</td>
<td>2x4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>000B</td>
<td>ORG</td>
<td>3x4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>000C</td>
<td>ORG</td>
<td>3x4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0014</td>
<td>ORG</td>
<td>5x4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>001A</td>
<td>ORG</td>
<td>5x4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>001D</td>
<td>ORG</td>
<td>6x4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0020</td>
<td>ORG</td>
<td>1OH4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0030</td>
<td>ORG</td>
<td>1CH4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0040</td>
<td>ORG</td>
<td>1OH4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0070</td>
<td>ORG</td>
<td>1CH4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0074</td>
<td>ORG</td>
<td>10H4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0074</td>
<td>ORG</td>
<td>10H4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0074</td>
<td>ORG</td>
<td>10H4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0074</td>
<td>ORG</td>
<td>10H4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0060</td>
<td>ORG</td>
<td>18H4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0060</td>
<td>ORG</td>
<td>18H4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0078</td>
<td>ORG</td>
<td>01EH4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0078</td>
<td>ORG</td>
<td>01EH4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>007C</td>
<td>ORG</td>
<td>01FH4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>007C</td>
<td>ORG</td>
<td>01FH4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0110</td>
<td>ORG</td>
<td>048H4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0110</td>
<td>ORG</td>
<td>048H4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0120</td>
<td>ORG</td>
<td>048H4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0120</td>
<td>ORG</td>
<td>048H4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0124</td>
<td>ORG</td>
<td>049H4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0124</td>
<td>ORG</td>
<td>049H4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0124</td>
<td>ORG</td>
<td>049H4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0124</td>
<td>ORG</td>
<td>049H4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0160</td>
<td>ORG</td>
<td>081H4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0160</td>
<td>ORG</td>
<td>081H4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0168</td>
<td>ORG</td>
<td>089H4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0168</td>
<td>ORG</td>
<td>089H4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0224</td>
<td>ORG</td>
<td>090H4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0224</td>
<td>ORG</td>
<td>090H4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0400</td>
<td>ORG</td>
<td>0A0H4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0400</td>
<td>ORG</td>
<td>0A0H4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0400</td>
<td>DATA_AREA</td>
<td>LABEL BYTE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0400</td>
<td>DATA_Word</td>
<td>LABEL Word</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0700</td>
<td>ORG</td>
<td>07C00H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0700</td>
<td>ORG</td>
<td>07C00H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0700</td>
<td>ORG</td>
<td>07C00H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

(CAVEAT EMPTOR):

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.

--

--

ROM BIOS A-3

Appendix A
0000 80 [ ]

0000 STACK SEGMENT AT 30H

0000 DW 128 DUP(?)

0100 TOS LABEL WORD

0100 STACK ENDS

0000 DATA SEGMENT AT 40H

0000 04 [ ]

0000 RS232_BASE DW 4 DUP(?); ADDRESSES OF RS232 ADAPTERS

0008 04 [ ]

0008 PRINTER_BASE DW 4 DUP(?); ADDRESSES OF PRINTERS

0010 ??

0012 ??

0013 ??

0015 ??

0017 ??

0019 ??

001A ??

001E 10 [ ]

001E 10 [ ]

003E ??

003F ??

0040 ??

0041 ??

0042 ??

0042 ??

0042 ??

0042 ??

0042 ??

A-4 ROM BIOS
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 (?)

0060 ?? CURSOR_MODE DW ? ; CURRENT CURSOR MODE SETTING
0062 ?? ACTIVE_PAGE DB ? ; CURRENT PAGE BEING DISPLAYED
0063 ?? ADDR_DBG45 DW ? ; BASE ADDRESS FOR ACTIVE DISPLAY
0065 ?? CRT_MODE_SET DB ? ; CURRENT SETTING OF THE CRT MODE REGISTER
0066 ?? CRT_PALETTE DB ? ; CURRENT PALETTE MASK SETTING

0067 ?? EDGE_CNT DW ? ; TIME COUNT AT DATA EDGE
0069 ?? CRC_REG DW ? ; CRC REGISTER
006B ?? LAST_VAL DB ? ; LAST INPUT VALUE

0071 ?? BIOS_BREAK DB ? ; BIT 7=1 IF BREAK KEY HAS BEEN HIT
0072 ?? RESET_FLAG DW ? ; WORD=1234H IF KEYBOARD RESET UNDERWAY

0074 ?? TRACK0 DB ?
0075 ?? TRACK1 DB ?
0076 ?? TRACK2 DB ?
0077 ??

0078 04 [ ?? ]

007C 04 [ ?? ]

0080 ?? BUFFER_START DW ?
0082 ?? BUFFER_END DW ?
0084 ?? INTR_FLAG DB ? ; FLAG TO INDICATE AN INTERRUPT HAPPENED

0085 ?? CUR_CHAR DB ? ; CURRENT CHARACTER FOR TYPAMATIC
0086 ?? VAL_DELAY DB ? ; DETERMINES WHEN INITIAL DELAY IS OVER

0087 ?? DELAY_RATE EQU OFH ; INCREASES INITIAL DELAY
0088 ?? CUR_FUNC DB ? ; CURRENT FUNCTION
0089 ?? KB_FLAG_2 DB ? ; 3RD BYTE OF KEYBOARD FLAGS

008A ?? PAGDAT DB ? ; IMAGE OF DATA WRITTEN TO PAGREG
008B DATA ENDS

0090 XXDATA SEGMENT AT 50H
0090 ?? STATUS_BYTE DB ? ; THE FOLLOWING AREA IS USED ONLY DURING DIAGNOSTICS

0091 ?? DCP_MENU_PAGE DB ? ; TO CURRENT PAGE FOR DIAG. MENU
0092 ?? DCP_ROW_COL DW ? ; CURRENT ROW/COLUMN COORDINATES FOR DIAG MENU
0094 ?? WRAP_FLAG DB ? ; INTERNAL/EXTERNAL 8250 WRAP INDICATOR
A-6 ROM BIOS

0005 ?? MFG_TST DB ? ; INITIALIZATION FLAG
0006 ?? MEM_TOT DW ? ; WORD EQUIV. TO HIGHEST SEGMENT IN MEMORY
0008 ?? MEM_DONE DB ? ; CURRENT SEGMENT VALUE FOR BACKGROUND MEM TEST
000A ?? MEM_DONEO DW ? ; CURRENT OFFSET VALUE FOR BACKGROUND MEM TEST
000C ?? INTICS DW ? ; SAVE AREA FOR INTERRUPT IC ROUTINE
000E ?? INTICS DW ? ; FLAG TO INDICATE WHETHER MENU IS
0010 ?? MENU_UP DB ? ; ON SCREEN (FF=YES, OFFNO)
0011 ?? DONE128 DB ? ; BLOCKS TESTED BY BGMEM
0012 ?? KBDONE DW ? ; TOTAL K OF MEMORY THAT HAS BEEN TESTED BY BACKGROUND MEM TEST

; POST DATA AREA

0114 ?? IO_ROM_INIT DW ? ; POINTER TO OPTIONAL I/O ROM INIT ROUTINE
0116 ?? IO_ROM_SEG DW ? ; POINTER TO IO ROM SEGMENT
0118 ?? POST_ERR DB ? ; FLAG TO INDICATE ERROR OCCURRED DURING POST

; SERIAL PRINTER DATA

0222 ?? MFG_RTN DW ? ; POINTER TO MFG. OUTPUT ROUTINE
0224 ?? MFG_RTN DW ? ; SERIAL PRINTER DATA

0256 ?? SP_FLAG DW ? ; THE FOLLOWING SIX ENTRIES ARE DATA PERTAINING TO NEW STICK RIGHT STICK DELAY
0258 ?? SP_CHAR DB ? ; RIGHT BUTTON A DELAY
025A ?? SP_Char DB ? ; LEFT STICK DELAY
025C ?? NEW_STICK DATA DW ? ; LEFT BUTTON B DELAY
025E ?? NEW_STICK DATA DW ? ; RIGHT STICK LOCATION
0260 ?? NEW_STICK DATA DW ? ; UNUSED
0262 ?? NEW_STICK DATA DW ? ; LEFT STICK POSTITION

XXDATA ENDS

; DISKETTE DATA AREA

0000 ?? DDATA SEGMENT AT 60H
0000 ?? NUM_DRIVE DB ? ; TRACK,HEAD,SECTOR,NUM OF SECTOR,BUFFER FOR READ AND WRITE OPERATION
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 ?

0009 0B [ 60H
0000 ?? OK_BUF_LEN DUP(0,0,0,0) ; TRACK,HEAD,SECTOR,NUM OF SECTOR, BUFFER FOR READ AND WRITE OPERATION
0002 0200 [ 60H
0000 ?? OK_BUF_LEN DUP(0)
0009 0B [ 60H
0000 ?? OK_BUF_LEN DUP(0)
0009 0B [ 60H
0000 ?? OK_BUF_LEN DUP(0)
0009 0B [ 60H
0000 ?? OK_BUF_LEN DUP(0)

; INFO FLAGS

0429 ?? REQUEST_IN DB ? ; SELECTION CHARACTER
042A ?? DK_EXISTED DB ?
042B ?? DK_FLAG DB ?
042C ?? RAN_NUM DB ?
042E ?? SLED DB ?

; SPEED TEST VARIABLES

0430 ?? DK_SPEED DB ?
0432 ?? TIM_1 DB ?
0434 ?? TIM_1 DUP(0)
0436 ?? TIM_2 DUP(0)
0438 ?? TIM_2 DUP(0)
043A ?? FRACT_H DB ?
043C ?? FRACT_L DB ?
043E ?? PART_CYCLE DB ?
0440 ?? WHOLE_CYCLE DB ?
0442 ?? HALF_CYCLE DB ?
A-8  ROM BIOS

```
006D  B4 05  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  7B 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  B1 05  MOV  CL, 5  ; LOAD CNT REG WITH SHIFT CNT
007B  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  B0 40  MOV  AL, 40H  ; SET THE OF FLAG ON
0081  D0  E0  SHL  AL, 1  ; SETUP FOR TESTING
0083  71 39  JNO  L4  ; GO TO ERR ROUTINE IF OF NOT SET
0085  32 E4  XOR  AH, AH  ; SET AH = 0
0087  9E  SAHF  ; CLEAR SF, CF, ZF, AND PF
0088  76 34  JBE  L4  ; GO TO ERR ROUTINE IF CF ON
008A  7B 32  JS  L4  ; GO TO ERR ROUTINE IF SF ON
008C  7A 30  JP  L4  ; GO TO ERR ROUTINE IF PF ON
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  SLH  AH, 1  ; CHECK THAT 'OF' IS CLEAR
0097  70 25  JZ  L4  ; GO TO ERR ROUTINE IF ON

; ----- READ/WRITE THE 8088 GENERAL AND SEGMENTATION REGISTERS
; WITH ALL ONE'S AND ZEROES
; MOV  AX, OFFF0H  ; SETUP ONE'S PATTERN IN AX
0099  B8 FFFF  STC
009B  8E 08  MOV  DS, AX  ; WRITE PATTERN TO ALL REGS
009D  8E 0B  MOV  DS, BS
009F  8C 0B  MOV  DS, DS
00A1  8E 0C  MOV  ES, BS
00A3  8C 0C  MOV  ES, DS
00A5  8E 0D  MOV  SS, CX
00A7  8C 0D  MOV  SS, DS
00A9  8E 0E  MOV  SP,DX
00AB  8C 0E  MOV  SP, DS
00AD  88 FF  MOV  DI, SI
00B1  73 07  JNC  L3
00B3  33 C7  XOR  AX, 01H  ; PATTERN MAKE IT THRU ALL REGS
00B5  7B 07  JNZ  L4  ; NO - GO TO ERR ROUTINE
00B7  F8  CLC
00BB  EB E3  JMP  L2
00BA  0B C7  L3  ; OR AX, 01H  ; ZERO PATTERN MAKE IT THRU?
00BC  74 0C  JZ  L5  ; YES - GO TO NEXT TEST
00BD  B8 0A 00  MOV  DX, 000H  ; HANDLE ERROR
00C1  B0 00  MOV  AL, 0
00C3  EE  OUT  DX, AL  ; ERROR 0001
00C4  42  INC  DX
00C5  EE  OUT  DX, AL
00C6  FE 0C  INC  AL
00CB  EE  OUT  DX, AL
00CF  F4  HLAL  ; HALT

;------------------------------------------------------------------------
; TEST 2  ;
; ; B255 INITIALIZATION AND TEST  ;
; ; DESCRIPTION  ;
; ; FIRST INITIALIZE B255 PROG.  ;
; ; PERIPHERAL INTERFACE PORTS A&B  ;
; ; ARE LATCHED OUTPUT  ;
; ; BUFFERS. C IS INPUT.  ;
; ; MFG. ERR. CODE =0002H  ;
; ;------------------------------------------------------------------------
; 00CA  B0 FE  MOV  AL, OFEH  ; SEND FE TO MFG
00CC  E6 10  OUT  10H, AL
00CE  B0 89  MOV  AL, MDE_B255
00D0  E6  E3  OUT  CMO_PORT, AL  ; CONFIGURES 1/0 PORTS
00D2  2B C0  SUB  AX, AX  ; TEST PATTERN SEED = 0000
00D4  BA CA  MOV  AL, AH
00D6  4A 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  7B 06  JNZ  L7  ; IF NOT, SOMETHING IS WRONG
00E2  4E C4  INC  AH  ; MAKE NEW DATA PATTERN
00E4  75 EE  JNZ  L6  ; LOOP TILL 255 PATTERNS DONE
00E6  EB 05  JMP  SHORT L6  ; CONTINUE IF DONE
00EB  B3 02  MOV  BL, 02H  ; SET ERROR FLAG (BH=00 NOW)
00EC  EAB 098BC  JMP  E_MSG  ; GO ERROR ROUTINE
00ED  32 C0  XOR  AX, AL
00EE  E6 60  OUT  KBPORT, AL  ; CLEAR KB PORT
00F0  E4 62  IN  AL, PORT_C
00F2  80 08  MOV  AL, 00001008B  ; 64K CARD PRESENT?
00F5  80 1B  MOV  AL, 1BH  ; PORT SETTING FOR 64K SYS
00F7  7B 02  JNZ  LB  ;
00F9  80 3F  MOV  AL, 3FH  ; PORT SETTING FOR 128K SYS
00FB  BA 03DF  MOV  DX, PAGREG
00FE  EE  OUT  DX, AL
00FF  B0 00  MOV  AL, 00001100B  ; INITIALIZE OUTPUT PORTS
0101  E6 61  OUT  PORT_B, AL
```
PART 3  SET UP VIDEO GATE ARRAY AND 6845 TO GET MEMORY WORKING

MOV AL, DFCH  
OUT 10H, AL  
MOV DX, 034H  
SET ADDRESS OF 6845  
MOV DX, OFFSET VIDEO_PARMS  
MOV CX, 0040H  
SET PARM LEN  
XOR AH, AH  
AH IS REG #  
MOV AL, AH  
GET 6845 REG #  
OUT DX, AL  
POINT TO DATA PORT  
MOV AL, CS (BX)  
GET TABLE VALUE  
OUT DX, AL  
OUT TO CHIP  
INC BX  
NEXT IN TABLE  
DEC DX  
BACK TO POINTER REG

START VGA WITHOUT VIDEO ENABLED

MOV DX, VGA_CTL  
SET ADDRESS OF VGA  
IN AL, DX  
BE SURE ADDR/DATA FLAG IS  
IN THE PROPER STATE  
MOV CX, 5  
# OF REGISTERS  
XOR AH, AH  
AH IS REG COUNTER  
MOV AL, AH  
GET REG #  
OUT DX, AL  
SELECT IT  
XOR AL, AL  
SET ZERO FOR DATA  
INC AH  
NEXT REG

LOOP L11

TEST 4  PLANAR BOARD ROM CHECKSUM TEST  
DESCRIPTION  
A CHECKSUM TEST IS DONE FOR EACH ROM  
MODULE ON THE PLANAR BOARD TO  
MFG ERROR CODE =0003H MODULE AT ADDRESS:  
F000 0000 ERROR  
0004H MODULE AT ADDRESS  
F000: 0000 ERROR

MOV AL, OFCH  
OUT 10H, AL  
MFG OUT=FC  
CHECK MODULE AT F000:0 (LENGTH 32K)  
XOR SI, SI  
INDEX OFFSET WITHIN SEGMENT OF  
FIRST BYTE  
MOV AX, CS  
SET UP STACK SEGMENT  
MOV SS, AX  
MOV DS, AX  
LOAD DS WITH SEGMENT OF ADDRESS  
MOV CX, 8000H  
NUMBER OF BYTES TO BE TESTED, 32K  
MOV SP, OFFSET 21  
SET UP STACK POINTER 50  
MOV AX, OFFSET RAM CHECKSUM  
RETURN WILL COME HERE  
JMP ROUTINE WHICH PERFORMS  
CRC CHECK

L12:   
MOV AX, BX, 0003H  
SET ERROR CODE  
JMP E_MSG  
INDICATE ERROR

L13:   
MOV CX, 8000H  
LOAD COUNT (SI POINTING TO START  
MOV E_MSG  
OF NEXT MODULE AT THIS POINT)  
JMP ROM_CHECKSUM  
OF NEXT MODULE  
L14:   
JMP L15  
PROCEED IF NO ERROR

MOV BX, 0004H  
INDICATE ERROR

L15:  
TEST 5  BASE 2K READ/WRITE STORAGE TEST  
DESCRIPTION  
WRITE/READ/VERIFY DATA PATTERNS  
AA, 55, AND 0D 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)

MOV AL, OFBH  
OUT 10H, AL  
SET MFG FLAG=FB  
MOV CX, 0400H  
SET FOR 1K WORDS, 2K BYTES  
MOV ES, AX  
LOAD ES WITH 0000 SEGMENT  
JMP POO STG

L16:   
MOV AL, OFAH  
BAD STORAGE FOUND  
OUT 10H, AL  
MOV CX, 400H  
1024 WORDS TO BE TESTED IN THE  
REGEN BUFFER

IN AL, PORT_A  
WHERE IS THE REGEN BUFFER?  
CMP AL, 1BH  
TOP OF 64K?  
MOV AX, OF80H  
SET POINTER TO THERE IF IT IS  
JE L18  
MOV AH, JFH  
OR SET POINTER TO TOP OF 128K

L18:   
MOV ES, AX  
JMP POO STG

MOV AL, OFBH  
OUT 10H, AL  
MOV CX, 0400H  
SET FOR 1K WORDS, 2K BYTES  
MOV ES, AX  
LOAD ES WITH 0000 SEGMENT  
JMP POO STG
A-10 ROM BIOS
023B 47    INC 01
023B 47    INC 01 ; POINT TO NEXT VECTOR ENTRY
023A E2 FB  LOOP 05 ; REPEAT FOR ALL 16 BIOS INTERRUPTS
023C BE 09  MOV DS,CX ; SET DS TO ZERO
023E C7 06 0204 R 1B63 R
0244 C7 06 0208 R 1A2A R
024A C7 06 0224 R 1BA5 R

------- SET UP DEFAULT EQUIPMENT DETERMINATION WORD -------

0250 BB 1118  MOV BX,1118H ; DEFAULT GAME10,40X25,NO DMA,4BK ON
0253 E4 62  IN AL,PORT_C
0254 24  AND AL,0BH ; 64K CARD PRESENT
0257 75 03  JNZ DS5 ; NO, JUMP
0259 80 CB 04  OR BL,4 ; SET 64K ON PLANAR
025C BA IE 0410 R
0255 ; MOV DATA_WORD:EQUIP_FLAGS,DX

TEST 7

0260 E8 E608 R  CALL MFG_UP ; MFG CODE=F7
0263 B0 13  ASSUME DS:ABS0,CODE
0266 E6 20  OUT INTA0,AL
0267 B0 08  MOV AL,B ; ICW2 - SET INTERRUPT TYPE B (B=F)
0269 E6 21  OUT INTA0,AL
026B B0 09  MOV AL,9 ; ICW4 - SET BUFFERED MODE/SLA VE
026D E6 21  OUT INTA0,AL

TEST ABILITY TO WRITE/READ THE MASK REGISTER

026F B0 00  MOV AL,0 ; WRITE ZEROS TO IMR
0271 B9 08  MOV BL,AL ; PRESET ERROR INDICATOR
0273 E6 21  OUT INTA0,AL ; DEVICE INTERRUPTS ENABLED
0275 E4 21  IN AL,INTA01 ; READ IMR
0277 0A 00  OR AL,AL ; !MR = 07
0279 75 18  JNZ GERROR ; NO - GO TO ERROR ROUTINE
027B B0 FF  MOV AL,OFFH ; DISABLE DEVICE INTERRUPTS
027D E4 21  OUT INTA0,AL ; WRITE ONES TO IMR
027F E4 21  IN AL,INTA01 ; READ IMR
0281 04 01  ADD AL,1 ; ALL IMR BITS ON?
0283 75 0E  JNZ GERROR ; (ADD SHOULD PRODUCE 0)

CHECK FOR HOT INTERRUPTS

0285 F8
0286 B9 0050  MOV CX,50H
0288 E2 FF  LOOP HOT1
028B BA IE 0484 R  MOV BL,DATA_AREA(INTR_FLAG-DATA) ; DID ANY INTERRUPTS OCCUR?
028F 0A 08  OR BL,BL
0281 74 05  JZ END_TESTG ; NO - GO TO NEXT TEST
0283 B7 07  GERROR: MOV BH,07H ; SET 07 SECTION OF ERROR MSG
0285 E9 08BC R  JMP E_MSG
0288

END_TESTG:

; FIRE THE DISKETTE WATCHDOG TIMER

0289 B0 EO  MOV AL,WD_ENABLE+WD_STROBE+FDC_RESET
029A E6 F2  OUT OF2H,AL
029C B0 AD  MOV AL,WD_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 RESPO NSES 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

ROM BIOS  A-11
A-12 ROM BIOS
0328 B4 02
0329 B4 02
0330 24 04
0331 B3 05
0332 B2 2C

CHECK FOR STUCK BITS IN TIMER 2

MOV AH, 2
CALL BITS_ON_OFF
JNB REINIT_T2
MOV BL, 5
STUCK BITS IN TIMER 2
JMP SHORT TIMER_ERROR

RE_INITIALIZE TIMER 2 WITH MODE 0 AND A SHORT COUNT

0336 E4 61
0338 E4 61
033A E6 61
033C B0 0B
033E 00 0A
0342 FFFE 0 R

DROP GATE TO TIMER 2
IN AL, PORT_B
AND AL, 11111110B
RESET BIT 0 - LEAVE OTHERS ALONE
OUT PORT_B, AL
MOV AX, 0000B
MOV BX, 0000B
SET TIMER 2 TO MODE 0 BINARY
MOV BX, 0000B
INITIAL COUNT OF 10
CALL INIT_TIMER

CHECK PCS OF PORT_C TO SEE IF THE OUTPUT OF TIMER 2 IS LOW

034D E4 62
034F D0 00
0351 E4 62
0353 B3 04
0354 EB 13
0355 B9 000A
0356 E2 00
0357 4A 19
0358 B3 0E
0359 D7 00
035A E4 62
035B 00 00
035C 24 20
035D 75 57
035E EB 0E
035F E4 61
0360 B3 06

CHECK PCS OF PORT_C TO SEE IF THE OUTPUT OF TIMER 2 GOES HIGH

MOV CX, 0000B
WAIT FOR OUTPUT TO GO HIGH, SHOULD
IN AL, PORT_C
CURRENT STATUS
AND AL, 00000000B
MASK OFF OTHER BITS
MOV BL, 4
PCS OF PORT_C WAS HIGH WHEN IT
JMP SHORT TIMER_ERROR
SHOULD HAVE BEEN LOW
TURN GATE BACK ON

0365 B9 000A
0366 E2 00
0367 00 00
0368 B3 0F
0369 D7 00
036A E4 62
036B 00 00
036C 24 20
036D 75 57
036E EB 0E
036F E4 61
0370 B3 0E
0371 D7 00
0372 B3 0F
0373 D7 00
0374 B3 0F
0375 BF 0369 R
0376 D7 00
0377 B3 0F
0378 D7 00
0379 B3 0F
037A B6 04
037B 00 00
037C 03 08
037D 03 08
037E 03 08
037F 03 08
0380 03 08
0381 03 08
0382 03 08
0383 03 08
0384 03 08
0385 03 08
0386 03 08
0387 03 08
0388 03 08
0389 03 08
038A 03 08
038B 03 08
038C 03 08
038D 03 08
038E 03 08
038F 03 08
0390 03 08
0391 03 08
0392 03 08
0393 03 08
0394 03 08
0395 03 08
0396 03 08
0397 03 08
0398 03 08
0399 03 08
039A 03 08
039B 03 08
039C 03 08
039D 03 08
039E 03 08
039F 03 08
03A0 03 08
03A1 03 08
03A2 03 08
03A3 03 08
03A4 03 08
03A5 03 08
03A6 03 08
03A7 03 08
03A8 03 08
03A9 03 08
03AA 03 08
03AB 03 08
03AC 03 08
03AD 03 08
03AE 03 08
03AF 03 08
03B0 03 08
03B1 03 08
03B2 03 08
03B3 03 08
03B4 03 08
03B5 03 08
03B6 03 08
03B7 03 08
03B8 03 08
03B9 03 08
03BA 03 08
03BB 03 08
03BC 03 08
03BD 03 08
03BE 03 08
03BF 03 08
03C0 03 08
03C1 03 08
03C2 03 08
03C3 03 08
03C4 03 08
03C5 03 08
03C6 03 08
03C7 03 08
03C8 03 08
03C9 03 08
03CA 03 08
03CB 03 08
03CC 03 08
03CD 03 08
03CE 03 08
03CF 03 08
03D0 03 08
03D1 03 08
03D2 03 08
03D3 03 08
03D4 03 08
03D5 03 08
03D6 03 08
03D7 03 08
03D8 03 08
03D9 03 08
03DA 03 08
03DB 03 08
03DC 03 08
03DD 03 08
03DE 03 08
03DF 03 08
03E0 03 08
03E1 03 08
03E2 03 08
03E3 03 08
03E4 03 08
03E5 03 08
03E6 03 08
03E7 03 08
03E8 03 08
03E9 03 08
03EA 03 08
03EB 03 08
03EC 03 08
03ED 03 08
03EE 03 08
03EF 03 08
03F0 03 08
03F1 03 08
03F2 03 08
03F3 03 08
03F4 03 08
03F5 03 08
03F6 03 08
03F7 03 08
03F8 03 08
03F9 03 08
03FA 03 08
03FB 03 08
03FC 03 08
03FD 03 08
03FE 03 08
03FF 03 08

TIMER_ERROR:
MOV BH, 0
TIMER_ERRORIndicator
CALL E_MSG
JMP SHORT PO01 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.

LATCHES LABEL BYTE

DB 00H
LATCH MASK FOR TIMER 0
DB 00H
LATCH MASK FOR TIMER 1
DB 00H
LATCH MASK FOR TIMER 2

BITS_ON_OFF PROC NEAR
XOR BX, BX
INITIALIZE BX REGISTER
XOR SI, SI
IST PASS - SI = 0
MOV DX, TIMER
BASE PORT ADDRESS FOR TIMERS
ADD DL, AH
MOV G1, OFFSET LATCHES; SELECT LATCH MASK
XOR AL, AL
CLEAR AL
ADD DL, AH
AH -> AL
XCHG AH, AL
ADD DI, AX

IST PASS - CHECKS FOR ALL BITS TO COME ON
2ND PASS - CHECKS FOR ALL BITS TO GO OFF

OUTER LOOP:
MOV CX, BX
OUTER LOOP COUNTER
INNER LOOP:
PUSH CX
SAVE OUTER LOOP COUNTER
MOV CX, OFFSET INNER LOOP COUNTER
TST_BITS:
MOV AL, CS:(011)
TIMER LATCH MASK
OUT TIM_CTL, AL
TIMER LATCH
ADD AX, AX
PAUSE

JMP SHORT TST_CMP
DON'T CHANGE 'FLAGS

MOV CX, BX
OUTER LOOP COUNTER
JMP SHORT TST_CMP
DON'T CHANGE 'FLAGS

MOV CX, BX
OUTER LOOP COUNTER
JMP SHORT TST_CMP
DON'T CHANGE 'FLAGS

MOV CX, BX
OUTER LOOP COUNTER
JMP SHORT TST_CMP
DON'T CHANGE 'FLAGS

MOV CX, BX
OUTER LOOP COUNTER
JMP SHORT TST_CMP
DON'T CHANGE 'FLAGS

MOV CX, BX
OUTER LOOP COUNTER
JMP SHORT TST_CMP
DON'T CHANGE 'FLAGS

MOV CX, BX
OUTER LOOP COUNTER
JMP SHORT TST_CMP
DON'T CHANGE 'FLAGS
A-14 ROM BIOS
; HAVE MADE COMPLETE VERTICAL-VERTICAL CYCLE, NOW TEST RESULTS

0421 B0 40  Q11: MOV AL,40H ; LATCH TIMER
0423 E6 43  OUT TIM_CTL,AL
0425 81 FB 00C8  CMP DX,EPF
0427 74 04  ; NUMBER OF ENABLES BETWEEN
0428 83 05  ; VERTICALS O.K.?
042A EB 74  Q12: MOV BL,05H
042D E4 41  JMP SHORT Q22
042F EA E0  ; WRONG # ENABLES = ERROR 0905
0431 99 90  ; GET TIMER VALUE LOW
0433 3D 41  MOV AH,AH,AL
0434 E6 41  SAVE IT
0436 FB 00  NOP ; GET TIMER HIGH
0438 FB 00  XCHG AH,AL
0439 90 00  ; INTERRUPTS BACK ON
043A 3D A0AC  CMP AX,M0VT
043D 70 04  STI ; ENABLE INTS.
043F B3 06  MOV BL,06H
0442 EB 60  JMP SHORT Q22
0444 3D C460  ; VERTICALS TOO FAR APART
0446 3E 04  Q13: MOV AX,M0VT
0448 B7 00  JLE Q14
044B 1B 50  MOV BL,07H
044E 07 02  JMP SHORT Q22
044F E6 21  ; VERTICALS TOO CLOSE TOGETHER
0452 9E 61  ; ERROR 0906
0454 20 06 0484 R 0456 1E 21  ; TIMING SEEMS O.K., NOW CHECK VERTICAL INTERRUPT (LEVEL 5)
0457 3D 0A4C  Q14: SUB CX,CX ; SET TIMOUT REG
0459 F6 06 0484 R 20  ; UNMASK INT. LEVEL 5
045B EA E6 21  OUT INTA01,AL
045E 20 06 0484 R 045F 1E 21  AND DATA_AREA INTR_FLAG-DATA), AL
045F 1E 21  ; ENABLE INTS.
0461 3D 0A4C  OUT INTA01,AL
0464 75 06  JMP SHORT Q22
0466 00 06 0484 R 0468 3E 04  ; NO VERTICAL INTERRUPT
0469 1E 21  ; ERROR 0906
046B 0C 20  OR AL,0010000B
046C 06 21  ; KEEP LOOKING IF IT DIDN'T
046E 1E 21  JMP SHORT Q22
046F 0C 20  ; DISABLE INTERRUPTS FOR LEVEL 5
0471 EB 8C ; OUT INTA01,AL
0473 EE EE  ; SEE IF RED, GREEN, BLUE AND INTENSIFY DOTS WORK
0475 EC 00  ; FIRST, SET A LINE OF REVERSE VIDEO, INTENSIFIED BLANKS INTO VIDEO
0477 BB 090B  ; BUFFER
0479 BB 077F  MOV AX,090BH
MOV BX,077FH
; WRITE CHARs, BLOCKS
047B BB 002B  MOV CX,40
; PAGE 7, REVERSE VIDEO,
047D CF 10  MOV AL,10H
; HIGH INTENSITY
047F 3D 0A4C  MOV AH,AX
; 40 CHARACTERS
0481 EB 6C 10  ; START WITH BLUE DOTS
0483 BD 00  MOV BL,10H
0485 0A 0C  OR BL,AL
; OR IN DOT BEING TESTED
0487 1E 21  JMP SHORT Q22
0489 1E 21  ; DOT NOT COMING ON = ERROR 091X
048B 28 09  IF (X=0, BLUE, X=1, GREEN,
048D EC 00  X=2, RED, X=3, INTENSITY)
048F AB 10  ; SEE IF DOT GOES OFF
0491 E2 F9  Q15: SUB CX,CX
0493 B3 10  Q20: IN AL,DX
0495 0A 0C  ; GET STATUS
0497 0A 0C  TEST AL,00010000B
0499 0A 0C  ; DOT THERE?
049B EC 00  MOV AL,10H
049D 0A 0C  ; GO LOOK FOR DOT TO TURN OFF
049F EC 00  LOOP Q18
04A1 EB 0B  ; CONTINUE TESTING FOR DOT ON
04A3 BD 00  MOV BL,10H
04A5 0A 0C  OR BL,AL
; OR IN DOT BEING TESTED
04A7 1E 21  JMP SHORT Q22
04A9 28 09  ; DOT STUCK ON = ERROR 092X
04AB EC 00  IF (X=0, BLUE, X=1, GREEN,
04AD 0A 0C  X=2, RED, X=3, INTENSITY)
04AF 01 0D  ; SEE IF DOT GOES OFF
04B1 EC 00  Q19: SUB CX,CX
04B3 BD 00  ; GET STATUS
04B5 0A 0C  TEST AL,00010000B
04B7 EC 00  ; IS DOT STILL ON?
04B9 EB 0B  MOV AL,10H
04BB 0A 0C  ; GO ON IF DOT OFF
04BD EC 00  LOOP Q20
04BF 0A 0C  ELSE, KEEP WAITING FOR DOT
04C1 EB 0B  ; TO GO OFF
04C3 BD 00  MOV BL,10H
04C5 0A 0C  OR BL,AL
; OR IN DOT BEING TESTED
04C7 1E 21  JMP SHORT Q22
04C9 28 09  ; DOT STUCK ON = ERROR 092X
04CB EB 0B  IF (X=0, BLUE, X=1, GREEN,
04CD 0A 0C  X=2, RED, X=3, INTENSITY)
04CF EC 00  ; ADJUST TO POINT TO NEXT DOT
04D1 EC 00  Q21: INC AH
04D3 EC 00  ; ALL 4 DOTS DONE?
04D5 EC 00  CMP AH,4
04D7 EC 00  ; GO END
04DB EC 00  JE Q23
04DA EC 00  MOV AL,AL
04DE EB 06  JMP Q17
04E2 EB 06  ; GO LOOK FOR ANOTHER DOT
04E6 BD 00  MOV BH,09H ; SET MSR OF ERROR CODE
04E8 E9 09BC R  ; JMP E_MSG
04EB 1E 21  ; DONE WITH TEST RESET TO 40X25 = COLOR
04F1 EC 00  ; ASSUME DS: DATA
04F3 EC 00  Q23: CALL DDS
04F5 EC 00  ; INIT TO 40X25 = COLOR
04F7 EC 00  MOV AX,0001H
04F9 EC 00  INT 10H
04FB EC 00  MOV AX,0507H
04FD EC 00  SET TO VIDEO PAGE 7
04FF EC 00  MOV BL,05H
0501 EC 00  ; SET VIDEO FLAG,1234H, WARM START?
0503 BD 00  JE Q24
0505 BD 00  BYPASS PUTTING UP POWER-ON SCREEN
0507 BD 00  CALL PUT_LOGO
0509 BD 00  ; PUT LOGO ON SCREEN

Appendix A
ROM BIOS A-15
04BD EB 0C21 R  CALL PUT_LOGO ; PUT LOGO ON SCREEN
04C0 B0 76  MOV AL,01110110B ; RE-INIT TIMER 1
04C4 B0 00  OUT TIM_CTRL,AL
04C6 EB 41  OUT TIMER+1,AL
04C8 90  NOP
04CA 90  NOP
04CE EB 41  OUT TIMER+1,AL
04CC EB E60B R  CALL MFG_UP ; MFG CHECKPOINT=F4
04C4 33 C0  XOR AX,AX
04D1 BE 0B  MOV DS,AX
04D3 C7 06 0008 R 0F78 R  MOV NMI_PTR,OFFSET KBDMNK ; SET INTERRUPT VECTOR
04D9 C7 06 0120 R F068 R  MOV KEY2D_PTR,OFFSET KEY_SCAN_SAVE ; SET VECTOR FOR POD INT HANDLER
04DF 0E  PUSH CS
04E0 5A  POP AX
04E1 A3 0122 R  MOV KEY62_PTR+2,AX ; IF A KEY IS STUCK, THE BUFFER SHOULD FILL WITH THAT KEY'S CODE
04E4 EB 138B R  CALL DDS ; SET DATA SEGMENT
04EE BE 001E R  MOV SI,OFFSET KBUFFER ; SET KEYBOARD PARTS
04F4 BE 89 36 001A R  MOV BUFFER_HEAD,SI
04F7 BE 89 36 001C R  MOV BUFFER_TAIL,SI
04F9 BE 89 36 0080 R  MOV BUFFER_START,SI
04F9 BE 89 36 0082 R  ADD SI,32 ; SET DEFAULT BUFFER OF 32 BYTES
04FA BE 89 36 0008 R  MOV BUFFER_END,SI
04FD E4 A0  IN AL,0A0H ; CLEAR NMI /F
04FF B0 B0  MOV AL,BOH ; ENABLE NMI
0501 E6 A0  OUT 0A0H,AL
0503 EB E60B R  CALL MFG_UP ; MFG CHECKPOINT=F3
0506 BB 0040  MOV BX,BX  ; START WITH BASE 64K
0509 E4 62  IN AL,PORT_C ; GET CONFIG BYTE
050B A8 08  TEST AL,00001000B ; SEE IF 64K CARD INSTALLED
050D 75 03  JNE Q25 ; (BIT 4 WILL BE 0 IF CARD PLUGGED)
050F B3 C3 40  ADD BX,64 ; ADD 64K
0512 53  Q25  PUSH BX  ; SAVE K COUNT
0513 B3 ED 10  SUB BX,16 ; SUBTRACT 16K CRT REFRESH SPACE
0516 89 E8 0013 R  MOV [MEMORY_SIZE],BX ; LOAD "CONTINUOUS MEMORY" WORD
051A 5A  POP BX
051B BA 2000  MOV DX,2000H ; SET POINTER TO JUST ABOVE 128K
051E 2B FF  SUB DI,DI ; SET DI TO POINT TO BEGINNING
0520 89 AA55  MOV CX,DATA BUFFER_START ; LOAD DATA PATTERN
0523 BE C2  Q26  MOV ES,DX ; SET SEGMENT TO POINT TO MEMORY SPACE
0525 26 89 00  MOV ES:[01H],CX ; SET DATA PATTERN TO MEMORY
0528 B0 0F  MOV AL,0FH ; SET AL TO 0 FF VALUE
052A 26 8B 05  MOV AX,E5H ; 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 B1 C2 1000  ADD DX,1000H ; POINT TO BEGINNING OF NEXT 64K
0535 83 C3 40  ADD BX,64 ; ADJUST TOTAL MEM. COUNTER
0538 B0 FE 40  CMP DH,0A0H ; PAST 640K YET?
053B 75 E6  JNE Q26 ; CHECK FOR ANOTHER BLOCK IF NOT
053D 89 IE 0015 R  Q27  MOV [TRUE_MEM],BX ; LOAD "TOTAL MEMORY" WORD ; SIZE HAS BEEN DETERMINED, NOW TEST OR CLEAN ALL OF MEMORY
0541 BB 0004  MOV AX,4 ; 4 KB KNOWN OK AT THIS POINT
0544 EB 058C R  CALL Q35
0547 B0 0080  MOV DX,0080H ; SET POINTER TO JUST ABOVE LOWER 2K
054A B9 7800  MOV CX,7800H ; TEST 30K WORDS (60KB)
054D BE C2  Q28  MOV E5,DX
054F 51  PUSH CX
0550 B3  PUSH BX
0551 50  PUSH AX
0552 EB 0859 R  CALL POPSTG ; TEST OR FILL MEM
0555 74 03  JMP Q28 ; CLEAR
0557 E9 0603 R  JMP Q39 ; JUMP IF ERROR
055A 5A  POP AX
055B 5B  POP BX
055C 59  POP CX ; RECOVER
055D 80 FD 78  CMP CH,78H ; WAS THIS A 60 K PASS
0560 9C  PUSHF
0561 05 003C  ADD AX,60 ; BUMP GOOD STORAGE BY 60 KB
0564 90  POPPF
0565 74 03  JE Q30
0567 05 0002  ADD AX,2 ; ADD 2 FOR A 62K PASS
056A EB 05BC R  Q30  CALL Q25
056B 3B C3  CMP AX,BX ; ARE WE DONE YET?
056F 75 03  JNE Q31
0571 E9 064A R  JMP Q43 ; ALL DONE, IF SO
Q31: CMP AX,128 ; DONE WITH 1ST 128K?
Q32: JE Q32 ; GO REST OF MEM.
Q33: CMP AX,0FH0H ; SET POINTER TO FINISH 1ST 64 KB
Q34: MOV CX,4000H ; SET FOR 32KB BLOCK
Q35: PROC NEAR
Q36: XOR DX,DX
Q37: MOV AH,2 ; SET CURSOR TOWARD THE END OF
Q38: MOV CX,3 ; SET UP FOR DECIMAL CONVERT
Q39: OR AL,3F2H ; MAKE INTO ASCII
Q40: MOV DL,3OH ; MAKE INTO ASCII
Q41: MOV DX,1000H ; SET POINTER TO 2ND 64K BLOCK
Q42: MOV CX,7C00H ; EPI X
Q43: JMP Q28 ; GO TEST IT
Q44: MOV DX,2000H ; POINT TO BLOCK ABOVE 128K
Q45: CMP BX,AX ; COMPARE GOOD MEM TO TOTAL MEM
Q46: JNZ Q34 ; JUMP GOOD MEMORY COUNT
Q47: JMP Q33 ; EXIT IF ALL DONE
Q48: POP AX ; AND MAKE ANOTHER PASS
Q49: POP BX
Q50: POP AX
Q51: POP CX
Q52: POP DX
Q53: POP ECX
Q54: POP EDX
Q55: POP EBX
Q56: POP ESP
Q57: POP EBP
Q58: POP ESI
Q59: POP EDI
Q60: MOV AX,65536 ; SET TOTAL GOOD MEMORY COUNT
Q61: MOV AH,2 ; JUMP GOOD MEMORY COUNT
Q62: INT 21H ; AND MAKE ANOTHER PASS
Q63: MOV AX,128K ; ESTABLISH ADDRESSING
Q64: CALL DDS ; CALL DSS : RESET_FLAG,1234H ; WARM START?
Q65: JNE Q35 ; NO PRINT ON WARM START
Q66: CALL DSS ; CALL DSS : RESET_FLAG,1234H ; WARM START?
Q67: JNZ Q35 ; NO PRINT ON WARM START
Q68: CALL DDS ; CALL DSS : RESET_FLAG,1234H ; WARM START?
Q69: CALL DDS ; CALL DSS : RESET_FLAG,1234H ; WARM START?
Q70: CALL DDS ; CALL DSS : RESET_FLAG,1234H ; WARM START?
Q71: CALL DDS ; CALL DSS : RESET_FLAG,1234H ; WARM START?
Q72: CALL DDS ; CALL DSS : RESET_FLAG,1234H ; WARM START?
Q73: CALL DDS ; CALL DSS : RESET_FLAG,1234H ; WARM START?
Q74: CALL DDS ; CALL DSS : RESET_FLAG,1234H ; WARM START?
Q75: CALL DDS ; CALL DSS : RESET_FLAG,1234H ; WARM START?
Q76: CALL DDS ; CALL DSS : RESET_FLAG,1234H ; WARM START?
Q77: CALL DDS ; CALL DSS : RESET_FLAG,1234H ; WARM START?
Q78: CALL DDS ; CALL DSS : RESET_FLAG,1234H ; WARM START?
Q79: CALL DDS ; CALL DSS : RESET_FLAG,1234H ; WARM START?
Q80: CALL DDS ; CALL DSS : RESET_FLAG,1234H ; WARM START?
Q81: CALL DDS ; CALL DSS : RESET_FLAG,1234H ; WARM START?
Q82: CALL DDS ; CALL DSS : RESET_FLAG,1234H ; WARM START?
Q83: CALL DDS ; CALL DSS : RESET_FLAG,1234H ; WARM START?
Q84: CALL DDS ; CALL DSS : RESET_FLAG,1234H ; WARM START?
0626 80 FC 02 Q41: CMP AH, 02 ; EVEN BYTE ERROR? ERR 0A0X
0629 BA D9 MOV BL, CL
062B 74 0B JE Q42
062D FE C7 INC BH ; MAKE INTO 0BXX ERR
062F 0A 0A MOVT BL, CH ; MOVE AND COMBINE ERROR BITS
0631 80 FC 01 CMP AH, 1 ; ODD BYTE ERROR
0634 74 02 JE Q42
0636 FE C7 INC BH ; MUST HAVE BEEN BOTH
0638 BE 00 35 R Q42: MOV SI, OFFSET MEM_ERR
063B E8 09 BC R CALL E_MSG ; LET ERROR ROUTINE FIGURE OUT
063E FA CLI ; WHAT TO DO
063F F4 HLT
0640 Q43:--------------------------------------------------------------------

; 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 CARD FAILURE
; XX01, KB DATA STUCK HIGH
; XX02, KB DATA STUCK LOW
; XX03, NO NMI INTERRUPT
; 22XX STUCK KEY (XX=SCAN CODE)
;--------------------------------------------------------------------

; CHECK FOR STUCK KEYS

0640 EB E60B R CALL MFG_UP ; MFG CODE=F2
0643 EB 13BB R CALL DOS5 ; ESTABLISH ADDRESSING
0646 BB 00 1E R MOV BX, OFFSET KB_BUFFER
0649 BA 07 MOV AL, [BX] ; CHECK FOR STUCK KEYS
064B 0A 0C OR AL, AL ; SCAN CODE = 0?
064D 74 06 JE F6_Y ; YES - CONTINUE TESTING
064F 87 22 MOV BH, 22H ; 22XX ERROR CODE
0651 LB BA 0B MOV BL, AL
0653 EB 0A JMP SHORT F6
0655 80 3E 00 12 R 00 F6_Y: CMP KB_ERR_B, OOH ; DID NMI'S HAPPEN WITH NO SCAN
0657 74 1C JE F7 ; CODE PASSED?
0659 88 1C MOV BX, 2000H ; SET ERROR CODE 2000
065B EF 00 3E R F6: MOV SI, OFFSET KEY_ERR ; GET MSG ADDR
065E 81 3E 0072 R 4321 CMP RESET_FLAG, 4321H ; WARM START TO DIAGS
0662 74 0B JE F6_Z ; DO NOT PUT UP MESSAGE
0664 8A 3E 0072 R 1234 CMP RESET_FLAG, 1234H ; WARM SYSTEM START
0666 74 03 JE F6_Z ; DO NOT PUT UP MESSAGE
0668 87 22 MOV BH, 22H ; 22XX ERROR CODE
066A EB 09 BC R CALL E_MSG ; PRINT MSG ON SCREEN
066C E9 0EF R F6_Z: JMP F6_X ; CHECK LINK CARD, IF PRESENT
066F 8A 02 01 R F7: MOV DX, 020H
0672 74 EC IN AL, DX ; CHECK FOR BURN-IN MODE
0675 6C 24 F0 AND AL, FF0H
0677 74 7F JZ F6_X ; BYPASS CHECK IN BURN-IN MODE
0679 E4 62 IN AL, PORT_C ; GET CONFIG. PORT DATA
067B 24 80 AND AL, 10000000B ; KEYBOARD CABLE ATTACHED?
067D 74 79 JZ F6_X ; BYPASS TEST IF IT IS
067F E4 61 IN AL, PORT_B
0681 E4 FC AND AL, 11111100B ; DROP SPEAKER DATA
0683 EA 61 E6 OUT PORT_B, AL
0685 8C 80 MOV AL, 06BH ; MODE SET TIMER 2
0687 8E 43 OUT TIM_CTL, AL
0689 80 40 MOV AL, 040H ; DISABLE NMI
068B 6E 40 OUT 040H, AL
068D 80 A0 MOV AL, 32 ; LSB TO TIMER 2
068F 8A 00 42 MOV DX, TIMER+2
0692 69 EE OUT DX, AL
0694 8A 2B C0 SUB AX, AX
0696 8B CB MOV CX, AX
0698 EE EE OUT DX, AL ; MSB TO TIMER 2 (START TIMER)
069A 4F 61 IN AL, PORT_B
069C 61 OC 01 OUT AL, 1
069E 61 E6 OUT PORT_B, AL ; ENABLE TIMER 2
069A 4F 62 F7_0: IN AL, PORT_C ; SEE IF KEYBOARD DATA ACTIVE
069C 62 40 F7_0 AND AL, 01000000B ; KEYBOARD CABLE ATTACHED?
069E 75 06 JNZ F7_1 ; EXIT LOOP IF DATA SHOWN UP
06A0 E2 F8 LOOP F7_0
06A2 BD 03 MOV BL, 02H ; SET NO KEYBOARD DATA ERROR
06A4 EB 49 JMP SHORT F6_1
06A6 06 F7_1: PUSH ES ; SAVE ES
06A8 2B C0 SUB AX, AX ; SET UP SEGMENT REG
06B0 BE 0C MOV ES, AX
06B2 8C 2F MOV PORT_X, PORT_X
06B4 26 0E 06 008 R PORT_Y 0F15 R MOV ES: [NMI_PTR+OFFSET_D1] ; SET UP NEW NMI VECTOR
06B8 A2 0084 R MOV INTR_FLAG, AL ; RESET INTR FLAG
06BC 6C 24 F0 IN AL, PORT_B ; DISABLE INTERNAL BEEPER TO
06BE 24 40 OR AL, 01010000B ; PREVENT ERROR BEEP
06C0 6E 61 OUT PORT_B, AL
06C2 80 C8 MOV AL, 0COH
06C4 6F 40 OUT 040H, AL ; ENABLE NMI
06C6 89 B0 0100 MOV CX, 0100H ;
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)
; (23FFH (RELAY FAILED TO PICK))

= OA9A
MIN_PERIOD EQU OA9AH ; NOM.*102

----- TURN THE CASSETTE MOTOR OFF

0703 E8 E680 R CALL MFG_UP     ; MFG CODE=F1
0706 E4 61 61 IN AL,PORT_B
0708 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 21 IN AL,INTA01
070E 0C 01 OR AL,01H     ; DISABLE TIMER INTERRUPTS
0710 E6 21 21 OUT INTA01,AL
0712 B0 B6 MOV AL,08BH ; SEL TIM 2, L5B, M5B, M0 3
0714 E6 43 43 OUT TIMER+3,AL ; WRITE 8223 CMD/MODE REG
0716 B8 0402 MOV AX,1234 ; SET TIMER 2 CNTR FOR 1000 USEC
0718 E9 42 42 OUT TIMER+2,AL ; WRITE TIMER 2 COUNTER REG
071B BA C4 MOV AL,6H
071D E6 42 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 62 IN AL,PORT_C ; READ VALUE OF CASS IN BIT
0725 24 10 AND AL,10H ; ISOLATE FROM OTHER BITS
0727 A2 006B R MOV AL,LAST_VAL,AL
0729 EB F96F R CALL READ_HALF_BIT ; TO SET UP CONDITIONS FOR CHECK
072B EB F96F R CALL READ_HALF_BIT
0730 E3 3E JCKX FB ; CAS_ERR
0732 53 PUSH BX ; SAVE HALF BIT TIME VALUE
0733 EB F96F R CALL READ_HALF_BIT
0736 79 POP AX ; SET TOTAL TIME
0737 E3 37 JCKX FB ; CAS_ERR
0739 03 C3 ADD AX,BX
073B 30 0A9A CMP AX,MAX_PERIOD
073E 73 30 JNC FB ; CAS_ERR
0740 30 0BAD CMP AX,MIN_PERIOD
0743 2B 2B JC FB
0745 BA 0201 MOV DX,201H
0748 E5 IN AL,DX
0749 24 F0 AND AL,0F0H ; DETERMINE MODE
074B 3C 10 CMP AL,00010000B ; MFG?
074D 74 04 JE G INDEX ; MFG
074F 3C 40 CMP AL,00100000B ; 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.)

----- CASSette INTERFACE TEST

0753 E4 61 IN AL,PORT_B
0755 8A D0 MOV DL,AL ; SAVE PORT B CONTENTS
0757 24 E5 AND AL,11100101B ; SET CASSETTE MOTOR ON
0759 EB 61 61 OUT PORT_B,AL
075B 33 C9 XOR CX,CX
075D E2 FE 81 LOOP F91 ; WAIT FOR RELAY TO SETTLE
075F EB F96F R CALL READ_HALF_BIT
0762 EB F96F R CALL READ_HALF_BIT
0767 BA C2 MOV AL,DL
0767 EB 61 OUT PORT_B,AL
0769 E3 0E JCKX T13_END ; READ HALF_BIT SHOULD TIME OUT IN
; THIS SITUATION
076B B8 23FF MOV BX,23FFFH ; ERROR 23FF
076F E8 03 JMP SHORT F81
0770 8B FB ; CAS_ERR
0770 BB 2300 MOV BX,2300H ; ERR CODE 2300H
0773 BE 0037 R MOV SI,OFFSET CAS_ERR ; CASSETTE WRAP FAILED
0776 EB 038C R CALL E_MSG ; GO PRINT ERROR MSG
0779 E4 21 IN AL,INTA01
077B 24 FE AND AL,OFER ; ENABLE TIMER INTS
077D 81 21 OUT INTA01,AL
077F 4A 00 IN AL,NMI_PORT ; CLEAR NMI: FLIP/FLIP
0781 B9 B0 MOV AL,08H ; ENABLE NMI INTERRUPTS
0783 E6 A0 OUT NMI_PORT,AL

----- CASSette INTERFACE TEST

06CD E2 FE F6_0: LOOP F6_0 ; WAIT A BIT
06CF E4 61 IN AL,PORT_B ; RE-ENABLE BEEPER
06D1 24 CF AND AL,11001111B
06D3 EB 61 OUT PORT_B,AL
06D5 A0 0084 R MOV AL,INTR_FLAG ; GET INTR FLAG
06DB 0A C0 MOV AL,AL ; WILL BE NON-ZERO IF NMI HAPPENED
06D8 E6 03 MOV BL,03H ; SET POSSIBLE ERROR CODE
06DC 26: C7 06 000B R OF78 R MOV ES:[NMI_PTR],OFFSET KBDMEM ; RESET NMI VECTOR
06DE 07 POP ES ; RESTORE ES
06E4 74 14 JZ F6_1 ; JUMP IF NO NMI
06E6 B0 00 MOV AL,00H ; DISABLE FEEDBACK CKT
06E8 E6 A0 OUT DA0H,AL
06EA E4 61 IN AL,PORT_B
06EC 24 FE AND AL,11111110B ; DROP GATE TO TIMER 2
06EE E4 61 OUT PORT_B,AL
06F0 E4 62 F6_2: IN AL,PORT_C ; SEE IF KEYBOARD DATA ACTIVE
06F2 24 40 AND AL,01000000B ; LOOP F6_2
06F4 74 09 JZ F6_X ; EXIT LOOP IF DATA WENT LOW
06F6 E2 F8 LOOP F6_2
06F8 B3 01 MOV BL,01H ; SET KEYBOARD DATA STUCK HIGH ERR
06FA 87 21 MOV BH,21H ; POST ERROR "21XX"
06FC E9 065F R JMP F6
06FF B0 00 F6_X: MOV AL,00H ; DISABLE FEEDBACK CKT
0701 E6 A0 OUT DA0H,AL
0785 EB E608 R CALL MFG_UP ; MFG ROUTINE INDICATOR=0F
0788 BA 02F8 MOV DX,02F8H ; ADDRESS OF SERIAL PRINTER CARD
078B EB EB31 R CALL UART ; ASYNCH. COMM. ADAPTER POD
078E 73 06 JNC TM ; PASSED
0790 BE 0038 R MOV SI,OFFSET COM1_ERR ; CODE FOR DISPLAY
0793 EB 09BC R CALL E_MSG ; REPORT ERROR

ASSUME CS:CODE,DS:DATA

; TEST SERIAL PRINTER INS8250 UART

0796 EB E608 R TM: CALL MFG_UP ; MFG ROUTINE INDICATOR = EF
0799 E4 62 IN AL,PORT_C ; TEST FOR MODERN CARD PRESENT
079B 24 02 AND AL,00000010B ; ONLY CONCERNED WITH BIT 1
079D 75 0E JNE TM1 ; IT'S NOT THERE - DONE WITH TEST
079F BA 03F8 MOV DX,03F8H ; ADDRESS OF MODERN CARD
07A2 EB EB31 R CALL UART ; ASYNCH. COMM. ADAPTER POD
07A5 73 06 JNC TM1 ; PASSED
07A7 BE 0038 R MOV SI,OFFSET COM2_ERR ; MODERN ERROR
07AA EB 09BC R CALL E_MSG ; REPORT ERROR
07AD

; TEST MODERN INS8250 UART

; SETUP HARDWARE INT. VECTOR TABLE

; ASSUME CS:CODE,DS:AB50

07B0 2B CO SUB AX,AX
07AF BE CO MOV ES,AX
07B1 B9 0008 MOV CX,0B ; GET VECTOR CNT
07B4 0E PUSH CS ; SETUP DS SEG REG
07B5 IF POP DS
07B6 BE FEF3 R MOV SI,OFFSET VECTOR_TABLE
07B9 BF 0020 R MOV DI,OFFSET INT_PTR
07BC A5 F7A MOVSW
07BD 47 INC DI ; SKIP OVER SEGMENT
07BE 47 INC DI
07BF E2 FB LOOP F7A ;----- SET UP OTHER INTERRUPTS AS NECESSARY

; CHECK FOR OPTIONAL ROM FROM 00000 TO F0000 IN 2K BLOCKS

; (A VALID MODULE HAS 'SSAA' IN THE FIRST 2 LOCATIONS, LENGTH INDICATOR (LENGTH/S12) IN THE 3D LOCATION AND
; TEST/INIT. CODE STARTING IN THE 4TH LOCATION)

; MFG ERR CODE 23XX (AX=MSB OF SEGMENT THAT HAS CRC CHECK)

; CHECK FOR OPTIONAL ROM FROM 00000 TO F0000 IN 2K BLOCKS

; (A VALID MODULE HAS 'SSAA' IN THE FIRST 2 LOCATIONS, LENGTH INDICATOR (LENGTH/S12) IN THE 3D LOCATION AND
; TEST/INIT. CODE STARTING IN THE 4TH LOCATION)

; MFG ERR CODE 23XX (AX=MSB OF SEGMENT THAT HAS CRC CHECK)

; CHECK FOR OPTIONAL ROM FROM 00000 TO F0000 IN 2K BLOCKS

; (A VALID MODULE HAS 'SSAA' IN THE FIRST 2 LOCATIONS, LENGTH INDICATOR (LENGTH/S12) IN THE 3D LOCATION AND
; TEST/INIT. CODE STARTING IN THE 4TH LOCATION)

; MFG ERR CODE 23XX (AX=MSB OF SEGMENT THAT HAS CRC CHECK)

; CHECK FOR OPTIONAL ROM FROM 00000 TO F0000 IN 2K BLOCKS

; (A VALID MODULE HAS 'SSAA' IN THE FIRST 2 LOCATIONS, LENGTH INDICATOR (LENGTH/S12) IN THE 3D LOCATION AND
; TEST/INIT. CODE STARTING IN THE 4TH LOCATION)

; MFG ERR CODE 23XX (AX=MSB OF SEGMENT THAT HAS CRC CHECK)

; CHECK FOR OPTIONAL ROM FROM 00000 TO F0000 IN 2K BLOCKS

; (A VALID MODULE HAS 'SSAA' IN THE FIRST 2 LOCATIONS, LENGTH INDICATOR (LENGTH/S12) IN THE 3D LOCATION AND
; TEST/INIT. CODE STARTING IN THE 4TH LOCATION)

; MFG ERR CODE 23XX (AX=MSB OF SEGMENT THAT HAS CRC CHECK)

; CHECK FOR OPTIONAL ROM FROM 00000 TO F0000 IN 2K BLOCKS

; (A VALID MODULE HAS 'SSAA' IN THE FIRST 2 LOCATIONS, LENGTH INDICATOR (LENGTH/S12) IN THE 3D LOCATION AND
; TEST/INIT. CODE STARTING IN THE 4TH LOCATION)

; MFG ERR CODE 23XX (AX=MSB OF SEGMENT THAT HAS CRC CHECK)
08C3 B0 3E 0018 R 00 CMP POST_ERR,00H ; CHECK FOR "POST_ERR" NON-ZERO
ASSUME DS:DATA
08C6 1F IF POP DS
08C9 74 10 JE F1SA_0 ; CONTINUE IF NO ERROR
08CE B2 02 MOV DL,2 ; 2 SHORT BEEPS (ERROR)
08CF EB 1AOC R CALL ERR_BEEP
08D0 ERWAIT: ERR: MOV AH,00 ; WAIT FOR "ENTER" KEY
08D2 80 04 MOV AH,1CH
08D4 80 FC 1C CMP AH,1CH
08D7 75 F7 JNE ERWAIT
08DB 08 05 JMP SHORT F15C
08DD B2 01 F1SA_0: MOV DL,1 ; 1 SHORT BEEP (NO ERRORS)
08DE EB 1AOC R CALL ERR_BEEP
08E0 80 0030 R MOV FISC: MOV BP,OFFSET F4 ; FISC_TBL
08E3 33 F6 XOR SI,SI
08E5 F16: MOV SI,SI ; FRO_BASE
08E7 2E: BB 56 00 MOV DX,C5: [BP] ; GET PRINT BASE ADDR
08E9 BE 00 AA MOV AL,OAH ; WRITE DATA TO PORT A
08EC BE EE OUT DL,AL
08ED BE 00 EC PUSH DS ; BUS SETTLING
08EE BE EC IN AL,DX ; READ PORT A
08EF BE 00 EE POP DS
08F0 BE 00 AA CMP AL,OAH ; DATA PATTERN SAME
08F4 F1 76 05 JNE F17 ; NO - CHECK NEXT PRT CD
08F8 89 B4 0008 R MOV PRINTER_BASE(SI),DX ; YES - STORE PRNT BASE ADDR
08FB 46 INC SI ; INCREMENT TO NEXT WORD
08FE 46 F17: INC SI
08FF 45 INC BP ; POINT TO NEXT BASE ADDR
0904 45 INC BP
090F 83 FD 41 CMP BP,OFFSET F4E ; ALL POSSIBLE ADDR CHECKED?
0912 F1C 76 E5 JNE F16 ; FRO_BASE
0915 30 00 DB XOR BX,BX ; SET ADDRESS BASE
0918 02 0A BF MOV DX,03FAH ; POINT TO INT 10 REGISTER
091A 0E EC IN AL,DX ; READ PORT
091C 06 AB F8 TEST AL,0FH ; SEEM TO BE AN 8250
091E 08 75 08 JNZ F1B
0920 89 C7 8000 R 03FB MOV RS232_BASE(BX),3FBH ; SETUP RS232 C0 #1 ADDR
0924 10 43 INC BX
0926 43 43 INC BX
0929 C7 87 0000 R 02FB MOV RS232_BASE(BX),2FBH ; SETUP RS232 #2
092C 43 4B INC BX ; (ALWAYS PRESENT)
092F 43 43 INC BX ; ---- SETUP EQUF FLAG TO INDICATE NUMBER OF PRINTERS AND RS232 ; CARDS
0932 BB 06 MOV AX,SI ; SI HAS 2*# OF PRINTERS
0934 81 B1 03 MOV CL,3 ; SHIFT COUNT
0936 81 D2 0B MOV CX,CL ; ROTATE RIGHT 3 POSITIONS
0938 82 E6 0A OR AL,CL ; OR IN THE RS232 COUNT
093A 02 08 06 0011 R OR BYTE PTR EQUF_FLAG+1,AL ; STORE AS SECOND BYTE ; ---- SETUP EQUF 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.
093C BB CB MOV CX,AX
093F 82 BB 02FE MOV BX,2FEH ; SET POINTER TO MODEM STATUS REG
0942 82 BA 02FC MOV DX,2FC ; XOR TO MODEM CONTROL REG
0945 82 2A 0C SUB AL,AL
0948 83 EE OUT DX,AL ; CLEAR IT
094B 83 8B 83 JMP *+2 ; DELAY
094E 83 BD 83 JCXZ DX,DX ; POINT TO MODTEM STATUS REG
0951 83 EC IN AL,DX ; CLEAR IT
0954 83 EE 00 JMP *+2 ; DELAY
0957 83 EB 00 JCXZ DX,DX ; POINT TO MODTEM STATUS REG
095A 83 EC IN AL,DX ; GET CONTENTS
095D 83 EE 08 TEST AL,00001000B ; HAS CARRIER DETECT CHANGED?
095F 84 7A 23 JZ F19_A ; NO, THEN NO PRINTER
0961 84 46 08 TEST AL,00000100B ; DID CTS CHANGE? (AG WITH WRAP ; CONNECTOR INSTALLED) ; CARRIER DETECT IS FOLLOWING RTS INDICATE SERIAL PRINTER ATTACHED
0964 80 75 20 OR CL,00100000B ; CHECK FOR NO PARALLEL PRINTERS
0967 80 75 0B JNZ F19_B ; DO NOTHING IF PARALLEL PRINTER ; ATTACHED
096A 80 80 40 OR CL,01000000B ; INDICATE 1 PRINTER ATTACHED
096D 83 C7 06 0008 R 02FB MOV PRINTER_BASE,2F8H ; STORE ON-BOARD RS232 BASE IN ; PRINTER BASE
0970 86 08 0E 0011 R F19_A: OR BYTE PTR EQUF_FLAG+1,CL ; STORE AS SECOND BYTE
0973 86 33 00 DB XOR DX,DX ; POINT TO FIRST SERIAL PORT
0976 86 F6 8F JZ CL,40H ; SERIAL PRINTER ATTACHED?
0979 87 74 1B TEST CL,004H ; SERIAL PRINTER ATTACHED?
097C 87 81 3E 0000 R 02FB CMP RS232_BASE,02FBH ; PRINTER IN FIRST SERIAL PORT
097F 87 7A 01 JNE F19_B ; YES, JUMP
0982 87 4C 42 INC DX ; NO POINT TO SECOND SERIAL PORT
0985 87 9B 0087 F19_B: MOV AX,87H ; INIT SERIAL PRINTER
0988 87 70 4F INT 14H ; SERIAL PRINTER
098B 87 9B 0011B MOV AX,011BH ; SEND CANCEL COMMAND TO
098E 87 94 0A INT 14H ; SERIAL PRINTER

A-22 ROM BIOS
**ROM BIOS A-23**

**F19_C**: MOV DX, 0201H  ; GET MFG. / SERVICE MODE INFO

**F19_0**: JMP START  ; ELSE GO TO BEGINNING OF POST

**F19_1**: CMP AL, 00100000B ; SERVICE MODE LOOP?

**F19_9**: BRANCH TO START

**F19_B**: CMP RESET_FLAG, 4321H ; DIAG. CONTROL PROGRAM RESTART?

**F19_3**: JE F19_0  ; NO, GO BOOT

**F19_5**: CMP AL, 00010000B ; MFG DCP RUN REQUEST

**F19_7**: JE F19_3  ; SET WARM START INDICATOR IN CASE

**F19_9**: INT 19H  ; OF CARTRIDGE RESET

**FA0**: SUB AX, AX  ; TO GO THE BOOT LOADER

**FA1**: ASSUME DS:ABS0

**FA3**: CL1

**FA5**: MOV DS, AX  ; RESET TIMER INT.

**FA7**: MOV INT_PTR, OFFSET TIMER_INT

**FA9**: INT 20H  ; ENTER DCP THROUGH INT. 20H

**FAA**: This subroutine is the general error handler for the POST.

**FAE**: ENTRY REQUIREMENTS:

**F19_3**: SI = OFFSET (ADDRESS) OF MESSAGE BUFFER

**FAH**: BX = ERROR CODE FOR MANUFACTURING OR SERVICE MODE

**FAE**: REGISTERS ARE NOT PRESERVED

**FAC**: LOCATION "POST_ERR" IS SET NON-ZERO IF AN ERROR OCCURES IN

**FAD**: CUSTOMER MODE

**FAB**: SERVICE / MANUFACTURING FLAGS AS FOLLOWS:

**FAC**: (HIGH NIBBLE OF

**FAD**: PORT 201)

**FAG**: 0000 = MANUFACTURING (BURN-IN) MODE

**FBE**: 0001 = MANUFACTURING (SYSTEM TEST) MODE

**FCD**: 0010 = SERVICE MODE (LOOP POST)

**FCE**: 0011 = SERVICE SINE (SYSTEM TEST)

**FA8**: PROC NEAR

**FAE**: MOV DX, 201H  ; GET MFG BITS

**FAC**: IN AL, DX  ; ISOLATE BITS OF INTEREST

**FAD**: AND AL, OFOH

**FAB**: JNZ EM0

**FAC**: JMP NMG, MFG, MFG, MFG

**FAD**: MOV AL, 00010000B ; MANUFACTURING MODE (BURN-IN)

**FAB**: JNE EM1

**FAC**: JMP NMG, MFG, MFG, MFG

**FAD**: MOV DH, AL  ; SERVICE MODE

**FAB**: JMP NMG, MFG, MFG, MFG

**FAC**: MOV AL, 00010000B ; ERROR CODE ABOVE OAH (CRT STARTED

**FAD**: OR SYSTEM TEST) DISPLAY POSSIBLE?  ; DO BEEP OUTPUT IF BELOW 10H

**FBE**: JLP

**FCA**: PUSH BX  ; SAVE ERROR AND MODE FLAGS

**FCD**: PUSH SI

**FCC**: MOV AH, 2  ; SET CURSOR

**FCE**: MOV DX, 1521H  ; ROW 21, COL. 33

**FCD**: MOV BH, 7 ; PAGE 7

**FCC**: INT 10H

**FCE**: MOV SI, OFFSET ERROR_ERR

**FCD**: MOV CX, 5  ; PRINT WORD "ERROR"

**FCC**: EM_O: MOV AL, CS:[SI]

**FCD**: INC SI

**FCC**: CALL PRT_HEX

**FCD**: LOOP EM_G

**FCE**: ; LOOK FOR A BLANK SPACE TO POSSIBLY PUT CUSTOMER LEVEL ERRORS (IN

**FCD**: ; CASE OF MULTI ERROR)

**FCC**: MOV DH, 16H

**FCD**: EM_1: MOV AH, 2  ; SET CURSOR

**FCC**: INT 10H

**FCD**: MOV AL, 207 ; MULTIPLE ERROS

**FCC**: MOV AH, B ; READ CHARACTER THIS POSITION

**FCD**: INC DL ; POINT TO NEXT POSITION

**FCC**: CMP AL, \"\" ; BLANK?

**FCD**: JNE EM_1 ; GO CHECK NEXT POSITION, IF NOT

**FCC**: POP DX

**FCD**: POP SI

**FCC**: POP BX

**FCD**: POP DX

**FCC**: CMP DH, 00100000B ; SERVICE MODE?

**FCD**: JE SERV_OUT

**FCC**: MOV AL, CS:[SI]  ; GET ERROR CHARACTER

**FCD**: CALL PRT_HEX ; DISPLAY IT0E

**FCC**: CMP BH, 20H ; ERROR BELOW 20? (NEG TROUBLE?)

**FCD**: JNL EM_2

**FCC**: JMP TOLTPO ; HALT SYSTEM IF 50.

**FCD**: ASSUME DS:XXDATA

**FCC**: EM_2: PUSH DS

**FCD**: PUSH AX

**FCC**: MOV AX, XXDATA

**FCD**: MOV DS, AX

**FCC**: MOV POST_ERR, BH  ; SET ERROR FLAG NON-ZERO

**FCD**: POP AX

**FCC**: POP DS

**FCD**: ASSUME DS:NOTHING

**FCC**: RET ; RETURN TO CALLER

**FCD**: ;
OA29 SERV_OUT: MOV AL,BH ; PRINT MSB
OA39 BA C7 MOV AX,CS ; SET CODE SEG: STACK SEG
OA99 53 PUSH BX CALL XPC_BYTE ; DISPLAY IT
OA2C EB 18A9 R POP BX CALL XPC_BYTE
OA2F 5B JMP AL,BL ; PRINT LSB
OA30 BA C3 MOV AL,BL ; PRINT LSB
OA32 EB 18A9 R CALL XPC_BYTE
OA35 E9 0ABB R JMP TOTLTP0
OA38 FA BEEPS: CLI
OA39 8C CB MOV AX,CS ; SET CODE SEG: STACK SEG
OA38 8E B0 MOV SS,AX ; (STACK IS LOST, BUT THINGS ARE
OA42 B3 01 EB: MOV BL,1 ; OVER, ANYWAY)
OA44 E9 FF31 R JMP BEEP ;
OA47 E2 FE EBO: LOOP EB0 ; WAIT (BEEPER OFF)
OA49 FE CA DEC DL ; DONE YET?
OA48 75 F5 MJNZ EB ; LOOP IF NOT
OA4D 80 FF 05 CMP BH,05H ; 64K CARD ERROR?
OA50 75 69 JNE TOTLTP0 ; END IF NOT
OA52 80 FE 20 CMP DX,00100000B ; SERVICE MODE?
OA55 74 05 JE EB1
OA57 80 FE 40 CMP DH,01000000B ;
OA54 75 6F JNE TOTLTP0 ; END IF NOT
OA5C 83 01 EB1: MOV BL,1 ; ONE MORE BEEP FOR 64K ERROR IF IN
OA5E E9 FF31 R JMP BEEP
OA61 MFG_OUT: CLI
OA64 E4 61 MOV AX,PORT_B ;
OA66 24 FC AND AL,0FCF ;
OA68 EB 61 OUT PORT_B,AL ;
OA6B BA 0011 MOV DX,11H ; SEND DATA TO ADDRESSES 11,12
OA68 BA C7 MOV AL,BH ;
OA69 EE OUT DX,AL ; SEND HIGH BYTE
OA6E 42 INC DX
OA6F BA C3 MOV AL,BL ;
OA71 EE OUT DX,AL ; SEND LOW BYTE
OA72 BB ---- R ; INIT. ON-BOARD RS232 PORT FOR COMMUNICATIONS W/MFG MONITOR
OA75 8E DB MOV DS,AX ; ASSUME DS:XXDATA
OA77 BC CB MOV AX,CS ; POINT TO DATA SEGMENT CONTAINING
OA79 BE D0 MOV SS,AX ; CHECKPOINT #
OA7E BA C2 MOV SP,OFFSET EX_O ; SET STACK FOR RTN
OA85 BA 028 R MOV SP,OFFSET EX1 ;
OA89 BA 02FB MOV DX,02FBH ; LINE CONTROL REG. ADDRESS
OA8B E9 F085 R JMP SB250 ; GO SET UP FOR 9600, 000, 2 STOP
OAAB BB CA MO1: MOV CX,DX ; DX CAME BACK WITH XMIT REG
OAAE BA 02FC MOV DX,02FC ; ADDRESS IN IT
OA99 2A C0 SUB AL,AL ; SET DTR AND RTS LOW SO POSSIBLE
OA9B EE OUT DX,AL ; WRAP PLUG Won'T CONFUSE THINGS
OA9C BA 02FE MOV AX,02FEH ; MODEM STATUS REG
OA9E EC OUT DX,AL ;
OA9F EC MOV AX,AL ;
OA90 24 10 AND AL,00010000B ; CTS UP YET?
OA92 74 FB JZ M02 ; LOOP TILL IT IS
OA94 4A 4A DEC DX ; SET DX=2FD (LINE STATUS REG)
OA95 B7 D1 XCHG DX,DX ; POINT TO XMIT DATA REG
OA97 A0 005 R MOV AX,MFG_TST ; GET MFG ROUTINE ERROR INDICATOR
OA9A EE OUT DX,AL ; (MAY BE WRONG FOR EARLY ERRORS)
OA9B EB 00 JMP *+2 ; DELAY
OA9D B7 D1 XCHG DX,DX ; POINT DX=2FD
OA9F EC MO3: IN AL,DX ; TRANSMIT EMPTY?
OAAB 24 20 AND AL,00100000B ;
OAAB EE JMP *+2 ; DELAY
OAAC 74 FB JZ M03 ; LOOP TILL IT IS
OAAD B7 D1 XCHG DX,DX ;
OAAE BA C7 MOV AL,BH ; GET MSB OF ERROR WORD
OA9A EE OUT DX,AL ;
OA9B EB 00 JMP *+2 ; DELAY
OA9D B7 D1 XCHG DX,DX ;
OA9F EC MO4: IN AL,DX ; WAIT FOR XMIT EMPTY
OA9B 24 20 AND AL,00100000B ;
OA9B EB 00 JMP *+2 ; DELAY
OA9D 74 F9 JZ M04 ;
OA9E BA C3 MOV AL,BL ; GET LSB OF ERROR WORD
OA9B B7 D1 XCHG DX,DX ;
OA9A EE OUT DX,AL ;
OA9B TOTLTP0.
OA9B FA CLI ; DISABLE INTS.
OA9C 2A C0 SUB AL,AL ;
OA9E 00 F2 OUT 0F2H,AL ; STOP DISKETTE MOTOR
OA9C 00 EE 00 OUT 0A0H,AL ; DISABLE NMI
OA9C 02 F4 HLT ;
OA9C 03 RET
OA9C 04 E_MSG ENDP

A-24 ROM BIOS
SUBROUTINE TO INITIALIZE INSB250 PORTS TO THE MASTER RESET STATUS. THIS ROUTINE ALSO TESTS THE PORTS' PERMANENT ZERO BITS.

EXPECTS TO BE PASSED:

(DX) = ADDRESS OF THE B250 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 MODM 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:

XF9 = INTR ENBL REG = 0 ALL INTERRUPTS DISABLED

XFA = INTR ID REG = 0000001B NO INTERRUPTS PENDING

XFB = LINE CTRL ALL BITS LOW

XFC = MODM CTRL REG = 0 ALL BITS LOW

XFD = LINE STAT REG = 01100000B TRANSMITTER HOLDING REGISTER AND TRANSMITTER EMPTY ON

XFE = MODM STAT REG = XXXX0000BB WHERE X'S REPRESENT INPUT SIGNALS

REGISTERS DX, AL, AND BL ARE ALTERED. NO OTHER REGISTERS USED.

I8250 PROC NEAR

IN AL,DX ; READ RCVR BUFFER BUT IGNORE CONTENTS

MOV BL,2 ; ERROR INDICATOR

CALL RR5 ; READ INTR ENBL REG

AND AL,11110000B ; BITS 4-7 OFF?

JNE AT20 ; NO - ERROR

CALL RR1 ; READ INTR ID REG

AND AL,11110000B ; BITS 3-7 OFF?

JNE AT20 ; NO

INC DX ; LINE CTRL REG

CALL RR1 ; READ MODM CTRL REG

AND AL,11100000B ; BITS 5-7 OFF?

JNE AT20 ; NO

INC RR1 ; READ LINE STAT REG

AND AL,10000000B ; BIT 7 OFF?

JNE AT20 ; NO

MOV AL,60H

OUT DX,AL ; 1/0 DELAY

JMP +2

INC DX ; MODM STATUS REG

XOR AL,AL

OUT DX,AL ; WIRED BITS WILL BE HIGH

CALL RR3 ; CLEAR BITS 0-3 IN CASE THEY'RE ON

AFTER WRITING TO STATUS REG

RECEIVER BUFFER

IN AL,DX ; IN CASE WRITING TO PORTS CAUSED DATA READY TO GO HIGH!

CLC

SUB DX,6

RET

I8250 ENDP

SUBROUTINE TO TEST A PARTICULAR B250 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 B250 INTERRUPT IS GENERATED.

IT EXPECTS TO BE PASSED:

(AH) = BIT # TO BE TESTED

(BL) = INTERRUPT IDENTIFIER

(D0) = RECEIVED DATA AVAILABLE OR TRANSMITTER HOLDING REGISTER EMPTY INTERRUPT TEST

(1) = RECEIVER LINE STATUS OR MODM STATUS INTERRUPT TEST

(BH) = BITS WHICH DETERMINE WHICH INTERRUPT IS TO BE CHECKED

(D0) = MODM STATUS

(2) = TRANSMITTER HOLDING REGISTER EMPTY

(4) = RECEIVED DATA AVAILABLE

(6) = RECEIVER LINE STATUS

(CK) = 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) = MODM STATUS INTERRUPT

(DX) = ADDRESS OF THE LINE STATUS OR MODM STATUS REGISTER IT RETURNS;

(AL) = 0 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 MODM STATUS REGISTER DEPENDING ON WHICH ONE WAS TESTED

(DX) = ADDRESS OF INTERRUPT ID REGISTER FOR RECEIVED DATA AVAILABLE ON 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.

ROM BIOS A-25
A-26 ROM BIOS
POSTG PROC NEAR
ASSUME DS:ABS0

CLD
SUB DX, DX
SUB CX, AX

MOV DS, AX
SET DS TO ABS0

MOV BX, DATA_Word[RESET_FLAG-Data]
MOV AX, DX
MOV DX, AX
.
.
.
.

PI: CMP BX, 4321H
JMP P2
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
INT 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)=6 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.COLUM (0,0) IS UPPER LEFT

(BH) = PAGE NUMBER (MUST BE 0 FOR GRAPHICS MODES)

READ CURSOR POSITION

(BH) = PAGE NUMBER (MUST BE 0 FOR GRAPHICS MODES)

ON EXIT (DH,DL) = ROW.COLUM OF CURRENT CURSOR
(CH,CL) = CURSOR MODE CURRENTLY SET

(AH)=3
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.COLUM 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 0,1, 0-3 FOR
MODES 2B3)

IF BIT 7 (BOH) OF AL=1
READ/WRITE CRT/CPU PAGE REGISTERS

(AL) = BOH READ CRT/CPU PAGE REGISTERS

(AL) = B1H SET CPU PAGE REGISTER

(BL) = VALUE TO SET

(AL) = B2H SET CRT PAGE REGISTER

(BH) = VALUE TO SET

(AL) = B3H 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 (BOH) 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.COLUM OF UPPER LEFT CORNER OF
SCROLL

(DH,DL) = ROW.COLUM 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.COLUM OF UPPER LEFT CORNER OF
SCROLL

(DH,DL) = ROW.COLUM 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
(WAH) 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 40H (location O010H) IS USED TO POINT TO THE 1K BYTE TABLE CONTAINING THE FIRST 128 CHAR (0-127).

interrupt 1F8H (location 0007CH) IS USED TO POINT TO THE 1K BYTE TABLE CONTAINING THE SECOND 128 CHAR (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'ED 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 GRAPHIC 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
(BH) = 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 BBOH SEGMENT VALUE ONLY REACHES THE FIRST 16K. BIOS USES THE CONTENTS OF THE CPU PAGE REG (BIT 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
<table>
<thead>
<tr>
<th>VIDEO GATE ARRAY REGISTERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>PORT 30A OUTPUT</td>
</tr>
<tr>
<td>REG 0 MODE CONTROL 1 REGISTER</td>
</tr>
<tr>
<td>01H +HI BANDWIDTH LOW BANDWIDTH</td>
</tr>
<tr>
<td>02H +GRAPHICS ALPHA</td>
</tr>
<tr>
<td>04H +BW</td>
</tr>
<tr>
<td>08H +VIDEO ENABLE</td>
</tr>
<tr>
<td>10H +16 COLOR GRAPHICS</td>
</tr>
<tr>
<td>REG 1 PALETTE MASK REGISTER</td>
</tr>
<tr>
<td>01H PALETTE MASK 0</td>
</tr>
<tr>
<td>02H PALETTE MASK 1</td>
</tr>
<tr>
<td>04H PALETTE MASK 2</td>
</tr>
<tr>
<td>0BH PALETTE MASK 3</td>
</tr>
<tr>
<td>REG 2 BORDER COLOR REGISTER</td>
</tr>
<tr>
<td>01H BLUE</td>
</tr>
<tr>
<td>02H GREEN</td>
</tr>
<tr>
<td>04H RED</td>
</tr>
<tr>
<td>08H INTENSITY</td>
</tr>
<tr>
<td>REG 3 MODE CONTROL 2 REGISTER</td>
</tr>
<tr>
<td>01H RESERVED -- MUST BE ZERO</td>
</tr>
<tr>
<td>02H +ENABLE BLINK</td>
</tr>
<tr>
<td>04H RESERVED -- MUST BE ZERO</td>
</tr>
<tr>
<td>08H +2 COLOR GRAPHICS (64X200 2 COLOR ONLY)</td>
</tr>
<tr>
<td>REG 4 RESET REGISTER</td>
</tr>
<tr>
<td>01H +ASYNCHRONOUS RESET</td>
</tr>
<tr>
<td>02H +SYNCHRONOUS RESET</td>
</tr>
<tr>
<td>REGS 10 TO 1F PALETTE REGISTERS</td>
</tr>
<tr>
<td>01H BLUE</td>
</tr>
<tr>
<td>02H GREEN</td>
</tr>
<tr>
<td>04H RED</td>
</tr>
<tr>
<td>08H INTENSITY</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VIDEO GATE ARRAY STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>PORT 30A INPUT</td>
</tr>
<tr>
<td>01H +DISPLAY ENABLE</td>
</tr>
<tr>
<td>02H +LIGHT PEN TRIGGER SET</td>
</tr>
<tr>
<td>04H +LIGHT PEN SWITCH MADE</td>
</tr>
<tr>
<td>08H +VERTICAL RETRACE</td>
</tr>
<tr>
<td>10H +VIDEO DOTS</td>
</tr>
</tbody>
</table>

ASSUME CS: CODE, DS: DATA, ES: VIDEO_RAM

<table>
<thead>
<tr>
<th>MO010 LABEL WORD</th>
<th>TABLE OF ROUTINES WITHIN VIDEO I/O</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCE9</td>
<td>DW OFFSET SET_MODE</td>
</tr>
<tr>
<td>OCE9 0DA5 R</td>
<td>DW OFFSET SET_TYPE</td>
</tr>
<tr>
<td>OCE9 E48E R</td>
<td>DW OFFSET SET_CPS</td>
</tr>
<tr>
<td>OCE9 E488 R</td>
<td>DW OFFSET READ_CURSOR</td>
</tr>
<tr>
<td>OCEF ES20 R</td>
<td>DW OFFSET READ_CURSOR</td>
</tr>
<tr>
<td>OCF1 F751 R</td>
<td>DW OFFSET ACT_DISP_PAGE</td>
</tr>
<tr>
<td>OCF3 E4B3 R</td>
<td>DW OFFSET ACT_SCROLL_UP</td>
</tr>
<tr>
<td>OCF5 ES53 R</td>
<td>DW OFFSET SCROLL_DOWN</td>
</tr>
<tr>
<td>OCF7 ES5F R</td>
<td>DW OFFSET SCROLL_UP</td>
</tr>
<tr>
<td>OCF9 F0E4 R</td>
<td>DW OFFSET READ_AC_CURRENT</td>
</tr>
<tr>
<td>OCFB F113 R</td>
<td>DW OFFSET WRITE_AC_CURRENT</td>
</tr>
<tr>
<td>OCFD F12C R</td>
<td>DW OFFSET WRITE_C_CURRENT</td>
</tr>
<tr>
<td>OCFF ES54 R</td>
<td>DW OFFSET SET_COLOR</td>
</tr>
<tr>
<td>00D1 F187 R</td>
<td>DW OFFSET WRITE_DOT</td>
</tr>
<tr>
<td>00D3 F146 R</td>
<td>DW OFFSET READ_DOT</td>
</tr>
<tr>
<td>00D5 1992 R</td>
<td>DW OFFSET WRITE_TTY</td>
</tr>
<tr>
<td>00D7 ES5B R</td>
<td>DW OFFSET VIDEO_STATE</td>
</tr>
<tr>
<td>00D9 ES55 R</td>
<td>DW OFFSET SET_PALETTE</td>
</tr>
</tbody>
</table>

= 0022

<table>
<thead>
<tr>
<th>MO010L EQU</th>
<th>$-MO010</th>
</tr>
</thead>
<tbody>
<tr>
<td>000B</td>
<td>VIDEO_10 PROC NEAR</td>
</tr>
<tr>
<td>000B FB</td>
<td>STI ; INTERRUPTS BACK ON</td>
</tr>
<tr>
<td>000C FC</td>
<td>CALL ; SET DIRECTION FORWARD</td>
</tr>
<tr>
<td>000D 06</td>
<td>PUSH ES</td>
</tr>
<tr>
<td>000E 01</td>
<td>CALL ; SAVE SEGMENT REGISTERS</td>
</tr>
<tr>
<td>000F 52</td>
<td>PUSH DX</td>
</tr>
<tr>
<td>0010 51</td>
<td>PUSH CX</td>
</tr>
<tr>
<td>0011 53</td>
<td>PUSH BX</td>
</tr>
<tr>
<td>0012 56</td>
<td>PUSH SI</td>
</tr>
<tr>
<td>0013 57</td>
<td>PUSH DI</td>
</tr>
<tr>
<td>0014 50</td>
<td>CALL DS</td>
</tr>
<tr>
<td>0015 8A C4</td>
<td>MOV AX, AL ; AX GET INTO LOW BYTE</td>
</tr>
<tr>
<td>0017 32 E4</td>
<td>XOR AH, AH ; ZERO TO HIGH BYTE</td>
</tr>
<tr>
<td>0019 D1 E0</td>
<td>MOV AX, 01 ; *2 FOR TABLE Lookup</td>
</tr>
<tr>
<td>001B BB F0</td>
<td>MOV SI, AX ; PUT INTO SI FOR BRANCH</td>
</tr>
<tr>
<td>001D 3D 0022</td>
<td>CMP AX, MO010 ; TEST FOR WITHIN RANGE</td>
</tr>
<tr>
<td>0020 72 04</td>
<td>JB C1 ; BRANCH AROUND BRANCH</td>
</tr>
<tr>
<td>0022 58</td>
<td>POP AX ; THROW AWAY THE PARAMETER</td>
</tr>
<tr>
<td>0023 E9 FF70 R</td>
<td>JMP VIDEO_RETURN ; DO NOTHING IF NOT IN RANGE</td>
</tr>
<tr>
<td>0026 E8 1388 R</td>
<td>CALL DS</td>
</tr>
<tr>
<td>0029 BB B000</td>
<td>MOV AX, OBD00H ; SEGMENT FOR COLOR CARD</td>
</tr>
<tr>
<td>0030 B0 3E 0049 R 09</td>
<td>CMP CRC_MODE, 9 ; IN MODE USING 32K REGEN</td>
</tr>
<tr>
<td>0031 72 09</td>
<td>CALL C2 ; NO_JUMP</td>
</tr>
<tr>
<td>0032 BA 26 008A R</td>
<td>MOV AH, PGSAT ; GET COPY OF PAGE REGS</td>
</tr>
<tr>
<td>0033 B0 E4 38</td>
<td>AND AH, CPUREG ; ISOLATE CPU REG</td>
</tr>
<tr>
<td>003A D0 EC</td>
<td>SHR AH, 1 ; SHIFT INTO MADE INTO SEGMENT VALUE</td>
</tr>
<tr>
<td>003C BE CO C2</td>
<td>MOV ES, AX ; SET UP TO POINT AT VIDEO_RAM AREA</td>
</tr>
<tr>
<td>003E 58</td>
<td>POP AX ; RECOVER VALUE</td>
</tr>
<tr>
<td>003F BA 26 0049 R</td>
<td>JMP MOV AH, CRC_MODE ; GET CURRENT MODE INTO AH</td>
</tr>
</tbody>
</table>

| Video_10 | ENDP |

---

ROM BIOS A-31

Appendix A
SET MODE

; THIS ROUTINE INITIALIZES THE ATTACHMENT TO
; THE SELECTED MODE. THE SCREEN IS BLANKED.
; INPUT
; (AL) = MODE SELECTED (RANGE 0-8)
; OUTPUT
; NONE

0048 M0050 LABEL WORD ; TABLE OF REGEN LENGTHS
0048 0800 DW 2048 ; MODE 0 40X25 BW
004A 0800 DW 2048 ; MODE 1 40X25 COLOR
004C 1000 DW 4096 ; MODE 2 80X25 BW
004E 1000 DW 4096 ; MODE 3 80X25 COLOR
0050 4000 DW 16384 ; MODE 4 320X200 4 COLOR
0052 4000 DW 16384 ; MODE 5 320X200 4 COLOR
0054 4000 DW 16384 ; MODE 6 640X200 BW
0056 0000 DW 0 ; MODE 7 INVALID
0058 4000 DW 16384 ; MODE 8 160X200 16 COLOR
005A 8000 DW 32768 ; MODE 9 320X200 16 COLOR
005C 8000 DW 32768 ; MODE A 640X200 4 COLOR

005E M0060 LABEL BYTE
005E 28 28 50 50 28 28 50 00 14 28 50

;-------- TABLE OF GATE ARRAY PARAMETERS FOR MODE SETTING
0069 M0070 LABEL BYTE
0069 0C 0F 00 02 DB OCH,OFH,0,2 ; GATE ARRAY PARMS
006D 0F 00 02 DB OCH,OFH,0,2 ; GATE ARRAY PARMS
0071 0F 00 02 DB OCH,OFH,0,2 ; GATE ARRAY PARMS
0075 0F 00 02 DB OCH,OFH,0,2 ; GATE ARRAY PARMS
0079 0A 03 00 00 DB OAH,03H,0,0 ; GATE ARRAY PARMS
007E 0E 03 00 00 DB OEH,03H,0,0 ; GATE ARRAY PARMS
0081 0E 01 00 08 DB OEH,01H,0,8 ; GATE ARRAY PARMS
0085 00 00 00 00 DB 00H,00H,0,0 ; GATE ARRAY PARMS
0089 1A 0F 00 00 DB 1AH,0FH,0,0 ; GATE ARRAY PARMS
008F 0B 03 00 00 DB 0BH,03H,0,0 ; GATE ARRAY PARMS

;-------- TABLES OF PALETTE COLORS FOR 2 AND 4 COLOR MODES

0095 M0072 LABEL BYTE
0095 00 0F 00 00 DB 0,0FH,0,0
= 0004 M0072L EQU $-M0072 ; ENTRY LENGTH
0099 0F 00 00 00 DB 0,0FH,0,0
009D 0F 00 00 00 DB 0,0FH,0,0

00A9 M0074 LABEL BYTE
00A9 02 04 06 DB 0,2,4,6

00AD M0075 LABEL BYTE
00AD 00 03 05 OF DB 0,3,5,OFH

00A5 50 SET_MODE PROC NEAR
00A5 50 PUSH AX ; SAVE INPUT MODE ON STACK
00A6 24 7F AND AL,7FH ; REMOVE CLEAR REGEN SWITCH
00A8 3C 07 CMP AL,7 ; CHECK FOR VALID MODES
00AA 74 04 JE C3 ; MODE 7 IS INVALID
00AC 3C 08 CMP AL,0BH
00AE 72 02 JC C4 ; GREATER THAN A IS INVALID
00B0 80 00 C3: MOV AL,0 ; CLEAR MODE 0
00B2 3C 02 C4: CMP AL,2 ; CHECK FOR MODES NEEDING 128K
00B4 74 08 JE C5
00B6 3C 03 CMP AL,3
00BB 74 04 JE C5
00BA 3C 09 CMP AL,0BH
00BC 72 0A JNC C6
00B8 81 0015 R 0080 DB TRUE_MEM,128 ; DO WE HAVE 128K?
00CC 73 02 JC C6 ; YES, JUMP
00C6 80 00 MOV AL,0 ; NO, DEFAULT TO MODE 0
00CB 0A 3D MOV AL,0 ; ADDRESS OF COLOR CARD
00CD 02 049 R MOV CRT_MODE,AL ; SAVE MODE IN AH
00D0 89 0E 0063 R MOV ADDR_BLA,DX ; SAVE ADDRESS OF BASE
00D4 88 FA MOV DI,AX ; SAVE MODE IN DI
00D6 BA 03A0 MOV DX,VGA_CTL ; POINT TO CONTROL REGISTER
00D9 8C 0065 R IN AL,DX ; SYNC CONTROL REG TO ADDRESS
00DA 24 00 MOV AL,CRT_MODE_SET ; GET LAST MODE SET
00DE 24 F7 AND AL,0FH ; TURN OFF VIDEO
00EE 00 OUT DX,AL ; SET IN GATE ARRAY

A-32 ROM BIOS
; ----- SET DEFAULT PALETTES
00E3 BB C7 MOV AX,01 ; SET MODE
00E5 BB 10 MOV AH,10H ; SET PALETTE REG 0
00E7 BB 00F R MOV BX,OFFSET M0072 ; POINT TO TABLE ENTRY
00EA 3C 06 CMP AL,6 ; 2 COLOR MODE?
00EC 74 0F JE C7 ; YES, JUMP
00EE BB 0DA 0 MOV BX,OFFSET M0075 ; POINT TO TABLE ENTRY
00F1 3C 05 CMP AL,5 ; CHECK FOR 4 COLOR MODE
00F3 74 0B JE C7 ; YES, JUMP
00F5 3C 04 CMP AL,4 ; CHECK FOR 4 COLOR MODE
00F7 74 04 JE C7 ; YES JUMP
00F9 3C 0A CMP AL,0AH ; CHECK FOR 4 COLOR MODE
00FB 75 11 JNE C9 ; NO, JUMP
00FD B9 0004 C7 MOV CX,4 ; NUMBER OF REGS TO SET
00ED RA C4 MOV AL,01 ; GET REG NUMBER
00F0 EE OUT DX,AL ; SELECT IT
00F2 2E BA 07 MOV AL,CS:[BX] ; GET DATA
00F4 EE OUT DX,AL ; SET IT
00F6 FE C4 INC AH ; NEXT REG
00F8 43 INC BX ; NEXT TABLE VALUE
00FA EA F4 LOOP CB
00FC EB 08 JMP SHORT C11
00FE B9 0010 ; ----- SET PALETTES FOR DEFAULT 16 COLOR
00F0 C9 MOV CX,16 ; NUMBER OF PALETTES, AH IS REG
00F2 BA C4 MOV AL,AH ; COUNTER
00F4 EE OUT DX,AL ; SELECT IT
00F6 EE OUT DX,AL ; SET PALETTE VALUE
00F8 17 E2 LOOP C10
; ----- SET UP M0 & M1 IN PAGREG
00FA BB C7 MOV AX,01 ; SET CURRENT MODE
00FB 32 DB XOR BL,BL ; SET UP FOR ALPHA MODE
00FD 3C 03 CMP AL,4 ; IN ALPHA MODE
00FF 72 0B JC C12 ; YES, JUMP
00F1 2B 40 MOV BL,40H ; SET UP FOR 16K REGEN
00F3 23 0C CMP AL,09H ; MODE USE 16K
00F5 72 02 JC C15 ; YES, JUMP
00F7 B3 C0 MOV BL,0CH ; SET UP FOR 32K REGEN
00F9 B4 03DF ; SET PORT ADDRESS OF PAGREG
00FA AO 008A R MOV AL,PAGDAT ; GET LAST DATA OUTPUT
00FB 2F 24 3F AND AL,3FH ; CLEAR M0 & M1 BITS
00FC 31 AA C3 OR AL,AL ; SET NEW BITS
00FD 3E EE OUT DX,AL ; STUFF BACK IN PORT
00FE 34 A2 008A R MOV PAGDAT,AL ; SAVE COPY IN RAM
; ----- ENABLE VIDEO AND CORRECT PORT SETTING
00F0 BB C7 MOV AX,01 ; GET CURRENT MODE
00F2 32 EA XOR AH,AH ; INTO AX REG
00F4 BB 0004 MOV CX,M0070L ; SET TABLE ENTRY LENGTH
00F6 F7 E1 MUL CX ; TIMES MODE FOR OFFSET INTO TABLE
00F8 40 MOV BX,AX ; TABLE OFFSET IN BX
00FA 01 C1 0659 MOV BX,OFFSET M0070 ; ADD BX,OFFSET M0070 ; ADD TABLE START TO OFFSET
00FC 46 2B BA 27 MOV AH,CS:[BX] ; SAVE MODE SET AND PALETTE
00FD 49 27 02 MOV AL,CS:[BX + 2] ; TILL WE CAN PUT THEM IN RAM
00FE 40 F0 MOV SI,AX ;
00FF FA CLI ; DISABLE INTERRUPTS
00F0 E8 E675 R CALL MODE_ALIVE ; KEEP MEMORY DATA VALID
00F2 B0 10 MOV AL,10H ; DISABLE NMI AND HOLD REQUEST
00F4 55 EE OUT NMI_PORT,AL ;
00F6 55 0A MOV DX,VGA_CLIP ;
00F8 BA 04 OUT DX,AL ;
00F9 EE 5C SE TO GATE ARRAY
00FA BD 02 MOV AL,2 ; SET SYNCHRONOUS_RESET
00FB EE 5E OUT DX,AL ; DO IT
; WHILE THE GATE ARRAY IS IN RESET STATE, WE CANNOT ACCESS RAM
00FC 60 BB C6 MOV AX,SI ; RESTORE NEW MODE SET
00FD 60 B4 F7 AND AH,OF7H ; TURN OFF VIDEO ENABLE
00FE 32 C0 XOR AL,AL ; SET UP TO SELECT VGA REG 0
00FF 67 EE OUT DX,AL ; SELECT IT
00F0 6B EE XCHG AH,AL ; AH IS VGA REG COUNTER
00F1 68 EE OUT DX,AL ; SET MODE
00F2 6B 04 MOV AL,4 ; POINT TO RESET REG
00F3 EE EXIT SEND TO GATE ARRAY
00F4 60 EE OUT DX,AL ; SELECT IT
00F5 EE 32 C0 XOR AL,AL ;
00F6 EE 70 OUT DX,AL ; REMOVE RESET FROM VGA
; NOW OKAY TO ACCESS RAM AGAIN
00F7 EE 80 MOV AL,80H ; ENABLE NMI AGAIN
00F8 EE OUT NMI_PORT,AL ;
00F9 EE CALL MODE_ALIVE ; KEEP MEMORY DATA VALID
00FA 6B 00 STI ; ENABLE INTERRUPTS
00FB FA JMP SHORT C14 ;
00FC 13 B7 MOV AH,AL ; GET VGA REG NUMBER
00FD 7E 2E BA 07 OUT DX,AL ; SELECT REG
00FE 78 EE MOV AL,CS:[BX] ; GET TABLE VALUE
00FF 81 EE OUT DX,AL ; PUT IN VGA REG
00F0 82 43 INC BX ; NEXT IN TABLE
00F1 83 EE INC AH ; NEXT REG
00F2 EB E2 F4 LOOP C13 ; DO ENTIRE ENTRY
; ----- SET UP CRT AND CPU PAGE REGS ACCORDING TO MODE & MEMORY SIZE
00F4 87 B5 030F MOV DX,PAGDAT ; SET TO ADDRESS OF PAGREG
00F6 8B A0 008A R MOV AL,PAGDAT ; GET LAST DATA OUTPUT
00F8 B0 24 0C AND AL,OC0H ; CLEAR REG BITS
00F9 DF 36 MOV BL,3EH ; SET UP FOR GRAPHICS MODE WITH 32K REGEN
00FA 91 AE B0 TEST AL,80H ; IN THIS MODE?
00FB 75 0C JNC C15 ; YES, JUMP
00FC 93 B3 3F MOV BL,3FH ; SET UP FOR 16K REGEN AND 128K MEMORY
00FD 97 B1 3E 0015 R 0080 CMP TRUE_MEM,128 ; DO WE HAVE 128K?
00FE 99 73 02 MOV BL,18H ; SET UP FOR 16K REGEN AND 64K MEMORY
00FF 9F 53 1B
OE0A 0A C3 C15: OR AL, BL ; COMBINE MODE BITS AND REG VALUES
OE0A EE OUT DX, AL ; SET PORT
OE0A A2 008A R MOV PAGDAT, AL ; SAVE COPY IN RAM
OE0A B8 26 0066 R MOV AX, SI ; PUT MODE SET & PALETTE IN RAM
OE0A ED A2 0066 R MOV CRTC_PALLETE, AL
OE0B E4 61 IN AL, PORT_B ; GET CURRENT VALUE OF 8255 PORT B
OE0B D4 78 MOV AX, OFBH ; SET UP GRAPHICS MODE
OE0B F4 C0 02 TEST AH, 2 ; JUST SET ALPHA MODE IN VGA?
OE0B F7 75 02 JNZ C16 ; YES, JUMP
OE0B OC 04 OR AL, 4 ; SET UP ALPHA MODE
OE0B E6 61 C16: OUT PORT_B, AL ; STUFF BACK IN 8255

----- SETUP 6845
OE0B 1E PUSH DS ; SAVE DATA SEGMENT VALUE
OE0C 33 C0 XOR AX, AX ; SET UP FOR ABS SEGMENT
OE0C 8E 08 MOV DS, AX ; ESTABLISH VECTOR TABLE ADDRESSING
OE0C A4 0099 ASSUME DS: ABSO
OE0C C5 0E 0074 R LDS BX, PARM_PTR ; GET POINTER TO VIDEO PARMS
OE0C C6 00 00 CODE ASSUME DS: CODE
OE0C CB B7 C7 MOV AX, DI ; GET CURRENT MODE IN AX
OE0C CB B9 0010 90 MOV CX, MO040 ; LENGTH OF EACH ROW OF TABLE
OE0C CB BO FC 02 CMP AH, 2 ; DETERMINE WHICH TO USE
OE0C CF 72 10 JC C17 ; MODE IS 0 OR 1
OE0D 03 09 ADD BX, CX ; MOVE TO NEXT ROW OF INIT TABLE
OE0D 03 FC 04 CMP AH, 4
OE0D 72 09 JC C17 ; MODE IS 2 OR 3
OE0D 03 09 ADD BX, CX ; MOVE TO GRAPHICS ROW OF INIT_TABLE

----- DX POINTS TO CORRECT ROW OF INITIALIZATION TABLE
OE11 50 PUSH AX ; SAVE MODE IN AH
OE12 47 02 MOV AL, DS: (BX+23) ; GET HORIZ. SYNC POSITION
OE12 B8 0C 0A MOV DI, WORD PTR DS: (BX+10) ; GET CURSOR TYPE
OE13 1E PUSH DS
OE13 EB 138B R CALL DDS

----- SETUP DATA
OE14 A2 0089 R MOV HORIZ_POS, AL ; SAVE HORIZ. POSITION VARIABLE
OE14 B9 03 06 R MOV CURSOR_MODE, DS: ; SAVE CURSOR MODE
OE15 F3 50 PUSH AX
OE16 F4 A0 0086 R MOV AL, VAR_DELAY ; SET DEFAULT OFFSET
OE16 F7 24 0F AND AL, OFH
OE16 F9 A2 0086 R MOV VAR_DELAY, AL
OE17 58 POP AX ; ASSUME DS: CODE
OE18 1F PUSH DS
OE18 E2 3E 44 XOR AH, AH ; AH WILL SERVE AS REGISTER NUMBER

----- LOOP THROUGH TABLE, PUTTING OUT REG ADDRESS, THEN VALUE FROM TABLE
OE0C BA 0304 MOV BX, OX304H ; POINT TO 6845 REG NUMBER

----- LOOP THROUGH TABLE, PUTTING OUT REG ADDRESS, THEN VALUE FROM TABLE
OE0D BA C4 C18: MOV AL, AH ; GET 6845 REGISTER NUMBER

----- LOOP THROUGH TABLE, PUTTING OUT REG ADDRESS, THEN VALUE FROM TABLE
OE0E EE OUT DX, AL
OE0F 42 INC DX ; POINT TO DATA PORT
OE0F CE 04 INC AH ; NEXT REGISTER VALUE

----- LOOP THROUGH TABLE, PUTTING OUT REG ADDRESS, THEN VALUE FROM TABLE
OE0F 0A 07 MOV AL, (BX) ; GET TABLE VALUE
OE0F 0E OUT DX, AL ; OUT TO CHIP

----- LOOP THROUGH TABLE, PUTTING OUT REG ADDRESS, THEN VALUE FROM TABLE
OE0F 43 INC BX ; NEXT IN TABLE

----- LOOP THROUGH TABLE, PUTTING OUT REG ADDRESS, THEN VALUE FROM TABLE
OE0F 4A DEC DX ; BACK TO POINTER REGISTER
OE0F E2 F3 LOOP C18 ; DO THE WHOLE TABLE
OE0F 1B 58 POP AX ; GET MODE BACK

----- LOOP THROUGH TABLE, PUTTING OUT REG ADDRESS, THEN VALUE FROM TABLE
OE0F 1F POP DS ; RECOVER SEGMENT VALUE

----- LOOP THROUGH TABLE, PUTTING OUT REG ADDRESS, THEN VALUE FROM TABLE
OE11 33 FF XOR DI, DI ; SET UP POINTER FOR REGEN

----- LOOP THROUGH TABLE, PUTTING OUT REG ADDRESS, THEN VALUE FROM TABLE
OE12 B9 03 06 R MOV CRSTART, D1 ; START ADDRESS SAVED IN GLOBAL
OE13 C6 06 0062 R 00 MOV ACTIVE_PAGE, 0 ; SET PAGE VALUE
OE14 10 5A POP DX ; GET ORIGINAL INPUT BACK
OE15 80 E2 80h AND DL, 80H ; NO CLEAR OF REGEN ?
OE16 75 1C JNZ C21 ; SKIP CLEARING REGEN

----- LOOP THROUGH TABLE, PUTTING OUT REG ADDRESS, THEN VALUE FROM TABLE
OE16 BA 8800h MOV DX, OX2800H ; SET UP SEGMENT FOR 16K REGEN AREA
OE16 B9 2000h MOV CX, B192H ; NUMBER OF WORDS TO CLEAR
OE16 3C 09 CMP AL, 0BH ; REQUIRE 32K BYTE REGEN ?
OE16 72 05 JC C19 ; SET REG SEGMENT

----- LOOP THROUGH TABLE, PUTTING OUT REG ADDRESS, THEN VALUE FROM TABLE
OE16 D1 E1 SHL CX, 1 ; SET 16K WORDS TO CLEAR
OE16 FA 1Bh MOV DX, OX180H ; SET UP SEGMENT FOR 32K REGEN AREA
OE16 EB C2 C2 MOV ES, DX ; SET REG SEGMENT
OE16 3C 04 CMP AX, 4 ; TEST FOR GRAPHICS

----- LOOP THROUGH TABLE, PUTTING OUT REG ADDRESS, THEN VALUE FROM TABLE
OE17 3B 02h MOV AX, ' 152k56S' ; FILL CHAR FOR ALPHA
OE17 3B 02h JC C20 ; NO GRAPHICS_INIT
OE17 3B 33 C0 XOR AX, AX ; FILL FOR GRAPHICS MODE

----- LOOP THROUGH TABLE, PUTTING OUT REG ADDRESS, THEN VALUE FROM TABLE
OE17 F3 3F/ A8 C20: REP STOSW ; FILL THE REGEN BUFFER WITH BLANKS

----- ENABLE VIDEO
OE17 F3 03A R MOV DX, VGA_CTRL ; SET PORT ADDRESS OF VGA
OE17 F2 03 C0 XOR AL, AL
OE17 44 EE OUT DX, AL ; SELECT VGA REG 0
OE17 40 06 65 R MOV AL, CR_MODESET ; GET MODE SET VALUE
OE17 48 EE OUT DX, AL ; SET MODE

----- DETERMINE NUMBER OF COLUMNS, BOTH FOR ENTIRE DISPLAY

----- AND THE NUMBER TO BE USED FOR TTY INTERFACE

----- ENABLE VIDEO
OE49 32 FF XOR BH, BH
OE49 BA 0E 049 R MOV BL, CR_MODE
OE4F 2E: BA 0B 065 R MOV AL, CS: [BX+ OFFSET M0600]
OE54 42 E4 XOR AH, AH
OE56 A3 004A R MOV CRT_COLS, AX ; NUMBER OF COLUMNS IN THIS SCREEN
;----- SET CURSOR POSITIONS
SHL BX, 1

; WORD OFFSET INTO CLEAR LENGTH

OF5B 2E: B8 BF 0048 R
MOV CX, CS: CBX + OFFSET M00503 ; LENGTH TO CLEAR

OF60 89 E0 004C R
MOV CNT_LEN, CX ; SAVE LENGTH OF CNT

OF64 B8 0008
MOV CX, B ; CLEAR ALL CURSOR POSITIONS

OF67 BF 0050 R
MOV DI, OFFSET CURSOR_P05N

OF6A 1E
PUSH DS ; ESTABLISH SEGMENT

OF6B 07
POP ES ; ADDRESSING

OF6C 33 C0
XOR AX, AX

OF6E F3/ AB
REP STOSW ; FILL WITH ZEROS

;----- NORMAL RETURN FROM ALL VIDEO RETURNS
VIDEO_RETURN

OF70

OF70 5F
POP DI

OF71 5E
POP SI

OF72 5B
POP BX

OF73 59
C22: POP CX

OF74 5A
POP DX

OF75 1F
POP DS

OF76 07
POP ES ; RECOVER SEGMENTS

OF77 CF
IRET

OF78

;----- return ALL DONE

SET_MODE ENDP

; KBDNMI - KEYBOARD NMI INTERRUPT ROUTINE

KBDMN1I PROC FAR

;---------------DISABLE INTERRUPTS
CLI

;-----------SAVE REGS & DISABLE NMI

PUSH SI

PUSH DI

PUSH AX ; SAVE REGS

OF83 32 E4
MOV CX, 5 ; SET COUNTER

OF88 B9 0005
11: IN AL, PORT_C ; GET SAMPLE

OF8B 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

OF95 B0 FC 03
CMP AX, 3 ; VALID DATA BIT?

OF98 73 03
JNB 125 ; JUMP IF OK

OF9A EB 5D 90
JMP IB ; INVALID (SYNC ERROR) NO AUDIO

;---------------VALID START BIT, LOOK FOR TRAILING EDGE

OF9B B9 0032
125: MOV CX, 50 ; SET UP WATCHDOG TIMEOUT

OFAB E4 62
I3: IN AL, PORT_C ; GET SAMPLE

OFAD AB 40
TEST AL, 40H ; TEST IF 0

OFAE 74 05
JZ 15 ; JMP IF TRAILING EDGE FOUND

OFAF E2 F8
LOOP 13 ; KEEP LOOKING FOR TRAILING EDGE

OFAB EF 90
JMP IB ; SYNC ERROR (STUCK ON 1'S)

;---------------READ CLOCK TO SET START OF BIT TIME

OFAC B0 40
I35: MOV AL, 40H ; READ CLOCK

OFAD E6 43
OUT TIM_CTL, AL ;

OFAF 90
NOP ;

OFB0 90
NOP ;

OFB1 E4 41
IN AL, TIMER+1 ;

OFB3 B0 E0
MOV AH, AL ;

OFB5 E4 41
IN AL, TIMER+1 ;

OFB7 B6 E0
XCHG AH, AL ;

OFBB B8 F8
MOV DI, AX ; SAVE CLOCK TIME IN DI

;---------------VERIFY VALID TRANSITION

OFBB B9 0004
MOV CX, 4 ; SET COUNTER

OFBE E4 62
I6: IN AL, PORT_C ; GET SAMPLE

OFCC AB 40
TEST AL, 40H ; TEST IF 0

OFCD 75 35
JNZ 18 ; JMP IF INVALID TRANSITION (SYNC)

OFCE E2 F8
KEEP LOOKING FOR VALID TRANSITION

;---------------SET UP DISTANCE TO MIDDLE OF 1ST DATA BIT

OFCF 50
MOV DX, 5A44 ; 310 USEC AWAV (33B US / CT)

OFC0 020E
; START LOOKING FOR TIME TO READ DATA BITS AND ASSEMBLE BYTE

OFC1 1A 30
CALL 130

OFC2 B0 020E
MOV DX, 526 ; SET NEW DISTANCE TO NEXT HALF BIT

OFC3 50
POP AX ; SAVE 1ST HALF BIT

OFC4 0B 130
CALL 130

OFC5 0A CB
MOV CL, AL ; PUT 2ND HALF BIT IN CL

OFC6 58
POP AX ; RESTORE 1ST HALF BIT

OFC7 3A CB
CMP CL, AL ; ARE THEY OPPOSITES?

OFC8 74 2A
JE 19 ; NO, PHASE ERROR
short BH, AL
or BH, AL ; or in new data bit
dec SI ; decrement data bit counter
mov 75H, E8  ; continue for more data bits

...---WAIT FOR TIME TO SAMPLE PARITY BIT

mov E8, 1031H  ; valid parity bit, check parity
and BL, 1 ; check if odd parity
mov 80, E3  ; jmp if parity error
sti ; enable interrupts
mov AL, BH ; place scan code in AL
mov 87H, E4  ; int 4BH ; character processing
...---RESTORE REGS AND RE-ENABLE NMI

mov E9, 07H  ; restore regs
mov 84H, 1F  ; pop DS
mov 85H, 5A  ; pop DX
mov 86H, 59  ; pop CX
mov 87H, 58  ; pop BX
mov 88H, 4A  ; in AL, 0A0H ; enable nmi
mov 89H, 5B  ; pop AX
mov 8AH, 5F  ; pop SI
mov 8BH, 00H  ; iret ; return to system

...---PARITY, SYNCH OR PHASE ERROR. OUTPUT MISSED KEY BEEP

...---SAMPLE LINE

; PORT C IS SAMPLED CX TIMES AND IF THERE ARE 3 OR MORE 1'S THEN B0H IS RETURNED IN AL ELSE 00H IS RETURNED IN AL.
; PARITY COUNTER IS MAINTAINED IN ES.

mov 80H, 0018H  ; xor AH, AH ; clear counter
mov 81H, 03H  ; test aL, port_c ; get sample
mov 82H, 04H  ; xor aL, aL ; test if 1
mov 83H, 02H  ; jz 133 ; jmp if 0
mov 84H, 00H  ; inc aH ; keep count of 1's
mov 85H, 02H  ; loop 132 ; keep sampling
mov 86H, 03H  ; cmp aH, 3 ; valid 1?
mov 87H, 05H  ; jz 134 ; if not valid 1
mov 88H, 00H  ; mov 80H, 08H ; return 80H in al (1)
mov 89H, 03H  ; inc bl ; increment parity counter
mov 8AH, 03H  ; ret ; return to caller
mov 8BH, 02H  ; xor al, al ; return 0 in al (0)
mov 8CH, 03H  ; ret ; return to caller
mov 8DH, 00H  ; endp

THE PURPOSE OF THIS ROUTINE IS TO TRANSLATE SCAN CODES AND SCAN CODE COMBINATIONS FROM THE 83 KEYBOARD TO THEIR
EQUIVALENTS ON THE 85 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 INTENTION 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 62 KEYBOARD TO LOOK AS IF IT WERE AN 83 KEY
VERSION.

IT IS ASSUMED THAT THIS ROUTINE IS CALLED FROM THE 81
DESERIALIZATION ROUTINE AND THAT ALL REGISTERS ARE
DESTROYED.

EQUATES
BREAK_BIT EQU 80H
FN_KEY EQU 54H
PHK EQU FN_KEY+1
EXT_SCAN EQU PHK+1 ; BASE CODE FOR SCAN CODES
OFF EQU 00H ; EXTENDING BEYOND 83
AND_MASK EQU OFFH ; USED TO SELECTIVELY REMOVE BITS
CLEAR_FLAGS EQU (FN_FLAG+FN_BREAK+FN_PENDING) ; SCAN CODES
B_KEY EQU 48
Q_KEY EQU 16
P_KEY EQU 25
E_KEY EQU 18
S_KEY EQU 31
N_KEY EQU 49
UP_ARROW EQU 72
DOWN_ARROW EQU 80
LEFT_ARROW EQU 75
RIGHT_ARROW EQU 77
MINUS EQU 12
EQUALS EQU 13
NUM_O EQU 11

; NEW TRANSLATED SCAN CODES

NOTE:
BREAK, PAUSE, ECHO, AND PRT_SCREEN ARE USED AS OFFSETS
INTO THE TABLE 'SCAN'. OFFSET = TABLE POSITION + 1.

= 0001
= 0002
= 0003
= 0004
= 0005
= 0006
= 0007
= 0008
=BREAK, Q_KEY, E_KEY, P_KEY, S_KEY, N_KEY
=UP_ARROW, DOWN_ARROW, LEFT_ARROW, RIGHT_ARROW, MINUS
=DB, ECHO, PAUSE, PRT_SCREEN, SCROLL_LOCK, NUM_LOCK,
=HOME, END_KEY, PAGE_UP, PAGE_DOWN, KEYPAD_MINUS, KEYPAD_PLUS

; TABLE OF VALID SCAN CODES

KB0 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

KB0LEN EQU % - KB0

; TABLE OF NEW SCAN CODES

KB1 LABEL BYTE
DB BREAK, PAUSE, ECHO, PRT_SCREEN, SCROLL_LOCK, NUM_LOCK,
= HOME, END_KEY, PAGE_UP, PAGE_DOWN, KEYPAD_MINUS, KEYPAD_PLUS

; TABLE OF NUMERIC KEYPAD SCAN CODES

NUM_CODES LABEL BYTE
DB 79, 80, 81, 75, 76, 77, 71, 72, 73, 82

; TABLE OF SIMULATED KEYSTROKES

; THIS TABLE REPRESENTS A 4x2 ARRAY. EACH ROW
; CONSISTS OF A SEQUENCE OF SCAN CODES WHICH
; WOULD HAVE BEEN GENERATED ON AN 83 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, 182, 157 ; CTRL + PRTSC
DB 29, 70, 198, 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

ALT_TABLE  Label BYTE
1093 35 2B 34 1A 1B DB 53,40,52,26,27
0 0006 ALT_LEN EQU 9 - ALT_TABLE

TABLE OF TRANSLATED SCAN CODES WITH ALT SHIFT

This table containing 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+ / \ 

NEW_ALT  Label BYTE
1098 2B 29 29 28 29 0B 43,41,55,43,41

TABLE OF SCAN CODES FOR MAPPING EXTENDED SET

Of scan codes (scan codes > 0F). This table allows other devices to use the keyboard interface, if the device generates a scan code > 0F 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 86 and not mapping 86 to the keyboard will have a 150,03 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 86, 87, B8, ... B6+n(n-1). The scan code bytes in the table correspond to this set with the first data byte matching B6, the second matching B7 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 scan codes in the extended set.

EXTAB

1090
1090 14 DB 20 ; LENGTH OF TABLE
109E 004B 0049 004D 0051 DW 72,73,77,81,80,79,75,71,57,28
005D 004F 0048 0047 0039 001C
1082 0011 0012 001F 0020 002C 002B 001E 0010 000F 0001

1006 KEY62_INT PROC FAR
1006 FB STI
1007 FC CLD ; FORWARD DIRECTION
100B EB 1388 R CALL DS ; SET UP ADDRESSING
100C BA 0E MOV AH,AL ; SAVE SCAN CODE
100D EB 131E R CALL TPM ; ADJUST OUTPUT FOR USER MODIFICATION
1000 73 01 JNC KBKO ; JUMP IF OK TO CONTINUE
1002 CF IRET ; RETURN FROM INTERRUPT.

1003 3C FF KBKO: CMP AL,OFFH ; IS THIS AN OVERRUN CHAR?
1005 74 6C JE KB4 ; IS THIS A SCAN CODE > B3?
1007 24 7F AND AL, AND_MASK-BREAK_BIT ; TURN OFF BREAK BIT
1009 3C 56 CMP AL, EXT_SCAN ; IS THIS A SCAN CODE > B3?
100B 7C 5F JB KBK4 ; REPLACE BREAK BIT

SCAN CODE IS IN EXTENDED SET

100D 6E PUSH DS
100E 33 F6 XOR SI,SI
100E 8E DE MOV DS,SI
100F 24 C4 0124 R LES SI,WORD PTR EXT ; GET THE POINTER TO THE EXTENDED SET
1016 66 2A 00 MOV CL, BYTE PTR ES:01 ; GET LENGTH BYTE
1018 IF 05 PDP DS
1019 IF DS 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
10E8 FE C9 DEC CL ; LENGTH - 1
10EE 3A C1 CMP AL, CL ; IS CODE IN TABLE?
10F0 7C 10 JG KBK1 ; JUMP IF SCAN CODE IS NOT IN TABLE

A-38 ROM BIOS
1047 8F 9B MOV BX,AH
1048 9C 76 SHL BX,1
1049 9D 2F ADD DL,BX
1050 F0 2B MOV AL,BYTE PTR ES:[1105]
1051 FF 3C CMP AL,00H
1052 FF 3D JZ KBO_2
1053 FF 3E JNZ KBO_3
1054 FF 40 MOV BX,BUFFER_TAIL
1055 FF 41 MOV SI,BX
1056 FF 42 CALL K4
1057 FF 43 CMP BX,BUFFER_HEAD
1058 FF 44 JZ KBX3
1059 FF 45 MOV AX,0
105A FF 46 MOV AX,0
105B FF 47 MOV AX,0
105C FF 48 MOV AX,0
105D FF 49 MOV AX,0
105E FF 4A MOV AX,0
105F FF 4B MOV AX,0
1060 FF 4C MOV AX,0
1061 FF 4D MOV AX,0
1062 FF 4E MOV AX,0
1063 FF 4F MOV AX,0
1064 FF 50 MOV AX,0
1065 FF 51 MOV AX,0
1066 FF 52 MOV AX,0
1067 FF 53 MOV AX,0
1068 FF 54 MOV AX,0
1069 FF 55 MOV AX,0
106A FF 56 MOV AX,0
106B FF 57 MOV AX,0
106C FF 58 MOV AX,0
106D FF 59 MOV AX,0
106E FF 5A MOV AX,0
106F FF 5B MOV AX,0
1070 FF 5C MOV AX,0
1071 FF 5D MOV AX,0
1072 FF 5E MOV AX,0
1073 FF 5F MOV AX,0
1074 FF 60 MOV AX,0
1075 FF 61 MOV AX,0
1076 FF 62 MOV AX,0
1077 FF 63 MOV AX,0
1078 FF 64 MOV AX,0
1079 FF 65 MOV AX,0
107A FF 66 MOV AX,0
107B FF 67 MOV AX,0
107C FF 68 MOV AX,0
107D FF 69 MOV AX,0
107E FF 6A MOV AX,0
107F FF 6B MOV AX,0
1080 FF 6C MOV AX,0
1081 FF 6D MOV AX,0
1082 FF 6E MOV AX,0
1083 FF 6F MOV AX,0
1084 FF 70 MOV AX,0
1085 FF 71 MOV AX,0
1086 FF 72 MOV AX,0
1087 FF 73 MOV AX,0
1088 FF 74 MOV AX,0
1089 FF 75 MOV AX,0
108A FF 76 MOV AX,0
108B FF 77 MOV AX,0
108C FF 78 MOV AX,0
108D FF 79 MOV AX,0
108E FF 7A MOV AX,0
108F FF 7B MOV AX,0
1090 FF 7C MOV AX,0
1091 FF 7D MOV AX,0
1092 FF 7E MOV AX,0
1093 FF 7F MOV AX,0
1094 FF 80 MOV AX,0
1095 FF 81 MOV AX,0
1096 FF 82 MOV AX,0
1097 FF 83 MOV AX,0
1098 FF 84 MOV AX,0
1099 FF 85 MOV AX,0
109A FF 86 MOV AX,0
109B FF 87 MOV AX,0
109C FF 88 MOV AX,0
109D FF 89 MOV AX,0
109E FF 8A MOV AX,0
109F FF 8B MOV AX,0
10A0 FF 8C MOV AX,0
10A1 FF 8D MOV AX,0
10A2 FF 8E MOV AX,0
10A3 FF 8F MOV AX,0
10A4 FF 90 MOV AX,0
10A5 FF 91 MOV AX,0
10A6 FF 92 MOV AX,0
10A7 FF 93 MOV AX,0
10A8 FF 94 MOV AX,0
10A9 FF 95 MOV AX,0
10AA FF 96 MOV AX,0
10AB FF 97 MOV AX,0
10AC FF 98 MOV AX,0
10AD FF 99 MOV AX,0
10AE FF 9A MOV AX,0
10AF FF 9B MOV AX,0
10B0 FF 9C MOV AX,0
10B1 FF 9D MOV AX,0
10B2 FF 9E MOV AX,0
10B3 FF 9F MOV AX,0
10B4 FF A0 MOV AX,0
10B5 FF A1 MOV AX,0
10B6 FF A2 MOV AX,0
10B7 FF A3 MOV AX,0
10B8 FF A4 MOV AX,0
10B9 FF A5 MOV AX,0
10BA FF A6 MOV AX,0
10BB FF A7 MOV AX,0
10BC FF A8 MOV AX,0
10BD FF A9 MOV AX,0
10BE FF AA MOV AX,0
10BF FF AB MOV AX,0
10C0 FF AC MOV AX,0
10C1 FF AD MOV AX,0
10C2 FF AE MOV AX,0
10C3 FF AF MOV AX,0
10C4 FF B0 MOV AX,0
10C5 FF B1 MOV AX,0
10C6 FF B2 MOV AX,0
10C7 FF B3 MOV AX,0
10C8 FF B4 MOV AX,0
10C9 FF B5 MOV AX,0
10CA FF B6 MOV AX,0
10CB FF B7 MOV AX,0
10CC FF B8 MOV AX,0
10CD FF B9 MOV AX,0
10CE FF BA MOV AX,0
10CF FF BB MOV AX,0
10D0 FF BC MOV AX,0
10D1 FF BD MOV AX,0
10D2 FF BE MOV AX,0
10D3 FF BF MOV AX,0
10D4 FF C0 MOV AX,0
10D5 FF C1 MOV AX,0
10D6 FF C2 MOV AX,0
10D7 FF C3 MOV AX,0
10D8 FF C4 MOV AX,0
10D9 FF C5 MOV AX,0
10DA FF C6 MOV AX,0
10DB FF C7 MOV AX,0
10DC FF C8 MOV AX,0
10DD FF C9 MOV AX,0
10DE FF CA MOV AX,0
10DF FF CB MOV AX,0
10E0 FF CC MOV AX,0
10E1 FF CD MOV AX,0
10E2 FF CE MOV AX,0
10E3 FF CF MOV AX,0
10E4 FF D0 MOV AX,0
10E5 FF D1 MOV AX,0
10E6 FF D2 MOV AX,0
10E7 FF D3 MOV AX,0
10E8 FF D4 MOV AX,0
10E9 FF D5 MOV AX,0
10EA FF D6 MOV AX,0
10EB FF D7 MOV AX,0
10EC FF D8 MOV AX,0
10ED FF D9 MOV AX,0
10EE FF DA MOV AX,0
10EF FF DB MOV AX,0
10F0 FF DC MOV AX,0
10F1 FF DD MOV AX,0
10F2 FF DE MOV AX,0
10F3 FF DF MOV AX,0
10F4 FF E0 MOV AX,0
10F5 FF E1 MOV AX,0
10F6 FF E2 MOV AX,0
10F7 FF E3 MOV AX,0
10F8 FF E4 MOV AX,0
10F9 FF E5 MOV AX,0
10FA FF E6 MOV AX,0
10FB FF E7 MOV AX,0
10FC FF E8 MOV AX,0
10FD FF E9 MOV AX,0
10FE FF EA MOV AX,0
10FF FF EB MOV AX,0
1100 FF EC MOV AX,0
1101 FF ED MOV AX,0
1102 FF EE MOV AX,0
1103 FF EF MOV AX,0
1104 FF F0 MOV AX,0
1105 FF F1 MOV AX,0
1106 FF F2 MOV AX,0
1107 FF F3 MOV AX,0
1108 FF F4 MOV AX,0
1109 FF F5 MOV AX,0
110A FF F6 MOV AX,0
110B FF F7 MOV AX,0
110C FF F8 MOV AX,0
110D FF F9 MOV AX,0
110E FF FA MOV AX,0
110F FF FB MOV AX,0
1110 FF FC MOV AX,0
1111 FF FD MOV AX,0
1112 FF FE MOV AX,0
1113 FF FF MOV AX,0
;----CHECK IF FUNCTION FLAG ALREADY SET
KB4: CMP AL, PKH ; IS THIS A PHANTOM KEY?
JZ KB3_2 ; JUMP IF PHANTOM SEQUENCE
KB4_0: TEST KB_FLAG_2, FN_FLAG+FN_LOCK ; ARE WE IN FUNCTION STATE?

11E4 7F 21 ;----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 0B CMP AL, NUM_0 ; ARE WE IN NUMERIC KEYPAD REGION?
11EF 77 12 JB KB3_1 ; JUMP IF NOT IN KEYPAD
11F1 FE CB DEC AL ; CHECK LOWER BOUND OF RANGE
11F3 7F 0E JZ KB4_1 ; JUMP IF NOT IN RANGE (ESC KEY)

;----TRANSLATE SCAN CODE TO NUMERIC KEYPAD
11F5 FE CB DEC AL ; AL IS OFFSET INTO TABLE
11F7 BB 1091 R MOV X, OFFSET NUM_CODES,
11FA 2E: 07 XLAT CS: NUM_CODES ; NEW SCAN CODE IS IN AL
11FC 80 E4 B0 AND AH, BREAK_BIT ; ISOLATE BREAK_BIT ON ORIGINAL SCAN CODE
11FF 0A C4 OR AL, AH ; UPDATE KEYPAD SCAN CODE
1201 EB 59 JMP SHORT CONT_INT ; CONTINUE WITH INTERRUPT
1203 F2 C4 KB4_1: MOV AL, AH ; GET BACK BREAK_BIT IF SET
1205 EB 55 JMP SHORT CONT_INT

;----ESCAPE KEY, LOCK KEYBOARD IN FUNCTION LOCK
1207 3C 0B CMP AL, NUM_0 ; CHECK FOR RANGE OF NUMBERS
1209 77 20 JB KB7 ; JUMP IF NOT IN RANGE
120B FE CB DEC AL ; ESC FOR ESC KEY (=1)
120D 75 25 JNZ KB6 ; NOT ESCAPE KEY, RANGE OF NUMBERS

1219 7F 28 TEST AH, BREAK_BIT ; IS THIS A BREAK KEY?
121A F6 06 0088 R 80 TEST KB_FLAG_2, FN_FLAG ; TOGGLES ONLY WHEN FN HELD
121D 75 30 JZ KB8 ; NOT HELD CONTINUALLY
121F 86 06 0088 R 40 TEST KB_FLAG_2, FN_BREAK ; HAS THE FUNCTION KEY BEEN RELEASED?
1220 75 22 JNZ KB8 ; CONTINUE IF RELEASED. PROCESS AS ESC
1222 F6 06 0017 R 03 TEST KB_FLAG, LEFT_SHIFT+RIGHT_SHIFT ; EITHER SHIFT?
1224 77 18 JB KB8 ; NOT HELD DOWN
1226 80 36 0088 R 10 XOR KB_FLAG_2, FN_FLAG ; TOGGLE STATE
1228 8E 26 0088 R IF 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 KB8: ADD AL, 5B ; GENERATE CORRECT SCAN CODE
1236 EB 3E JMP SHORT KB12 ; CLEAR-UP BEFORE RETURN TO KB_INT

;----CHECK TABLE FOR OTHER VALID SCAN CODES
1238 0E KB9: PUSH CS
1239 07 MOV ES ; ESTABLISHED ADDRESS OF TABLE
123A BF 1069 R MOV D1, OFFSET KB8 ; BASE OF TABLE
123B 89 00DC MOV CX, KBOLON ; LENGTH OF TABLE
123D F2 3E AESD REPNE SCASB ; SEARCH TABLE FOR A MATCH
1242 74 1D JE KB10 ; JUMP IF MATCH

;----ILLEGAL CHARACTER
1244 F6 06 0088 R 40 KB8: TEST KB_FLAG_2, FN_BREAK ; HAS BREAK_OCCURRED?
1247 7F 0F JZ KB9 ; FUNCTION KEY HAS NOT BEEN RELEASED
1249 F6 C4 B0 TEST AH, BREAK_BIT ; IS THIS A BREAK OF AN ILLEGAL
124E 7E 05 KB9: JNZ KB9 ; DON'T RESET FLAGS ON ILLEGAL

1250 80 26 0088 R IF KB8: AND KB_FLAG_2, CLEAR_FLAGS ; NORMAL STATE
1255 C6 06 0087 R 00 MOV CUR_FUNC, 0 ; RETRIEVE ORIGINAL SCAN CODE

;----FUNCTION BREAK IS NOT SET
125A 8A C4 KB9: MOV AL, AH ; RETRIEVE ORIGINAL SCAN CODE
125C E6 60 OUT KBPORT, AL
125E CD 09 INT 9H ; ISSUE KEYBOARD INTERRUPT

1260 CF IRET ; ;----BEFORE TRANSLATION CHECK FOR ALT+F+N_KEY AS NUM LOCK
1261 3C 31 KB10: CMP AL, N_KEY ; IS THIS A POTENTIAL NUMLOCK?
1263 75 07 JNE KB10_1 ; NOT A NUMKEY, TRANSLATE IT
1265 F6 06 0017 R 08 TEST KB_FLAG, ALT_SHIFT ; ALT HELD DOWN ALSO?
1266 7A 74 JB KB10_2 ; TREAT AS ILLEGAL COMBINATION
1268 BC 10 06A R KB10_1: MOV CX, OFFSET KB8+1 ; GET OFFSET TO TABLE
126B FF 2F SUB D1, CX ; UPDATE INDEX TO NEW SCAN_CODE TABLE

1271 2E: 8A B5 1075 R MOV AL, CS:KB10DI ; MOV NEW SCAN_CODE INTO REGISTER
1276 F6 C4 B0 KB12: TEST AH, BREAK_BIT ; IS THIS A BREAK CHAIN?
1279 74 35 JZ KB3_2 ; 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 46 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, B0H ; TURN ON BREAK_BIT
1285 E6 60 OUT KBPORT, AL
1287 CD 09 INT 9H ; TOGGLE STATE
1289 24 70 AND AL, AND_MASK-BREAK_BIT ; TURN OFF BREAK_BIT
128B 8F 06 0088 R 40 KB12_2: TEST KB_FLAG_2, FN_BREAK ; HAS FUNCTION BREAK_OCCURRED?
128C 74 11 JZ KB3_2 ; JUMP IF BREAK HAS NOT_OCCURRED
128D 92 06 0087 R MOV AL, CUR_FUNC ; IS THIS A BREAK OF OLD VALID FUNCTION
128F 75 C8 JNE RET_INT ; ALLOW FURTHER CURRENT FUNCTIONS
1290 KB2_20: MOV KB_FLAG_2, CLEAR_FLAGS
1292 3A 06 0087 R JNE KB12_20 ; RET_INT ; RETURN FROM INTERRUPT
1294 DA 06 0087 R 00 MOV CUR_FUNC, 0 ; CLEAR CURRENT FUNCTION
12A2 CF IRET ;
12A3 3A 0E 0087 R

12A7 75 B7

12A9 B0 2E 0088 R DF

12AE EB ED

; ----VALID MAKE KEY HAS BEEN PRESS

12B0 F6 0E 0088 R 40

12B5 74 00

; ----FUNCTION BREAK FUNCTION HAS ALREADY OCCURED

12B7 80 3E 0087 R 00

12B8 7C 0E 0087 R

12C2 75 8C

; ----CHECK FOR SCAN CODE GENERATION SEQUENCE

12C4 A2 0087 R

12C7 3C 04

; ----BREAK OR ECHO

12D0 FE CB

12D3 D0 E0

12D5 E0 D0

12D7 9B

12D8 8E 0E 36 1088 R

12DD 03 F0

12DE B9 0004

12EA CF

; ----PUT KEYBOARD IN HOLD STATE

12EB F6 0E 0018 R 08

12F0 75 0E

12F2 B0 0E 0018 R 08

12F7 E4 A0

12F9 F6 0E 0018 R 08

12FE 75 F9

1300 CF

; ----PRINT SCREEN FUNCTION

1301 F6 0E 0018 R 08

1306 74 0E

130B 80 26 0018 R F7

130F CF

1310 E3 C4 06

1311 07

1312 IF

1313 5A

1314 59

1315 5B

1316 E4 A0

131A 5B

131B 5F

131C 5E

131D CF

131E KEY62_END_INT

; --------------------------

; TYPOMATIC
;
; 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 SHOULD 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.

131F PROD NEAR

1320 53

1321 3F 0E 0085 R

1323 74 31

1325 A8 80

1327 74 12

132B 74 7F

132F 8A C4

1331 75 05

1333 C6 0E 0085 R 00

133B FB

1339 5B

133A C3

; ----NEW CHARACTER CHECK FOR BREAK SEQUENCES

1352 A8 80

1357 74 12

135B 74 7F

135F 8A C4

1361 75 05

1363 C6 0E 0085 R 00

136B FB

1369 5B

136A C3
---INITIALIZE A NEW CHARACTER

TP0: MOV CUR_CHAR, AL ; SAVE NEW CHARACTER
   AND VAR_DELAY,FOFH ; CLEAR VARIABLE DELAY
   TEST KB_FLAG,OFEH ; INITIAL PUTCHAR BIT AS ZERO
   TEST KB_FLAG,INIT_DELAY ; ARE WE INCREASING THE
   ; INITIAL DELAY?
   JZ TP ; DEFAULT DELAY
   JMP VAR_DELAY_DELAY_RATE ; INCREASE DELAY BY 2X

1354 EB E2

---CHECK IF WE ARE IN TYPAMATIC MODE AND IF DELAY IS OVER

TP2: TEST KB_FLAG_2,TYPE_OFF ; IS TYPAMATIC TURNED OFF?

1356 F6 06 0088 R 0B

1358 75 2B

JNZ TP4 ; JUMP IF TYPAMATIC RATE IS OFF

135B 8A 1E 0086 R

MOV BL,VAR_DELAY ; GET VAR_DELAY
   AND BL,FOFH ; MASK OFF HIGH ORDER(SCREEN RANGE)

1360 A4 82

OR BL,BL ; IS INITIAL DELAY OVER?
   JZ TP3 ; JUMP IF DELAY IS OVER
   DEC BL ; DECREASE DELAY WAIT BY ANOTHER
   ; CHARACTER

1364 FA CB

---CHECK IF TIME TO OUTPUT CHAR

TP3: TEST KB_FLAG_2,HALF_RATE ; ARE WE IN HALF RATE MODE

1366 74 00

JZ TP3 ; JUMP IF WE ARE IN NORMAL MODE
   XOR KB_FLAG_2,PUTCHAR ; TOGGLE BIT

136F 80 36 0088 R 01

TEST KB_FLAG_2,PUTCHAR ; IS IT TIME TO PUT OUT A CHAR

1371 75 80

JNZ TP ; NOT TIME TO OUTPUT CHARACTER
   JMP SHORT TP4 ; SKIP THIS CHARACTER

1373 FA 9F

STC ; SET CARRY FLAG

1376 FA CB

POP BX

1379 FA C3

RET

1382 E9 E9

TPM ENDP

---THIS SUBROUTINE SETS DS TO POINT TO THE BIOS DATA AREA

INPUT: NONE

OUTPUT: DS IS SET

138B 46 5D 01 0B 80 8D 0F 14

DD5 PROC NEAR

138C B9 0040

MOV AX,40H

138F B0 DB

MOV DS,AX

1391 SB

POP AX

1393 C3

RET

ENDP

---INT IA-------------------------------

TIME_OF_DAY/SOUND SOURCE SELECT

THIS ROUTINE ALLOWS THE CLOCK TO BE SET/READ
   AN 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) = B0M SET UP SOUND MULTIPLEXER
   AL = (SOURCE OF SOUND) --> "AUDIO OUT" OR RF MODULATOR
   00 = B253 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 1193180/65536 COUNTS/SEC
   (OR ABOUT 18.2 PER SECOND -- SEE EQUATES BELOW)

ASSUME CS, CODE, DS: DATA

TIME_OF_DAY PROC FAR

1399 STI ; INTERRUPTS BACK ON

139D IE

PUSH DS ; SAVE SEGMENT

139E EA 1388 R

CALL DS5

139F B0 FC 80

CMP AH,BOH ; AH=BOH

13A0 74 2E

JE T4A ; MUX_SET_UP

13A1 0A E4

OR AH,AH

13A3 74 07

JZ T2 ; READ_TIME

13A4 FB

BEC AH ; AH=1

13A5 74 16

JZ T3 ; SET_TIME

13A8 FA T2: CLI ; NO TIMER INTERRUPTS WHILE READING

13A9 A0 0070 R

MOV AL,TIMER_OFL

13AC 8A 00 0070 R 00

MOV TIMER_OFL,0 ; GET OVERFLOW, AND RESET THE FLAG

13B1 B0 06 006E R

MOV CX,TIMER_HIGH

13B5 88 16 006C R

MOV DX,TIMER_LOW

13B9 EA 82

JMP T1 ; IOD_RETURN

13BB FA T3: CLI ; NO INTERRUPTS WHILE WRITING

13BC 89 16 006C R

MOV TIMER_LOW,DX

13C0 B9 06 006E R

MOV TIMER_HIGH,CX ; SET THE TIME

13C3 C6 06 0070 R 00

MOV TIMER_OFL,0 ; RESET OVERFLOW

13C9 EB 0A

JMP T1 ; IOD_RETURN

A-42  ROM BIOS
NO CODE IS ASCII

13CB 51 T4A: PUSH CX ;
13CC B1 05 MOV CL, 5 ;
13CE D2 E0 SAL AL, CL ; SHIFT PARM BITS LEFT 5 POSITIONS
13D0 B6 C4 SCC AH, AL ; SAVE PARM
13D2 E4 61 IN AL, PORT_B ; GET CURRENT PORT SETTINGS
13D4 24 9F AND AL, 10011111B ; ISOLATE Nx BIT
13D6 OA C4 OR AL, AH ; COMBINE PORT BITS/PARM BITS
13D8 E6 61 OUT PORT_B, AL ; SET PORT TO NEW VALUE
13DA 58 POP CX
13DB EB CB JMP TI ; T0D_RETURN
13DD TIME_OF_DAY ENDP

THESE ROUTINES PROVIDE KEYBOARD SUPPORT

INPUT

(AH)=O 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.

(AH)=2 RETURN THE CURRENT SHIFT STATUS IN AL REGISTER THE BIT SETTINGS FOR THIS CODE ARE INDICATED IN THE THE EQUATUSES FOR KB_FLAG.

(AH)=3 SET TYPAMATIC RATES. THE TYPAMATIC RATE CAN

BE CHANGED USING THE FOLLOWING FUNCTIONS:

(AL)=O 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 CHAR.

(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.

***NOTE*** 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.

(AH)=4 ADJUST KEYBOARD BY THE VALUE IN AL AS FOLLOWS:

(AL)=0 T0FF 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

13D0 KEYBOARD_IO PROC FAR
ASSUME CS:CODE, DS:DATA

13D1 ; INTERRUPTS BACK ON
13D2 STI

13D3 ; S\AVE CURRENT DS
13D4 PUSH DS

13D5 ; S\AVE BX TEMPORARILY
13D6 PUSH BX

13D7 ; POINT 5$ AT BIOS DATA SEGMENT
13D8 CALL DDB

13D9 OR AH, AH

13DA AH=1

13DB DEC AH

13DC AJ=2

13DD DEC AM

13DE J=3

13DF JT.SHIFT_STATUS

13E0 ---- READ THE KEY TO FIGURE OUT WHAT TO DO

13E1 ; ASCII READ
13E2 K1: STI

13E3 ; INTERRUPTS BACK ON DURING LOOP
13E4 HOP

13E5 ; ALLOW AN INTERRUPT TO OCCUR
13E6 HOP

13E7 ; INTERRUPTS BACK OFF
13E8 CLF

13E9 ; INTERRUPTS BACK OFF
13EA MOV BX, BUFFER_HEAD

13EB ; GET POINTER TO HEAD OF BUFFER
13EC CMP BX, BUFFER_TAIL ; TEST END OF BUFFER
13ED J=4

13EE MOV AX,[BX] ; GET SCAN CODE AND ASCII CODE
13EF CALL K4

13F0 ; MOVE POINTER TO NEXT POSITION
13F1 MOV BUFFER_HEAD,BX ; STORE VALUE IN VARIABLE
13F2 JMP RET_INT16

13F3 ---- ASCII STATUS

13F4 ; INTERRUPTS OFF
13F5 K2: CLF

13F6 ; INTERRUPTS BACK ON
13F7 MOV BX, BUFFER_HEAD

13F8 ; GET HEAD POINTER
13F9 CMP BX, BUFFER_TAIL ; IF EQUAL (Z=1) THEN NOTHING THERE
13FA MOV AX,[BX]

13FB ; INTERRUPTS BACK ON
13FC STI

13FD ; RECOVER REGISTER
13FE POP BX

13FF ; RECOVER SEGMENT
1300 POP DS

1301 RET 2 ; THROW AWAY FLAGS

1302 ---- SHIFT STATUS

1303 MOV AL, KB_FLAG ; GET THE SHIFT STATUS FLAGS
1304 JMP RET_INT16

1305 ----

1306 K3: A0 0017 R
1307 JMP SHORT RET_INT16

1308 A-43

A-43

A-ROM BIOS
1533
1533
153A

54 55 56 57 58 59
5A
58 5e 50

1530
1530
1542

68 69 6A 69 6e
6D BE SF 70 71

1547
1547

37 38 39 2D 34 35
3628313233'30
2E

1S54
1554

47 48 49 FF 48 FF

1558

40
FF 4F 50 51 52 53
Fa
50
53
51

1565

52

US6B
1567
1568
1569

56
57
IE
06

156A

Fe

156B
156E

£8 1388 R
8A EO

1570
1572
1514
1577
151 A

3t FF
75 19
BS 0080
ss 0048
£8 E035 R

1570

80 26 0017 R Fa

1582

80 26 0018 R OF

1587

80 26 0088 R IF

15Be

£9 164A R

IsaF
IsaF

24 7F

1591

DE

1592

07

1593

SF 14!5C R

1596
1599

89 0009
F21 AE

159B

8A C4

1590

74 03

159F

£9 163A R

15A2
15A6
15A8
15AD

81 EF 1450 R
2£: 8A A5 1464 R
A8 80
75 51

15AF

80 Fe 10

1582

73 07

15B4
15B8

08 26 0017 R
£9 164A R

1598
I!5BB

F6 06 0017 R 04

15cO

75 78

15C2

3C 52

15C4

75 22

15C6
15C8
15CD
15D2
15D4

F6
75
Fa
75
F6

1509
160B

74 00

15DB

98 5230
E9 17EC R

I!5DE

15F2
l!iF6
15F8
15FA
15FD

08
; ------ AL T TABLE SCAN
LABEL
BYTE
K 13
DB
DB
; ------ NUM STATE TABLE
K 14
LABEL
BYTE
DB

84,85,86,87, B8, 89,'0
91,92,93

104, 105, 106, 107, 108
109, 110,111,112,113

'789-456+1230. '

BASE CASE TABLE
LABEL
BYTE
DB
71,72,73,-1,75,-1,77
DB
-1,79, SO, 81, 82, 83
; ------ KEYBOARD I NTERRUPT ROUT I NE
PRoe
FAR
STI
; ALLOW FURTHER INTERRUPTS
PUSH
AX
ex
PUSH
cx
PUSH
OX
PUSH
SI
PUSH
01
PUSH
OS
PUSH
ES
PUSH
i FORWARD 0 I RECTI ON
CLO
CALL
DDS
MOY
AH, AL
i SAVE SCAN CODE IN AH
TEST fOR OVERRUN SCAN CODE FROM KEYBOARD
IS TH I S AN OVERRUN CHAR?
CMP
AL,OFFH
JNZ
K16
NO, TEST FOR SHIFT KEY
DURATION OF ERROR BEEP
MOY
SK, BOH
FREQUENCY OF TONE
MOV
CK,48H
BUFFER FULL BEEP
CALL
KB_NOISE
CLEAR AL T , CLRL, LEFT AND RIGHT
AND
KB_FLAG,OFOH
SHIFTS
AND
CLEAR POTENTI AL BREAK OF INS j CAPS
,NUM AND SCROLL SH I FT
CLEAR FUNCTION STATES
AND
KB_FLAG_2, IFH
END OF INTERRUPT
..IMP
K2&
; ------ TEST FOR SHI FT KEYS
TEST_SHIFT
K16:
TURN OFF THE BREAK BIT
AND
AL,07FH
CS
PUSH
ESTABLISH ADDRESS OF SHIFT· TABLE
POP
ES
SHI FT KEY TA8lE
MOV
D I, OFFSET K6
LENGTH
HOV
CK, K6L
LOOK THROUGH THE TABLE FOR A
REPNE
SCASB
MATCH
RECOVER SCAN CODE
MOV
AL,AH
JUMP I F MATCH FOUND
JE
K17
I~ NO MATCH, THEN SHIFT NOT FOUND
JMP
K25
; ------ SHIFT KEY FOUND
K17:
SUB
DI,OFFSET K6+1
ADJUST PTR TO SCAN tODE MATCH
MOV
AH,CS:K7[Dll
GET MASK INTO AH
TEST
AL,80H
TEST FOR BREAK KEY
JNZ
K23
; BREAK_SHIFT_FOUND
;------ sHIFT MAKE FOUND, DETERMINE SET OR TOGGLE
CMP
AH,SCROLL_SHIFT
; IF SCROLL SHIFT OR ABOVE, TOGGLE
JAE
K18
; KEY
;------ PLAIN SHIFT KEY, SET SHIFT ON
OR
KB_FLAG, AH
; TURN ON SHI FT SIT
JMP
K26
i INTERRUPT_RETURN
i - - - - - - TOGGLED SHI FT KEY, TEST FOR 1ST MAKE OR NOT
K18:
; SHIFT-TOGGLE
TEST
KB_FlAG, tTL_SHIFT i CHECK CTL SHIFT STATE
JNZ
K25
i JUMP I F eTl STATE
CMP
AL, INS_KEY
i CHECK FOR I NSERT KEY
JNZ
K22
; JUMP I F Nor I NSERT KEY
TEST
KB_FLAG, ALT_SHIFT ; CHECK FOR ALTERNATE SHIFT
JNZ
K25
i JUMP IF ALTERNATE SHIFT
TEST
KS_FLAG, NUM_STATE ; CHECK FOR BASE STATE
JNZ
K21
; JUMP IF NUM LOCK IS ON
TEST
KB_FLAG, LEFT_SHIFT+ RIGHT_SHIFT
JZ
K22
JUMP I F BASE STATE
; NUMERIC ZERO, NOT INSERT KEY
K20:
; PUT OUT AN ASCI I ZERO
AX, 5230H
HOV
JHP
K57
; BUFFER_FILL
; MIGHT 8E NUMERIC
K21:
KS_FLAG, LEFT_SHIFT+ RIGHT_SHIFT
TEST
K20
JUMP NUMERIC, NOT INSERT
JZ
SHIFT TOGGLE KEY HIT; PROCESS IT
K22:
IS· KEY· ALREADY DEPRESSED
TEST
AH, KB_FLAG_l
JUMP I F KEY ALREAOY OEPRESSED
JHZ
K26
INDICATE THAT THE KEY IS
OR
KB_FLAG_l, AH
DEPRESSED
TOGGLE THE SHIFT STATE
XOR
KB_FlAG, AH
TEST FOR 1ST MAKE OF I NSERT KEY
AL, INS_KEY
CHP
JUMP~ 'iF NOT I NSERT KEY
JHE
K26
SET SCAN CODE INTO AH, 0 INTO AL
HOV
AX,INS_KEY.2S6
PUT J NTO OUTPUT BUFFER
JHP
K57

teB INT

1561
1561
1562
1563
1564

15£1
15£1
15E6
15£8
15£8
15Ee
15EE

; ------ UC TABLE SCAN
K 12
LABEL
BVTE
D8

06 0017 R 08
60
06 0017 R 20
00
06 0017 R 03

F6 06 0017 R 03
74 F3
84 26 0018 R
75 5e
os 2S 0018 R
30 2& 0017 R
3C 52
75 50
88 5200
£9 17EC R

ROM BIOS A-45


----- BREAK SHIFT FOUND

1600 80 FC 10  CMP AH, SCROLL_SHIFT ; IS THIS A TOGGLE KEY
1600 73 1A  JA K24 ; YES, HANDLE BREAK TOGGLE
1605 F6 D4  OR AH ; INVERT MASK
1607 20 26 0017 R AND KB_FLAG, AH ; TURN OFF SHIFT BIT
160C 00 BB  CMP AL, ALT_KEY+BOH ; IS THIS ALTERNATE SHIFT RELEASE
160C 75 3B  INTERRUPT_RET

----- ALTERNATE SHIFT KEY RELEASED, GET THE VALUE INTO BUFFER

160F A0 001B R MOV AL, AL_INPUT
1612 32 E4  XOR AH, AH ; SCAN CODE OF 0
1614 B8 26 0019 R MOV ALT_INPUT, AH ; ZERO OUT THE FIELD
1618 0A 0C  OR AL, AL ; HAS THE INPUT?0
161A 74 2E  JE K26 ; INTERRUPT_RET
161C E9 17F5 R JMP K5B ; IT WASN'T, SO PUT IN BUFFER

161F 3C BA  CMP AL, CAPS_KEY+BREAK_BIT ; SPECIAL CASE OF TOGGLE KEY
1622 75 0F  JNE K24 1 ; JUMP AROUND POTENTIAL UPDATE
1623 F6 06 0018 R 0  TEST KB_FLAG, CLICK_SEQUENCE
1627 74 0B  JZ K24 1 ; JUMP IF NOT SPECIAL CASE
162A 80 26 0018 R FD AND KB_FLAG, AND_MASK+CLICK_SEQUENCE ; MASK OFF MAKE

162F EB 19 90  JMP KB26 ; INTERRUPT IS OVER

----- BREAK OF NORMAL TOGGLE

1632 F6 D4 1 , NOT AH ; INVERT MASK
1634 20 26 001B R AND KB_FLAG, AH ; INICATE NO LONGER DEPRESSED
1638 EB 10  JMP SHORT KB26 ; INTERRUPT_RETURN

----- TEST FOR HOLD STATE

163A 3C 80  CMP AL, BOH ; TEST FOR BREAK KEY
163C 73 0C  JAE K26 ; NOTHING FOR BREAK CHARs FROM HERE ON

163E F6 06 0018 R 08 TEST KB_FLAG, HOLD_STATE ; ARE WE IN HOLD STATE?
1643 74 0E  JZ K2B ; BRANCH AROUND TEST IF NOT
164B 80 26 0018 R F7 AND KB_FLAG, NOT_HOLD_STATE ; TURN OFF THE HOLD STATE

164A K26 ; INTERRUPT_RETURN

164A 07  POP ES
164B 1F  POP DS
164C 5F  POP SI
164D 5E  POP DI
164E 5A  POP DX
164F 59  POP CX
1650 5B  POP BX
1651 5B  POP AX ; RESTORE STATE
1652 CF  IRET ; RETURN, INTERRUPTS BACK ON WITH FLAG CHANGE

----- NOT IN HOLD STATE, TEST FOR SPECIAL CHARs

1653 K2B ; NO-HOLD-STATE
1653 F6 06 0017 R 08 TEST KB_FLAG, ALT_SHIFT ; ARE WE IN ALTERNATE SHIFT
1658 75 03  JN2 K29 ; JUMP IF ALTERNATE SHIFT
165A E9 1749 R JMP K3B ; JUMP IF NOT ALTERNATE

----- TEST FOR ALT+CTRL KEY SEQUENCES

165D K29 ; TEST-RESET
165F F6 06 0017 R 04 TEST KB_FLAG, CTRL_SHIFT ; ARE WE IN CONTROL SHIFT ALSO
1662 74 69  JZ K3I ; NO_RESET
1664 3C 53  CMP AL, 0EL_KEY ; SHIFT STATE IS THERE, TEST KEY
1666 75 09  JNE K29 1 ; NO_RESET

1668 C7 06 072 R 1234 MOV RESET_FLAG, 1234H ; SET FLAG FOR RESET FUNCTION
166E EE 0043 R JMP NEAR PTR RESET ; JUMP TO POWER ON DIAGNOSTICS
1671 3C 52  K29 1 , CMP AL, INS_KEY ; CHECK FOR RESET WITH DIAGNOSTICS
1673 75 09  JNE K29 2 ; CHECK FOR OTHER

----- ALT-CTRL-SEQ UENCES

1675 C7 06 072 R 4321 MOV RESET_FLAG, 4321H ; SET FLAG FOR DIAGNOSTICS
167B EE 0043 R JMP NEAR PTR 1DIAGNOSTICS
167E 3C 3A  K29 2 , CMP AL, CAPS_KEY ; CHECK FOR KEVBOARD CLICK TOGGLE
1680 75 19  K29 3 ; CHECK FOR SCREEN ADJUSTMENT

1682 F6 06 018 R 02 TEST KB_FLAG, CLICK_SEQUENCE
1687 75 CL  JNZ K26 ; JUMP IF SEQUENCE HAS ALREADY OCCURRED

1689 80 36 0018 R 04 XOR KB_FLAG, CLICK_ON ; TOGGLE BIT FOR AUDIO KEYSTROKE

168E 80 0E 0018 R 02 OR KB_FLAG, CLICK_SEQUENCE ; SET CLICK_SEQUENCE STATE
1683 EB BS  JMP SHORT KB26 ; INTERRUPT IS OVER
1695 3C 4D  K29 3 , CMP AL, RIGHT_ARROW ; ADJJUST SCREEN TO THE RIGHT?
1697 75 12  JNE K29 4 ; LOOK FOR RIGHT ADJUSTMENT
1699 EB 186E R CALL GET_POS ; GET THE # OF POSITIONS SCREEN IS SHIFTED

169C 3C FC  CMP AL, 0-RANGE ; IS SCREEN SHIFTED AS FAR AS POSSIBLE?
169E 7C AA  JL K26 ; OUT OF RANGE
16A0 FE 00 0089 R DEC HOH_POS ; SHIFT VALUE TO THE RIGHT
16A4 FE CB  DEC AL ; DECREASE RANGE VALUE
16A6 EE 187A R CALL PUT_POS ; RESTORE STORAGE LOCATION
16A8 EB 14  JMP SHORT K29 5 ; ADJUST
16AB 3C 4B  K29 4 , CMP AL, LEFT_ARROW ; ADJUST SCREEN TO THE LEFT?
16A9 75 1E  JNE K3I ; NOT AN ALT_CTRL_SEQUENCE
16AF EB 186E R CALL GET_POS ; GET NUMBER OF POSITIONS SCREEN IS SHIFTED

16B2 3C 04  CMP AL, RANGE ; IS SCREEN SHIFTED AS FAR AS POSSIBLE?
16B4 7F 94  JG K26 ; OUT OF RANGE
16B6 EE 06 0089 R INC HOH_POS ; SHIFT SCREEN TO THE LEFT
16BA EB CO  INC AL ; INCREASE NUMBER OF POSITIONS
16BC EE 187A R CALL PUT_POS ; PUT POSITION BACK IN STORAGE
16BF 00 02 K29_5: MOV AL, 2 ; ADJUST
16C1 BA 0304 MOV DX, 304H ; ADDRESS TO CRT CONTROLLER
16C4 EE OUT DX, AL
16C6 00 0089 R MOV AL, MORP_LOC ; COLUMN POSITION
16C8 42 INC DX ; POINT AT DATA REGISTER
16C9 EE OUT DX, AL ; MOV POSITION
16CA E9 1644 R

;;;;;;;;;; IN ALTERNATE SHIFT, RESET NOT FOUND
K31: 
CMP AL, 57 ; NOT-RESET
JNE K32 ; TEST FOR SPACE KEY
JMP K37 ; NOT THERE

;;;;;;;;;; ALT-INPUT-TABLE
K30: LABEL BYTE
DB 02, 79, 80, 81, 75, 76, 77

16DD 47 48 49 DB 71, 72, 73 ; 10 NUMBERS ON KEYPAD
16DE 10 11 12 13 14 15 DB 16, 17, 18, 19, 20, 21, 22, 23 ; A-Z TYPEWRITER CHAR
16DF 16 17 DB 24, 25, 30, 31, 32, 33, 34, 35
16E0 22 23 DB 36, 37, 38, 44, 45, 46, 47, 48
16E1 31 32 DB 49, 50

16FA 0B 1606 R MOV DI, OFFSET K30 ; ALT-KEY-PAD
16FB B9 000A MOV CX, 10 ; LOOK FOR ENTRY USING KEYPAD
1700 F2/ AE REPNE SCASB ; LOOK FOR MATCH
1702 75 12 JNE K33 ; NO-ALT_KEYPAD
1704 B1 EF 16D7 R SUB DI, OFFSET K30+1 ; DI NOW HAS ENTRY VALUE
1708 A0 0019 R MOV AL, ALT_INPUT ; GET THE CURRENT BYTE
170B B4 0A MOV AH, 10 ; MULTIPLY BY 10
170F 6E E4 MUL AL
1710 F3 C7 ADD AX, 01 ; ADD IN THE LATEST ENTRY
1711 A2 0019 R MOV ALT_INPUT, AL ; STORE IT AWAY
1714 E9 1644 R JMP K26 ; THROW AWAY THAT KEYSRROKE

;;;;;;;;;; LOOK FOR SUPERSHIFT ENTRY
K32: 
MOV ALT_INPUT, 0 ; NO-ALT-KEYPAD
MOV CX, 26 ; DI, ES ALREADY POINTING
REPNE SCASB ; LOOK FOR MATCH IN ALPHABET
JNE K34 ; NOT FOUND, FUNCTION KEY OR OTHER
JMP K37 ; ASCI CODE of ZERO

;;;;;;;;;; LOOK FOR TOP ROW OF ALTERNATE SHIFT
K33: 
CMP AL, 2 ; ALT-TOP-ROW
JMP K34 ; KEY WITH ‘1’ ON IT

;;;;;;;;;; TRANSLATE ALTERNATE SHIFT PSEUDO SCAN CODES
K34: 
CMP AL, 14 ; IS IT IN THE REGION?
JAE K35 ; ALT-FUNCTION
JMP K57 ; PUT IT IN THE BUFFER

;;;;;;;;;; CLOSE-RETURN
K35: 
CMP AL, 118 ; CONVERT PRUDEO SCAN CODE TO RANGE
JNE K36 ; NOT ONE OF INTERESTING KEYS

;;;;;;;;;; Dump Key with ALPHABET
K36: 
JMP K26 ; ALT-CONTINUE
JMP K37 ; IN KEYPAD REGION

;;;;;;;;;; NOT IN ALTERNATE SHIFT
K37: 
JMP K63 ; ALTSFT PSEUDO SCAN TABLE
JMP K36 ; TRANSLATE THAT

;;;;;;;;;; TEST KB_FLAGS, CTL_SHIFT, ARE WE IN CONTROL SHIFT?
K38: 
TEST KB_FLAGS, 0 ; NOT-ALTSFT
J2 K48 ; NOT-CTL-SHIFT

;;;;;;;;;; CONTROL SHIFT, TEST SPECIAL CHARACTERS
K39: 
CMP AL, SCROLL_KEY ; TEST FOR BREAK
JNE K41 ; NO-BREAK

;;;;;;;;;; TEST FOR BREAK AND PAUSE KEYS
K40: 
CMP AL, 0 ; TEST FOR BREAK
JNE K41

;;;;;;;;;; TEST FOR BREAK AND PAUSE KEYS
1750 3C 46 MOV BX, BUFFER_HEAD ; GET CURRENT BUFFER HEAD
1752 B8 06 0071 R 80 MOV BX, BIOS_BREAK, 80H ; TURN ON BIOS_BREAK BIT
1755 CD 1B INT 1BH ; BREAK INTERRUPT VECOR
175F 2B C0 SUB BX, AX ; PUT OUT DUMMY CHARACTER
1761 05 07 MOV IBX3, AX ; PUT DUMMY CHAR AT BUFFER HEAD
1763 E9 144F R CALL K4 ; UPDATE BUFFER POINTER
1766 B9 01 001C R MOV BUFFER_TAIL, BX ; UPDATE TAIL
1769 E9 1644 R JMP K26 ; DONE WITH INTERRUPT
176D K41: ; TEST SPECIAL CASE KEY 55

1760 3C 37 CMP AL, 55
176F 07 B5 MOV AX, 114#256 ; START/STOP PRINTING SWITCH
1771 B2 72 MOV K57 ; BUFFER_FILL
1774 EB 76 90
;------ SET UP TO TRANSLATE CONTROL SHIFT
  K42: MOV BX, OFFSET K8 ; SET UP TO TRANSLATE CTRL
        ; NOT-KEY=59
  K43: CMP AL, 59 ; IS IT IN TABLE?
        JB K45 ; YES, GO TRANSLATE CHAR
        ; CTRL-TABLE-TRANSLATE
  K45: MOV BX, OFFSET K9 ; CTRL TABLE SCAN
        JMP K63 ; TRANSLATE SCAN

  K46: ; NOT IN CONTROL SHIFT
  K47: CMP AL, 71 ; TEST FOR KEYPAD REGION
        JAE K48 ; HANDLE Keypad REGION
  K48: ; TEST KB_FLAG,LEFT_SHIFT,RIGHT_SHIFT
        MOV K2,KB_FLAG,LEFT_SHIFT,RIGHT_SHIFT
        JZ K54 ; TEST FOR SHIFT STATE

  K49: ; UPPER CASE, HANDLE SPECIAL CASES
  K50: CMP AL, 15 ; BACK TAB KEY
        JNE K46 ; NOT-BACK-TAB
  K51: MOV AX, 15*256 ; SET PSEUDO SCAN CODE
        JMP SHORT K57 ; BUFFER_FILL
  K52: ; NOT-PRINT-SCREEN
  K53: CMP AL, 59 ; FUNCTION KEYS
        JB K47 ; NOT-UPPER-FUNCTION
  K54: ; TEST KB_FLAG,LEFT_SHIFT,RIGHT_SHIFT
  K55: MOV BX, OFFSET K12 ; UPPER CASE PSEUDO SCAN CODES
  K56: JMP K63 ; TRANSLATE SCAN
  K57: JMP SHORT K56 ; OK, TRANSLATE THE CHAR

  K58: ; KEYPAD KEYS, MUST TEST NUM LOCK FOR DETERMINATION
  K59: KEYPAD-REGION
  K60: ; TEST KB_FLAG,NUM_STATE, ARE WE IN NUMLOCK?
  K61: JNZ K52 ; TEST FOR SURE
  K62: ; TEST KB_FLAG,LEFT_SHIFT,RIGHT_SHIFT, ARE WE IN SHIFT
  K63: JZ K53 ; IF SHIFTED, REALLY NUM STATE

  K64: ; BASE CASE FOR KEYPAD
  K65: CMP AL, 74 ; BASE-CASE
        JNE K56 ; SPECIAL CASE FOR A COUPLE OF KEYS
  K66: JE K50 ; MINUS
  K67: CMP AL, 78 ; NUMBERS
  K68: JE K51 ; PLUS
  K69: SUB AL, 71 ; CONVERT ORIGIN
        MOV BX, OFFSET K15 ; BASE CASE TABLE
  K70: JMP K66 ; MINUS
  K71: MOV AX, 74*256+-'.' ; CONVERT TO PSEUDO SCAN
  K72: JMP SHORT K57 ; BUFFER_FILL
  K73: JMPSHORT K56 ; PLUS

  K74: ; MIGHT BE NUM LOCK, TEST SHIFT STATUS
  K75: TEST KB_FLAG,LEFT_SHIFT,RIGHT_SHIFT
  K76: TEST NUM_STATE
  K77: JE K59 ; SHIFTS TEMP OUT OF NUM STATE
  K78: SUB AL, 70 ; CONVERT ORIGIN
        MOV BX, OFFSET K14 ; NUM STATE TABLE
  K79: JMP SHORT K57 ; TRANSLATE CHAR

  K80: ; PLAIN OLD LOWER CASE
  K81: CMP AL, 59 ; NOT-SHIFT
        JB K55 ; NOT-LOWER-FUNCTION
  K82: XOR AL, AL ; SCAN CODE IN AM READY
        JNZ K80 ; BUFFER_FILL
  K83: ; NOT-LOWER-FUNCTION
  K84: MOV BX, OFFSET K10 ; LC TABLE
       转型字符的字符
  K85: DEC AL ; CONVERT ORIGIN
        XLAT CS:K11 ; CONVERT THE SCAN CODE TO ASCII

  K86: ; BUFFER-FILL
  K87: CMP AL, -1 ; IS THIS AN IGNORE CHAR?
        JE K59 ; YES, DO NOTHING WITH IT
        XOR AL, -1 ; LOOK FOR -1 PSEUDO SCAN
        JE K59 ; NEAR_INTERRUPT_RETURN

  K88: ; HANDLE THE CAPS LOCK PROBLEM
  K89: BUFFER-FILL-NOTEST
  K90: TEST KB_FLAG,CAPS_STATE ; ARE WE IN CAPS LOCK STATE?
  K91: JZ K61 ; IF NOT, CONVERT LOWER TO UPP.

  K92: ; CONVERT ANY UPPER CASE TO LOWER CASE
  K93: CMP AL, 'A' ; FIND OUT IF ALPHABETIC
        JB K61 ; NOT_CAPS_STATE
  K94: CMP AL, 'Z' ; NOT_CAPS_STATE
        JMP SHORT K61 ; CONVERT TO LOWER CASE

  K95: ; NEAR-INTERRUPT-RETURN
  K96: JMP K26 ; INTERRUPT_RETURN

  K97: ; CONVERT ANY LOWER CASE TO UPPER CASE
  K98: CMP AL, 'a' ; FIND OUT IF ALPHABETIC
        JB K61 ; NOT_CAPS_STATE
  K99: CMP AL, 'z' ; NOT_CAPS_STATE
        JMP SHORT K61 ; CONVERT TO UPPER CASE
MOV BX, BUFFER_TAIL
GET THE END POINTER TO THE BUFFER
MOV SI, BX
SAVE THE VALUE
CALL K4
ADVANCE THE TAIL
CMP BX, BUFFER_HEAD
HAS THE BUFFER WRAPPED AROUND?
JNE K61_1
BUFFER_FULL_BEEP
PUSH BX
SAVE BUFFER_TAIL
MOV BX, 0B8H
DURATION OF ERROR BEEP
MOV CX, 48H
FREQUENCY OF ERROR BEEP HALF TONE
CALL K48_NOISE
OUTPUT NOISE
AND KB_FLAG, FOFH
CLEAR ALT, CLR, LEFT AND RIGHT
SHIFTS
AND KB_FLAG_1, OFH
CLEAR POTENTIAL BREAK OF INS,CAPS
NUM AND SCROLL SHIFT
AND KB_FLAG_2, IFH
CLEAR FUNCTION STATES
POP BX
RETRIEVE BUFFER TAIL
JMP K26
RETURN FROM INTERRUPT
K61_1: 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 K48_NOISE
OUTPUT AUDIO FEEDBACK OF KEY
STROKE
POP BX
RETRIEVE BUFFER_TAIL VALUE
JMP K26
INTERRUPT_RETURN
------ TRANSLATE SCAN FOR PSEUDO SCAN CODES
--- TRANSLATE-SCAN
SUB AL, 59
CONVERT ORIGIN TO FUNCTION KEYS
SUB AX, 80
TRANSLATE-SCAN-ORDG
XLAT CS: K9
CTL TABLE SCAN
MOV AH, AL
PUT VALUE INTO AH
SAR AL, CL
ZERO ASCII CODE EXTENDED
JMP K57
PUT IT INTO THE BUFFER

--- 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 05 POINTS AT THE BIOS DATA AREA

--- OUTPUT ---
AL CONTAINS THE SHIFTED VALUE.

--- GET_POS_PROC NEAR ---
PUSH CX
SAVE SHIFT REGISTER
MOV AX, BX
MOV SBYTE PTR VAR_DELAY, GET STORAGE LOCATION
AND AL, OFH
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.

--- PUT_POS_PROC NEAR ---
PUSH CX
SAVE REGISTER
MOV CL, 4
SHIFT COUNT
SPL AL, CL
PUT IN HIGH ORDER NIBBLE
MOV CL, BYTE PTR VAR_DELAY
GET DATA BYTE
AND CL, OFH
CLEAR OLD VALUE IN HIGH NIBLE
OR AL, CL
COMBINE HIGH AND LOW NIBLES
MOV BYTE PTR VAR_DELAY, AL
PUT IN POSITION
POP CX
RESTORE REGISTER

--- PUT_POS_ENDP ---

--- MANUFACTURING ACTIVITY SIGNAL ROUTINE ---- INVOKE THRU THE TIMER
--- TICK ROUTINE DURING MANUFACTURING ACTIVITIES ---- (ACCESS THRU INT 1CH)

--- MFG_TICK_PROC FAR ---
PUSH AX
SAVE AX, AX
OUT 13H, AL
SEND A 00 TO PORT 13 AS A
IN AL, PORT_B
ACTIVITY SIGNAL
MOV AH, AL
SAVE ORIG SETTING
MOV AH, 10011101B
MAKE SURE MUX IS -> RIGHT AND
NOT AL
ISOLATE SPEAKER BIT
AND AL, 00000001B
FLIP ALL BITS
OR AL, AH
ISOLATE SPEAKER DATA BIT (NOW IN
OR AL, 00001000B
OPPOSITE SENSE)
OUT PORT_B, AL
COMBINE WITH ORIG DATA FROM
PORT_B
OUT PORT_B, AL
AND DISABLE INTERNAL SPEAKER
MOV AL, 20H
EOI TO INTR. CHIRP
MOV AX, 20H
POP AX
IRET

--- MFG_TICK_ENDP ---
;-----------------------------------------------------------------------------------------;
; CONVERT AND PRINT ASCII CODE
; AL MUST CONTAIN NUMBER TO BE CONVERTED.
; AX AND BX DESTROYED
;-----------------------------------------------------------------------------------------;

XPC_BYTE PROC NEAR

XPC_BYTE

PUSH AX

SAVE FOR LOW NIBBLE DISPLAY

MOV CL, 4

SHIFT COUNT

SHR AL, CL

NIBBLE SNAP

CALL XLAT_PR

DO THE HIGH NIBBLE DISPLAY

PUS AX

RECOVER THE NIBBLE

AND AL, OFH

ISOLATE TO LOW NIBBLE

XLAT_PR PROC NEAR

CONVERT 00-0F TO ASCII CHARACTER

ADD AL, 090H

ADD FIRST CONVERSION FACTOR

DAA

ADJUST FOR NUMERIC AND ALPHA RANGE

ADC AL, 040H

ADD CONVERSION AND ADJUST LOW NIBBLE

DAA

ADJUST HIGH NIBBLE TO ASCII RANGE

PRT_HEX PROC NEAR

PUS BX

DISPLAY CHARACTER IN AL

MOV AH, 14

PRT HEX ENDP

BX

CALL VIDEO_10

IBCF

MFM IN

ADD AL, 0

CALL VIDEO_10

IF NOT VALID, RETURN ELSE

DEC AH

TEST FOR AH = 0

MOV AL, OFH

ISOLATE TO LOW NIBBLE

AND AL, OFH

TEST FOR AH = 1

TEST FOR AH = 2

ADD AX, 0

1 OR AH, AH

TEST FOR 'CORRECT'

INC DK

IN WHICH CASE, USE CARD 1

SUB DX, DX

ASSUME TO USE CARD 0

TEST CH, 00000100B

UNLESS THERE ARE TWO CARDS

CALL FAKE

FAKE WILL MAP ERROR BITS FROM RS232 TO CORRESPONDING ONES

FOR THE PRINTER

CHECK IF ANY FLAGS WERE SET

MOVE FSED ERROR CONDITION TO AH

MOVE FAILED ERROR CONDITION TO AH

MOVE IN STATUS FOR 'CORRECT'

JMP SHORT B10_3

AAAAAAAAAA

THEN RETURN

MOV AL, 090H

IN STATUS FOR 'CORRECT'

MOV AH, OFH

HERE

MOV AH, AH

HERE

IF DH FREE INIT

FOR THAT'S

FREE INIT

JMP SHORT B10_3

THEN RETURN

rts

ENDP

END

A-50 ROM BIOS
;INITIALIZE QUEUE SO MESSAGE

;1910 50 B12: PUSH AX ;SAVE AL
;1911 64 01 MOV AH, 01 ;IS SEND A CHAR DOWN COMM L1NE
;1913 CD 14 INT 014H ;SEND THE CHAR
;1915 EB 1925 R CALL FAKE ;FAKE WILL MAP ERROR BITS FROM
;RS232 TO CORRESPONDING ONES
;FOR THE PRINTER
;191B 59 POP AX ;RESTORE AL
;1919 68 F6 OR DH, DH ;SEE IF NO ERRORS WERE RETURNED
;191B 7B 02 JZ B12_1
;191D 8B 69 MOV AH, DH ;IF THERE WERE ERRORS, RETURN THEM
;191F EB CC JMP SHORT B10_3 ;AND RETURN
;1921 8D 10 MOV AH, 010H _B12_1 ;PUT 'CORRECT' RETURN STATUS IN AH
;1923 EB C8 JMP SHORT B10_3 ;AND RETURN
;REPRINT ENDP

;THIS PROC MAPS THE ERRORS RETURNED FROM A BIOS INT14 CALL
;TO THOSE 'LIKE THAT' OF AN INT17 CALL.
;BREAK, FRAMING, PARITY, OVERRUN ERRORS ARE LOGGED AS 1/0
;ERRORS AND A TIME OUT IS MOVED TO THE APPROPRIATE BIT
;FAKE PROC NEAR
;1925 32 F6 XOR DH, DH ;CLEAR FAKED STATUS FLAGS
;1927 F6 C4 IE TEST AH, 011110B ;CHECK FOR BREAK, FRAMING, PARITY
;OVERRUN
;192A 74 03 JZ B13_1, ERRORS, IF NOT THEN CHECK FOR
;TIME OUT.
;192C 86 08 MOV DH, 01000B ;SET BIT 3 TO INDICATE '1/0 ERROR'
;192E C3 RET ;AND RETURN
;B13_1: TEST AH, 080H
;192F 66 C4 80 B13_2 TEST FOR TIME OUT RETURNED
;1932 74 02 JZ B13_2 ;IF NOT TIME OUT, RETURN
;1934 86 09 MOV DH, 09H ;IF TIME OUT
;1936 C3 RET FAKE ENDP

;NEW_INT9
;THIS ROUTINE IS THE INTERRUPT 9 HANDLER WHEN THE MACHINE IS
FIRST POWERED ON AND CASSETTE BASIC IS GIVEN CONTROL. IT
HANDLES THE FIRST KEystrokes ENTERed FROM THE KEYBOARD AND
PERFORMS "SPECIAL" ACTIONS AS FOLLOWS:
;IF ESC IS THE FIRST KEY ENTERED MINI-WELCOME IS EXECUTED
;IF CTRL-ESC IS THE FIRST SEQUENCE "LOAD CAS1:R," IS EXECUTED GIVING THE USER THE ABILITY TO BOOT
FROM CASSETTE
;AFTER THESE KEystrokes OR AFTER ANY OTHER KEystrokes THE
INTERRUPT 9 VECTOR IS CHANGED TO POINT AT THE REAL
INTERRUPT 9 ROUTINE.

;NEW_INT9 PROC FAR
;1937 3C 01 CMP AL, 1 ;IS THIS AN ESCAPE KEY?
;1939 7A 10 JE ESC_KEY ;JUMP IF AL=ESCAPE KEY
;193B 3C 10 CMP AL, 9B ;ELSE, IS THIS A CTRL KEY?
;193D 7A 06 JE CTRL_KEY ;JUMP IF AL=CTRL KEY
;193F EB E01B R CALL REAL_VECTOR_SETUP ;OTHERWISE, INITIALIZE REAL
;INT 9 VECTOR
;1942 CD 09 INT 9H ;PASS THE SCAN CODE IN AL
;1944 CF IRET ;RETURN TO INTERRUPT 4BH
;1945 ESC_KEY:
;1945 80 0E 0017 R 04 OR KB_FLAG, 04H ;TURN ON CTRL SHIFT IN KB_FLAG
;1948 CF IRET ;RETURN TO INTERRUPT 4BH
;194B ESC_FLAG:
;194B F6 06 0017 R 04 TEST KB_FLAG, 04H ;HAS CONTROL SHIFT OCCURED?
;1950 7A 29 JZ ESC_ONLY ;JE ESCONLY
;1952 C6 06 0017 R 00 MOV KB_FLAG, 0 ;CONTROL ESCAPE HAS OCCURED, PUT MESSAGE IN BUFFER FOR CASSETTE
;LOAD
;1954 6E 00 CF MOV KB_FLAG, 0 ;ZERO OUT CONTROL STATE
;1956 9E 07 PUSH DS ;SMALL INTERRUPT
;1958 07 POP ES ;INITIALIZE ES FOR BIOS DATA
;195A DE PUSH DS ;SAVE OLD DS
;195C 05 POP DS ;POINT DS AT CODE SEGMENT
;195F EB 1983 R MOV SI, OFFSET CAS_LOAD ;GET MESSAGE
;1965 BF 001E R MOV DI, OFFSET KB_BUFFER ;POINT AT KEYBOARD BUFFER
;1969 89 000F 90 MOV CX, CAS_LENGTH ;LENGTH OF CASSETTE MESSAGE
;196D 6E 02 FC LOOP T_LOOP ;BEGIN LOOP
;196F 6A 0F POP DS ;RETRIEVE BIOS DATA SEGMENT
;1970 C7 06 001A R 001E R T_LOOP: LODSB ;GET ASCII CHARACTER FROM MESSAGE
;1973 C7 06 001C R 003C R STOSB ;PUT IN KEYBOARD BUFFER
;1976 66 02 FC LOOP T_LOOP
;1978 6F 03 POP DS ;RETRIEVE BIOS DATA SEGMENT
;197A C7 06 001A R 001E R ;----- INITIALIZE QUEUE 50 MESSAGE WILL BE REMOVED FROM BUFFER
;197D C7 06 001C R 003C R MOV BUFFER_TAIL, OFFSET KB_BUFFER+(CAS_LENGTH+2)
;197F 87 EB E01B R CALL REAL_VECTOR_SETUP
;1981 CF IRET
;ESC_ONLY:
;1977 EB E01B R CALL REAL_VECTOR_SETUP
;197A CF IRET
;197B ESC_KEY:
;1978 EB E01B R CALL REAL_VECTOR_SETUP
;197E BF 90 MOV CX, MINI
;1981 FF E1 JMP CX ;ENTER THE WORLD OF KEYBOARD CAPER
;----- MESSAGE FOR OUTPUT WHEN CONTROL-ESCAPE IS ENTERED AS FIRST KEY SEQUENCE
;CAS_LOAD LABEL BYTE
;DB 'LOAD "CAS1:R",'
;1983 3C 4F 4F 41 44 20 22 MOV BUFFERS HEAD, OFFSET KB_BUFFER
;43 41 53 31 3A 22
;2C 52
;1989 0D DB 'LOAD "CAS1:R",'
;198B 000F CAS_LENGTH EQU $ - CAS_LOAD
;198D NEW_INT9 ENDP
THIS INTERFACE PROVIDES A TELETYPewriter 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 INCREased IF THE ROW
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

1992

WRITE_TTY PROC NEAR

1993

PUSH AX ; SAVE REGISTERS

1994

PUSH AX ; SAVE CHAR TO WRITE

1995

BA E0 0062 R MOV BH,ACTIVE_PAGE ; GET CURRENT PAGE SETTING

1996

53

PUSH BX ; SAVE IT

1997

9A 0F

MOV BL,BH ; IN BL

1998

32 FF

XOR BH,BH

1999

D1 E3

SAL BX,1 ; CONVERT TO WORD OFFSET

19A0

BB FE 0050 R MOV DX,[(BX)+OFFSET CURSORPOSN] ; GET CURSOR POSITION

19A1

5B

POP BX ; RECOVER CURRENT PAGE

19A2

5A

POP AX ; RECOVER CHAR

19A3

58

DX NOW HAS THE CURRENT CURSOR POSITION

19A4

3C 0B

CMP AL,B ; IS IT A BACKSPACE?

19A5

74 50

JE UB ; BACKSPACE

19A6

3C 00

CMP AL,0DH ; IS IT A CARRIAGE RETURN?

19A7

74 54

JE UB ; CARRET

19A8

3C 0A

CMP AL,0AH ; IS IT A LINE FEED

19A9

74 10

JE U10 ; LINE_FEED

19AA

3C 07

CMP AL,07H ; IS IT A BELL

19AB

74 50

JE U01 ; BELL

19AC

58

WRITE THE CHAR TO THE SCREEN

19AD

B4 0A

MOV AH,10 ; WRITE CHAR ONLY

19AE

B9 0001

MOV CX,1 ; ONLY ONE CHAR

19AF

CA CD

MOV CX,10H ; WRITE THE CHAR

19B0

B0 FC2

INC DL

19B1

3A 1E 004A R CMP DL,BYTE PTR CRT_COLS ; TEST FOR COLUMN OVERFLOW

19B2

75 31

JNZ U7 ; SET_CURSOR

19B3

32 02

XOR DL,DL ; COLUMN FOR CURSOR

19B4

--- LINE FEED

19B5

U0:

19B6

80 FE 1B

CMP DH,24

19B7

9C 2B

JNZ U6 ; SET_CURSOR_INC

19B8

B4 02

; SCROLL REQUIRED

19B9

MOV AH,2

19BA

CD 10

INT 10H ; SET CURSOR

19BB

--- DETERMINE VALUE TO FILL WITH DURING SCROLL

19BC

A0 0049 R MOV AL,CRT_MODE ; GET THE CURRENT MODE

19BD

3C 04

CMP AL,4 ; READ-CURSOR

19BE

74 04

JC U2 ; READ CURSOR

19BF

32 FF

XOR BH,BH ; FILL WITH BACKGROUND

19C0

EB 06

JMP SHORT U3 ; SCROLL-UP

19C1

B4 08

U2: MOV AH,8

19C2

CD 10

INT 10H ; READ CHAR/ATTR AT CURRENT CURSOR

19C3

8A FC

MOV BH,AH ; STORE IN BH

19C4

BB 0601

MOV AX,60H ; SCROLL ONE LINE

19C5

2B C9

SUB CX,CX ; UPPER LEFT CORNER

19C6

B6 18

MOV DH,24 ; LOWER RIGHT ROW

19C7

BA 1E 004A R MOV DL,BYTE PTR CRT_COLS ; LOWER RIGHT COLUMN

19C8

EB CA

DEC DL

19C9

50 10

INT 10H ; SCROLL UP THE SCREEN

19CA

56

POP AX ; RESTORE THE CHARACTER

19CB

E9 0F70 R JMP VIDEO_RETURN ; RETURN TO CALLER

19CC

FE C6

JMP BX

19CD

F5 B4 02

U7: MOV AH,2

19CE

7B F4

JMP U4 ; ESTABLISH THE NEW CURSOR

19CF

--- BACK SPACE FOUND

19D0

A0 02

U8: OR DL,DL ; ALREADY AT END OF LINE

19D1

FE C6

JMP BX

19D2

F0 FE CA

DEC DL ; NO -- JUST MOVE IT BACK

19D3

EB F4

JMP U7 ; SET_CURSOR

19D4

CARRIAGE RETURN FOUND

19D5

A0 32 02

U9: XOR DL,DL ; MOVE TO FIRST COLUMN

19D6

EB F0

U7 ; SET_CURSOR

19D7

--- BELL FOUND

19D8

B3 02

U11: MOV BL,2 ; SET UP COUNT FOR BEEP

19D9

EB FF31 R CALL BEEP ; SOUND THE PO0 BELL

19DA

EB E3

U5 ; ITT_RETURN

19DB

WRITE_TTY ENDP

A-52 ROM BIOS
THIS PROCEDURE WILL ISSUE SHORT TONES TO INDICATE FAILURES
THAT 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

ERR_BEEP PROC NEAR
PUSHF ; SAVE FLAGS
PUSH BX
G3: ; DISABLE SYSTEM INTERRUPTS
MOV BL, 1 ; COUNTER FOR A SHORT BEEP
CALL BEEP ; DO THE SOUND
G4: ; DELAY BETWEEN BEEPS
LOOP G4
DEC DL
JNZ G3
DO SOME MORE
G5: ; LONG DELAY BEFORE RETURN
LOOP G5
POP BX
POPFD ; RESTORE ORIG CONTENTS OF BX
POPFD ; RESTORE FLAGS TO ORIG SETTINGS
RET ; RETURN TO CALLER
ERR_BEEP ENDP

REAL_VECTOR_SETUP

; THIS ROUTINE WILL INITIALIZE THE INTERRUPT 9 VECTOR TO
; POINT AT THE REAL INTERRUPT ROUTINE.

REAL_VECTOR_SETUP PROC NEAR
PUSH AX ; SAVE THE SCAN CODE
PUSH BX
PUSH ES
XOR AX, AX ; INITIALIZE TO POINT AT VECTOR
SECTOR(0)
MOV ES, AX
MOV BX, 0024H ; POINT AT INTERRUPT 9
MOV WORD PTR ES: (BX), OFFSET KB_INT ; MOVE IN OFFSET OF 
ROUTINE
INC BX
ADD 2 TO BX
MOV CX
PUSH CS
GET CODE SEGMENT OF BIOS (SEGMENT 
RELOCATEABLE)
POP AX
MOV WORD PTR ES: (BX), AX ; MOVE IN SEGMENT OF ROUTINE
POP ES
POP BX
POP AX
RET
REAL_VECTOR_SETUP ENDP

KB_NOISE PROC NEAR

; 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.

STI
PUSH AX
PUSH BX
PUSH CX
IN AL, 061H ; GET CONTROL INFO
PUSH AX ; SAVE
LOOP01:
AMO AL, OFCH ; TURN OFF TIMER GATE AND SPEAKER
DATA
OUT 061H, AL
OUT 061H, AL ; OUTPUT TO CONTROL
HALF CYCLE TIME FOR TONE
PUSH CX
LOOP02: LOOP LOOP02 ; SPEAKER OFF
OR AL, 2 ; TURN ON SPEAKER BIT
OUT 061H, AL
OUT 061H, AL ; OUTPUT TO CONTROL
POP CX
LOOP03: LOOP LOOP03 ; ANOTHER HALF CYCLE
DEC BX ; TOTAL TIME COUNT
RET
LOOP04: LOOP LOOP04 ; RETRIEVE FREQUENCY
RET

KB_NOISE ENDP
A-56 ROM BIOS
; 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 6845 IF DISPLAY PAGE IS CURRENT DISPLAY

E4B8 SET_CPOS PROC NEAR
E4B9 MOV CL,BH
E4BA OR CH,CH ; ESTABLISH LOOP COUNTER
E4BB SAL CX,1 ; WORD OFFSET
E4BC MOV SI,CX ; USE INDEX REGISTER
E4BD MOV EBX,0050 R ; CMP ACTIVE PAGE,BH
E4BE JNZ C24 ; SET_CPOS RETURN
E4BF MOV AX,DX ; GET ROW/COLUMN TO AX
E4C0 CALL C25 ; CURSOR_SET
E4C1 JMP VIDEO_RETURN
E4C2 C24 ; VIDEO_RETURN
E4C3 SET_CPOS ENDP

E4C4 C25 PROC NEAR
E4C5 MOV EAX,E5C2 R ; CALL POSITION ; DETERMINE LOCATION IN REGEN
E4C6 MOV CX,AH ; BUFFER
E4C7 MOV EAX,E004E R ; ADD CX,CRT_START ; ADD IN THE START ADDRESS FOR THIS
E4C8 SAR CX,1 ; PAGE
E4C9 ADD_LOOP BY 2 FOR CHAR ONLY COUNT
E4CA MOV AX,AL4 ; REGISTER NUMBER FOR CURSOR
E4CB MOV EAX,E472 R ; CALL C23 ; OUTPUT THE VALUE TO THE 6845
E4CC MOV EAX,E3 ; RET
E4CD C26 ENDP

E4D1 ACT_DISP_PAGE PROC NEAR
E4D2 MOV AL,0B0H ; CRT/CPU PAGE REG FUNCTION
E4D3 JNZ SET_CRTCPU ; YES, GO HANDLE IT
E4D4 MOV EAX,ACTIVE_PAGE,AL ; SAVE ACTIVE PAGE VALUE
E4D5 MOV CX,CRT_LEN ; CONVERT AL TO WORD
E4D6 MOV EAX,ACTIVE_PAGE,AL ; SAVE ACTIVE PAGE VALUE
E4D7 MOV AL,CRT_START,AX ; SAVE START ADDRESS FOR LATER USE
E4D8 MOV EAX,CRT,AX ; START ADDRESS TO AX
E4D9 SAR CX,1 ; DIVIDE BY 2 FOR 6845 HANDLING
E4DA MOV EAX,CRT,AX ; DIVIDE BY 2 FOR 6845 HANDLING
E4DB MOV AH,E12 ; 6845 REGISTER FOR START ADDRESS
E4DC MOV EAX,E472 R ; CALL C23 ; RECOVER PAGE VALUE
E4DD MOV EAX,SAL BX,1 ; #2 FOR WORD OFFSET
E4DE MOV EAX,EBX,0050 R ; MOV AX,EBX + OFFSET_CURSOR_POSN1; GET CURSOR FOR THIS
E4DF MOV EAX,E0E2 R ; CALL C25 ; SET THE CURSOR POSITION
E4E0 MOV EAX,E0F70 R ; JMP VIDEO_RETURN

E4E1 SET_CRTCPU PROC NEAR
E4E2 MOV EAX,AL ; CRT/CPU PAGE REG READ/WRITE REQUEST
E4E3 MOV AL,0B0H ; CRT/CPU PAGE REG FUNCTION
E4E4 JNZ SET_CRTCPU ; YES, GO HANDLE IT
E4E5 MOV EAX,CRT,AL ; SET BOTH CRT AND CPU PAGE REGS
E4E6 MOV AX,VALUE TO SET IN CRT PAGE REG
E4E7 MOV EAX,CRT,AL ; SET BOTH CRT AND CPU PAGE REGS
E4E8 MOV AX,VALUE TO SET IN CRT PAGE REG
E4E9 MOV EAX,CRT,AL ; SET BOTH CRT AND CPU PAGE REGS
E4EA MOV AX,VALUE TO SET IN CRT PAGE REG
E4EB MOV EAX,CRT,AL ; SET BOTH CRT AND CPU PAGE REGS
E4EC MOV AL,CRT,AL ; CURRENT CONTENTS OF CRT/CPU PAGE REG
E4ED MOV EAX,CRT,AL ; CURRENT CONTENTS OF CRT/CPU PAGE REG

E4F0 MOV EAX,0A0 ; SAVE REQUEST IN AH
E4F1 MOV EAX,DX,VGA_CTL ; SET ADDRESS OF GATE ARRAY
E4F2 MOV AL,DL ; GET STATUS
E4F3 MOV AL,0B0H ; VERTICAL RETRACE?
E4F4 JNZ C26 ; NO, WAIT FOR IT
E4F5 MOV DX,PAGREG ; SET 10 ADDRESS OF PAGE REG
E4F6 MOV EAX,0080A R ; MOV AL,PAGREG ; GET DATA LAST OUTPUT TO REG
E4F7 CMP AH,BOH ; READ FUNCTION REQUESTED?
E4F8 JZ C29 ; YES, DON'T SET ANYTHING
E4F9 CMP AH,BOH ; VALID REQUEST?
E4FA JNC C29 ; NO, PRETEND IT WAS A READ REQUEST
E4FB JNZ C27 ; SET CPU REG?
E4FC JNC C27 ; SET CPU REG?
E4FD JZ C27 ; NO, GO SEE ABOUT CRT REG
E4FE CPL EAX ; SHIFT VALUE TO RIGHT BIT POSITION
E4FF MOV EAX,CPL EAX ; CLEAR OLD CPU VALUE
E500 MOV EAX,AL ; CPU,Reg
E501 MOV EAX,CPL EAX ; CLEAR OLD CPU VALUE
E502 OR AL,AL ; AL READY IN NEW VALUE

E4F0 SET_CRTCPU ENDP

ROM BIOS A-57
This is a page from the ROM BIOS documentation. The text appears to be a part of a program listing in machine code, with comments and explanations in English. The program appears to be related to cursor and color settings in a graphics interface.
;---------- HANDLE COLOR 0 BY SETTING THE BACKGROUND COLOR
; AND BORDER COLOR
E54E B0 3E 0049 R 04
CMP CRT_MODE,4 ; IN ALPHA MODE?
E554 72 0E ; YES, JUST SET BORDER REG
E556 B0 10
MOV AL,10H ; SET PALETTE REG 0
E558 EE
OUT DL,AL ; SELECT VGA REG
E559 8A C3
MOV AL,BL ; GET COLOR
E55B EE
OUT DX,AL ; SET IT
E55C B0 02
C35C 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 0066 R MOV CRT_PALLETTE,AL ; SAVE THE COLOR VALUE
E565 E9 0F70 R JMP VIDEO_RETURN
 ;---------- HANDLE COLOR 1 BY CHANGING PALETTE REGISTERS
E568 A0 0049 R C31 MOV AL,CRT_MODE ; GET CURRENT MODE
E56B B9 0095 R MOV CX,OFFSET M0072 ; POINT TO 2 COLOR TABLE ENTRY
E56E 3C 0F ; 2 COLOR MODE?
E570 74 0F
JE C33 ; YES, JUMP
E571 3C 04
CMP AL,4 ; 4 COLOR MODE?
E574 74 0F
JE C32 ; YES, JUMP
E576 3C 05
CMP AL,5 ; 5 COLOR MODE?
E578 74 04
JE C33 ; YES, JUMP
E57A 3C 0A
CMP AL,0AH ; 4 COLOR MODE?
E57C 75 20
JNE C30 ; NO, GO TO 16 COLOR SET ENTRY
E57E B9 0030 R C32 MOV CX,OFFSET M0074 ; POINT TO 4 COLOR TABLE ENTRY
E580 D0 C8
ROR AX,1 ; FOR 2 COLOR MODE
E583 73 03
ADD AL,AX
E585 BA C1 04
INC CS ; NEXT ENTRY
E588 B8 D9
MOV BX,CX ; TABLE ADDRESS IN BX
E58A 43
INC BX ; SKIP OVER BACKGROUND COLOR
E58B B9 0003
MOV CX,OFFSET M0072-1 ; SET NUMBER OF REGS TO FILL
E58E B4 11
MOV AL,11H ; AH IS REGISTER COUNTER
E590 BA C4
C35F MOV AL,AX ; GET REG NUMBER
E592 EE
OUT DX,AL ; SELECT IT
E593 2E ; EA 07
OUT 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 C33
E59C E8 00
JMP SHORT C38
E59E B4 11
C36 ; MOV AH,11H ; AH IS REGISTER COUNTER
E5A0 B9 000F
MOV CX,15 ; NUMBER OF PALETTES
E5A3 8A C4
C37 ; MOV AL,AH ; GET REG NUMBER
E5A5 EE
OUT DX,AL ; SELECT IT
E5A6 EE
OUT DX,AL ; SET PALETTE VALUE
E5A7 FE C4
INC AH ; NEXT REG
E5A9 E2 F8
LOOP C37
E5AB 32 C0
C3B ; XOR AL,AL ; SELECT LOW REG TO ENABLE VIDEO
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 8A 26 004A R MOV AH,BYTE PTR CRT_COLS ; GET NUMBER OF COLUMNS
E5B5 A0 0049 R MOV AL,CRT_MODE ; CURRENT MODE
E5B9 B8 3E 0062 R MOV BH,ACTIVE_PAGE ; GET CURRENT ACTIVE PAGE
E5BC 5F
POP D1 ; 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 53
PUSH BX ; SAVE REGISTER
E5C3 8A 00
MOV BX,AX
E5C5 8A C4
MOV AL,AX ; ROWS TO AL
E5C7 F6 26 004A R MUL BYTE PTR CRT_COLS ; DETERMINE BYTES TO ROW
E5CB 32 FF
XOR BH,BH
E5CD 03 C3
ADD AX,BX ; ADD IN COLUMN VALUE
E5CF D1 E0
POP AX,1 ; = 2 FOR ATTRIBUTE BYTES
E5D1 8D 0D
POP BX
E5D2 02 C3
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:CODE, DS:DATA, ES:DATA

SCROLL_UP  PROC  NEAR

E503 8A 08  MOV BL, AL ; SAVE LINE COUNT IN BL
E505 8C 04  CMP AH, 4 ; TEST FOR GRAPHICS MODE
E508 72 03  JC C39 ; HANDLE SEPARATELY
E50A E9 F259 R  JMP GRAPHICS_UP
E50D  C39:  UP_CONTINUE

E50F 53  PUSH BX ; SAVE FULL ATTRIBUTE IN BH
E510 E8 E609 R  CALL SCROLL_POSITION ; GET SETUP FOR SCROLL
E513 74 20  JZ C44 ; BLANK_FIELD
E515 E3 F0  ADD SI, AX ; FROM ADDRESS
E517 8A E6  MOV AH, DH ; # ROWS IN BLOCK
E519 E9 2A E3  SUB AH, BL ; # ROWS TO BE MOVED
E51B EB E62F R  C40: CALL C45 ; MOVE ONE ROW
E51D E3 F5  ADD SI, BP
E51E E0 F3  ADD DI, BP ; POINT TO NEXT LINE IN BLOCK
E51F E5 2F CC  DEC AH ; COUNT OF LINES TO MOVE
E522 E6 75 F5  JMZ C40 ; ROW_LOOP
E525 5B  C41: POP AX ; RECOVER ATTRIBUTE IN AH
E526 E9 80 20  MOV AL, ; FILL WITH BLANKS
E528 E9 E63B R  CALL C46 ; CLEAR THE ROW
E52B E5 F0  ADD DI, BP ; POINT TO NEXT LINE
E52C FE C8  DEC BL ; COUNTER OF LINES TO SCROLL
E52D 75 F7  JNZ C42 ; CLEAR_0DP
E52F E9 0F70 R  C43: JMP VIDEO_RETURN
E531 8A DE  MOV BL, DH ; GET ROW COUNT
E533 EB ED  JMP C41 ; GO CLEAR THAT AREA

SCROLL_UP  ENDP

SCROLL_POSITION  PROC  NEAR

E609 E8 E5C2 R  CALL POSITION ; CONVERT TO REGEN POINTER
E60C 03 00 004E R ADD AX, CRT_START ; OFFSET OF ACTIVE PAGE
E610 8B F8  MOV DI, AX ; TO ADDRESS FOR SCROLL
E612 8B F0  MOV SI, AX ; FROM ADDRESS FOR SCROLL
E614 2B D1  SUB DX, CX ; DX = #ROWS, RCOLS IN BLOCK
E616 FE C6  INC DH
E617 EB F2  INC DL ; INCREMENT FOR 0 ORIGIN
E618 A1 32 ED  XOR CR, CH ; SET HIGH BYTE OF COUNT TO ZERO
E61A 8C 00 4A R MOV BP, CRT_COLS ; GET NUMBER OF COLUMNS IN DISPLAY
E61C 0B E8  ADD BP, DI ; TIMES 2 FOR ATTRIBUTE BYTE
E61E 8A C3  MOV AL, BL ; GET LINE COUNT
E620 F6 26 004A R MUL BYTE PTR CRT_COLS ; DETERMINE OFFSET TO FROM
E622 03 C0  ADD AX, AX ; #2 FOR ATTRIBUTE BYTE
E624 06  PUSH ES ; ESTABLISH ADDRESSING TO REGEN BUFFER
E626 1F  POP DS ; FOR BOTH POINTERS
E628 0A 0B  OR BL, BL ; 0 SCROLL MEANS BLANK FIELD
E62A E6 C3  RET ; RETURN WITH FLAGS SET

SCROLL_POSITION  ENDP

; ------ MOVE_ROW

E62F C45  PROC  NEAR
E62F 8A CA  MOV CL, DL ; GET # OF COLS TO MOVE
E631 56  PUSH SI
E632 57  PUSH DI ; SAVE START ADDRESS
E633 E9 3F A5  REP MOVSW ; MOVE THAT LINE ON SCREEN
E635 5F  POP DI ; SET ATTRIBUTE IN AH
E636 5E  POP SI ; RECOVER ADDRESSES
E637 C3  RET ; RETURN WITH FLAGS SET

C45  ENDP

; ------ CLEAR_ROW

E638 C46  PROC  NEAR
E638 8A CA  MOV CL, DL ; GET # COLUMNS TO CLEAR
E63A 57  PUSH DI
E63B F3/ AB  REP STOSW ; STORE THE EMPTY CHARACTER
E63D 5F  POP DI
E63E C3  RET

C46  ENDP

SCROLL_DOWN  PROC  NEAR

E63F 5D  STD ; DIRECTION FOR SCROLL DOWN
E640 8A D8  MOV BL, AL ; LINE COUNT TO BL
E642 8C 04  CMP AH, 4 ; TEST FOR GRAPHICS
E645 72 03  JC C47 ; JMP GRAPHICS_DOWN
E647 E9 F305 R  C47: PUSH BX ; SAVE ATTRIBUTE IN BH
E649 5B  C48  MOV AX, DX ; LOWER RIGHT CORNER
E64A E8 E609 R  CALL SCROLL_POSITION ; GET REGEN LOCATION
E64B 72 C2  MOV AX, DX ; LOWER RIGHT CORNER
E64C E9 E609 R  CALL SCROLL_POSITION ; GET REGEN LOCATION
E64E 74 F1  JS C51 ; SI IS FROM ADDRESS
E650 2B F0  SUB SI, AX
E652 EA E6  MOV AH, DH ; GET TOTAL # ROWS
E654 2A E3  SUB AH, BL ; COUNT TO MOVE IN SCROLL

A-60  ROM BIOS
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.

```assembly
E65B EB E62F R
E65B 2B F5
E650 2B F0
E65F FE CC
E651 7F F5
E653 5B
E664 B0 20
E66B FE CD
E680 75 F7
E66F EB 91
E671 BA DE
E673 EB EE
E675 SCROLL_DOWN ENDP

MODE_ALIVE PROC NEAR
    ; SAVE USED REGS
    PUSH AX
    MOV SI,SP
    MOV AX, SS:[SI+12] ; GET SEG FROM STACK
    MOV ES, AX
    MOV SI, DX ; OFFSET IN SI
    MOV DX, VGA_CTL ; SET VGA CONTROL PORT
    IN AL, DX ; GET VGA STATUS
    AND AL, 0BH ; IN VERTICAL RETRACE?
    JNZ CS3 ; YES, WAIT FOR IT TO GO AWAY
    IN AL, DX ; GET VGA STATUS
    AND AL, 0BH ; IN VERTICAL RETRACE?
    JZ CS4 ; NO, WAIT FOR IT
    POP AX
    OR AL, AL ; SET PALETTE REG?
    JZ CS5 ; YES, GO DO IT
    CMP AL, 2 ; SET ALL REGS?
    JNE CS6 ; SET BORDER COLOR REG?
    CMP AL, 1 ; SET BORDER COLOR REG?
    JNE CS9 ; NO, DON'T DO ANYTHING
    MOV AL, 2 ; SET BORDER COLOR REG NUMBER
    JMP SHORT CS6
    CMP AL, BL ; GET DESIRED REG NUMBER IN AL
    AND AL, 0FH ; STRIP UNUSED BITS
    OR AL, 10H ; MAKE INTO REAL REG NUMBER
    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
    XOR CO, AL
    JMP SHORT CS9
    MOV AH, 10H ; AH IS REG COUNTER
    MOV AH, 0 ; 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
    INC AH ; NEXT REG
    CMP AH, 20H ; LAST PALETTE REG?
    JB CS8 ; 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
```

E684 EE
E604 EE   OUT DX,AL ; PUT IN VGA REG
E605 E9 0F70 R CS9:    JMP VIDEO_RETURN ; ALL DONE
E606      SET_PALLETTE ENDP
E607 MFG_UP PROC NEAR
E608 50   PUSH AX
E609 9E   MOV AL, DS:AX
E60A BB ---- R MOV AX, DS:DATA
E60B 8E D8   MOV DX, AX
E60C A0 0005 R MOV AL, MFG_TST ; GET MFG CHECKPOINT
E60D 40 10   OUT DX, AL ; OUTPUT IT TO TESTER
E60E E6 10   MOV MFG_TST, AL
E60F 42 0005 R DEC AL ; DROP IT BY 1 FOR THE NEXT TEST
E610 FE CB   MOV DS, A850
E611 IF   POP DS
E612 5B   POP AX
E613 C3   RET
E614 C3   RET
E615 MFG_UP ENDP
E616 ORG O6F2H
E617 JMP NEAR PTR BOOTSTRAP

E618 SUBROUTINE TO SET UP CONDITIONS FOR THE TESTING OF B250 AND B259 INTERRUPTS. ENABLES MASKABLE EXTERNAL INTERRUPTS, CLEAR THE B259 INTR RECEIVED FLAG BIT, AND ENABLES THE DEVICE'S B259 INTR (WHICHEVER IS BEING TESTED). IT EXPECTS TO BE PASSED:

(05) = ADDRESS OF SEGMENT WHERE INTR_FLAG IS DEFINED
(01) = OFFSET OF THE INTERRUPT BIT MASK

UPON RETURN:

- INTERRUPT BIT FOR THE DEVICE = 0
- NO REGISTERS ARE ALTERED.

E619

E620 SUI PROC NEAR
E621 50   PUSH AX
E622 FB   STI ; ENABLE MASKABLE EXTERNAL INTERRUPTS
E623 E4 21 MOV AH, CS:[013] ; GET INTERRUPT BIT MASK
E624 20 26 0084 R AND INTR_FLAG, AH ; CLEAR B259 INTERRUPT REG' D FLAG
E625 E6 21 MOV AL, INTA01 ; CLEAR CURRENT INTERRUPTS
E626 22 C4 AND AL, AH ; ENABLE THIS INTERRUPT, TOO
E627 E0 21 OUT INTA01, AL ; WRITE TO B259 (INTERRUPT CONTROLLER)
E628 5B   POP AX
E629 C3   RET
E630 C3   RET
E631 SUI ENDP

E632 SUBROUTINE WHICH CHECKS IF A B259 INTERRUPT IS GENERATED BY THE B250 INTERRUPT.

IT EXPECTS TO BE PASSED:

- (01) = OFFSET OF THE INTERRUPT BIT MASK
- (05) = ADDRESS OF SEGMENT WHERE INTR_FLAG IS DEFINED

IT RETURNS:

- (CF) = 1 IF NO INTERRUPT IS GENERATED
- 0 IF THE INTERRUPT OCCURRED
- POP INTERRUPT = COMPLEMENT OF THE INTERRUPT MASK
- NO OTHER REGISTERS ARE ALTERED.

E633

E634 E706 CS059 PROC NEAR
E635 51   PUSH CX
E636 2B C9 SUB BX, CX ; SET PROGRAM LOOP COUNT
E637 0A 48 0045 MOV AL, CS:[013] ; GET INTERRUPT MASK
E638 34 FF XOR AL, OFFH ; COMPLEMENT MASK SO ONLY THE INTR
E639 E70E 84 06 0084 R AT25: TEST INTR_FLAG, AL ; B259 INTERRUPT OCCUR?
E640 E712 75 03 JNE AT27 ; YES - CONTINUE
E641 E714 E2 F8 LOOP AT25 ; WAIT SOME MORE
E642 E716 F9 STC ; TIME'S UP - FAILED
E643 E717 59 AT27: POP CX
E644 E718 C3 RET
E645 CS059 ENDP

E646 SUBROUTINE TO WAIT FOR ALL ENABLED B250 INTERRUPTS TO CLEAR (50 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.

IT EXPECTS TO BE PASSED:

- (0X) = ADDRESS OF THE INTERRUPT ID REGISTER

IT RETURNS:

- (0X) = CONTENTS OF THE INTERRUPT ID REGISTER
- (CF) = 1 IF INTERRUPTS ARE STILL PENDING
- 0 IF NO INTERRUPTS ARE PENDING (ALL CLEAR)
- NO OTHER REGISTERS ARE ALTERED.

E647

E648 E719 WB250C PROC NEAR
E649 51   PUSH CX
E64A 2B C9 SUB CX, CX ; READ INTERRUPT REG
E64B 3C 01 CMP AL, 1 ; INTERRUPTS STILL PENDING?
E64C 74 05 JE AT29 ; NO - GOOD FINISH
E64D E2 F9 LOOP AT28 ; KEEP TRYING
E64E E723 F8 STM ; TIME'S UP - ERROR
E64F E724 EB 01 JMP SHORT AT30
E650 E726 8F AT29: JC LC
E651 E727 59 AT30: POP CX
E652 E728 C3 RET
E653 WB250C ENDP

A-62 ROM BIOS
---INT 14---

RS232 I/O

This routine provides byte stream I/O to the communications port according to the parameters:

(AH)=0: Initialize the communications port
(AH) has parameters for initialization

---8---7---6---5---4---3---2---1---0---
BAUD RATE ----PC----STOP BIT---- WORD LENGTH---

-------------------------------------------------------------
000 - 110: BAUD = 10 - 7 BITS
001 - 150: 01 - 000  1- 2 11 - 8 BITS
010 - 300: 11 - EVEN
011 - 600: 010 - 1200
100 - 1200: 010 - 2400
101 - 2400: 010 - 4800
111 - 4800: 010 - 9600

On return, the RS232 interrupts are disabled and conditions are set as in call to COMM status (AH)=3

(AH)=1: Send the character in (AL) over the COM port line.
(ALL 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 COM port 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 COM port status in (AX).
AH contains the line control status

BIT 7 = TIME-OUT
BIT 6 = TRAN 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 8250 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

ASSUME CS:CODE,DS:DATA
ORG 03729H

A1 LABEL WORD

DW 1017 ; 110 BAUD ; TABLE OF INIT VALUE
    746 ;
    373 ; 300
    186 ; 600
    93  ; 1200
    47  ; 2400
    23  ; 4800
    23  ; 4800
RS232_I0 PROC FAR

E729 FB
E72A IE
E72B 56
E72C 56
E72D 57
E72E SI
E72F 53
E730 BB F2
E731 BB FA
E732 01 E6
E733 0089
E734 00BD
E735 0017
E736 0017
E737 0017
RS232_10 PROC

E739 FB
E73A IE
E73B 52
E73C 56
E73D 57
E73E SI
E73F 53
E740 BB F2
E741 BB FA
E742 01 E6
E743 E6 13BH
E744 0000 R
E745 00 BB 0000 R
E746 0B 02
E747 74 13
E748 0A E4
E749 74 16
E750 74 CC
E751 74 47
E752 74 CC
E753 74 6C
E754 75 03
E755 75 03
E756 E9 E7F3 R

--- VECTOR TO APPROPRIATE ROUTINE
STI ; INTERRUPTS BACK ON
PUSH DS ; SAVE SEGMENT
PUSH DX
PUSH SI
PUSH DI
PUSH CX
PUSH BX
MOV SI,DX ; RS232 VALUE TO SI
MOV BX,DS ; RS232 BASE ADDRESS
SML SI,1 ; WORD OFFSET
CALL DDS ; CALL DDS TO BIOS DATA SEGMENT
MOV DX,RS232_BASE+1 ; GET BASE ADDRESS
OR DX,DX ; TEST FOR 0 BASE ADDRESS
JZ A3 ; RETURN
OR AH,AH ; TEST FOR (AH)=0
JZ A4 ; COMMUN INIT
DEC AH ; TEST FOR (AH)=1
JZ A5 ; SEND AL
DEC AH ; TEST FOR (AH)=2
JZ A12 ; RECEIVE INTO AL
DEC AH ; TEST FOR (AH)=3
JMP A1B ; COMMUNICATION STATUS
E764 BA E0
E765 B3 C2 03
E766 B0 00
E767 EE
E768 EE
E769 EE
E770 BF E729 R
E771 03 F4
E772 BB 94 0000 R
E773 42
E774 2E: BA 45 01
E775 2E: BA 05
E776 EE
E777 EE
E778 EE
E779 EE
E77A EE
E77B EE
E77C EE
E77D EE
E77E EE
E77F EE
E780 EE
E781 EE
E782 EE
E783 EE
E784 EE
E785 EE
E786 EE
E787 EE
E788 EE
E789 E8 E002 R
E78A 7F 09
E78B 6A 01
E78C 75 09
E78D 4A
E78E EB E002 R
E78F 74 F4
E790 EE
E791 EA C1
E792 EE
E793 EE
E794 EE
E795 EE
E796 EE
E797 EE
E798 EE
E799 EE
E79A EE
E79B EE
E79C EE
E79D EE
E79E EE
E79F EE
E7A0 59
E7A1 59
E7A2 59
E7A3 59
E7A4 59
E7A5 59
E7A6 59
E7A7 59
E7A8 59
E7A9 59
E7AA 59
E7AB 59
E7AC 59
E7AD 59
E7AE 59
E7AF 59
E7B0 59
E7B1 59
E7B2 59
E7B3 59
E7B4 59
E7B5 59
E7B6 59
E7B7 59
E7B8 59
E7B9 59
E7BA 59
E7BB 59
E7BC 59
E7BD 59
E7BE 59
E7BF 59
E7C0 59
E7C1 59
E7C2 59
E7C3 59
E7C4 59
E7C5 59
E7C6 59
E7C7 59
E7C8 59
E7C9 59
E7CA 59
E7CB 59
E7CC 59
E7CD 59
E7CE 59
E7CF 59
E7D0 59
E7D1 59
E7D2 59
E7D3 59
E7D4 59
E7D5 59
E7D6 59
E7D7 59
E7D8 59
E7D9 59
E7DA 59
E7DB 59
E7DC 59
E7DD 59
E7DE 59
E7DF 59
E7E0 59
E7E1 59
E7E2 59
E7E3 59
E7E4 59
E7E5 59
E7E6 59
E7E7 59
E7E8 59
E7E9 59
E7EA 59
E7EB 59
E7EC 59
E7ED 59
E7EE 59
E7EF 59
E7F0 59
E7F1 59
E7F2 59
E7F3 59
E7F4 59
E7F5 59
E7F6 59
E7F7 59
E7F8 59
E7F9 59
E7FA 59
E7FB 59
E7FC 59
E7FD 59
E7FE 59
E7FF 59

E773 04
E774 04
E775 04
E776 04
E777 04
E778 04
E779 04
E77A 04
E77B 04
E77C 04
E77D 04
E77E 04
E77F 04
E780 04
E781 04
E782 04
E783 04
E784 04
E785 04
E786 04
E787 04
E788 04
E789 04
E78A 04
E78B 04
E78C 04
E78D 04
E78E 04
E78F 04
E790 04
E791 04
E792 04
E793 04
E794 04
E795 04
E796 04
E797 04
E798 04
E799 04
E79A 04
E79B 04
E79C 04
E79D 04
E79E 04
E79F 04
E7A0 04
E7A1 04
E7A2 04
E7A3 04
E7A4 04
E7A5 04
E7A6 04
E7A7 04
E7A8 04
E7A9 04
E7AA 04
E7AB 04
E7AC 04
E7AD 04
E7AE 04
E7AF 04
E7B0 04
E7B1 04
E7B2 04
E7B3 04
E7B4 04
E7B5 04
E7B6 04
E7B7 04
E7B8 04
E7B9 04
E7BA 04
E7BB 04
E7BC 04
E7BD 04
E7DE 04
E7DF 04
E7E0 04
E7E1 04
E7E2 04
E7E3 04
E7E4 04
E7E5 04
E7E6 04
E7E7 04
E7E8 04
E7E9 04
E7EA 04
E7EB 04
E7EC 04
E7ED 04
E7EE 04
E7F0 04
E7F1 04
E7F2 04
E7F3 04
E7F4 04
E7F5 04
E7F6 04

A-64 ROM BIOS
This routine will read timer1. The value read is returned in ax.

---

**WAIT FOR STATUS**

PROC NEAR

**MOV** BL, $5232 Tim_out(01), Load outer loop count

**ENDP**

**DESCRIPTION:**

This subroutine performs a thorough check out of an INS8250 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 INS8250 interrupt system and that the $8250
   interrupts trigger an $8250 (interrupt controller) interrupt.
4. Performing the loop back test:
   a) Testing what was written/read and that the transmitter
      holding register 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 INS8250 card to test.

**NOTE:** The assumption has been made that the modem adapter is

**located at 03F8H; the serial printer at O2FBH.**

**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)
= 23H: Serial printer adapter test failure
= 24H: MODEM adapter test failure

**BL = 2**

- Permanent zero bits in interrupt enable register were incorrect
- Permanent zero bits in interrupt identification register were incorrect
- Permanent zero bits in data set control register were incorrect
- Permanent zero bits in the line status register were incorrect
- Received data available interrupt test failed
- (the interrupt was not generated)
- Received data available interrupt failed to clear
- 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)
- Transmitter holding register empty interrupt test failed
- Receiver line status interrupt test failed
- (the interrupt was not generated)
- Overflow error
- Parity error
- Framing error
- Break interrupt error
- Receiver line status interrupt failed to clear
- C-F: MODEM status interrupt test failed
- (the interrupt was not generated)
- Delta clear to send error
- Delta data set ready error
- Trailing edge ring indicator error
- Delta receive line signal detect error
IC-IF MODEM STATUS INTERRUPT FAILED TO CLEAR

10H AN 8259 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 B259 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)

EB31 UART PROC NEAR
ASSUME CS:CODE,DS:DATA

EB31 JE PUSH DS
EB32 E4 21 IN AL,INTA01 ; CURRENT ENABLED INTERRUPTS
EB34 50 PUSH AX ; SAVE FOR EXIT
EB35 0C 01 OR AL,00000001B ; DISABLE TIMER INTR DURING THIS
EB37 E6 21 OUT INTA01,AL
EB39 9C PUSHF ; SAVE CALLER'S FLAGS (SAVE INTR
EB3A 52 PUSH DX ;フラグ) SAFE BASE ADDRESS OF ADAPTER CARD
EB3B EB 138B R CALL DDS ; SET UP 'DATA' AS DATA SEGMENT
EB3C EB 004C R CALL 18250
EB41 73 03 JNC AT1 ; ALL OK
EB43 E9 EB4B R JMP AT14 ; A PORT'S ZERO BITS WERE NOT ZERO!

= 0084
INITIALIZE PORTS FOR MASTER RESET STATES AND TEST PERMANENT
ZERO DATA BITS FOR CERTAIN PORTS.

EB46 BF 0041 R CALL 18250
EB47 73 02 JNC AT1 ; ALL OK
EB49 E9 EB4B R JMP AT14 ; A PORT'S ZERO BITS WERE NOT ZERO!

IMB290 INTERRUPT SYSTEM TEST
ONLY THE INTERRUPT BEING TESTED WILL BE ENABLED.

EB52 E8 06F5 R CALL SUI ; SET UP FOR INTERRUPTS
EB55 FE C3 INC BL ; ERROR REPORTER (INIT IN 18250)
EB57 42 INC DX ; POINT TO INTERRUPT ENABLE
EB58 B0 01 MOV AL,1 ; REGISTER
EB5A EE OUT DX,AL
EB5B 53 Push BX ; SAVE ERROR REPORTER
EB5C 03 C2 04 ADD DX,4 ; POINT TO LINE STATUS REGISTER
EB5F B4 01 MOV AH,1 ; SET RECEIVER DATA READY BIT
EB61 BB 0400 MOV BX,0400H ; INTR TO CHECK, INTR IDENTIFIER
EB64 BB 0003 MOV CX,3 ; INTERRUPT RD REG 'INTR'
EB67 E8 0AFB R CALL ICT ; PERFORM TEST FOR INTERRUPT
EB6A 5B POP DX ; RESTORE ERROR INDICATOR
EB6B 3C FF CMP AL,OFFH ; INTERRUPT ERROR OCCUR?
EB6D 74 36 JE AT4 ; YES
EB6F E8 E706 R CALL C5059 ; GENERATE B295 INTERRUPT?
EB72 72 33 JC AT5 ; NO
EB74 4A DEC DX
EB75 4A DEC DX ; RESET INTR BY READING REC REG BUFR
EB76 EC IN AL,DX ; DON'T CARE ABOUT THE CONTENTS!
EB77 42 INC DX
EB78 42 INC DX ; INTR RD REG
EB79 E8 E719 R CALL W8250C ; WAIT FOR INTR TO CLEAR
EB7C 73 03 JNC AT3 ; OK
EB7E E9 EB4B R JMP AT12 ; IDN'T CLEAR

TRANSMITTER HOLDING REGISTER EMPTY INTERRUPT TEST
THIS TEST HAS BEEN MODIFIED BECAUSE THE DIFFERENT B259'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 B259 INTERRUPTS). IT IS STILL VALID FOR TESTING
IF AN 8259 INTERRUPT IS GENERATED AS EXPECTED TO THE B259
INTERRUPT AND THAT THE B259 INTERRUPT CLEAR 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 8259 INTERRUPT AND CLEAR PROPERLY TO PASS
THIS TEST.
E914 EB E719 R          CALL W8250C ; WAIT FOR INTERRUPT TO CLEAR
E917 72 2F              JC AT12 ; IT DIDN'T
E919 4D                DEC, BP
E91A 74 09               JE AT12 ; ALL FOUR BITS TESTED -- GO ON
E91C 90 E4             SLL AH, 1 ; SET READY FOR NEXT BIT
E91E B3 C2 04          ADD DX, 4 ; MODEM STATUS REGISTER
E921 EB 06               JMP AT9 ; TEST NEXT BIT

POSSIBLE 8259 INTERRUPT CONTROLLER PROBLEM

E923 B3 C2 04
E925 EB 24
AT10: MOV BL, 10H ; SET ERROR REPORTER
AT11: JMP SHORT AT14

SET 9600 BAUD RATE AND DEFINE DATA WORD AS HAVING 8
BITS/WORD, 2 STOP BITS, AND ODD PARITY

E929 EB 00           JMP #*2 ; I/O DELAY
E93B B3 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 ; INTR EMBL REG
E940 B0 00                MOV AL, 0 ; SET FOR INTERNAL WRAP TEST
E942 C0 B4            INT WRAP ; DO LOOP BACK TRANSMISSION TEST
E944 B1 00                MOV CL, 0 ; ASSUME NO ERRORS
E946 73 05              JNC AT15 ; WRAP TEST PASSED
E948 B0 C3 10         ADD BL, 10H ; ERROR INDICATOR
AT13:

AN ERROR WAS ENCOUNTERED SOMEWHERE DURING THE TEST

E94B B1 01                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 ON THE
                         ; CARRY FLAG.

E94F 5A          PUSH BX ; GET BASE ADDRESS OF 8250 ADAPTER
E94E 53                CALL I8250 ; RE-INITIALIZE 8250 PORTS
E952 8C          POP BX
E953 2E BA 25          MOV AH, CS : EDI3 ; GET DEVICE INTERRUPT MASK
E955 20 2E 0084 R     AND INTR_FLAG, AH ; CLEAR DEVICE'S INTERRUPT FLAG BIT
E955 80 F4 FF          AND XOR AH, OFFH ; FLIP BITS
E956 E4 21               IN AL, INTA01 ; GET CURRENT INTERRUPT PORT
E959 8A C4                OR AL, AH ; DISABLE THIS DEVICE INTERRUPT
E961 E6 21              OUT INTA01, AL
E963 90            POPF ; RE-ESTABLISH CALLER'S INTERRUPT
                         ; FLAG
                         ; RE-ESTABLISH CALLER'S INTERRUPT
                         ; OR CL, CL ; ANY ERRORS?
                         ; JE AT17 ; NO
E964 0A C9                MOV BM, 24H ; ASSUME MODEM ERRORS
E966 74 0C             JMP BH, 2 ; OR IS IT SERIAL?
E968 B7 24                MOV BM, 24H ; IT'S MODEM
E96A 80 FE 02             CMP DH, 2 ; IT'S SERIAL PRINTER
E96C 75 02          JMP AT16 ; SET CARRY FLAG TO INDICATE ERROR
E96F B7 23               MOV BM, 23H ; SET CARRY FLAG TO INDICATE ERROR
E971 F9          AT16: STC
E972 EB 01             POP DX ; RE-ESTABLISH CARRY FLAG
E974 F8              AT17: CLC
E975 8B                          CLC, SHORT AT18
E976 89 E6 21            POP AX ; RESTORE ENABLER INTERRUPTS
E978 83 85              OR INTA01, AL ; DEVICE INTS RE-ESTABLISHED
E979 8F                  POP DS ; RESTORE REGISTER
E979 C3          RET
E986 UART ENDP
E987 ORG OE867H
E987 EB 1561 R          JMP NEAR PTR KB_INT

NEC_OUTPUT

THIS ROUTINE SENSORS 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

(4) DESTROYED

A-68  ROM BIOS
NEC_OUTPUT  PROC  NEAR

E9BA 52  PUSH  DX  ; SAVE REGISTERS
E9BB 51  PUSH  CX  
E9BC 6A 00F4  MOV  DX,NEC_STAT  ; STATUS PORT
E9BF 33 C9  XOR  CX,CX  ; COUNT FOR TIME OUT

E991 EC  JZ:  IN  AL,DX  ; GET STATUS
E992 48 40  TEST  AL,DIO  ; TEST DIRECTION BIT
E994 7C 0C  JZ  J25  ; DIRECTION OK
E996 E2 F9  LOOP  J23

E998  J24:  ; TIME ERROR
E99B 80 0E 0041 R 80  OR  DISKETTE_STATUS,TIMJE_OUT
E99D 59  POP  CX  
E99E 5A  POP  DX  ; SET ERROR CODE AND RESTORE REGS
E99F 58  POP  AX  ; DISCARD THE RETURN ADDRESS
E9A0 F9  STC  ,  INDICATE ERROR TO CALLER
E9A1 C3  RET

E9A2 33 C9  J25:  XOR  CX,CX  ; RESET THE COUNT
E9A4 EC  J26:  IN  AL,DX  ; GET THE STATUS
E9A5 AB 80  TEST  AL,RM  ; IS IT READY?
E9A7 75 04  JNZ  J27  ; YES, GO OUTPUT
E9AB E2 F9  LOOP  J26  ; COUNT DOWN AND TRY AGAIN
E9AF EB EB  JMP  J24  ; ERROR CONDITION

E9A0  BA C4  MOV  AL,AH  ; GET BYTE TO OUTPUT
E9A1 42  INC  DX  ; DATA PORT IS 1 GREATER THAN
E9B0 EE  OUT  DX,AL  ; STATUS PORT
E9B1 59  POP  CX  ; OUTPUT THE BYTE
E9B2 5A  POP  DX  ; RECOVER REGISTERS
E9B3 C3  RET  ; CY = 0 FROM TEST INSTRUCTION

NEC_OUTPUT  ENDP

GET_PARM  PROC  NEAR

E9B4 1E  PUSH  DS  ; SAVE SEGMENT
E9B5 56  PUSH  SI  ; SAVE REGISTER
E9B6 2B C0  SUB  AX,AX  ; ZERO TO AX
E9B8 32 FF  XOR  BH,BH  ; ZERO BH
E9BA 8E DB  MOV  DS,AX  
E9BC C5 36 0078 R  ASSUME  DS:ABS0
E9BD 01 EB  LDS  SI,DISK_POINTER ; POINT TO BLOCK
E9C0 0B B1  SHR  BX,1  ; DIVIDE BX BY 2, AND SET FLAG FOR
E9C2 9C  PUSHF  ; SAVE OUTPUT BIT
E9C3 8A 20  MOV  AH,(SI+BX)  ; GET THE BYTE
E9C5 B3 FB 01  CMP  BX,1  ; IS THIS THE PARM WITH DNA
E9C8 75 05  JNZ  J27_1  
E9CA 60 CC 01  OR  AH,1  ; TURN ON NO DMA BIT
E9CB EB 0C  JMP  SHORT J27_2
E9CC 63 FB 0A  CMP  BX,10  ; MOTOR STARTUP DELAY?
E9CF 75 07  JNE  J27_2  
E9D0 80 FC 04  CMP  AH,4  ; GREATER THAN OR EQUAL TO 1/2 SEC?
E9D4 70 02  JGE  J27_2  ; YES, OKAY
E9DA 84 04  CMP  AH,4  ; NO, FORCE 1/2 SECOND DELAY
E9DB 9D J27_2:  POPF  ; GET OUTPUT BIT
E9DC 5E  POP  SI  ; RESTORE REGISTER
E9DD 5F  POP  DS  ; RESTORE SEGMENT
E9DE 72 AA  JC  NEC_OUTPUT  ; IF FLAG SET, OUTPUT TO CONTROLLER
E9ED C3  RET  ; RETURN TO CALLER

GET_PARM  ENDP

BOUND_SETUP  PROC  NEAR

; 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:DI AND DS:SI POINT TO THE BUFFER. THIS
; CALCULATED ADDRESS WILL ALWAYS ACCESS 64K OF MEMORY.
; BX DESTROYED

BOUND_SETUP  ENDP

ROM BIOS  A-69
BOUND_SETUP PROC NEAR
    ; RECALIBRATION
    ; EXECUTION

    ; SEEK
    ; SEEK THIS ROUTINE WILL MOVE THE HEAD ON THE NAME DRIVE
    ; TO THE NAME TRACK IF THE DRIVE HAS NOT BEEN ACCESSSED
    ; 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

SEEK PROC NEAR
    ; I CONTAINS OFFSET FOR CORRECT DRIVE, AL CONTAINS BIT MASK
    ; IN POSITION 0, 1 OR 2
    ; EAX 09
    ; POP CX
    ; RESTORE PARAMETER REGISTER
    ; EAOE E8 EA66 R
    ; MOV BX,OFFSET J32
    ; SET UP ERROR RECOVERY ADDRESS
    ; EALI S3
    ; PUSH BX
    ; NECESSARY FOR ROUTINE NEC_OUTPUT
    ; EALI 04
    ; MOV AL,1
    ; ESTABLISH MASK FOR RECAL
    ; EALI 03 CA
    ; USE DRIVE AS A SHIFT COUNT
    ; EA05 B1 E1 OFFF
    ; AND CX,OFFH
    ; MASK OFF HIGH BYTE
    ; EA0E 09 F3
    ; ADD SI,CX
    ; POINT SI AT CORRECT DRIVE
    ; EA0E 02 C0
    ; GET MASK FOR DRIVE
    ; EADB E8 EA8A R
    ; ----- HEAD IS MOVING TO CORRECT TRACK
    ; CALL CHK_STAT_2
    ; GET THE STATUS OF RECALIBRATED
    ; EAEF E8 EA6F R
    ; ----- DRIVE IS IN SYNCH WITH CONTROLLER, SEEK TO TRACK
    ; CALL CHK_STAT_2
    ; GET THE STATUS OF RECALIBRATED
    ; EAF9 9C
    ; PUSHF
    ; SAVE STATUS FLAGS
    ; EAAF 51
    ; PUSH CX
    ; SAVE REGISTERS
    ; EA40 B3 12
    ; MOV BL,1B
    ; HEAD SETTLE PARAMETER
    ; EA4F E8 EA84 R
    ; J29:
    ; HEAD_SETTLE
    ; EA52 B9 0226
    ; MOV CX,550
    ; 1 MS LOOP
    ; EA55 0A E4
    ; OR AH,AE
    ; TEST FOR TIME EXPIRED
    ; EA67 74 06
    ; JZ J31
    ; EA59 E2 FE
    ; LOOP J30
    ; DELAY FOR 1 MS
    ; EA5B FE CC
    ; DEC AH
    ; DECREMENT THE COUNT
    ; EA5D E8 F3
    ; JMP J29
    ; DO IT SOME MORE
    ; EA5F 59
    ; J31:
    ; POP CX
    ; RESTORE REGISTER
    ; EA60 90
    ; POPF
    ; EA61 72 06
    ; JC J32_2
    ; EA63 B8 20
    ; MOV BYTE PTR[SI],CH
    ; EA65 58
    ; J31_1:
    ; POP BX
    ; GET RID OF DUMMY RETURN
    ; EA66 J32
    ; POP BX
    ; RESTORE REGISTER
    ; EA67 5E
    ; POP SI
    ; UPDATE CORRECT
    ; EA68 C3
    ; RET
    ; RETURN TO CALLER
    ; EA69 C6 04 FF
    ; J32_2:
    ; MOV BYTE PTR[SI],OFFH
    ; UNKNOWN STATUS ABOUT SEEK
    ; RET
    ; OPERATION
    ; EA6C 58
    ; POP BX
    ; GET RID OF DUMMY RETURN
    ; EA6D E8 F7
    ; JMP SHORT J32
    ; EA6F SEEK
ENDP

BOUND_SETUP
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

chk_stat_2 proc near

 EA8F
 EA8F 53 push bx ; save registers
 EA70 56 push si
 EA71 33 db xor bx, bx ; number of sense interrupts to
 EA73 BE ea8b r mov si, offset j33_3 ; set up dummy return from
 j33_3: push bx
 EA76 56 push si
 EA77 B4 0B mov ah, 0Bh ; sense interrupt status
 EA79 E8 0Aa r call nec_output ; issue sense interrupt status
 EA7C E8 EA0 r call results ;
 EA7F 72 10 jc j35 ; nec time out, flags set in results

 EA81 A0 0042 r mov al, nec_status ; get status
 EA84 48 20 test al, seek_end ; is seek or recall operation done?
 EA86 75 0D jnz j35_1 ; jump if execution of seek or
 EA88 4B j33_3: dec bx ; dec loop counter

 EA89 75 EC dec bx
 EA8B 80 0E 0041 R 80 mov dx, 041h
 EA90 F9 j34: stc ; return error indication for

 EA91 5E pop si ; caller
 EA92 SE pop si
 EA93 SB pop bx
 EA94 C3 ret

 SEEK END HAS OCCURRED, CHECK FOR NORMAL TERMINATION

 J35_1: and al, 0coh ; mask normal termination bits
 EA95 74 F8 jz j35 ; jump if normal termination
 EA99 80 0E 0041 R 40 or diskette_status, bad_seek
 EA9E EB F0 jmp j34

 EA90

 result proc near

 EA00

 EA00 FC cld
 EA1A BF 0042 R mov dl, offset nec_status ; pointer to data area
 EA44 B1 05 push cx ; save counter
 EA45 52 push dx
 EA46 53 push bx
 EA47 B3 07 mov bl, 7 ; max status bytes

 EA48

 J3B: ; input_loop
 EA49

 EA49 33 C9 xor cx, cx ; counter
 EA4A BA 00F4 mov dx, nec_status ; status port

 EA4E AC

 EA51 A8 80 test al, 080h ; master ready
 EA51 75 0C jnz j40a
 EA53 E2 F9 loop j39 ; wait_master
 EA55 80 0E 0041 R 80 or diskette_status, time_out

 EA5A

 EA60 FC

 EA6A 79 007 ; max status bytes

 EA6B F9 stc ; set error return

 EA6B 5B pop bx
 EA6C 5A pop dx
 EA6B 59 pop cx
 EA6C 3B ret

 EA6F EC

 EA70 A8 40 test al, 040h ; test direction bit
 EA72 C6 75 07 jnz j42 ; ok to read status

 EA74

 EA7C 80 0E 0041 R 20 or diskette_status, bad_nec

 EA83 EF

 EA8B 42 inc dx ; point at data port
 EA8C EC in al, dx ; get the data
 EA8D B9 05 mov (01h), al ; store the byte
 EA8F 47 inc di ; increment the pointer
 EA9D 09 000A mov cx, 10 ; loop to kill time for nec
 EA93 E2 FE loop j43
 EA95 4A dec dx ; point at status port
 EA96 EC in al, dx ; get status
 EA97 A8 10 test al, 010h ; test for nec still busy
 EA99 7A 0D jz j44 ; results done

 EA9B FE CB dec bl ; decrement the status counter
 EA9D 75 CA jnz j30 ; go back for more

 EA9F EB E3 jmp j41 ; chip has failed

 Appendix A

 ROM BIOS A-71
; NUM_TRANS PROC NEAR
EAEI A0 0041 R MOV AL, NEC_STATUS+3 ; GET CYLINDER END ON
EAE4 3A 46 0B CMP AL, [BP+11] ; SAME AS WE STARTED
EAE7 A0 0047 R MOV AL, NEC_STATUS+5 ; GET ENDING SECTOR
EAE9 74 07 JZ J4S ; IF ON SAME CYL, THEN NO ADJUST
EAE9 B3 0B MOV BL, 8
EAEF EB 89 H4 R CALL GET_PHP ; GET EOT VALUE
EAF1 BA 04 MOV AL, AH ; INTO AL
EAF3 FE CO J4S: INC AL ; USE EOT+1 FOR CALCULATION
EAF5 2A 46 0A SUB AL, [BP+10] ; SUBTRACT FROM END
EAF8 BB 46 0E MOV [BP+143], AL
EAFB C3 RET
EAFC NUM_TRANS ENDP
EAFC RESULTS ENDP

; DISABLE PROC NEAR
EADF 42 21 IN AL, INTA01 ; READ CURRENT MASK
EAFB 89 46 10 MOV [BP+16], AX ; SAVE MASK ON THE SPACE ALLOCATED
EB02 B0 BF MOV AL, 0BHF ; MASK OFF ALL INTERRUPTS EXCEPT
EB04 E6 21 OUT INTA01, AL ; OUTPUT MASK TO THE B259
EB06 EB E9 E1 R CALL BOUND_SET ; SETUP REGISTERS TO ACCESS BUFFER
EB09 5B POP AX
EB0A C3 RET
EB0B DISABLE ENDP

; ENABLE PROC NEAR
EB0B 52 PUSH AX ; SAVE AX
EB0C B0 76 MOV AL, .0111010B ;
EB0E 06 43 OUT TIM_CTL, AL
EB10 50 PUSH AX
EB11 5B POP AX ; WAIT FOR B253 TO INITIALIZE
EB12 B0 F0 MOV AL, 0FFH ; INITIAL VALUE FOR B253
EB14 E6 41 OUT TIMER+1, AL ; LSB
EB16 50 PUSH AX
EB17 5B POP AX ; WAIT
EB1B E6 41 OUT TIMER+1, AL ; LSB
EB1A BE 46 10 MOV E5, [BP+16] ; GET ORIGINAL E5 VALUE FROM THE
EB1D E4 62 IN AL, 62H ; READ PORT C OF B255
EB1F 24 01 AND AL, 01H ; BIT = 1 MEANS KESTROKE HAS OCCURRED
EB21 50 PUSH AX ; SAVE IT ON THE STACK
EB22 E4 A0 MOV AL, NMI_PORT ; ENABLE NMI INTERUPTS
EB24 B0 B0 MOV AL, 00H ; MASK TO ENABLE NMI
EB26 E6 A0 OUT NMI_PORT, AL ; ENABLE NMI
EB28 BB 46 10 MOV AX, [BP+16] ; GET MASK FROM THE STACK
EB2E 5A POP DX
EB2F FB STI
EB30 C3 RET
EB31 ENABLE ENDP
CLOCK_WAIT PROC NEAR
; 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
; TIMERO.

INPUT NONE.
OUTPUT NONE. AX IS DESTROYED.

CLOCK_WAIT PROC ENDP

GET_DRIVE PROC NEAR
; 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 OHM 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.
OUTPUT AL CONTAINS THE BIT MASK. ALL OTHER REGISTERS ARE INTACT.

GET_DRIVE PROC ENDP

ROM_CHECK PROC NEAR
; 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 ENDP

ROM BIOS A-73
DISKETTE I/O

THIS INTERFACE PROVIDES ACCESS TO THE 5 1/4" DISKETTE DRIVES

INPUT

(AH)=0
RESET DISKETTE SYSTEM

(AH)=1
READ THE STATUS OF THE SYSTEM INTO (AL)

REGISTERS FOR READ/VERIFY/FORMAT

(DL) - DRIVE NUMBER (0-3 ALLOWED, 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 DESIRSED 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 QUANTES 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/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,DS,ES:DATA,ES:DATA

EC59 DISKETTE_I0 PROC FAR

ORG OEC59H

EC59 STI ; INTERRUPTS BACK ON

EC5A PUSH ES ; SAVE ES

EC5B PUSH AX ; SAVE COMMAND AND N_SECTORS

EC5C STI ; TIMER INITIAL VALUE

EC5D PUSH AX ; ALLOCATE ONE WORD ON STACK FOR

EC5E PUSH BX ; USE IN PROCS ENABLE AND DISABLE.

EC5F PUSH CX ; WILL HOLD 8259 MASK.

EC60 PUSH DX ; SAVE COMMAND AND N_SECTORS

EC61 PUSH SI ; SAVE SEGMENT REGISTER VALUE

EC62 PUSH DS ; SAVE ALL REGISTERS DURING

EC63 POP DI ; OPERATION

EC64 POP BP

EC65 MOV BP,SP ; SET UP POINTER TO HEAD PARM

EC66 CALL DDS ; SET DS=DATA

EC67 CALL J1 ; CALL THE REST TO ENSURE DS

EC68 CALL RESTORED

EC69 MOV BL,4 ; GET THE MOTOR WAIT PARAMETER

EC6A MOV BP+181,AH ; RETURN STATUS IN AL

EC6B MOV BP+181,AL ; STORE STATUS IN AL

EC6C MOV DS:DDS ; KEEP THE TIME COUNT FOR THE MOTOR

EC6D CALL GET_PARM

EC6E CALL GET_PARM

EC6F CALL GET_PARM

EC70 POP BX ; RESTORE SAVED FLAGS

EC71 POP AX

EC72 ADD SP,4 ; DISCARD DUMMY SPACE FOR 8259 MASK

EC73 MOV AX,ES ; RECOVER SEGMENT

EC74 MOV AH,1 ; SET THE CARRY FLAG TO INDICATE

EC75 MOV CX,CMC

EC76 MOV POP AX

EC77 RET 2 ; THROW AWAY SAVED FLAGS

A-74 ROM BIOS
RESET DISKETTE STATUS ROUTINE

DISKETTE_IO ENDP

DISK_RESET PROC NEAR

BA F0 MOV DH,AL ; SAVE 8 SECTORS IN DH
BA 26 003F R 7F
B0 OR AH,AH ; AH=0
74 JZ DISK_RESET
DISK_RESET
DEC AH
AH=1
74 JZ DISK_STATUS
EC9F MOV DISKETTE_STATUS,0 ; RESET THE STATUS INDICATOR
C6 06 0041 R 00
CMP DL,2 ; TEST FOR DRIVE IN 0-2 RANGE
FA JA J3 ; ERROR IF ABOVE
73 J3
EC9B DEC AH
26 AH=2
74 JZ DISK_READ
60 DEC AH
AH=3
ECAF JNZ J2 ; TEST_DISK_VERF
75 J2

EB91 E9 ED 30 R
DISKETTE_WRITE
JMP DISK_WRITE

ECB4 MOV FS,SEEK_STATUS
ECB8 MOV FS,AL

ECBC MOV FS,DISKETTE_STATUS
C6 06 0041 R 01
ECBF MOV FS,DISKETTE_STATUS,BAD_CMD ; ERROR CODE, NO SECTORS
BAD_CMD

ECC3 RET ; UNDEFINED OPERATION

disco

----- DISK_RESET PROC NEAR

----- RESET THE DISKETTE SYSTEM

DISK_FORMAT PROC

DISK_RESET

E19E B0 0010 MOV CX,10H ; NUMBER OF SENSE INTERRUPTS TO ISSUE

E1B4 B4 08 MOV AH,08H ; COMMAND FOR SENSE INTERRUPT
E1BB CALL NEC_OUTPUT ; COMMAND FOR SENSE INTERRUPT
E1C3 EB 89B8 R CALL NEC_OUTPUT

E1E6 EB EAA0 R MOV AL,MOTOR_STATUS

E1E9 AA 0042 R MOV AL,MOTOR_STATUS

E1EC 3C 00 MOV AL,MOTOR_STATUS

E1E7 74 12 MOV AL,DISKETTE_STATUS

E1EE 0E 1B MOV AL,DISKETTE_STATUS

E1EF 80 0E 1B MOV AL,DISKETTE_STATUS,SEEK_STATUS

E1F0 0E 1B MOV AL,DISKETTE_STATUS

E1F8 JMP SHORT J18

ECFA BE ECFA R MOV SI,OFFSET J4_2 ; DUMMY RETURN FOR

ECFD 56 MOV AH,00H ; BAD_WRITE

ECFE E2 1E MOV AL,DISKETTE_STATUS

ED00 EB F0 MOV AH,08H ; COMMAND FOR SENSE INTERRUPT

ED02 5E MOV AH,03H ; SPECIFY COMMAND

ED0B 03 MOV BL,1 ; STEP RATE TIME AND HEAD UNLOAD

ED0E B3 01 MOV BL,1 ; HEAD LOAD AND NO DMA

ED1A J8 ; TO THE NEC CONTROLLER

ED1B RET ; RETURN TO CALLER

ED1C RET ; RETURN TO CALLER

ENDP
DISKETTE DO TEST FOR FORMAT COMMAND ALLOW WAIT FOR MOTOR BOTH

PUSH AX SEND OUT TURN ON

ED74 E9 ED8D R JMP J18 TO THE CONTROLLER

DISK_FORMAT ENDP

-------- DISKETTE WRITE ROUTINE

DISK_WRITE PROC NEAR

80 0E 003F R B0 OR MOTOR_STATUS,B0H; INDICATE A WRITE OPERATION

B8 00 11 MOV BX,17 TO THE FILLER BYTE

ED93 53 PUSH BX SAVE PARAMETER INDEX ON STACK

ED9A E9 ECCD R JMP J18 TO THE CONTROLLER

DISK_WRITE ENDP

;-------- ALLOW WRITE ROUTEINE 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

ED45 51 PUSH CX SAVE THE T/S PARMS

ED46 FA CLI ; NO INTERRUPTS WHILE DETERMINING

ED47 C6 06 040 R FF MOV MOTOR_COUNT,OFFH; SET LARGE COUNT DURING OPERATION

ED4C E8 EB45 R CALL GET_DRIVE; GET THE DRIVE PARAMETER FROM THE STACK

ED4F B4 06 003F R CALL TEST MOTOR_STATUS,AL; TEST MOTOR FOR OPERATING

ED53 75 1F JNZ J14 IF RUNNING, SKIP THE WAIT

ED55 B0 06 003F R AND MOTOR_STATUS,OFH; TURN OFF RUNNING DRIVE

ED5A 08 06 003F R OR OR MOTOR_STATUS,AL; TURN ON THE CURRENT MOTOR

ED5E FB STI ; INTERRUPTS BACK ON

ED60 0C B0 OR,AL,FOC_RESET OR RESET TURN ON MOTOR

ED61 E6 F2 OUT NEC_CTL,AL WAIT FOR MOTOR BOTH READ AND WRITE

ED63 B3 14 MOV BL,20 GET MOTOR START TIME

ED65 E8 EB94 R CALL GET_PARM

ED68 0A E4 OR AH,AN; TEST FOR NO WAIT

ED6A J12: JZ J14 ; EXIT WITH TIME EXPIRED

ED6A 74 08 MOV CX,CX ; SET UP 1/8 SECOND LOOP TIME

ED6E 02 FE SUB CX,CX ; DEC A VALUE IN CX

ED70 FE CC DEC AH ; DECREMENT TIME VALUE

ED72 EB F6 JMP J12 ; ARE WE DONE YET

ED74 J14: STI ; INTERRUPTS BACK ON FOR BYPASS

ED75 59 POP CX WAIT

ED76 E8 EB9F R CALL SEEK ; DO THE SEEK OPERATION

ED79 58 POP AX ; MOVE TO CORRECT TRACK

ED7A 8A FC MOV BH,AH ; RECOVER COMMAND

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

ED7F EA EED? R JMP J17 CARRY SET JUMP TO MOTOR WAIT

ED7F EB EED? R JMP J17 CARRY SET JUMP TO MOTOR WAIT

ED86 56 PUSH SI ; SO THAT IT WILL RETURN TO MOTOR

ED87 EB EB9A R CALL NEC_OUTPUT ; OFF LOCATION

ED88 8A 66 01 MOV AH,[BP+1] ; OUTPUT THE OPERATION COMMAND

ED8B 0D E4 SAL AH,1 ; GET THE CURRENT HEAD NUMBER

ED8D 0D E4 SAL AH,1 ; MOVE IT TO BIT 2

ED91 80 E4 04 AND AH,4 ; ISOLATE THAT BIT

ED94 0A E2 OR AH,DL ; OR IN THE DRIVE NUMBER

ED96 E8 EB9A R CALL NEC_OUTPUT ; SEND OUT THE PARAMETERS TO THE CONTROLLER

ED99 8D FF 40 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 SQ, HANDLE SPECIAL

ED9A B8 E5 MOV AH,CH ; CYLINDER NUMBER

ED9E B8 E5 MOV AH,CH ; CYLINDER NUMBER

ED9A B8 E5 MOV AH,CH ; CYLINDER NUMBER

ED98 E8 EB9A R CALL NEC_OUTPUT ; HEAD NUMBER FROM STACK

ED9A 8A 66 01 MOV AH,[BP+1] ; GET THE CURRENT HEAD NUMBER

ED9B 0D E4 SAL AH,1 ; MOVE IT TO BIT 2

ED9D 0D E4 SAL AH,1 ; ISOLATE THAT BIT

ED9F 0A E2 OR AH,DL ; OR IN THE DRIVE NUMBER

ED9H E8 EB9A R CALL NEC_OUTPUT ; TEST FOR FORMAT COMMAND

ED9F 8D FF 40 CMP BH,04DH ; IS THIS A FORMAT OPERATION?

EDA2 B8 E5 MOV AH,CH ; CYLINDER NUMBER

EDA5 8A 66 01 MOV AH,[BP+1] ; HEAD NUMBER FROM STACK

EDA8 8A E1 MOV AH,CL ; SECTOR NUMBER

EDAD E8 EB9A R CALL NEC_OUTPUT ; SECTOR NUMBER

EB00 07 MOV BL,7 ; BYTES/SECTOR FROM BLOCK

EB02 E8 EB94 R CALL GET_PARM ; TO THE NEC

EB05 B3 0B MOV BL,B Bulk

EB07 EB EB94 R CALL GET_PARM ; TO THE NEC

EB09 02 4E 01 ADD CL,[BP+1]; ADD CURRENT SECTOR TO NUMBER IN TRANSFER

EB0D FE C9 DEC CL ; CURRENT_SECTOR + N_SECTORS - 1

EB10 BF EA 0E MOV AH,CL ; EOT PARAMETER IS THE CALCULATED

EB14 E8 EB9A R CALL NEC_OUTPUT ; ONE

EB16 E8 EB9A R CALL NEC_OUTPUT ; ONE

EB18 E8 EB9A R CALL NEC_OUTPUT ; ONE

EB8C 8B 000D MOV BX,BX ; DTL PARM FROM BLOCK

EBCC 53 PUSH BX ; SAVE INDEX TO DISK PARAMETER ON
EDCD FC J16: CLD ; FORWARD DIRECTION
EDCE 80 70 ;------- START TIMER1 WITH INITIAL
MOV AL, 0110000B ; VALUE OF FFFF
SELECT TIMER1, LSB-MSB, MODE 0,
; BINARY COUNTER
EDD0 E6 43 OUT TIM_CTL, AL ; INITIALIZE THE COUNTER
EDD2 B0 00 PUSH AX
EDD3 8B POP AX ; ALLOW ENOUGH TIME FOR THE 8253 TO
; INITIALIZE ITSELF
EDD4 B0 FF MOV AL, OFFH ; INITIAL COUNT VALUE FOR THE 8253
EDD6 E6 41 OUT TIMER1, AL ; OUTPUT LEAST SIGNIFICANT BYTE
EDD8 B0 50 PUSH AX
EDD9 8B POP AX ; WAIT
EDDA E6 41 OUT TIMER1, AL ; OUTPUT MOST SIGNIFICANT BYTE
EDDC 8A 46 0F MOV AL, 1BP+163 ; RETRIEVE COMMAND PARAMETER
EDDF A0 01 TEST AL, 01H ; IS THIS AN ODD NUMBERED FUNCTION?
EED1 74 05 JZ J16_1 ; JUMP IF NOT ODD NUMBERED
EED3 B9 EEE R MOV CX, OFFSET WRITE_LOOP
EED6 EB 0C JMP SHORT J16_3
EED8 3C 02 CMP AL, 2 ; IS THIS A READ?
EED9 7E 05 JNZ J16_2 ; JUMP IF VERIFY
EEDC B9 EE3 A MOV CX, OFFSET READ_LOOP
EEDF EB 03 JMP SHORT J16_3
EF01 B9 EE20 R J16_2: MOV CX, OFFSET VERIFY LOOP
;------- FINISH INITIALIZATION
EF04

;---------- INTERRUPTS

;----------  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.

EF04 B0 10 MOV AL, IOH ; DISABLE NMI
EF05 EB 60 OUT NLI_PORT, AL ; NO KEYBOARD INTERRUPT
EF0B EB EBD1 R CALL CLOCK_WAIT ; WAIT IF TIMERO IS ABOUT TO
; ENABLE WATCHDOG TIMER

;----------

;---------- GIVEN 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 B259 INTERRUPT 6 LINE IS ENABLED.
;---------- THIS IS BECAUSE OF A GLITCH ON THE LINE LARGE ENOUGH TO
;---------- TRIGGER AN INTERRUPT.

EF0B EB EB45 R CALL GET_DRIVE ; GET BIT MASK FOR DRIVE
EF0F BA 00F2 MOV DX, NEC_CTL ; CONTROL PORT TO NEC
EF01 0C E0 OR AL, FDC_RESET+WD_ENABLE+WD_STROBE
EF03 EE OUT DX, AL ; OUTPUT CONTROL INFO FOR
; WATCHDOG(WD) ENABLE/STROBE
EF04 24 A7 AND AL, FDC_RESET+WD_ENABLE+WD_STROBE
EF06 EE OUT DX, AL ; OUTPUT CONTROL INFO TO STROBE
EF07 BA 00F4 MOV DX, NEC_STAT ; PORT TO NEC STATUS
EF0A B0 20 MOV AL, 2OH ; SELECT TIMERO INPUT FROM TIMERO
EF0C E6 A0 OUT NLI_PORT, AL ; READ TIMERO NOW AND SAVE THE INITIAL VALUE
EF0E EB EB1A R CALL READ_TIME ; CALL READ_TIME
EF11 B9 46 1D MOV IBP=181, AX ; SAVE INITIAL VALUE FOR CLOCK
EF14 EB EAFC R CALL DISABLE ; DISABLE ALL INTERRUPTS
EF17 0B POP DX ; GET PARAMETER FROM STACK
EF18 EB EB94 R CALL GET_PARAMETER ; OUTPUT LAST PARAMETER TO THE NEC
EF1B 8B POP AX ; CAN NOW DISCARD THAT DUMMY RETURN
EF1C 06 PUSH ES
EF1D 1F POP DS ; INITIALIZE DS FOR WRITE
EF1E 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 LOOP

EE20
EE20 EC IN AL, DX ; READ STATUS
EE21 A8 20 TEST AL, BUSY_BIT ; HAS NEC ENTERED EXECUTION PHASE
; YET?
EE23 74 F8 JZ VERIFY_LOOP ; NO, CONTINUE SAMPLING
EE25
EE25
EE27 B8 00 TEST AL, ROM ; IS DATA READY?
EE27 75 07 JNZ J22_4 ; JUMP IF DATA TRANSFER IS READY
EE29 BC IN AL, DX ; READ STATUS PORT
EE2A A8 20 TEST AL, BUSY_BIT ; ARE WE DONE?
EE2C 75 F7 JNZ J22_2 ; JUMP IF MORE TRANSFERS
EE2E EB 35 JMP SHORT OP_END ; TRANSFER DONE
EE30 42 INC DX ;がない
EE31 4A DEC DX ; POINT AT NEC DATA REGISTER
EE33 42 IN AL, DX ; READ DATA
EE34 7E A8 20 TEST AL, BUSY_BIT ; ARE WE DONE?
EE36 7E 8B JMP J22_2 ; CONTINUE
EE38 EB 28 JMP SHORT OP_END ; WE ARE DONE

ROM BIOS A-77
Appendix A
;-----READ OPERATION
READ_LOOP:

EE3A EC IN AL,DX ; READ STATUS REGISTER
EE3B AB 20 ; TEST AL,BUSY_BIT ; HAS NEC STARTED THE EXECUTION
EE3D 74 FB JZ READ_LOOP ; HAS NOT STARTED YET
EE3F EC IN AL,DX ; READ STATUS PORT
EE40 AB 20 ; TEST AL,BUSY_BIT ; HAS NEC COMPLETED EXECUTION
EE42 74 B1 JZ OP_END ; JUMP IF EXECUTION PHASE IS OVER
EE44 AB B0 TEST AL,ROM ; IS DATA READY?
EE46 74 F7 INC DX ; POINT AT NEC_DATA
EE48 42 IN AL,DX ; READ DATA
EE49 EC STOSB ; TRANSFER DATA
EE4A 4A DEC DX ; POINT AT NEC_STATUS
EE4C EB F1 JMP J22_5 ; CONTINUE WITH READ OPERATION

;-----WRITE AND FORMAT OPERATION
WRITE_LOOP:

EE4E EC IN AL,DX ; READ NEC STATUS PORT
EE4F AB 20 ; TEST AL,BUSY_BIT ; HAS THE NEC ENTERED EXECUTION
EE51 74 FB JZ WRITE_LOOP ; NO, CONTINUE LOOPING
EE53 B9 20B0 MOV CX,BUSY_BIT+256*ROM
EE56 J22_7: IN AL,DX ; READ STATUS PORT
EE57 B4 C5 TEST AL,CH ; IS THE FEC STILL IN THE EXECUTION
EE59 74 0A JZ OP_END ; JUMP IF EXECUTION PHASE IS DONE
EE5B B4 C1 TEST AL,CL ; IS THE DATA PORT READY FOR THE
EE5D 74 F7 JZ J22_7 ; TRANSFER?
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

EE65 9C OP_END: PUSHF ; SAVE THE CARRY BIT SET IN
EE66 EB EB45 R CALL GET_DRIVE ; GET BIT MASK FOR DRIVE SELECTION
EE69 BC 00 OR AL,FDC_RESET ; NO RESET, KEEP DRIVE SPINNING
EE6B BA 00F2 MOV DX,NEC_CTL
EE6E EE OUT DX,AL ; DISABLE WATCHDOG
EE72 EB EB39 R CALL CLOCK_WAIT ; WALT IF TIMERO IS CLOSE TO
EE75 EB EB1A R CALL READ_TIME ; WRAPPING
EE78 BB 5E 12 MOV BX,[BP+1B] ; GET THE INITIAL VALUE OF TIMER1
EE7B 2B C3 SUB AX,BX ; UPDATE NUMBER OF INTERRUPTS
EE7D F7 D8 NEG AX ; MISSED
EE7F 50 PUSH AX ; SAVE IT FOR REUSE IN ISSUING
EE80 01 06 006C R ADD TIMER_LOW,AX ; TIMER INTERRUPTS
EE84 73 04 JNC J16_4 ; ADD NUMBER OF TIMER INTERRUPTS TO
EE86 FF 06 006E R INC TIMER_HI ; TIME
EE8A B3 3E 006E R 1B J16_4: CMP TIMER_HI,01Bh ; TEST FOR COUNTING 24 HOURS
EE8B 75 19 JNZ J16_5 ; JUMP IF NOT 24 HOURS
EE8F B1 3E 006C R 0080 CMP TIMER_LO,0080h ; CMP LOW_VALUE = 24 HOUR VALUE?
EE93 7C 11 JL J16_5 ; NOT 24 HOUR VALUE?
EE99 C7 06 006E R 0000 MOV TIMER_HI,0 ; ZERO OUT TIMER_HI VALUE
EE9F 81 2E 006C R 0080 SUB TIMER_LOW,0080h ; VALUE REFLECTS CORRECT TICKS PAST
EEA5 C6 00 0070 R 01 MOV TIMER_OFL,1 ; 0080h
EEA6 EB 80B0 R J16_5: CALL ENABLE ; INDICATES 24 HOUR THRESHOLD
EEA9 5D POP CX ; ENABLE ALL INTERRUPTS
EEAA E3 26 CALL IC ENABLE ; TIME INTERRUPTS
EEAD EB 50 PUSH DS ; IF ZER0 DO NOT ISSUE ANY
EEAF 1E PUSH AX ; INTERRUPTS
EEB3 52 PUSH DX ; THIS PROVIDES A COMPATIBLE
EEB5 52 INT 1CH ; INTERFACE TO IC
EEB7 5A LOOP J16_6 ; TRANSFER CONTROL TO USER
EEB8 5B POP AX ; ALL USER TIMER INTERRUPTS
EEB9 1F POP DS ; RESTORE REGISTERS
EEB3 CD 1C J16_6: MOV CX,08H ; CLOCK IS UPDATED AND USER INTERRUPTS IC HAVE BEEN ISSUED.
EEB4 00 E0 CALL KB_NOISE ; CHECK IF KEYSTROKE OCCURRED
EEB7 74 1B OR AL,AL ; AL WAS DURING CALL TO ENABLE
EEBC J16_7 ; NO KEY WAS PRESSED WHILE SYSTEM
EEBD 00 080 MOV BX,0080h ; WAS MASKED
EEC1 B9 004B MOV CX,04Bh ; DURATION OF TONE
EEC4 EB E039 R MOV BX,00BH ; NOTIFY USER OF MISSED KEYBOARD
EEC9 00 CALL KB_NOISE ; INPUT

A-78 ROM BIOS
A-80 ROM BIOS
--- INT 17 ----------------------------------------------------------

/ **PRINTER IO** /
/ **THIS ROUTINE PROVIDES COMMUNICATION WITH THE PRINTER** /
/ **(AH)=0** PRINT THE CHARACTER IN [AL] /
/ **(AH)=1** INITIALIZE THE PRINTER PORT /
/ **(AH)=2** READ THE PRINTER STATUS INTO [AH] /
/ **(AH)=3** TEST FOR [AH]=0, 1, 2 CORRESPONDING TO ACTUAL VALUES IN PRINTER_BASE AREA /
/ **DATA AREA** PRINTER_BASE CONTAINS THE BASE ADDRESS OF THE PRINTER CARD(S) AVAILABLE (LOCATED AT BEGINNING OF DATA SEGMENT, 408H, ABSOLUTE, 3 WORDS), UNLESS THERE IS ONLY A SERIAL PRINTER ATTACHED, IN WHICH CASE THE WORD AT 40:9 WILL CONTAIN A 02FH. / **REGISTERS** AH IS MODIFIED /
/ **ALL OTHERS UNCHANGED** ----------------------------------------------------------

```
ASSUME CS:CODE, DS:DATA

EFO2 ORG EFO2H
EFO2 PROC FAR

EFO2 FB STI , INTERRUPTS BACK ON
EFO2 EB PUSH DS , SAVE SEGMENT
EFO2 EC PUSH DX
EFO2 ED PUSH SI
EFO2 EE PUSH CX
EFO2 EF PUSH BX
EFO2 F0 CALL DDS

EFO8 ES 138H R

REDIRECT TO SERIAL ONLY IF:
1) SERIAL PRINTER IS ATTACHED, AND...
2) WORD AT PRINT BASE = 02FH
POWER ONS WILL ONLY PUT A 02FH IN THE PRINTER BASE IF THERE'S NO PARALLEL PRINTER ATTACHED.

EFO8 BB 00 0010 R MOV CX, EQUIP_FLAG ; GET FLAG IN CX
EFO8 FF F6 CS 20 TEST CH, 00010000B ; SERIAL ATTACHED?
EFOF 74 00 JB Z0 , NO HANDLE NORMALLY
EFOE BB 0E 0008 R MOV BX, PRINTER_BASE ; SEE IF THERE'S AN RS232
EFOF BI 01 FF 02FB CMP BX, 02FH ; BASE IN THE PRINTER BASE.
EFOG 75 03 JNE Z0
EFOH E9 18C3 R B00 JMP Z1 ; IF THERE IS REDIRECT

CONTROL IS PASSED TO THIS POINT IF THERE IS A PARALLEL OR THERE'S NO SERIAL PRINTER ATTACHED.
B0 MOV SI, DX ; GET PRINTER PARAM
EFOI 8A 9C 007B R MOV BL, PRINT_TIME_OUT ; LOAD TIMEOUT VALUE
EFOJ DI 06 SHL SI, 1 ; WORD OFFSET INTO TABLE
EFOK BB 94 0008 R MOV DX, PRINTER_BASE[] ; GET BASE ADDRESS FOR PRINTER CARD
EFL0 0B D2 OR DX, DX ; TEST DX FOR ZERO, INDICATING NO PRINTER
EFL1 74 0C JZ Z1 ; IF NO PARALLEL, RETURN
EFL2 0A E4 OR AH, AH ; TEST FOR [AH]=0
EFL3 74 0E JB Z2 , PRINT_AL
EFL4 FE CC DEC AH ; TEST FOR [AH]=1
EFL5 47 40 JZ B0 ; INIT_PTR
EFL6 FE CC DEC AH ; TEST FOR [AH]=2
EFL7 74 2B JZ B5 ; PRINTER STATUS
EFL8 F0 8B POP BX
EFL9 59 POP CX
EFLA 9E POP SI ; RECOVER REGISTERS
EFLB 5A POP DX ; RECOVER REGISTERS
EFLC 1F POP DS
EFLD CF IRET

-------- PRINT THE CHARACTER IN [AL] --------
F015 50 B2 PUSH AX ; SAVE VALUE TO PRINT
F016 EE OUT DX, AL ; OUTPUT CHAR TO PORT
F017 42 INC DX ; POINT TO STATUS PORT

-------- WAIT BUSY --------
F018 2B C9 B3: SUB CX, CX ; INNER LOOP (64K)
F019 BB 0E IN AL, DX ; GET STATUS
F01A BB 00 MOV AH, AL ; STATUS TO AH ALSO
F01B A8 70 TEST AL, 00H ; IS THE PRINTER CURRENTLY BUSY
F01C 75 0E JNZ OUT_STROBE
F01D 0F E2 F7 LOOP B3, 1 ; LOOP IF NOT
F01E 02 FE DEC BL ; DROP OUTER LOOP COUNT
F01F 22 7F JNZ B3 ; MAKE ANOTHER PASS IF NOT ZERO
F020 80 CC 01 OR AH, 1 ; SET ERROR FLAG
F021 80 E4 9F AND AH, 0FH ; TURN OFF THE UNSED BITS
F022 0E 14 JMP SHORT B7 ; RETURN WITH ERROR FLAG SET
F023 F0 B4: OUT_STROBE
F024 80 00 MOV AL, 0DH ; SET THE STROBE HIGH
F025 42 INC DX
F026 EE OUT DX, AL
F027 80 0C MOV AL, 0CH
F028 33 EE OUT DX, AL ; SET THE STROBE LOW
F029 5B POP AX ; RECOVER THE OUTPUT CHAR
```

---C-APPENDIX A---

ROM BIOS A-81
;--------- PRINTER STATUS
F035 50 B5: PUSH AX ; SAVE AL REG
F036 5B 04 0008 B R B6: MOV DX, PRINTER_BASE + 01
F03A 04 42 INC DX
F03B EC IN AL,DX ; GET PRINTER STATUS
F03C 8A E0 MOV AH, AL
F03E E0 84 F8 AND AH, OFH
F041 47 B7: ; ORG OFOA4H ; STATUS_SET
F04A 45 5A POP DX ; RECOVER AL REG
F042 8A C2 MOV AL, DL ; GET CHARACTER INTO AL
F044 80 F4 4B XOR AL, AH ; CLEAR DUPLICATE BITS
F047 E4 CB JMP B1 ; RETURN FROM INITIALIZE ROUTINE
;--------- INITIALIZE THE PRINTER PORT
F049 50 BB: PUSH AX ; SAVE AL
F04A 42 INC DX ; POINT TO OUTPUT PORT
F04B 42 INC DX
F04C 80 08 MOV AL, B ; SET INITIAL Baud Rate
F04E EE OUT DX, AL
F04F BB 03 EE MOV AX, 0000H
F052 89 ; INIT_LOOP
F052 4B DEC AX ; LOOP FOR RESET TO TAKE EFFECT
F053 75 FD JNZ B9 ; INIT_LOOP
F055 B0 0C MOV AL, 0CH ; NO INTERRUPTS, NON AUTO LF, INIT
F057 EE OUT DX, AL
F05B 8B 0C JMP BB ; PRT_STATUS_1
F05A PRINTER_IO ENDP
F065 ORG OF0E8H
F06E BB 09 DD R JMP NEAR PTR VIDEO_IO ; SUBROUTINE TO SAVE ANY SCAN CODE RECEIVED ; BY THE NMI ROUTINE (PASSED IN AL) ; DURING POST IN THE KEYBOARD BUFFER ; CALLED THROUGH INT 48H
;
F06B KEY_SCAN_SAVE PROC FAR
ASSUME DS:DATA
F06B EB 13 BB R CALL DDS ; POINT DS TO DATA AREA
F06B BB 0E 04 MOV SI, OFFSET KB_BUFFER ; POINT TO FIRST LOC. IN BUFFER
F06E BB 04 MOV [SI], AL ; SAVE SCAN CODE
F070 BB C4 MOV AX, SP ; CHECK FOR STACK UNDERFLOW
F072 80 E4 ED AND AH, 11100000B (THESE BITS WILL BE 111 IF UNDERFLOW HAPPENED)
F075 74 0D JZ KS_1
F077 32 CD XOR AL, AL
F079 E6 AD OUT 0AOH, AL ; SHUT OFF NMI
F07B BB 2000 MOV BX, 2000H ; ERROR CODE 200H
F07D BB 0E 03 MOV SI, OFFSET KEY_ERR ; POST MESSAGE
F07E BE 0036 R MOV SI, OFFSET KB_BUFFER ; POINT TO FIRST LOC. IN BUFFER
F081 EB 09 BC R CALL E_MSG ; AND HALT SYSTEM
F084 CF JMP KS_1: ; RETURN TO CALLER
F085 KEY_SCAN_SAVE ENDP
; KEY_SCAN_SAVE PROC
;
; 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
; ODD PARITY.
; EXPECTS TO BE PASSED
; (DX) = LINE CONTROL REGISTER
; (DI) = TRANSMIT/RECEIVE BUFFER ADDRESS
; ALSO, ALTERS REGISTER AL. ALL OTHERS REMAIN INTACT.
;
F085 SB250 PROC NEAR
F085 F8 B0 0D MOV AL, BX ; SET DBL = 1
F087 EE OUT DX, AL
F08B EB 00 JMP *+2 ; 1/0 DELAY
F08A BB 02 03 MOV DX, 3 ; LSB OF DIVISOR LATCH
F08B BB 00 OC MOV AL, 12 ; DIVISOR = 12 PRODUCES 9600 BPS
F08F EE OUT DX, AL ; SET LSB
F090 E9 00 OB JMP *+2 ; 1/0 DELAY
F092 42 INC DX ; MSB OF DIVISOR LATCH
F093 B0 00 MOV AL, 0 ; HIGH ORDER OF DIVISORS
F095 EE OUT DX, AL ; SET MSB
F096 EB 00 JMP *+2 ; 1/0 DELAY
F098 42 INC DX
F099 42 INC DX ; LINE CONTROL REGISTER
F09A B0 0F MOV AL, 0000111B ; 8 BITS/WORD, 2 STOP BITS, ODD
; PARITY
F09C EE OUT DX, AL
F09D EB 00 JMP *+2 ; 1/0 DELAY
F09F 8A 02 03 SUB DX, 3 ; RECEIVER BUFFER
F0A2 EE 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 ORG 0F0A4H
F0A4 VIDEO_PARMS LABEL BYTE
; INIT_TABLE
F0A4 38 2B 2C 06 1F 06 DB 3BH, 2BH, 2CH, 06H, 1FH, 6, 19H ; SETUP FOR 40X25
F0A4 1C 02 07 06 07 DB 1CH, 2, 7, 6, 7
F0A0 00 00 00 00 DB 0, 0, 0, 0

A-82 ROM BIOS
```
F0E4 08 0C 04
F0E7 72 03
F0E9 E9 F531 R
F0EC
F0E5 EB FO77 R
F0EF 88 FF
F0F0 01 0E
F0F6 00 00
F0F7 84 F0
D3 DB
F0F9 3D 06
F0FB 88 F1
F0FF 84 0050 R
F103 3D DB
F105 E3 06
F107
F10B 03 0E
F10D 88 E52 R
F110 3D DB
F112 C3
F113
F115 80 FC 04
F116 72 03
F118 E9 F3F1 R
F11B
F11E 8A E3
F11F EB FO77 R
F122 88 FB
F124 59
F125 5B
F126 84
F127 E2 F0
F129 E9 0F70 R
F12C

F0E4 DB 71H,50H,5AH,0CH,1FH,6,19H ; SETUP FOR 80X25
F0E7 DB 1CH,2,7,6,7
F0E9 DB 0,0,0,0
F0EC DB 3BH,28H,28H,06H,7FH,6,64H ; SET UP FOR GRAPHICS
F0F0 DB 70H,2,1,26H,7
F0F4 DB 3BH,2,3,26H,7 ; USING 32K OF MEMORY
F0F8 DB 0,0,0,0

; 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) = RESEG SEGMENT
; OUTPUT
; (AL) = CHAR ATTR
; (AH) = ATTRIBUTE READ

; ALIAS
ASSUME CS:CODE,DS:DATA,ES:DATA
READ_AC_CURRENT PROC NEAR
  CMP AH, 4 ; IS THIS GRAPHICS?
  JC C60
  JMP GRAPHICS_READ
  ; READ_AC_CONTINUE
READ_AC_CURRENT ENDP

; FIND_POSITION
FIND_POSITION PROC NEAR
  MOV CL,BH ; DISPLAY PAGE TO CK
  XOR CH,CH
  MOV SI,BX ; ESTABLISH ADDRESSING IN SI
  PUSH ES
  POP DS
  GET SEGMENT FOR QUICK ACCESS
  LODSW ; GET THE CHAR/ATTR
  JMP VIDEO_RETURN
FIND_POSITION ENDP

; READ_AC_CURRENT
; THIS ROUTINE READS 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) = RESEG SEGMENT
; OUTPUT
; NONE
WRITE_AC_CURRENT PROC NEAR
  CMP AH, 4 ; IS THIS GRAPHICS?
  JC C63
  JMP GRAPHICS_WRITE
  ; WRITE_AC_CONTINUE
WRITE_AC_CURRENT ENDP
```

---

ROM BIOS A-83
READ_DOT PROC NEAR

CMP CRT_MODE, OAH ; 640X200 4 COLOR?
JE READ_ODD ; YES, HANDLE SEPARATELY

CALL C72 ; DETERMINE POSITION OF DOT
MOV AL, ES:[SI] ; GET THE BYTE
AND AL, AH ; MASK OFF THE OTHER BITS IN THE BYTE
SML AL, CL ; LEFT JUSTIFY THE VALUE
MOV CL, DH ; GET NUMBER OF BITS IN RESULT
ROL AL, CL ; RIGHT JUSTIFY THE RESULT
JMP VIDEO_RETURN ; RETURN FROM VIDEO 10

READ_ODD:

CALL C72 ; DETERMINE POSITION OF DOT
MOV AL, ES:[SI+1] ; GET CI COLOR BIT FROM ODD BYTE
AND AL, AH ; MASK OFF OTHER BITS
SML AL, CL ; LEFT JUSTIFY THE VALUE
MOV CL, DH ; GET NUMBER OF BITS IN RESULT
INC CL
ROL AL, CL ; RIGHT JUSTIFY THE RESULT
MOV BX, AX ; SAVE IN BX REG
POP AX ; RESTORE POSITION INFO
POP CX
POP DX

MOV AL, ES:[SI] ; GET CO COLOR BIT FROM EVEN BYTE
AND AL, AH ; MASK OFF OTHER BITS
SML AL, CL ; LEFT JUSTIFY THE VALUE
MOV CL, DH ; GET NUMBER OF BITS IN RESULT
ROL AL, CL ; RIGHT JUSTIFY THE RESULT
OR AL, BL ; COMBINE CI & CO
JMP VIDEO_RETURN

READ_DOT ENDP

WRITE_C_CURRENT PROC NEAR

CMP AH, 4 ; IS THIS GRAPHICS?
JC C65
PUSH AX ; SAVE ON STACK
PUSH CX ; SAVE WRITE COUNT
CALL FIND_POSITION
MOV DI, BX ; ADDRESS TO DI
POP CX ; WRITE COUNT
POP BX ; BL HAS CHAR TO WRITE
WRITE_LOOP:
MOV AL, BL ; RECOVER CHAR
INC DI ; BUMP POINTER PAST ATTRIBUTE
CALL GRAPH
JMP VIDEO_RETURN

WRITE_C_CURRENT ENDP

SAVE_BOTH CMP 7-4
F134 50
F135 51
F136 E8 F077 R
F137 BB FB
F138 59
F139 5B
F13A 07
F13B 07
F13C 07
F13D 07
F13E 07
F13F 07
F140 07
F141 E2 FA
F142 07
F143 E9 0F70 R
F144 07
F145 07
F146 07
F147 07
F148 E9 0F71 R
F149 07
F14A 07
F14B 07
F14C 07
F14D 07
F14E 07
F14F 07
F150 07
F151 07
F152 07
F153 07
F154 E9 0F72 R
F155 07
F156 E9 0F73 R
F157 07
F158 07
F159 07
F15A 07
F15B 07
F15C 07
F15D 07
F15E E9 0F74 R
F15F 07
F160 07
F161 07
F162 07
F163 07
F164 26 : BA 44 01
F165 26 : BA 04 04
F166 22 : C4
F167 E2 00
F168 01
F169 02
F16A 07
F16B 07
F16C 07
F16D 07
F16E 07
F16F 07
F170 07
F171 07
F172 07
F173 07
F174 07
F175 07
F176 07
F177 26 : BA 04
F178 22 : C4
F179 E2 00
F17A 07
F17B 07
F17C 07
F17D 07
F17E 07
F17F 07
F180 07
F181 07
F182 07
F183 07
F184 07

A-84 ROM BIOS
F 187
F 187
F 187
F 188
F 189
F 18A
Fl8S

51
52

50
50
E8 FlOg R

READ DOT
WRITE_DOT
PUSH
PUSH
PUSH
PUSH
CALL

ENDP
PROC

NEAR

CX

OX
AX
AX

C72

SAVE COL
SAVE ROW
SAVE DOT VALUE
TWICE
DETERMINE BYTE POSITION OF THE
DOT

F18E

02 Ea

SHR

AL,CL

Fl90

22 C4

AND

F 192
FI9S

26, 8A DC
58
F6 C3 80
75 36
F6 04

HOV
POP

Al, AH
CL, ES: [SI 1
BX
Bl, eOH
C70

NOT

AH

22 CC

AND
OR

CL, AH
AL, CL

26: 88 04
58
SA
59
80 3E 0049 R OA

HOV
POP
POP
POP
CMP

ES: [SI1,Al

75 20

JNE
PUSH
PUSH

C6'

50
DO E8

SHR

E8 F109 R

CALL

AL,l
C72

F 185

02 E8

SHR

AL, CL

F 187
F 189
F 180
F IBE
FIC!
F lC3

22 C4
26; SA 4C O!
58
F6 C3 80
75 !2
F6 04

AND

AL, AH
Cl, E5; [SI+U
BX
eL, BOH
C71

F lC5
F IC7
F lC9

22 CC
OA Cl

FI96
Fl99
Fl9B

F19C
F19F
FIAI
FIA1
F lA4
F lA5
F IA6

F lA7
FIAC
F IAE
F IAF
F 180
Fl82

TEST
JNZ

OA C1
C67:

50

SHIFT TO SET UP THE BITS FOR
OUTPUT
STRIP OFF THE OTHER BITS
GET THE CURRENT BYTE
RECOVER XOR FLAG
IS IT ON
YES, XOR THE DOT
SET THE MASK TO REMOVE THE
INDICATED BITS
OR I N THE NEW VALUE OF THOSE BITS
FINISH DOT
RESTORE THE BYTE I N MEMORY

AX

OX
CX

CRT_MODE,OAH
AX
AX

RECOVER ROW
RECOVER COL
640X200 4 COLOR?
NO, JUMP
SAVE DOT VALUE
TWICE
SHIFT cl BIT INTO cO POSITION
DETERMINE BYTE POSITION OF THE
DOT

F lC9
FICO

F ICE
FI01
FlO!
FI03

F 105
FI05
FID?
F 109

HOV
POP

TEST
JNZ
NOT

AH

AND
OR

eL, AH
AL,CL

HOV
POP

ES;[SI+1J,AL

JMP

V IDEO_RETURN

XOR

AL,CL

JMP

C67

C6828:

88 44 01

58
E9 OF70 R

C69:

C71:

32 Cl
EB FO

XOR
JMP

WRITE_DOT

OR I N THE NEW VALUE OF THOSE BITS
FINISH DOT
RESTORE THE BYTE I N MEMORY

AX

C70;
32 Cl
EB CC

SHIFT TO SET UP THE BITS FOR
OUTPUT
STRIP OFF THE OTHER BITS
GET THE CURRENT BYTE
RECOVER XOR FLAG
IS IT ON
YES, XOR THE DOT
SET THE MASK TO REMOVE THE
I NO I CATED BITS

AL,CL

C68

RETURN FROM VIDEO 10
><OR DOT
EXCLUSI VE OR THE DOTS
FINISH UP THE WRITING
XOR_DOT
EXCLUSI VE OR THE DOTS
FINISH UP ·THE·WRITING

ENDP

THIS SUBROUTINE DETERMINES THE REGEN BYTE LOCATION OF THE
INDICATED ROW COLUMN VALUE IN GRAPHICS MODE.
ENTRY -ox :::: ROW VALUE (0,..1991
ex :::: COLUMN VALUE (0-6391
EXIT -S I :::: OFFSET I NTO REGEN BUFFER FOR BYTE OF 1NTEREST
AH :::: MASK TO STRIP OFF THE BITS OF INTEREST
CL :::: BITS TO SHIFT TO RIGHT .JUSTIFY THE MASK IN AH
OH :::: # BITS IN RESULT

F 109
F 109
F IDA

53
50

F loB
F 100
F IDE
F lEI
F lE6

BO
52
SO
80
72

28
E2 FE
3E 0049 R 09
03

F lE8

80 E2 Fe

F lEB

F6 E2

F lED
FlEE
FIFI
F IF3
F IF6
F IF6
F IF8
FIFO

SA
F6 C2 01
7403
05 2000

80 3E 0049 R 09
72 08
F6 C2 02
F200 74 03
F202 05 4000
F205 . 88 FO
F207
58
F208 88 01

PROC
NEAR
PUSH
8X
; SAVE BX DURING OPERATION
PUSH
AX
, WilL SAVE AL DURING OPERATION
; ------ DETERMI NE 1ST BYTE IN IOICATED ROW BY MULTIPLYING ROW VALUE
BY 40( LOW BIT OF ROW DETERMINES EVEN/ODD, 80 BYTES/ROW
MOV
AL,40
PUSH
OX
SAVE ROW VALUE
ANO
DL,OFEH
STRIP OFF ODD/EVEN BIT
CMP
CRT_MODE,09H
MODE USING 32K REGEN?
JC
C73
NO, JUMP
AND
OL,OFCH
STRI P OFF LOW 2 BI TS
C73:
MUL
OL
AX HAS ADDRESS OF 1ST BYTE OF
INOI CATED ROW
POP
OX
RECOVER 1 T
TEST
Dl, 1
TEST FOR EVEN/ODD
JZ
C74
JUMP I F EVEN ROW
ADD
AX,2000H
OFFSET TO LOCATION OF ODD ROWS
C74·
EVEN ROW
MODE-USING 32K REGEN?
CMP
CRT_MODE,09H
JC
C75
NO, JUMP
TEST
TEST FOR ROW 2 OR ROW 3
DL,2
C75
JZ
JUMP IF ROW 0 OR 1
ADO
AX,4000H
OFFSET TO LOCATION OF ROW 2 OR 3
MOV
51,
AX
MOVE
POINTER TO 51
C75:
AX
RECOVER AL VALUE
POP
MOV
COLUMN VALUE TO ox
ox, CX

ROM BIOS A-85


--- DETERMINE GRAPHICS MODE CURRENTLY IN EFFECT
SET UP THE REGISTERS ACCORDING TO THE MODE
CH = MASK FOR LOW OF COLUMN ADDRESS (73/2/1 FOR HIGH/MED/LOW RES)
CL = # OF ADDRESS BITS IN COLUMN VALUE (2/3/2 FOR H/M/L)
BL = MASK TO SELECT BITS FROM POINTED BYTE (BH/COL/FOL FOR H/M/L)
BH = NUMBER OF VALID BITS IN POINTED BYTE (1/2/4 FOR H/M/L)

F28A BB 020C
MOV BX, 200H
F28B BB 0302
MOV CX, 302H
F290 80 3E 0049 R 04
CMP CRT_MODE, 4
F295 74 21
JE C77
F297 80 3E 0049 R 05
CMP CRT_MODE, 5
F29C 74 1A
Je C77
F29E 80 04 F0
MOV BX, 0FH
F2A9 80 01 01
MOV CX, 101H
F2AC 80 3E 0049 R 0A
CMP CRT_MODE, 0AH
F2B9 74 07
Je C76
F2BB 80 3E 0049 R 06
CMP CRT_MODE, 6
F2C0 76 05
JNE C77
F2C2 BB 0180
MOV BX, 180H
F2C5 BB 0703
MOV CX, 703H
SET PARMS FOR H/M/L RES

F2C8 22 EA
----- DETERMINE BIT OFFSET FROM BYTE IN COLUMN MASK
C77, AND CH, DL
----- DETERMINE BYTE OFFSET FOR THIS LOCATION IN COLUMN
F2C9 03 EA
SHR DX, CL
----- SHIFT BY CORRECT AMOUNT
F2C3 03 F2
ADD SI, DX
----- INCREMENT THE POINTER
F2C8 80 3E 0049 R 0A
CMP CRT_MODE, 0AH
----- 640X240 = 4 COLOR?
F2CD 75 02
JNC C78
----- NO JUMP
F2CF 03 F2
ADD SI, DX
----- INCREMENT THE POINTER
F2D4 8A F7
----- GET THE # OF BITS IN RESULT TO DH
C78: MOV DH, BH
----- MULTIPLE AX BY (VALID BITS IN BYTE) BY CH
F2D9 2A C9
SUB CL, CL
----- ZERO INTO STORAGE LOCATION
F2DF 80 C8
ROR AL, 1
----- LEFT JUSTIFY THE VALUE IN AL
( FOR WRITE )
F2E0 02 CD
ADD CL, CH
----- ADD IN THE BIT OFFSET VALUE
F2E4 FF CF
DEC BH
----- LOOP CONTROL
F2E8 75 FB
JNZ C79
----- ON EXIT, CL HAS SHIFT COUNT TO
F2EB 8A E3
MOV AH, BL
----- GET MASK TO AH
F2EE 02 EC
SHR AH, CL
----- MOVE THE MASK TO CORRECT LOCATION
F2F1 58
POP BX
----- RECOVER REG
F2F3 C3
RET
----- RETURN WITH EVERYTHING SET UP
F2F5

C72 ENDP

; 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

F2F9

GRAPHICS_UP PROC NEAR
F2FA 8A DB
MOV BL, AL
----- SAVE LINE COUNT IN BL
F2FB BB C1
----- USE CHARACTER SUBROUTINE FOR POSITIONING
F2FC BB 01
----- ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE
F2FD EB F72C R
CALL GRAPH_POSN
F2FE BB 0F
MOV DI, AX
----- SAVE RESULT AS DESTINATION
F2FF 80 E6
----- DETERMINE SIZE OF WINDOW
F2FC 2B D1
SUB DX, CX
----- GET UPPER LEFT POSITION INTO AX REG
F2F4 81 C2 0101
ADD DI, 101H
----- ADJUST VALUES
F2F8 D0 E6
SAL DH, 1
----- MULTIPLY # ROWS BY 4 SINCE 8 VERT
F2FA D0 E6
SAL DH, 1
----- AND EVEN/ODD ROWS
F2FB
----- DETERMINE CRT MODE
F2FC 80 3E 0049 R 06
CMP CRT_MODE, 6
----- TEST FOR HIGH RES
F2FD 74 1D
----- MEDIUM RES UP
F2FE 30 E2
SAL DL, 1
----- # COLUMNS = 2, SINCE 2 BYTES/CHAR
F2FF D1 E7
SAL DI, 1
----- OFFSET #2 SINCE 2 BYTES/CHAR
F2E2 80 3E 0049 R 04
CMP CRT_MODE, 4
----- TEST FOR MEDIUM RES
F2E7 74 12
Je CB0
F2E8 80 3E 0049 R 05
CMP CRT_MODE, 5
----- TEST FOR MEDIUM RES
F2EB 74 0B
Je CB0
F2EC 80 3E 0049 R 0A
CMP CRT_MODE, 0AH
----- TEST FOR MEDIUM RES
F2F0 74 04
Je CB0
----- LOW RES UP
F2F1 D0 E2
SAL DL, 1
----- # COLUMNS = 2 AGAIN, SINCE 4
F2F2 D1 E7
SAL DI, 1
----- OFFSET #2 AGAIN, SINCE 4

A-86 ROM BIOS
; ------ DETERMINE THE SOURCE ADDRESS IN THE BUFFER
CB0:  ; FIND_SOURCE
F290 06  ; GET SEGMENTS BOTH POINTING TO
         ; REGEN
F291 1F  ; LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD
F292 2A E3
F294 DD E3  ; ZERO TO HIGH OF COUNT REG
F296 DD E3
F298 74 67  ; GET NUMBER OF LINES IN AL
F29A BA C3  ; BIT COUNT TO NEXT LINE
F29C BA 50
F29E FE E4
F2AE AF F7  ; SET UP SOURCE
F2A2 03 F0  ; ADD 1, AX
F2A4 8A E6
F2AE 2A E3  ; DETERMINE NUMBER TO MOVE
         ; ------ LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD

; ------ FILL IN THE VACATED LINES
CB1:  ; FIELDS
         ; ROU_LOOP
F2AB E8 F3C7 R
F2AC 8A E7  ; MOVE ONE ROW
F2AD 1F  ; MOVE THAT ROW
F2AE E8 138B R
F2AF 80 3E 0049 R 09  ; REGEN
F2B0 1F
F2B2 72 15  ; CALL CRT_MODE, 9
F2B7 81 C6 2000
F2BB 81 C7 2000
F2BF E8 F3C7 R
F2C1 81 EE 3F80  ; MOVE 2 MORE ROWS
F2C6 81 EF 3F80
F2CA FF CC  ; BACK UP POINTERS
F2CC 81 EE 1F80  ; MOVE TO NEXT ROW
F2D0 81 EF 1F80
F2D4 FE CC
F2D6 75 D0  ; NUMBER OF ROWS TO MOVE
         ; ------ SCROLL DOWN, THE INFORMATION ON THE CRT
         ; ENTRY --
F2D8 8A C7  ; CH, CL = UPPER LEFT CORNER OF REGION TO SCROLL
F2DA E8 F3E0 R  ; DH, DL = LOWER RIGHT CORNER OF REGION TO SCROLL
F2DE EB 138B R  ; BOTH OF THE ABOVE ARE IN CHARACTER POSITIONS
F2E1 80 3E 0049 R 09
F2E7 72 00  ; BH = FILL VALUE FOR BLANKED LINES
F2E8 81 C7 2000
F2ED EB F3E0 R  ; AL = LINES TO SCROLL (AL=0 MEANS BLANK THE ENTIRE FIELD)
F2F0 81 EF 3F80
F2F4 FE CB
F2F6 81 EF 1F80  ; BLANK FIELD
F2FA 75 DC  ; SET BLANK COUNT TO EVERYTHING
F301 8A DE  ; NEXT ROWS WITH BLANK FIELD
F303 EB D3  ; CLEAR THE FIELD
F305 GRAPHICS_DOWN PROC
F307 F0  ; STD
F30E 8A D8  ; MOVE BL, AL
F30F 88 C2  ; SAVE LINE COUNT IN BL
F310 88 F72C R  ; MOV AX, DX
F311 81 C2 0101  ; GET LOWER RIGHT POSITION INTO AX REG
F315 00 E6
F317 00 E6  ; DETERMINE SIZE OF WINDOW
F31B 80 3E 0049 R 06  ; SUB DX, CX
F31D 00 E6  ; ADD AX, DX
F31F 01 E6  ; JNZ CB8
F31B 80 3E 0049 R 06  ; CMP CRT_MODE, 6
F321 74 22  ; TEST FOR HIGH RES
         ; ------ USE CHARACTER SUBROUTINE FOR POSITIONING
F325 8B 1F  ; CALL GRAPH_POSN
F329 05  ; MOVE DI, AX
F32D 1F  ; SAVE RESULT AS DESTINATION
         ; ------ DETERMINE SIZE OF WINDOW
F32F 2B D1
F331 81 C2 0101
F335 00 E6
F337 00 E6
F33B 80 3E 0049 R 06
         ; ------ DETERMINE CTR MODE
F33F 8B 1F  ; CMP CRT_MODE, 6
F337 0F 74 22  ; JZ CB7
         ; FIND_SOURCE_DOWN
--- CLEAR A SINGLE ROW

C96

PROC NEAR

MOV CL, DL ; NUMBER OF BYTES IN FIELD

F3E0

F3E0 8A CA

F3E2 57

F3E3 F3/ AA

F3E5 5F

F3E6 81 C7 2000

F3E8 8A CA

F3ED F3/ AA

F3EF 5F

F3F0 C3

F3F1

----- 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 CHARACTERS TO WRITE
DS = DATA SEGMENT
ES = REGSEG 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 CHARACTERS ARE CONTAINED IN
ROM. INTERRUPT 4AH IS USED TO POINT TO THE TABLE FOR THE FIRST
128 CHARACTERS. INTERRUPT 17H IS USED TO POINT TO THE TABLE FOR THE
SECOND 128 CHARACTERS.

ASSUME CS, CODE, DS, DATA, ES, DATA

--- GRAPHICS WRITE PROC NEAR

XOR AH, AH ; ZERO TO HIGH OF CODE POINT

F3F1

F3F1 32 E4

F3F3 50

F3F4 E8 F729 R

F3F7 BB FB

F3F9 5B

F3FA BE 0100 R

F3FD 3C 80

F3FF BB 05

F401 BE 007C R

F404 2C 80

F406 1E

F407 33 D2

F409 BE DA

F40B C5 34

F40D BC DA

F40F 1F

F410 52

----- DETERMINE GRAPHICS MODE IN OPERATION

F411 D1 E0

F413 D1 E0

F415 D1 E0

F417 03 F0

F419 B0 3E 0049 R 04

F41E 74 45

F420 B0 3E 0049 R 05

F422 74 3E

F427 B0 3E 0049 R 0A

F42C 75 03

F42E EA F404 R

F431 B0 3E 0049 R 06

F436 75 53

R1: PUSH DS

XOR DX, DX

F440 8E DA

F442 C5 34

F44B C5 34

F44E AC

F451 5E

F452 5F

F453 47

F454 E2 E3

----- DETERMINE GRAPHICS MODE IN OPERATION

SAL AX, 1 ; MULTIPLY CODE POINT

SAL AX, 1

F455 5E

F456 B6 85 IFFF

F444 B3 C7 4F

F44D FE CE

F44F 75 EC

F451 5E

F452 5F

F453 47

F454 E2 E3

----- DETERMINE GRAPHICS MODE IN OPERATION

SAL AX, 1 ; VALUE BY B

ADD SI, AX ; SI HAS OFFSET OF DESIRED CODES

F45A 5E

F45B 6C 52

F45C 47

F45D 5E

F45E 6C 85

F460 47

F461 5E

F462 5F

F463 47

F464 E2 E3

----- DETERMINE GRAPHICS MODE IN OPERATION

SAL AX, 1 ; MULTIPLY CODE POINT

SAL AX, 1

F465 5E

F466 B6 85 IFFF

F444 B3 C7 4F

F44D FE CE

F44F 75 EC

F451 5E

F452 5F

F453 47

F454 E2 E3

----- DETERMINE GRAPHICS MODE IN OPERATION

SAL AX, 1 ; VALUE BY B

ADD SI, AX ; SI HAS OFFSET OF DESIRED CODES

F46A 5E

F46B 6C 52

F46C 47

F46D 5E

F46E 6C 85

F470 47

F471 5E

F472 5F

F473 47

F474 E2 E3

----- DETERMINE GRAPHICS MODE IN OPERATION

SAL AX, 1 ; MULTIPLY CODE POINT

SAL AX, 1

F475 5E

F476 B6 85 IFFF

F444 B3 C7 4F

F44D FE CE

F44F 75 EC

F451 5E

F452 5F

F453 47

F454 E2 E3

----- DETERMINE GRAPHICS MODE IN OPERATION

SAL AX, 1 ; VALUE BY B

ADD SI, AX ; SI HAS OFFSET OF DESIRED CODES

F47A 5E

F47B 6C 52

F47C 47

F47D 5E

F47E 6C 85

F480 47

F481 5E

F482 5F

F483 47

F484 E2 E3

----- DETERMINE GRAPHICS MODE IN OPERATION

SAL AX, 1 ; MULTIPLY CODE POINT

SAL AX, 1

F485 5E

F486 B6 85 IFFF

F444 B3 C7 4F

F44D FE CE

F44F 75 EC

F451 5E

F452 5F

F453 47

F454 E2 E3

----- DETERMINE GRAPHICS MODE IN OPERATION

SAL AX, 1 ; VALUE BY B

ADD SI, AX ; SI HAS OFFSET OF DESIRED CODES

F48A 5E

F48B 6C 52

F48C 47

F48D 5E

F48E 6C 85

F490 47

F491 5E

F492 5F

F493 47

F494 E2 E3

----- DETERMINE GRAPHICS MODE IN OPERATION

SAL AX, 1 ; MULTIPLY CODE POINT

SAL AX, 1

F495 5E

F496 B6 85 IFFF

F444 B3 C7 4F

F44D FE CE

F44F 75 EC

F451 5E

F452 5F

F453 47

F454 E2 E3

----- DETERMINE GRAPHICS MODE IN OPERATION

SAL AX, 1 ; VALUE BY B

ADD SI, AX ; SI HAS OFFSET OF DESIRED CODES

F49A 5E

F49B 6C 52

F49C 47

F49D 5E

F49E 6C 85

F4A0 47

F4A1 5E

F4A2 5F

F4A3 47

F4A4 E2 E3

----- DETERMINE GRAPHICS MODE IN OPERATION

SAL AX, 1 ; MULTIPLY CODE POINT

SAL AX, 1

F4A5 5E

F4A6 B6 85 IFFF

F444 B3 C7 4F

F44D FE CE

F44F 75 EC

F451 5E

F452 5F

F453 47

F454 E2 E3

----- DETERMINE GRAPHICS MODE IN OPERATION

SAL AX, 1 ; VALUE BY B

ADD SI, AX ; SI HAS OFFSET OF DESIRED CODES

F4A9 5E

F4AA 6C 52

F4AB 47

F4AC 5E

F4AD 5F

F4AE 47

F4AF E2 E3

----- DETERMINE GRAPHICS MODE IN OPERATION

SAL AX, 1 ; MULTIPLY CODE POINT

SAL AX, 1

F4B0 5E

F4B1 B6 85 IFFF

F444 B3 C7 4F

F44D FE CE

F44F 75 EC

F451 5E

F452 5F

F453 47

F454 E2 E3

----- DETERMINE GRAPHICS MODE IN OPERATION

SAL AX, 1 ; VALUE BY B

ADD SI, AX ; SI HAS OFFSET OF DESIRED CODES

F4B4 5E

F4B5 6C 52

F4B6 47

F4B7 5E

F4B8 5F

F4B9 47

F4BA E2 E3
F518
F518  AC
F518  8A 00
F518  23  C3
F510  F6  C2  80
F520  74  03
F522  26: 32 25
F525  26: 32 45 01
F528  26  BB 25
F52C  26: 88 45 01
F530  C3
F531  R21
F531  END
F531  R21
GRAPHICS_WRITE  ENDP

F531  F532: EB 7 F29  R
F534: BB  F0
F536: B3  EC  08
F539: BB  EC
F53B: 06
F53C: B6  04
F543: 17
F545: 80 3E  0049  R  06
F547: 84  61
F551: 74  5A
F553: 80 3E  0049  R  05
F555: 74  53
F558: 74  53
F55A: EB  18
F55C  IF
F55D: 80  04
F562: 45
F563: 80  A4  2000
F567: 88  46  00
F56A: 45
F56B: B3  C6  50
F56F: FE  CE
F570: 75  EB
F572: EB  6E
F574: IF
F575: D1  E6
F577: D1  E6
F579: EB  F6FC  R
F57C: B1  C6  2000
F580: EB  F6FC  R
F583: 1E
F584: EB  1388  R
F587: 80 3E  0049  R  09
F58C: IF
F58D: 75  14
F590: B1  C6  2000
F593: EB  F6FC  R
F596: B1  C6  2000
F599: EB  F6FC  R
F59D: 01  EE  3FBO
F5A1: FE  CE
F5A3: B1  EE  1FBO
F5A7: FE  CE
F5A9: 75  CE
F5AE: D1  E6
F5B0: EB  F6FC  R
F5B3: B1  C6  2000
F5B7: EB  F63C  R
F5BA: IF
F5BB: EB  1388  R
F5BE: 80 3E  0049  R  0A
F5C3: IF
F5C4: 75  14
F5CE: B1  C6  2000
F5CA: EB  F63C  R
F5CD: B1  C6  2000
F5D1: EB  F63C  R
F5DB: FE  CE
F5DD: B1  EE  1FBO
F5DE: FE  CE
F5E0: 75  CE

R21
PROC
NEAR

GET_CODE_POINT

MOV
AH, AL
COPY INTO AH

AND
AX, BX
SET COLOR

TEST
DL, 80H
XOR FUNCTION?

JZ
R22
NO_JUMP

XOR
AH, ES:[DI]
EXCLUSIVE OR WITH CURRENT DATA

MOV
ES:[DI], AH
STORE IN REGEN BUFFER

MOV
ES:[DI+1], AL

R22

RET

GRAPHICS_READ

PROC
NEAR

;; CONVERTED TO OFFSET IN REGEN

CALL
R59

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

JZ
R23
HIGH_RESOLUTION

CMP
CRT_MODE, 4

JZ
R28
MEDIUM_RESOLUTION

CMP
CRT_MODE, 5

JZ
R29
MEDIUM_RESOLUTION

CMP
CRT_MODE_OAH

JZ
R28
MEDIUM_RESOLUTION

JMP
SHORT R25
LOW_RESOLUTION

-----

HIGH_RESOLUTION_READ

GET VALUES FROM REGEN BUFFER AND CONVERT TO CODE POINT

POP
DS
POINT TO REGEN SEGMENT

POP
AL,[SI]
GET FIRST BYTE

MOV
[BP], AL
SAVE IN STORAGE AREA

INC
BP
NEXT LOCATION

INC
BP

MOV
AL,[SI+2000H]
LOWER_REGION_BYTE

MOV
[BP], AL
ADJUST AND STORE

ADD
SI, 80
POINTER INTO REGEN

DEC
DH
LOOP CONTROL

JNZ
DO IT SOME MORE

JMP
SHORT R31
LOW_RESOLUTION_READ

-----

LOW_RESOLUTION_READ

POP
DS
POINT TO REGEN SEGMENT

POP
SI, 1
OFFSET+4 SINCE 4 BYTES/CHAR

ADD
SI, 2000H
GET 4 BYTES FROM REGEN INTO

CALL
R55
GET 4 BYTES FROM REGEN INTO

CALL
R55
SINGLE_SAVE

CALL
R55
SINGLE_SAVE

INC
BP

ADD
SI, 2000H
GOTO LOWER_REGION

CALL
R55
GET 4 BYTES FROM REGEN INTO

CALL
R55
SINGLE_SAVE

ADD
SI, 2000H
GOTO LOWER_REGION

CALL
R55
LESS THAN 4K

ADD
SI, 4000H-80
OFFSET LESS THAN 4K

SUB
SI, 20000H-80
SHORT ADDRESS?

SUB
SI, 20000H-80
ADJUST POINTER BACK TO UPPER

DEC
DH

JNZ
R26
DO IT SOME MORE

JMP
SHORT R31

-----

MEDIUM_RESOLUTION_READ

POP
DS
MET_RES_READ

POP
SI, 1
OFFSET+2 SINCE 2 BYTES/CHAR

POP
DS
GET PAIR BYTES FROM REGEN INTO

POP
DS
SINGLE_SAVE

POP
DS
GO TO LOWER_REGION

DEC
DH
GET THIS PAIR INTO SAVE

ADD
SI, 2000H
POINT TO BIOS DATA AREA

CALL
R50
DO WE HAVE A 32K REGEN AREA?

CALL
R50
SINGLE_SAVE

CALL
R50
SINGLE_SAVE

ADD
SI, 2000H
GOTO LOWER_REGION

CALL
R50
GET PAIR BYTES FROM REGEN INTO

CALL
R50
SINGLE_SAVE

ADD
SI, 2000H
GOTO LOWER_REGION

CALL
R50
ADJUST_POINTER

ADD
SI, 4000H-80
SHORT ADDRESS?

DEC
DH

JNZ
R29
KEEP GOING UNTIL ALL 8 DONE

DEC
DH

JNZ
R29
KEEP GOING UNTIL ALL 8 DONE

KEEP GOING UNTIL ALL 8 DONE
```assembly
F5E2 33 C0  ; FND_CHAR
F5E4 3E D8  ; ESTABLISH ADDRESSING TO VECTOR
F5E6 C4 3E 0110 R  ; GET POINTER TO FIRST HALF
F5EA B3 E0 08  ; NEXT POINTER TO BEGINNING OF
F5E9 B8 F5  ; SAVE AREA
F5EF FC  ; ENSURE DIRECTION
F5F0 32 C0  ; CURRENT CODE POINT BEING MATCHED
F5F2 16  ; ESTABLISH ADDRESSING TO STACK
F5F3 1F  ; FOR THE STRING COMPARE
F5F4 BA 0080  ; NUMBER TO TEST AGAINST
F5F7 56  ; SAVE AREA POINTER
F5F8 57  ; SAVE CODE POINTER
F5F9 B9 000B  ; NUMBER OF BYTES TO MATCH
F5FC F3 A6  ; COMPARE THE B BYTES
F5FE 5F  ; RECOVER THE POINTERS
F5FF 5E  ; POP SI
F600 74 1E  ; IF ZERO FLAG SET, THEN MATCH
F602 FE C0  ; OCCURRED
F604 B3 C7 08  ; NO MATCH, MOVE ON TO NEXT
F607 4A  ; NEXT CODE POINT
F608 75 E0  ; LOOP CONTROL
F60A 0A C0  ; DO IN SECOND HALF
F60C 74 12  ; OR IF ONLY 1ST HALF SCANNED
F60E 2B C0  ; AL = 0, THEN ALL HAS BEEN SCANNED
F610 BE D8  ; ESTABLISH ADDRESSING TO VECTOR
F612 C4 3E 007C R  ; GET POINTER
F616 BC C0  ; SEE IF THE POINTER REALLY EXISTS
F61B 0B C7  ; IF ALL 0, THEN DOESN'T EXIST
F61A 74 04  ; NO SENSE LOOKING
F61C 80 80  ; ORIGIN FOR SECOND HALF
F61E EB 92  ; GO BACK AND TRY FOR IT
F620 B3 C4 08  ; DO IN SECOND HALF
F623 E9 0F70 R  ; IF NOT FOUND
F626  ; ADD 256, REGENERATE THE STACK, THROW AWAY
F629 E9 VIDEO_RETURN  ; WORK AREA
F62A GRAPHICS_READ ENDP

F626 AC  ; GET CODE POINT
F627 EB F67E R  ; DOUBLE UP ALL THE BITS
F62A 23 C3  ; CONVERT THEM TO FOREGROUND COLOR
F62C F6 C2 00  ; IS THIS XOR FUNCTION?
F62F 74 07  ; NO, STORE IT IN AS IT IS
F631 26: 32 25  ; FUNCTION WITH HALF
F634 26: 32 45 01  ; AND WITH OTHER HALF
F633 26: 88 00  ; STORE SECOND BYTE
F63F C3  ; STORE SECOND BYTE
F640 R35  ; EXPAND 1 DOT ROW OF A CHAR INTO 4 BYTES IN THE REGEN BUFFER
F644 EB F6A0 R  ; EXPAND THE 2 BYTES & PUT IN REGEN
F643 EB E5  ; RECOVER CODE POINT
F645 R38  ; ADJUST REGEN POINTER
F646 AC  ; GET CODE POINT
F647 51  ; SAVE
F648 B1 04  ; MOV HIGH NIBBLE TO LOW
F64A D2 E8  ; SHR AL,CL
F64C 59  ; POP CX
F64E EB F640 R  ; EXPAND LOW NIBBLE & PUT IN REGEN
F650 59  ; RECOVER CODE POINT
F651 4F  ; ADJUST REGEN POINTER
F652 47  ; RESTORE REGEN POINTER
F653 EB F640 R  ; EXPAND LOW NIBBLE & PUT IN REGEN
F654 4F  ; RECOVER CODE POINT
F655 4F  ; ADJUST REGEN POINTER
F656 C3  ; RESTORE REGEN POINTER

F659 R39  ; EXPAND LOW COLOR
F65A EB  ; FILL THE ENTIRE BX REGISTER
F65B  ; BL = COLOR TO BE USED (LOW 2 BITS)
F65C  ; EXIT
F65D  ; BX = COLOR TO BE USED (8 REPLICATIONS OF THE 2 COLOR BITS)

A-92  ROM BIOS
F659 PROC NEAR
F659 80 E3 03
F65C 8A C3
F65E 51
F65F B9 0003
F662 D0 E0
F664 D0 E0
F666 0A DB
F668 E2 FB
F66A 8A FB
F66C 59
F66D 03
F66E ENDP
F66F

R40 PROC NEAR
R41 SAL AL, 1
R42 PROC NEAR
R43 PROC NEAR
R44: MOV BX, AX
R45 PROC NEAR

AND BL, 3
MOV AL, BL
POP CX
MOV CX, 3
NUMBER OF TIMES TO DO THIS
LEFT SHIFT BY 2
ANOTHER COLOR VERSION INTO BL
FILL ALL OF BL
FILL UPPER PORTION
REGISTER BACK
ALL DONE

EXPAND_LOW_COLOR:
THIS ROUTINE EXPANDS THE LOW 4 BITS IN BL TO
FILL THE ENTIRE BX REGISTER
ENTRY --
BL = COLOR TO BE USED ( LOW 4 BITS )
EXIT --
BX = COLOR TO BE USED ( 4 REPLICATIONS OF THE 4 COLOR BITS )

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

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
F6C3 BA 24  
F6C5 BA 44 01  
F6CB 1E  
F6CB EB 13B8 R  
F6CC B0 3E 0049 R 0A  
F6D1 IF  
F6D2 78 11  
F6D4 53  
F6D5 B0 0008  
F6DB 00 FC  
F6DA D1 DB  
F6DE D1 DB  
F6E0 E2 F6  
F6E2 BB C3  
F6E4 BB  
F6E5 B9 C000  
F6E6 02 02  
F6E8 B5 C1  
F6E9 74 01  
F6EC 9F 9  
F6F1 D1 E9  
F6F3 D1 E9  
F6F5 73 F3  
F6F7 BB 5E 00  
F6FA 45  
F6FB C3  
F6FC  
F6FC BA 24  
F6FA BA 44 01  
F701 32 D2  
F703 EB F714 R  
F705 BA 64 02  
F707 BA 44 03  
F70C EB F714 R  
F70F BB 56 00  
F712 45  
F713 C3  
F715 BB F000  
F717 BB 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  

--- 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  
MOV AH, [SI]  
GET FIRST BYTE  
MOV AL, [SI+1]  
GET SECOND BYTE  
PUSH DS  
SAVE DS  
CALL DS  
POINT TO BIOS DATA AREA  
CMP CRT_MODE, OAH  
IN 640X200 4 COLOR MODE?  
PUSH DS  
RESTORE REGEN SEG  
JNE RS2  
NO JUMP  
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 CIC0 ADJACENT PAIRS.  
R50 ENDP  
--- 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  
R55 PROC NEAR  
MOV AH, [SI]  
GET FIRST 2 BYTES  
MOV AL, [SI+1]  
BUILD HIGH NIBBLE  
XOR DL, DL  
BUILD HIGH NIBBLE  
CALL R56  
SET SECOND BTYTES  
MOV AH, [SI+2]  
CALL R56  
BUILD LOW NIBBLE  
MOV [BP], DL  
STORE RESULT IN SAVE AREA  
INC BP  
ADJUST POINTER  
R55 ENDP  
R56 PROC NEAR  
MOV CX, [FO00H]  
4 BIT MASK TO TEST THE ENTRIES  
R57: TEST AX, CX  
IS THIS SECTION BACKGROUND?  
JZ R58  
IS ZERO, IT IS BACKGROUND  
R58: RCL DL, 1  
MOVE THAT BIT INTO RESULT  
SHR CX, 1  
MOVE MASK RIGHg 4 BITS  
JNC R57  
DO IT AGAIN IF MASK DIDN'T FALL OUT  
R56 ENDP  
--- 
A-94 ROM BIOS
; 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
; (CH),DL = ROW,COLUMN OF CURRENT LIGHT PEN POSITION
; (CH) = RASTER POSITION
; (BX) = BEST GUESS AT PIXEL HORIZONTAL POSITION

ASSUME CS:CODE,DS:DATA

; ------------------------
; SUBTRACT_TABLE
; V1 LABEL BYTE
; DB 3,3,5,5,3,3,0,2,3,4 ;

READ_LPEN PROC NEAR
; --------
; WAIT FOR LIGHT PEN TO BE DEPRESSED
; MOV AL, AH ; SET NO LIGHT PEN RETURN CODE
; MOV DX, VGA_CTL ; GET ADDRESS OF VGA CONTROL REG
; INC AL ; GET STATUS REGISTER
; TEST AL, 4 ; TEST LIGHT PEN SWITCH
; JZ V7B ; NOT Set, RETURN
; JMP V6 ; NOT SET, RETURN
;------------------
; MOV TEST FOR LIGHT PEN TRIGGER
; V7B: TEST AL, 2 ; TEST LIGHT PEN TRIGGER
; JNZ V7A ; RETURN WITHOUT RESETING TRIGGER
;---------------------
; TRIGGER HAS BEEN SET, READ THE VALUE IN
; VTA: MOV AH, 16 ; 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
; MOV AX, AL ; SET IT UP
; MOV DL, AL ; DATA REGISTER
; MOV CH, AL ; SAVE IN CX
; MOV CL, DL ; ADDRESS REGISTER
; MOV CX, DL ; LIGHT PEN TRIGGER
; MOV CH, AL ; SECOND DATA REGISTER
; MOV AH, CH ; AX HAS INPUT VALUE
;-------------------
; AX HAS THE VALUE READ IN FROM THE 6845
; MOV BL, CRT_MODE ; MODE VALUE TO BX
; MOV AL, CX ; DETERMINE AMOUNT TO SUBTRACT
; MOV AH, BX ; TAKE IT AWAY
; MOV CX, 4000 ; IN TOP OR BOTTOM BORDER?
; JB V15 ; NO, OKAY
; XOR AH, AK ; YES, SET TO ZERO
; MOV BX, CRT_START ;
; SHR BX, 1 ;
; SUB AH, BX ; CONVERT TO CORRECT PAGE ORIGIN
; JNS V2 ; IF POSITIVE, DETERMINE MODE
; SUB AX, AK ; TO PLAYS AS 0
;------------------
; DETERMINE MODE OF OPERATION
; V2: DETERMINE MODE
; MOV CL, 3 ; SET #B SHIFT COUNT
; CMP CRT_MODE, 4 ; DETERMINE IF GRAPHICS OR ALPHA
; JB V4 ; ALPHAPEN
;---------------------
; GRAPHICS MODE
; MOV DL, 40 ; DIVISOR FOR GRAPHICS
; CMP CRT_MODE, 9 ; USING 32K REGEN?
; CMP AJ, V2 ; NO, JUMP
; MOV DL, 80 ; YES, SET RIGHT DIVISOR
;------------------
; AX CONTAINS OFFSET INTO REGEN BUFFER
;------------------------
; 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--
;----- DETERMINE GRAPHIC ROW POSITION
F7B2 8A EB
F7B4 02 ED
F7B6 80 3E 0049 R 09
F7B7 72 06
F7B9 D0 ED
F7B0 D0 ED
F7C1 02 ED
F7C3 8A DC
F7C5 2A FF
F7C7 80 3E 0049 R 06
F7CC 72 15
F7CE 77 06
F7D0 B1 04
F7D2 D0 E4
F7D4 EB 0D
F7D6 80 3E 0049 R 09
F7D7 77 F3
F7D9 74 04
F7DF B1 02
F7E1 D0 EC
F7E3 D3 E3
F7E5 8A D4
F7E7 8A F0
F7E9 D0 EE
F7EB D0 EE
F7E2 EB 12

;----- DETERMINE ALPHA CHAR POSITION
F7E3 0D 1H
F7E7 60 40
F7E9 60 00
F7EB 60 00

;----- ALPHA MODE ON LIGHT PEN
F7E8 F6 36 004A R
F7E9 8A F0
F7F0 8A D4
F7F7 D2 E0
F7F9 8A EB
F7FB 8A DC
F7FC 32 FF
F7FF D3 E3

;----- EXECUTE POWER ON DIAGNOSTICS TO SERVICE UNUSED
F801 60 00
F803 52
F804 BB 16 0063 R
F806 B3 C2 07
F808 EE
F80C 5A
F80D 5F
F80E 5E
F80F 1F
F810 1F
F811 1F
F812 1F
F813 07
F814 CF

READ_LPEN ENDP

;----- TEMPORARY INTERRUPT SERVICE ROUTINE
F815 IE
F816 50
F817 EB 13BB R
F818 8B 08
F81C E6 20
F81E 80
F81F E4 20
F820 8A E0
F823 0A C4
F825 70 04
F827 B4 FF
F829 80 0A
F82B E4 21
F82D 0A C4
F82F E6 21
F831 B0 20
F833 E6 20
F835 88 26 0084 R
F837 8B
F838 1F
F83B FB
F83C 0F
F83D CF

D11 PROC NEAR
ASSUME DS:DATA
F815 IE
PUSH DS
PUSH AX
SAVE REG AX CONTENTS
F816 50
CALL D5S
F817 EB 13BB R
F818 8B 08
MOV AL,0B8
READ IN-SERVICE REG
F81C E6 20
OUT INTAO0,AL
(FIND OUT WHAT LEVEL BEING
SERVICED)
F81E 80
F81F E4 20
IN AL,INTAO0
GET LEVEL
F820 8A E0
MOV AH,AL
SAVE IT
F823 0A C4
OR AL,AH
00? (NO HARDWARE ISR ACTIVE)
F825 70 04
JNZ HW_INT
F827 B4 FF
MOV AH,OFF
F829 80 0A
JMP SHORT_INT_FLAG ; SET FLAG TO FF IF NON-HOWARE
F82B E4 21
HW_INT: IN AL,INTAO1
GET MASK VALUE
F82D 0A C4
OR AL, AH
MASK OFF LVL BEING SERVICED
F82F E6 21
OUT INTAO1,AL
F831 B0 20
MOV AL,EO1
F833 E6 20
OUT INTAO0,AL
F835 88 26 0084 R
MOV PW_PROFILE,AL
SET FLAG
F837 8B
POP AX
RESTORE REG AX CONTENTS
F838 1F
F83B FB
STI
INTERRUPTS BACK ON
F83C 0F
DUMMY_RETURN
NEED IRET FOR VECTOR TABLE
F83D CF
IRET
F83D D11 ENDP

A-96 ROM BIOS
; --- INT 12 ------------------------------------------
; MEMORY_SIZE_DETERMINE
; INPUT
; NO REGISTERS
; THE MEMORY_SIZE VARIABLE IS SET DURING POWER ON DIAGNOSTICS
; OUTPUT
; (AX) = NUMBER OF CONTIGUOUS 1K BLOCKS OF MEMORY
;---------------------------------------------------------

FB41
ORG 0FB41H

FB41
MEMORY_SIZE_DETERMINE PROC FAR

FB41
STI ; INTERRUPTS BACK ON

FB42
TEST DS,DS ; SAVE SEGMENT

FB43
MOV AX,DATA ; ESTABLISH ADDRESSING

FB46
MOV DS,AX

FB48
MOV AX,MEMORY_SIZE ; GET VALUE

FB4C
IRET ; RETURN TO CALLER

MEMORY_SIZE_DETERMINE ENDP
;---------------------------------------------------------

; --- INT 11 ------------------------------------------
; EQUIPMENT_DETERMINE
; 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 (00H) = LOW ORDER BYTE OF EQUIPMENT
; PORT 3FA = INTERRUPT 10 REGISTER OF 8250
; PORT 37B = OUTPUT PORT OF PORT 8255 PORT THAT
; CAN BE READ AS WELL AS WRITTEN
; OUTPUT
; (AX) IS SET, BIT SIGNIFICANT, TO INDICATE ATTACHED I/O
; BIT 15,14 = NUMBER OF PRINTERS ATTACHED
; BIT 13 = 1 SERIAL PORT ATTACHED
; BIT 12 = GAME I/O ATTACHED
; BIT 11,10,9 = NUMBER OF RS232 CARDS ATTACHED
; BIT 8 = DMA CHIP PRESENT ON SYSTEM, 1 = DMA ON SYSTEM
; BIT 7,6 = NUMBER OF DISKETTE DRIVES
; O=1, 1=2, 10=3, 11=4 ONLY IF BIT O = 1
; BIT 5,4 = INITIAL VIDEO MODE
; 00 = UNUSED
; 01 = 40X255 BW USING COLOR CARD
; 10 = 80X255 BW USING COLOR CARD
; 11 = 80X255 BW USING BW CARD
; BIT 3,2 = PLANAR RAM SIZE (1024, 4096, 164K)
; BIT 1 NOT USED
; BIT 0 = 1 (IPL DISKETTE INSTALLED)
; NO OTHER REGISTERS AFFECTED
;---------------------------------------------------------

FB4D
ORG 0FB4DH

FB4D
EQUIPMENT PROC FAR

FB4E
STI ; INTERRUPTS BACK ON

FB4F
TEST DS,DS ; SAVE SEGMENT REGISTER

FB52
MOV AX,DATA ; ESTABLISH ADDRESSING

FB54
MOV DS,AX

FB57
MOV AX, EQUIP_FLAG ; GET THE CURRENT SETTINGS

FB58
IRET ; RETURN TO CALLER

EQUIPMENT ENDP
;---------------------------------------------------------

; --- INT 5 ------------------------------------------
; 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
; (DX) = COUNT OF BYTES ACTUALLY READ
; (CY) = 0 IF NO ERROR OCCURRED
; (AH) = ERROR RETURN IF (CY)=1
; (AH) = 01 IF CRC ERROR WAS DETECTED
; (AH) = 02 IF DATA TRANSITIONS ARE LOST
; (AH) = 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
; (ES, 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).
;---------------------------------------------------------

FB59
ORG 0FB59H

FB59
CASSETTE IO PROC FAR

FB59
STI ; INTERRUPTS BACK ON

FB5A
TEST DS,DS ; ESTABLISH ADDRESSING TO DATA

FB5B
E8 1280h R ; CALL BIOS

FB5C
E8 26 0071h R ; CALL BIOS_BREAK, 7FH

FB63
E8 F86A R ; MAKE SURE BREAK FLAG IS OFF

FB66
IRET ; INTERRUPT RETURN

FB67
CASSETTE IO ENDP

FB6A
ORG NEAR

Appendix A

ROM BIOS A-97
PURPOSE:
TO CALL APPROPRIATE ROUTINE DEPENDING ON REG AH
AH ROUTINE

0  MOTOR ON
1  MOTOR OFF
2  READ CASSETTE BLOCK
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 1B JZ MOTOR_OFF ; YES, DO IT
F872 FE CC DEC AH; READ CASSETTE BLOCK?
F874 7A 1A JZ READ_BLOCK ; YES, DO IT
F876 FE CC DEC AH; WRITE CASSETTE BLOCK?
F878 75 03 JNZ W2; NOT_DEFINED
F87A E9 F97 R JMP WRITE_BLOCK ; YES, DO IT
F87D W2: MOV AH, 0B0H ; ERROR, UNDEFINED OPERATION
F87F F9 ; ERROR FLAG
FB80 C3 RET

----------

W1: ENDP
F8B1 MOTOR_ON PROC NEAR

----------

F8B1 E4 61 IN AL, PORT_B ; READ CASSETTE OUTPUT
F8B3 24 F7 AND AL, NOT 0BH ; CLEAR BIT TO TURN ON MOTOR
F8B5 E6 61 W3: OUT PORT_B, AL; WRITE IT OUT
F8B7 2A E4 SUB AH, AH ; CLEAR AH
F8B9 C3 RET

----------

F8BA MOTOR_ON ENDP
F8BA MOTOR_OFF PROC NEAR

----------

F8B9 MOTOR_OFF ENDP
F8B9 READ_BLOCK PROC NEAR

----------

PURPOSE:
TO TURN ON CASSETTE MOTOR

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 TO BYTE PAST LAST BYTE PUT IN MEM
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 S6 PUSH SI ; SAVE SI
F893 BE 0007 MOV SI, 7 ; SET UP RETRY COUNT FOR LEADER
F896 EB F850 R CALL BEGIN_OP ; BEGIN BY STARTING MOTOR
F899 W4: SEARCH FOR LEADER
F899 E4 62 IN AL, PORT_C ; GET INITIAL VALUE
F89A 24 10 AND AL, 010H ; MASK OFF EXTRANEOUS BITS
F89B A0 066B R MOV LAST_VAL, AL ; SAVE IN LOC_LAST_VAL
F89D BA 3F7A MOV DX, 16250 ; # OF TRANSITIONS TO LOOK FOR
F89F ED F5 JMP W3 ; WRITE IT, CLEAR ERROR, RETURN

----------

W5: WAIT_FOR_EDGE
F8A3 F6 0E 0071 R 80 TEST BIOS_BREAK, BOH ; CHECK FOR BREAK KEY
F8AB 75 03 JNZ W6A ; JUMP IF NO BREAK KEY
F8AA 4A DEC DX ; JUMP IF BREAK KEY HIT
F8AB 75 03 JNZ W7 ; JUMP IF BEGINNING OF LEADER
F8AD E9 F82F R W6A: JMP W17 ; JUMP IF NO LEADER FOUND
F8BE EB F86F R W7: CALL READ_HALF_BIT ; IGNORE FIRST EDGE
F8BB E3 EE JCXZ W5 ; JUMP IF NO EDGE DETECTED
F8BE BA 037B MOV DX, 0378H ; CHECK FOR HALF BITS
F8BB 89 0200 MOV CX, 200H ; MUST HAVE AT LEAST THIS MANY ONE
F8BB 6D 1F SUB DX, 1F ; SIZE PULSES BEFORE CHECKING FOR
F8BB FA 10 CLI ; SYNC BIT (0)
F8BC WB: ; DISABLE INTERRUPTS
F8BC SEARCH-DR
F8BC F6 0E 0071 R 80 TEST BIOS_BREAK, BOH ; CHECK FOR BREAK KEY
F8BC 75 6C JNZ W17 ; JUMP IF BREAK KEY HIT
F8BC F3 S1 PUSH CX ; SAVE REG CX
F8BC EB F86F R CALL READ_HALF_BIT ; GET PULSE WIDTH
F8C7 6B C9 OR CX, CX ; CHECK FOR TRANSITION
F8CB 99 59 POP CX ; RESTORE ONE BIT COUNTER
F8CA 74 C0 JZ W4 ; JUMP IF NO TRANSITION
F8CC 38 03 CMP DX, BX ; CHECK PULSE WIDTH
F8CD E3 04 JCXZ W5 ; IF CX=0 THEN WE CAN LOOK
F8D0 73 C7 JNC W4 ; JUMP IF ZERO BIT (NOT GOOD)
F8D2 E2 EB LOOP WB ; DEC CX AND READ ANOTHER HALF ONE
F8DA 4A JC W8 ; JUMP IF ONE BIT (STILL LEADER)

----------

A-98 ROM BIOS
FB06  E8  F96F  R
FB09  E8  F941  R
FB0C  3C  16
FB0E  75  49

----- A SYNCH BIT HAS BEEN FOUND. READ SYNCHR.-----
CALL  READ_HALF_BIT  ; SKIP OTHER HALF OF SYNCH BIT (0)
CALL  READ_BYTE  ; READ SYNCH BYTE
CMP  AL, 1DH  ; SYNCHRONIZATION CHARACTER
JNE  W16  ; JUMP IF BAD LEADER FOUND

----- GOOD CRC 80 READ DATA BLOCK(S)-----

F8E0  5E  ; POP SI, RESTORE REGS
F8E1  59  ; POP CX
F8E2  58  ; POP BX

-----------------------------
READ 1 OR MORE 256 BYTES 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
-----------------------------

F8E3  51  ; PUSH CX ; SAVE BYTE COUNT
F8E4  C7  06  0069  R  FFFF
F8E8  BA  0100
F8E9  26  F6  0071  R  80
F8FE  75  23
F8F4  EB  F941  R
F8F7  72  1E
F8F9  E3  05

F8FB  26  BB  07
F8FE  43
F8FF  49
F900  D0  00

W12:  ; LOOP UNTIL DATA BLOCK HAS BEEN READ FROM CASSETTE
DEC  DX  ; DEC BLOCK CNT
JNZ  W13  ; لم يتم قراءة جميع النقاط

F90C  7F  EA
F90E  EB  F941  R
F90F  2A  E4
F90B  81  3E  0069  R  100F
F911  75  06
F913  E3  06

F915  EB  CD
F917  B4  01
F919  B4  01
F91B  F0  01
F91C  2B  D1

W13:  ; STILL MORE, SO READ ANOTHER BLOCK
JMP  W10

F91E  50
F91F  F6  C4  90
F922  75  13
F924  EB  F941  R
F927  EB  0E
F929  4E
F92A  74  03
F92C  EB  F999  R
F92F  01

W14:  ; DATA TIMEOUT
MOV  AH, 01H
INC  AL
MOV  ES:[BX], AL
IN  CX, AL
INC  BUFFER PTR

W15:  ; EXIT EARLY ON ERROR
MOV  DX, 0000h
SUB  CX, 0000h
POP  DX
; BAD-CRC

F930  5B
F932  2B  D2
F934  B4  04
F936  50

W16:  ; BAD-LEADER
DEC  SI
JNO  W17

F938  5B
F93A  74  03
F93C  E9  F999  R
F93F  03

W17:  ; NO VALID DATA FOUND
; NO DATA FROM CASSETTE ERROR, I.E. TIMOUT

F940  5E
F943  5B
F945  5B
F947  5B

W18:  ; NOT-OFF
STI  ; REENABLE INTERRUPTS
CALL  MOTOR_OFF  ; TURN OFF MOTOR
POP  AX  ; RESTORE RETURN CODE
CMP  AH, 01H  ; SET CARRY IF ERROR (AH=0)
JNZ  F941  ; FINISHED

ROM BIOS  A-99

Appendix A
PURPOSE:
TO READ A BYTE FROM CASSETTE
ON EXIT:
REG AL CONTAINS READ BYTE

READ_BYTE PROC NEAR
  PUSH BX  ; SAVE REGS BX, CX
  MOV CX, 08
  MOV CL, 8H
  SET BIT COUNTER FOR 8 BITS
  MOV AL, 40H
  OUT 21H, AL
  SUB AL, XCHG AL, AL
  AND 0FH
  MOV AL, 40H
  OUT 21H, AL
  PUSH CX  ; SAVE CX

READ_DATA_BIT_FROM_CASSETTE

READ_HALF_BIT PROC NEAR
  MOV CX, 08
  MOV AL, 40H
  OUT 21H, AL
  SUB AL, XCHG AL, AL
  AND 0FH
  MOV AL, 40H
  OUT 21H, AL
  PUSH CX  ; SAVE CX

READ_HALF_BIT ENDP

ENDP

__END__
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

WRITE_BLOCK PROC NEAR

F997
F997 53
F998 51
F999 E4 61
F99B 24 F0
F99C 0C 01
F99F E6 61
FA01 00 B6
FA03 E6 43
FA05 EB FA00 R
FA08 00 04A0
FA0B EB FA35 R
FA0E 00 0B00

FA01 F9
FA02 EB FAIF R
FA05 E2 FA
FA07 FA
FA08 FB
FA09 EB FAIF R
FA0C 59
FA0D 5B
FA0E 16
FA0F EB FA08 R

W23:
STC
CALL WRITE_BIT
LOOP W23
; LOOP 'TIL LEADER IS WRITTEN
CLC
CALL WRITE_BIT
POP BX
; RESTORE REGS CX,BX
POP CX
; WRITE 1'S CHARACTER
MOV AL,16H
; WRITE SYNC CHARACTER
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

WR_BLOCK:

F9C3
F9C3 C7 06 0069 R FFFF
F9CB BA 0100
F9CC 26 BA 07
F9CF EB FA08 R
F9D0 E3 02
FA04 43
FA05 49
FA06
FA06 4A
FA07 7F F3

FA09 A1 0069 R
FA0C F7 D0
FA0E 50
FA0F B6 E0
FA0E EB FA00 R
FA0F EB FA00 R
FA0E EB FA00 R
FA0E 0B C9
FA0A 75 07
FA0C 51
FA0D 08
FA0E 00 020
FA0F
FA0F 09
FA0F EB FAIF R
FA0F E2 FA
FA0F 80 00
FA0F 66 43
FA0F EB FA01
FA0F EB FA35 R
FA0E EB FA08 R
FA0E 2B 0C
FA07 C3
FA08

W24:
MOV CRC_REG,0FFFFH
; INIT CRC
MOV DX,256
; FOR 256 BYTES
CALL WRITE_BYTE
; WRITE IT TO CASSETTE
JCRX W25
; UNLESS CX=0, ADVANCE PTRS & DEC COUNT
INC BX
; INC BUFFER POINTER
DEC CX
; DEC BYTE COUNTER
SWI-ADV
DEC DX
; DEC BLOCK CNT
JG W24
; LOOP TILL 256 BYTE BLOCK 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 A1 0069 R
F9DC F7 D0
F9DF 50
FA00 B6 E0
FA0E EB FA00 R
FA0F EB FA00 R
FA0E EB FA00 R
FA0E 0B C9
FA0A 75 07
FA0C 51
FA0D 08
FA0E 00 020
FA0F
FA0F 09
FA0F EB FAIF R
FA0F E2 FA
FA0F 80 00
FA0F 66 43
FA0F EB FA01
FA0F EB FA35 R
FA0E EB FA08 R
FA0E 2B 0C
FA07 C3
FA08

W26:
MOV AX,CRC_REG
; WRITE THE ONE'S COMPLEMENT OF THE
; TWO BYTE CRC TO TAPE
NOT AX
; FOR 1'S COMPLEMENT
PUSH AX
; SAVE IT
XCHG AH,AL
; WRITE MS BYTE FIRST
CALL WRITE_BYTE
; WRITE IT
POP AX
; GET IT BACK
CALL WRITE_BYTE
; NOW WRITE LS BYTE
OR CX,CX
; IS BYTE COUNT EXHAUSTED?
PUSH CX
; SAVE REG CX
; SET -CX FOR
STI
; RE-ENABLE INTERRUPTS
MOV CX,32
; WRITE OUT TRAILER BITS
TRAIL-LOOP

FA97 F9
FA9F EB FAIF R
FA9F E2 FA
FA9F 80 00
FA9FA 66 43
FA9FC EB FA01
FA9F EB FA35 R
FA02 EB FA08 R
FA02 2B 0C
FA07 C3
FA08

WRITE_BLOCK ENDP
WRITE A BYTE TO CASSETTE.
BYTE TO WRITE IS IN REG AL.

FA08
FA08 51 WRITE_BYTE PROC NEAR
FA09 50 PUSH CX, SAVE REGS CX, AX
FA0A BA EB MOV CH, AL, AL=BYTE TO WRITE
FA0B B1 0B MOV CL, B, FOR B DATA BITS IN BYTE.
FA0C B1 0B NOTE: TWO EDGES PER BIT
FA0D 00 05 RCL CH, 1, ROTATE MS BIT INTO CARRY
FA0E 9C PUSHF, SAVE FLAGS.
FA0F B1 0B CALL WRITE_BIT, WRITE DATA
FA10 B1 0B CALL CRC_GEN, COMPUTE CRC ON DATA BIT
FA11 B1 0B DEC CL, LOOP TILL ALL B DATA BITS
FA12 B1 0B ADD AL, CL, JUMP IF NOT DONE YET
FA13 59 POP AX, RESTORE REGS AX, CX
FA14 B9 POP CX, SET
FA15 B3 07 WRITE_BYTE ENDP
FA16 07 WRITE_BIT PROC NEAR
FA17 07 TO WRITE A DATA BIT TO CASSETTE
FA18 07 CARRY FLAG CONTAINS DATA BIT
FA19 07 I.E. IF SET DATA BIT IS A ONE
FA1A 07 IF CLEAR DATA BIT IS A ZERO
FA1B 07 NOTE: TWO EDGES ARE WRITTEN PER BIT
FA1C 07 ONE BIT HAS 500 USEC BETWEEN EDGES
FA1D 07 FOR A 1000 USEC PERIOD (1 MILLISEC)
FA1E 07 ZERO BIT HAS 250 USEC BETWEEN EDGES
FA1F 07 FOR A 500 USEC PERIOD (.5 MILLISEC)
FA20 07 CARRY FLAG IS DATA BIT

FA20 BB 0440 MOV AX, 1184, SET AX TO NOMINAL ONE SIZE
FA21 B2 03 JC W28, JUMP IF ONE BIT
FA22 BB 0250 MOV AX, 592, NO, SET TO NOMINAL ZERO SIZE
FA23 07 WRITE-BIT-AX
FA24 07
FA25 B1 0B W28: PUSH AX, WRITE BIT WITH PERIOD EQ TO VALUE
FA26 07 AX
FA27 50 MOV AX, 05, WRITE BIT WITH PERIOD EQ TO VALUE
FA28 B1 0B W29: IN AL, PORT_C, INPUT TIMER_0 OUTPUT
FA29 E4 62 AND AL, 020H, ONE BIT HAS 200 USEC BETWEEN EDGES
FA2A 24 20 W2A: MOV AX, 0592, FOR A 1000 USEC PERIOD (1 MILLISEC)
FA2B B1 0B W30: IN AL, PORT_C, LOOP TILL HIGH
FA2C 74 FA AND AL, 020H, WAIT TILL TIMER'S OUTPUT IS
FA2D E4 62 W31: MOV AX, 0592, HIGH
FA2E 75 FA IN AL, 020H, NOW WAIT TILL TIMER'S OUTPUT IS
FA2F B1 0B W32: AND AL, 020H, LOOP TILL HIGH
FA30 24 20 JNZ W30, LOW
FA31 75 FA MOV AX, 0592, NOW WAIT TILL TIMER'S OUTPUT IS
FA32 24 20 JNZ W30, LOW
FA33 58 MOV AX, 0592, NOW WAIT TILL TIMER'S OUTPUT IS
FA34 75 FA MOV AX, 0592, NOW WAIT TILL TIMER'S OUTPUT IS
FA35 07
FA36 E6 42 OUT 042H, AL, Reload Timer with Period
FA37 B4 C4 MOV AL, AH, Period
FA38 E6 42 OUT 042H, AL, Period
FA39 C3 07 RET, Period
FA3A C3 07 WRITE_BIT ENDP
FA3B 07
FA3C CRC_GEN PROC NEAR
FA3D 07 UPDATE CRC REGISTER WITH NEXT DATA BIT
FA3E 07 CRC IS USED TO DETECT READ ERRORS
FA3F 07 ASSUMES DATA BIT IS IN CARRY
FA40 07 REG AX IS MODIFIED
FA41 07 FLAGS ARE MODIFIED
FA42 A1 0069 R MOV AX, CRC_REG, THE FOLLOWING INSTRUCTIONS
FA43 07 WILL SET THE OVERFLOW FLAG
FA44 07 IF CARRY AND MS BIT OF CRC
FA45 07 ARE UNEQUAL
FA46 07
FA47 D1 0B RCR AX, 1, CLEAR CARRY
FA48 D0 00 RCL AX, 1, IF DATA BIT XORED WITH
FA49 F8 CLC, CRC REG BIT IS ONE
FA4A 71 04 JNO W32, THEN XOR CRC REG WITH
FA4B 35 0810 XOR AX, 0810H, CRC REG BIT IS ONE
FA4C 07
FA4D 07 XOR AX, 0810H, THEN XOR CRC REG WITH
FA4E 07 CRC_REG
FA4F C3 07 XOR AX, 0810H, CRC_REG
FA50 07 CRC_GEN ENDP

A-102 ROM BIOS
CRC_CHECK PROC NEAR

ASSUME DS:NOTHING

MOV BX,CX ; Save count
MOV DX,OFFFH ; Init. encode register
CLD ; Set dir flag to increment
XOR AH, AH ; Init. work reg high
XOR CL, CL ; Init. work reg low
MOV CL, 4 ; Set rotate count

CRC_1: LODSB ; Get a byte
XOR DL, AH ; Form AJ+CJ+1
MOV AL, DL

FEB0 03 C0 ; ROL AX,CL ; Shift work reg back 4
FEB2 33 00 ; XOR DX,AX ; Add into result reg
FEB4 D1 C0 ; ROL AX, 1 ; Shift work reg back 1
FEB6 A6 F2 ; XCHG DH, DL ; Swap partial sum into result reg
FEB8 33 00 ; XOR DX, AX ; Add work reg into results
FEBA D3 CB ; ROR AX, CL ; Shift work reg over 4
FEBC 24 E0 ; AND AL, 1110000B ; Clear off (EFH)
FEBE 33 00 ; XOR DX, AX ; Add (ABCD) into results
FEB0 D1 CB ; ROR AX, 1 ; Shift work reg over on (AH:O for next pass)
FEB2 32 F0 ; XOR DX, DL ; Add (ABCD) into results low
FEB4 4B ; DEC BX ; Decrement count
FEB7 75 E4 ; JNZ CRC_1 ; Loop till count = 0000
FEB7 08 D2 ; OR DX, DX ; DX 5/8 = 0000 if O.K.
FEB9 C3 ; RET ; Return to caller

CRC_CHECK ENDP

SUBROUTINE TO READ AN 8250 REGISTER. MAY ALSO BUMP ERROR REPORTER (BL) AND/OR REG DX (PORT ADDRESS) DEPENDING ON WHICH ENTRY POINT IS CHOSEN.
THIS SUBROUTINE WAS WRITTEN TO AVOID MULTIPLE USE OF 1/0 TIME DELAYS FOR THE 8250. IT WAS THE MOST EFFICIENT WAY TO INCLUDE THE DELAYS IN EVERY CASE, UPON RETURN, REG AL WILL CONTAIN THE CONTENTS OF PORT(0DX)!

RR1 PROC NEAR

XOR AL, AL
OUT DX, AL ; Disable all interrupts
INC BL ; Bump error reporter
INC DX ; Increm port addr
INC AH ; Read register
RET

RR2 PROC FAR

BUMP ERR REPORTER

RR3 PROC NEAR

IN AL, DX ; Read register
RET

TEST LOW

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, 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 CLEAR 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 OFEASH

FEA5 ORG OFEASH

ASSUME DS:DATA

TIMER_INTERRUPT FAR PROC NEAR

STI ; Interrupts back on

FEA5 FF B8 ; Push DS
FEA6 8E 1E ; Push AX
FEA7 50 ; Push DX
FEA9 13B R 8 ; Save machine state
FEAC FF 06 006C R ; INC TIMER_LOW, increment time
FEB0 70 T4 ; TEST DAY
FEB2 FF 06 006E R ; INC TIMER_HIGH, increment high word of time
FEB6 83 0E R ; CMP TIMER_HIGH,01BH, test for count equaling 24 hours
FEB8 75 15 ; JS DISKETTE_CTL
FEBD 81 E3 006C R 0080 ; CMP TIMER_LOW,00DH, test for count equaling 24 hours
FEC3 75 0D ; JS DISKETTE_CTL

A-106 ROM BIOS
FF53  ORG OFF53H
FF54  PROC PRINT_SCREEN
STI
FF55  PUSH DS
FF56  ASSUME CS:CODE, DS:XXDATA
FF57  PUSH AX
FF58  PUSH CX
FF59  MOV AX, 0000H
FF59  MOV BX, 0000H
FF60  MOV AX, 0000H
FF61  MOV BX, 0000H
FF62  MOV AX, 0000H
FF63  MOV BX, 0000H
FF64  MOV AX, 0000H
FF65  MOV BX, 0000H
FF66  MOV AX, 0000H
FF67  MOV BX, 0000H
FF68  MOV AX, 0000H
FF69  MOV BX, 0000H
FF70  MOV AX, 0000H
FF71  MOV BX, 0000H
FF72  MOV AX, 0000H
FF73  MOV BX, 0000H
FF74  MOV AX, 0000H
FF75  MOV BX, 0000H
FF76  MOV AX, 0000H
FF77  MOV BX, 0000H
FF78  MOV AX, 0000H
FF79  MOV BX, 0000H
FF80  MOV AX, 0000H
FF81  MOV BX, 0000H
FF82  MOV AX, 0000H
FF83  MOV BX, 0000H
FF84  MOV AX, 0000H
FF85  MOV BX, 0000H
FF86  MOV AX, 0000H
FF87  MOV BX, 0000H
FF88  MOV AX, 0000H
FF89  MOV BX, 0000H
FF90  MOV AX, 0000H
FF91  MOV BX, 0000H
FF92  MOV AX, 0000H
FF93  MOV BX, 0000H
FF94  MOV AX, 0000H
FF95  MOV BX, 0000H
FF96  MOV AX, 0000H
FF97  MOV BX, 0000H
FF98  MOV AX, 0000H
FF99  MOV BX, 0000H
FF9A  MOV AX, 0000H
FF9B  MOV AX, 0000H
FF9C  MOV AX, 0000H
FF9D  MOV AX, 0000H
FF9E  MOV AX, 0000H
FF9F  MOV AX, 0000H
FFA0  MOV AX, 0000H
FFA1  MOV AX, 0000H
FFA2  MOV AX, 0000H
FFA3  MOV AX, 0000H
FFA4  MOV AX, 0000H
FFA5  MOV AX, 0000H
FFA6  MOV AX, 0000H
FFA7  MOV AX, 0000H
FFA8  MOV AX, 0000H
FFA9  MOV AX, 0000H
FFAA  MOV AX, 0000H
FFAB  MOV AX, 0000H
FFAC  MOV AX, 0000H
FFAD  MOV AX, 0000H
FFAE  MOV AX, 0000H
FFAF  MOV AX, 0000H
FFB0  MOV AX, 0000H
FFB1  MOV AX, 0000H
FFB2  MOV AX, 0000H
FFB3  MOV AX, 0000H
FFB4  MOV AX, 0000H
FFB5  MOV AX, 0000H
FFB6  MOV AX, 0000H
FFB7  MOV AX, 0000H
FFB8  MOV AX, 0000H
FFB9  MOV AX, 0000H
FFBA  MOV AX, 0000H
FFBB  MOV AX, 0000H
FFBC  MOV AX, 0000H
FFBD  MOV AX, 0000H
FFBE  MOV AX, 0000H
FFBF  MOV AX, 0000H
FFC0  MOV AX, 0000H
FFC1  MOV AX, 0000H
FFC2  MOV AX, 0000H
FFC3  MOV AX, 0000H
FFC4  MOV AX, 0000H
FFC5  MOV AX, 0000H
FFC6  MOV AX, 0000H
FFC7  MOV AX, 0000H
FFC8  MOV AX, 0000H
FFC9  MOV AX, 0000H
FFCA  MOV AX, 0000H
FFCB  MOV AX, 0000H
FFCD  MOV AX, 0000H
FFCE  MOV AX, 0000H
FFCF  MOV AX, 0000H
FFD0  MOV AX, 0000H
FFD1  MOV AX, 0000H
FFD2  MOV AX, 0000H
FFD3  MOV AX, 0000H
FFD4  MOV AX, 0000H
FFD5  MOV AX, 0000H
FFD6  MOV AX, 0000H
FFD7  MOV AX, 0000H
FFD8  MOV AX, 0000H
FFD9  MOV AX, 0000H
FFDA  MOV AX, 0000H
FFDB  MOV AX, 0000H
FFDC  MOV AX, 0000H
FFDD  MOV AX, 0000H
FFDE  MOV AX, 0000H
FFDF  MOV AX, 0000H
FFE0  MOV AX, 0000H
FFE1  MOV AX, 0000H
FFE2  MOV AX, 0000H
FFE3  MOV AX, 0000H
FFE4  MOV AX, 0000H
FFE5  MOV AX, 0000H
FFE6  MOV AX, 0000H
FFE7  MOV AX, 0000H
FFE8  MOV AX, 0000H
FFE9  MOV AX, 0000H
FFEA  MOV AX, 0000H
FFEB  MOV AX, 0000H
FFEC  MOV AX, 0000H
FFED  MOV AX, 0000H
FFEE  MOV AX, 0000H
FFEF  MOV AX, 0000H
FFF0  MOV AX, 0000H
FFF1  MOV AX, 0000H
FFF2  MOV AX, 0000H
FFF3  MOV AX, 0000H
FFF4  MOV AX, 0000H
FFF5  MOV AX, 0000H
FFF6  MOV AX, 0000H
FFF7  MOV AX, 0000H
FFF8  MOV AX, 0000H
FFF9  MOV AX, 0000H
FFFA  MOV AX, 0000H
FFFB  MOV AX, 0000H
FFFC  MOV AX, 0000H
FFFD  MOV AX, 0000H
FFFE  MOV AX, 0000H
FFFF  MOV AX, 0000H

; DUMMY RETURN FOR ADDRESS COMPATIBILITY
---

; 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 WITHOUT INTERUPTS ENABLED.
; IF A SUBSEQUENT PRINT SCREEN KEY IS DEPRESSED DURING THE
; TIME THIS ROUTINE IS PRINTING IT WILL BE GROUNDED.
; 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
---

; AT THIS POINT WE KNOW THE COLUMNS/LINE ARE IN
; (AX) AND THE PAGE IF APPLICABLE IS IN (BX), THE
; STACK ; HAS DS, AX, BX, CX, DX PUSHED [AL] HAS VIDEO MODE
---

; MOV CL, AH, WILL MAKE USE OF (CX) REGISTER TO
; MOV CH, 25, control row & columns
; CALL CLRFLR, 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, will set cursor position to (0, 0)
; POP DX, will set cursor position to (0, 0)
; MOV CX, will read each cursor position from the screen
; AND PRINT.
---

; THE LOOP FROM PR10 TO THE INSTRUCTION PRIOR TO PR10
; IS THE LOOP TO READ EACH CURSOR POSITION FROM THE SCREEN
; AND PRINT.
---

A-108  ROM BIOS
; 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

BAS_ENT PROC FAR
ASSUME DS:ABS0

FFCB 2B C0  SUB  AX, AX
FFCD BE DB  MOV  DS, AX ; SET ADDRESSING
FFCF C7 06 0024 R 1937 R
    MOV  PTR INT_PTR+4, OFFSET NEW_INT_9
FFD5 A3 0060 R
    MOV  BASIC_PTR, AX ; SET INT IB=F600:0
FFDB C7 06 0062 R F600
    MOV  BASIC_PTR+2, OFFSET NEW_INT_9
FFDE CD 1B  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
;-------------------------------------------------------------

FFEO

INIT_TIMER PROC NEAR
TIM_CTL, AL ; OUTPUT INITIAL CONTROL WORD
FFE2 BA 0040
    MOV  DX, TIMER ; BASE PORT ADDR FOR TIMERS
FFE5 O2 04
    ADD  DL, AH ; ADD IN THE TIMER #
FFE7 BA C3
    MOV  AL, BL ; LOAD LSB
FFEB EE
    OUT  DX, AL
FFEA 52
    PUSH DX ; PAUSE
FFEB 5A
    POP  DX
FFEE EE
    MOV  AL, BH ; LOAD MSB
FFEF C3
    OUT  DX, AL

INIT_TIMER ENDP

; POWER ON RESET VECTOR
;-------------------------------------------------------------

FFFO

ORG OFFFOH

;----- POWER ON RESET

FFFO EA
    DB  OEAH ; JUMP FAR
FFF1 0043 R
    DW  OFFSET RESET
FFF3 F000
    DW  0F000H
FFF5 30 36 2F 30 31 2F 3B 33
    DB  '06/01/93' ; RELEASE MARKER
FFFD FF
    DB  OFFH ; FILLER
FFFE F0
    DB  OFDH ; SYSTEM IDENTIFIER

; CODE ENDS
END

ROM BIOS   A-109

Appendix A
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
System Board (Sheet 4 of 17)

B-6 System Board
System Board (Sheet 8 of 17)
System Board (Sheet 14 of 17)
Program Cartridge (Sheet 1 of 3)
Program Cartridge (Sheet 3 of 3)

B-22  Program Cartridge
Power Supply Board (Sheet 2 of 2)

B-24  Power Board
64KB Memory and Display Expansion (Sheet 2 of 4)

B-26 Memory Expansion
64KB Memory and Display Expansion (Sheet 4 of 4)

B-28 Memory Expansion
Diskette Drive Adapter (Sheet 1 of 6)
Diskette Drive Adapter (Sheet 3 of 6)
Internal Modem
Parallel Printer Attachment (Sheet 2 of 5)
Parallel Printer Attachment (Sheet 4 of 5)

B-40 Printer Attachment
Infra-Red Receiver Board
PC Compact Printer

Compact Printer  B-47/B-48
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.


## APPENDIX C: CHARACTERS, KEYSSTROKES, AND COLOR

<table>
<thead>
<tr>
<th>Value</th>
<th>As Characters</th>
<th>Color/Graphics Text Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hex Dec</td>
<td>Symbol</td>
<td>Keystrokes</td>
</tr>
<tr>
<td>00 0</td>
<td>Blank (Null)</td>
<td>Ctrl 2</td>
</tr>
<tr>
<td>01 1</td>
<td></td>
<td>Ctrl A</td>
</tr>
<tr>
<td>02 2</td>
<td></td>
<td>Ctrl B</td>
</tr>
<tr>
<td>03 3</td>
<td></td>
<td>Ctrl C</td>
</tr>
<tr>
<td>04 4</td>
<td></td>
<td>Ctrl D</td>
</tr>
<tr>
<td>05 5</td>
<td></td>
<td>Ctrl E</td>
</tr>
<tr>
<td>06 6</td>
<td></td>
<td>Ctrl F</td>
</tr>
<tr>
<td>07 7</td>
<td></td>
<td>Ctrl G</td>
</tr>
<tr>
<td>08 8</td>
<td></td>
<td>Ctrl H, Backspace, Shift Backspace</td>
</tr>
<tr>
<td>09 9</td>
<td></td>
<td>Ctrl I</td>
</tr>
<tr>
<td>0A 10</td>
<td></td>
<td>Ctrl J, Ctrl ↓</td>
</tr>
<tr>
<td>0B 11</td>
<td></td>
<td>Ctrl K</td>
</tr>
<tr>
<td>0C 12</td>
<td></td>
<td>Ctrl L</td>
</tr>
<tr>
<td>0D 13</td>
<td></td>
<td>Ctrl M, Shift ↓</td>
</tr>
<tr>
<td>0E 14</td>
<td></td>
<td>Ctrl N</td>
</tr>
<tr>
<td>0F 15</td>
<td></td>
<td>Ctrl O</td>
</tr>
<tr>
<td>10 16</td>
<td></td>
<td>Ctrl P</td>
</tr>
<tr>
<td>11 17</td>
<td></td>
<td>Ctrl Q</td>
</tr>
<tr>
<td>12 18</td>
<td></td>
<td>Ctrl R</td>
</tr>
<tr>
<td>13 19</td>
<td></td>
<td>Ctrl S</td>
</tr>
<tr>
<td>14 20</td>
<td></td>
<td>Ctrl T</td>
</tr>
<tr>
<td>15 21</td>
<td></td>
<td>Ctrl U</td>
</tr>
<tr>
<td>16 22</td>
<td></td>
<td>Ctrl V</td>
</tr>
<tr>
<td>17 23</td>
<td></td>
<td>Ctrl W</td>
</tr>
<tr>
<td>Hex</td>
<td>Dec</td>
<td>Symbol</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>--------</td>
</tr>
<tr>
<td>18</td>
<td>24</td>
<td>↑</td>
</tr>
<tr>
<td>19</td>
<td>25</td>
<td>↓</td>
</tr>
<tr>
<td>1A</td>
<td>26</td>
<td>←</td>
</tr>
<tr>
<td>1B</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>1C</td>
<td>28</td>
<td>L</td>
</tr>
<tr>
<td>1D</td>
<td>29</td>
<td>↔</td>
</tr>
<tr>
<td>1E</td>
<td>30</td>
<td>▲</td>
</tr>
<tr>
<td>1F</td>
<td>31</td>
<td>▼</td>
</tr>
<tr>
<td>20</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>33</td>
<td>!</td>
</tr>
<tr>
<td>22</td>
<td>34</td>
<td>&quot;</td>
</tr>
<tr>
<td>23</td>
<td>35</td>
<td>#</td>
</tr>
<tr>
<td>24</td>
<td>36</td>
<td>$</td>
</tr>
<tr>
<td>25</td>
<td>37</td>
<td>%</td>
</tr>
<tr>
<td>26</td>
<td>38</td>
<td>&amp;</td>
</tr>
<tr>
<td>27</td>
<td>39</td>
<td>'</td>
</tr>
<tr>
<td>28</td>
<td>40</td>
<td>(</td>
</tr>
<tr>
<td>29</td>
<td>41</td>
<td>)</td>
</tr>
<tr>
<td>2A</td>
<td>42</td>
<td>*</td>
</tr>
<tr>
<td>2B</td>
<td>43</td>
<td>+</td>
</tr>
<tr>
<td>2C</td>
<td>44</td>
<td>’</td>
</tr>
<tr>
<td>2D</td>
<td>45</td>
<td>—</td>
</tr>
<tr>
<td>2E</td>
<td>46</td>
<td>.</td>
</tr>
<tr>
<td>2F</td>
<td>47</td>
<td>/</td>
</tr>
<tr>
<td>30</td>
<td>48</td>
<td>0</td>
</tr>
<tr>
<td>31</td>
<td>49</td>
<td>1</td>
</tr>
<tr>
<td>32</td>
<td>50</td>
<td>2</td>
</tr>
<tr>
<td>33</td>
<td>51</td>
<td>3</td>
</tr>
</tbody>
</table>

**C-2 Characters, Keystrokes, and Color**
<table>
<thead>
<tr>
<th>Hex</th>
<th>Dec</th>
<th>Symbol</th>
<th>Keystrokes</th>
<th>Modes</th>
<th>Background</th>
<th>Foreground</th>
</tr>
</thead>
<tbody>
<tr>
<td>34</td>
<td>52</td>
<td>4</td>
<td>4</td>
<td>Note 3</td>
<td>Cyan</td>
<td>Red</td>
</tr>
<tr>
<td>35</td>
<td>53</td>
<td>5</td>
<td>5</td>
<td>Note 3</td>
<td>Cyan</td>
<td>Magenta</td>
</tr>
<tr>
<td>36</td>
<td>54</td>
<td>6</td>
<td>6</td>
<td>Note 3</td>
<td>Cyan</td>
<td>Brown</td>
</tr>
<tr>
<td>37</td>
<td>55</td>
<td>7</td>
<td>7</td>
<td>Note 3</td>
<td>Cyan</td>
<td>Light Grey</td>
</tr>
<tr>
<td>38</td>
<td>56</td>
<td>8</td>
<td>8</td>
<td>Note 3</td>
<td>Cyan</td>
<td>Dark Grey</td>
</tr>
<tr>
<td>39</td>
<td>57</td>
<td>9</td>
<td>9</td>
<td>Note 3</td>
<td>Cyan</td>
<td>Light Blue</td>
</tr>
<tr>
<td>3A</td>
<td>58</td>
<td>:</td>
<td>:</td>
<td>Shift</td>
<td>Cyan</td>
<td>Light Green</td>
</tr>
<tr>
<td>3B</td>
<td>59</td>
<td>;</td>
<td>;</td>
<td></td>
<td>Cyan</td>
<td>Light Cyan</td>
</tr>
<tr>
<td>3C</td>
<td>60</td>
<td>&lt;</td>
<td>&lt;</td>
<td>Shift</td>
<td>Cyan</td>
<td>Light Red</td>
</tr>
<tr>
<td>3D</td>
<td>61</td>
<td>=</td>
<td>=</td>
<td></td>
<td>Cyan</td>
<td>Light Magenta</td>
</tr>
<tr>
<td>3E</td>
<td>62</td>
<td>&gt;</td>
<td>&gt;</td>
<td>Shift</td>
<td>Cyan</td>
<td>Yellow</td>
</tr>
<tr>
<td>3F</td>
<td>63</td>
<td>?</td>
<td>?</td>
<td>Shift</td>
<td>Cyan</td>
<td>White</td>
</tr>
<tr>
<td>40</td>
<td>64</td>
<td>@</td>
<td>@</td>
<td>Shift</td>
<td>Red</td>
<td>Black</td>
</tr>
<tr>
<td>41</td>
<td>65</td>
<td>A</td>
<td>A</td>
<td>Note 4</td>
<td>Red</td>
<td>Blue</td>
</tr>
<tr>
<td>42</td>
<td>66</td>
<td>B</td>
<td>B</td>
<td>Note 4</td>
<td>Red</td>
<td>Green</td>
</tr>
<tr>
<td>43</td>
<td>67</td>
<td>C</td>
<td>C</td>
<td>Note 4</td>
<td>Red</td>
<td>Cyan</td>
</tr>
<tr>
<td>44</td>
<td>68</td>
<td>D</td>
<td>D</td>
<td>Note 4</td>
<td>Red</td>
<td>Red</td>
</tr>
<tr>
<td>45</td>
<td>69</td>
<td>E</td>
<td>E</td>
<td>Note 4</td>
<td>Red</td>
<td>Magenta</td>
</tr>
<tr>
<td>46</td>
<td>70</td>
<td>F</td>
<td>F</td>
<td>Note 4</td>
<td>Red</td>
<td>Brown</td>
</tr>
<tr>
<td>47</td>
<td>71</td>
<td>G</td>
<td>G</td>
<td>Note 4</td>
<td>Red</td>
<td>Light Grey</td>
</tr>
<tr>
<td>48</td>
<td>72</td>
<td>H</td>
<td>H</td>
<td>Note 4</td>
<td>Red</td>
<td>Dark Grey</td>
</tr>
<tr>
<td>49</td>
<td>73</td>
<td>I</td>
<td>I</td>
<td>Note 4</td>
<td>Red</td>
<td>Light Blue</td>
</tr>
<tr>
<td>4A</td>
<td>74</td>
<td>J</td>
<td>J</td>
<td>Note 4</td>
<td>Red</td>
<td>Light Green</td>
</tr>
<tr>
<td>4B</td>
<td>75</td>
<td>K</td>
<td>K</td>
<td>Note 4</td>
<td>Red</td>
<td>Light Cyan</td>
</tr>
<tr>
<td>4C</td>
<td>76</td>
<td>L</td>
<td>L</td>
<td>Note 4</td>
<td>Red</td>
<td>Light Red</td>
</tr>
<tr>
<td>4D</td>
<td>77</td>
<td>M</td>
<td>M</td>
<td>Note 4</td>
<td>Red</td>
<td>Light Magenta</td>
</tr>
<tr>
<td>4E</td>
<td>78</td>
<td>N</td>
<td>N</td>
<td>Note 4</td>
<td>Red</td>
<td>Yellow</td>
</tr>
<tr>
<td>4F</td>
<td>79</td>
<td>O</td>
<td>O</td>
<td>Note 4</td>
<td>Red</td>
<td>White</td>
</tr>
<tr>
<td>50</td>
<td>80</td>
<td>P</td>
<td>P</td>
<td>Note 4</td>
<td>Magenta</td>
<td>Black</td>
</tr>
<tr>
<td>51</td>
<td>81</td>
<td>Q</td>
<td>Q</td>
<td>Note 4</td>
<td>Magenta</td>
<td>Blue</td>
</tr>
<tr>
<td>52</td>
<td>82</td>
<td>R</td>
<td>R</td>
<td>Note 4</td>
<td>Magenta</td>
<td>Green</td>
</tr>
<tr>
<td>53</td>
<td>83</td>
<td>S</td>
<td>S</td>
<td>Note 4</td>
<td>Magenta</td>
<td>Cyan</td>
</tr>
<tr>
<td>54</td>
<td>84</td>
<td>T</td>
<td>T</td>
<td>Note 4</td>
<td>Magenta</td>
<td>Red</td>
</tr>
<tr>
<td>Value</td>
<td>As Characters</td>
<td>Color/Graphics Text Attributes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>--------------</td>
<td>-------------------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hex Dec</td>
<td>Symbol</td>
<td>Keystrokes</td>
<td>Modes</td>
<td>Background</td>
<td>Foreground</td>
<td></td>
</tr>
<tr>
<td>55 85</td>
<td>U</td>
<td>U</td>
<td>Note 4</td>
<td>Magenta</td>
<td>Magenta</td>
<td></td>
</tr>
<tr>
<td>56 86</td>
<td>V</td>
<td>V</td>
<td>Note 4</td>
<td>Magenta</td>
<td>Brown</td>
<td></td>
</tr>
<tr>
<td>57 57</td>
<td>W</td>
<td>W</td>
<td>Note 4</td>
<td>Magenta</td>
<td>Light Grey</td>
<td></td>
</tr>
<tr>
<td>58 88</td>
<td>X</td>
<td>X</td>
<td>Note 4</td>
<td>Magenta</td>
<td>Dark Grey</td>
<td></td>
</tr>
<tr>
<td>59 89</td>
<td>Y</td>
<td>Y</td>
<td>Note 4</td>
<td>Magenta</td>
<td>Light Blue</td>
<td></td>
</tr>
<tr>
<td>5A 90</td>
<td>Z</td>
<td>Z</td>
<td>Note 4</td>
<td>Magenta</td>
<td>Light Green</td>
<td></td>
</tr>
<tr>
<td>5B 91</td>
<td>[</td>
<td>[</td>
<td></td>
<td>Magenta</td>
<td>Light Cyan</td>
<td></td>
</tr>
<tr>
<td>5C 92</td>
<td>\</td>
<td>\</td>
<td></td>
<td>Magenta</td>
<td>Light Red</td>
<td></td>
</tr>
<tr>
<td>5D 93</td>
<td>]</td>
<td>]</td>
<td></td>
<td>Magenta</td>
<td>Light Magenta</td>
<td></td>
</tr>
<tr>
<td>5E 94</td>
<td>^</td>
<td>^</td>
<td>Shift</td>
<td>Magenta</td>
<td>Yellow</td>
<td></td>
</tr>
<tr>
<td>5F 95</td>
<td>−</td>
<td>−</td>
<td>Shift</td>
<td>Magenta</td>
<td>White</td>
<td></td>
</tr>
<tr>
<td>60 96</td>
<td>\</td>
<td>\</td>
<td></td>
<td>Yellow</td>
<td>Black</td>
<td></td>
</tr>
<tr>
<td>61 97</td>
<td>a</td>
<td>a</td>
<td>Note 5</td>
<td>Yellow</td>
<td>Blue</td>
<td></td>
</tr>
<tr>
<td>62 98</td>
<td>b</td>
<td>b</td>
<td>Note 5</td>
<td>Yellow</td>
<td>Green</td>
<td></td>
</tr>
<tr>
<td>63 99</td>
<td>c</td>
<td>c</td>
<td>Note 5</td>
<td>Yellow</td>
<td>Cyan</td>
<td></td>
</tr>
<tr>
<td>64 100</td>
<td>d</td>
<td>d</td>
<td>Note 5</td>
<td>Yellow</td>
<td>Red</td>
<td></td>
</tr>
<tr>
<td>65 101</td>
<td>e</td>
<td>e</td>
<td>Note 5</td>
<td>Yellow</td>
<td>Magenta</td>
<td></td>
</tr>
<tr>
<td>66 102</td>
<td>f</td>
<td>f</td>
<td>Note 5</td>
<td>Yellow</td>
<td>Brown</td>
<td></td>
</tr>
<tr>
<td>67 103</td>
<td>g</td>
<td>g</td>
<td>Note 5</td>
<td>Yellow</td>
<td>Light Grey</td>
<td></td>
</tr>
<tr>
<td>68 104</td>
<td>h</td>
<td>h</td>
<td>Note 5</td>
<td>Yellow</td>
<td>Dark Grey</td>
<td></td>
</tr>
<tr>
<td>69 105</td>
<td>i</td>
<td>i</td>
<td>Note 5</td>
<td>Yellow</td>
<td>Light Blue</td>
<td></td>
</tr>
<tr>
<td>6A 106</td>
<td>j</td>
<td>j</td>
<td>Note 5</td>
<td>Yellow</td>
<td>Light Green</td>
<td></td>
</tr>
<tr>
<td>6B 107</td>
<td>k</td>
<td>k</td>
<td>Note 5</td>
<td>Yellow</td>
<td>Light Cyan</td>
<td></td>
</tr>
<tr>
<td>6C 108</td>
<td>l</td>
<td>l</td>
<td>Note 5</td>
<td>Yellow</td>
<td>Light Red</td>
<td></td>
</tr>
<tr>
<td>6D 109</td>
<td>m</td>
<td>m</td>
<td>Note 5</td>
<td>Yellow</td>
<td>Light Magenta</td>
<td></td>
</tr>
<tr>
<td>6E 110</td>
<td>n</td>
<td>n</td>
<td>Note 5</td>
<td>Yellow</td>
<td>Yellow</td>
<td></td>
</tr>
<tr>
<td>6F 111</td>
<td>o</td>
<td>o</td>
<td>Note 5</td>
<td>Yellow</td>
<td>White</td>
<td></td>
</tr>
<tr>
<td>70 112</td>
<td>p</td>
<td>p</td>
<td>Note 5</td>
<td>White</td>
<td>Black</td>
<td></td>
</tr>
<tr>
<td>71 113</td>
<td>q</td>
<td>q</td>
<td>Note 5</td>
<td>White</td>
<td>Blue</td>
<td></td>
</tr>
<tr>
<td>72 114</td>
<td>r</td>
<td>r</td>
<td>Note 5</td>
<td>White</td>
<td>Green</td>
<td></td>
</tr>
<tr>
<td>73 115</td>
<td>s</td>
<td>s</td>
<td>Note 5</td>
<td>White</td>
<td>Cyan</td>
<td></td>
</tr>
<tr>
<td>74 116</td>
<td>f</td>
<td>f</td>
<td>Note 5</td>
<td>White</td>
<td>Red</td>
<td></td>
</tr>
<tr>
<td>75 117</td>
<td>u</td>
<td>u</td>
<td>Note 5</td>
<td>White</td>
<td>Magenta</td>
<td></td>
</tr>
<tr>
<td>76 118</td>
<td>v</td>
<td>v</td>
<td>Note 5</td>
<td>White</td>
<td>Brown</td>
<td></td>
</tr>
</tbody>
</table>

C-4 Characters, Keystrokes, and Color
<table>
<thead>
<tr>
<th>Value</th>
<th>As Characters</th>
<th>Color/Graphics Text Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hex</td>
<td>Dec</td>
<td>Symbol</td>
</tr>
<tr>
<td>77</td>
<td>119</td>
<td>w</td>
</tr>
<tr>
<td>78</td>
<td>120</td>
<td>x</td>
</tr>
<tr>
<td>79</td>
<td>121</td>
<td>y</td>
</tr>
<tr>
<td>7A</td>
<td>122</td>
<td>z</td>
</tr>
<tr>
<td>7B</td>
<td>123</td>
<td>{</td>
</tr>
<tr>
<td>7C</td>
<td>124</td>
<td>{</td>
</tr>
<tr>
<td>7D</td>
<td>125</td>
<td>~</td>
</tr>
<tr>
<td>7E</td>
<td>126</td>
<td>Δ</td>
</tr>
<tr>
<td>7F</td>
<td>127</td>
<td></td>
</tr>
</tbody>
</table>

* * * * 80 to FF Hex are Flashing if Blink is Enabled * * * *

<table>
<thead>
<tr>
<th>Value</th>
<th>As Characters</th>
<th>Color/Graphics Text Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hex</td>
<td>Dec</td>
<td>Symbol</td>
</tr>
<tr>
<td>80</td>
<td>128</td>
<td>Ç</td>
</tr>
<tr>
<td>81</td>
<td>129</td>
<td>û</td>
</tr>
<tr>
<td>82</td>
<td>130</td>
<td>é</td>
</tr>
<tr>
<td>83</td>
<td>131</td>
<td>â</td>
</tr>
<tr>
<td>84</td>
<td>132</td>
<td>ã</td>
</tr>
<tr>
<td>85</td>
<td>133</td>
<td>à</td>
</tr>
<tr>
<td>86</td>
<td>134</td>
<td>à</td>
</tr>
<tr>
<td>87</td>
<td>135</td>
<td>ç</td>
</tr>
<tr>
<td>88</td>
<td>136</td>
<td>è</td>
</tr>
<tr>
<td>89</td>
<td>137</td>
<td>ë</td>
</tr>
<tr>
<td>9A</td>
<td>138</td>
<td>è</td>
</tr>
<tr>
<td>9B</td>
<td>139</td>
<td>ï</td>
</tr>
<tr>
<td>9C</td>
<td>140</td>
<td>ï</td>
</tr>
<tr>
<td>9D</td>
<td>141</td>
<td>ï</td>
</tr>
<tr>
<td>9E</td>
<td>142</td>
<td>À</td>
</tr>
<tr>
<td>9F</td>
<td>143</td>
<td>À</td>
</tr>
<tr>
<td>90</td>
<td>144</td>
<td>É</td>
</tr>
<tr>
<td>91</td>
<td>145</td>
<td>æ</td>
</tr>
<tr>
<td>92</td>
<td>146</td>
<td>æ</td>
</tr>
<tr>
<td>93</td>
<td>147</td>
<td>ô</td>
</tr>
<tr>
<td>94</td>
<td>148</td>
<td>ô</td>
</tr>
<tr>
<td>95</td>
<td>149</td>
<td>ô</td>
</tr>
<tr>
<td>Value</td>
<td>As Characters</td>
<td>Color/Graphics Text Attributes</td>
</tr>
<tr>
<td>-------</td>
<td>--------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>Hex</td>
<td>Dec</td>
<td>Symbol</td>
</tr>
<tr>
<td>96</td>
<td>150</td>
<td>ū</td>
</tr>
<tr>
<td>97</td>
<td>151</td>
<td>ů</td>
</tr>
<tr>
<td>98</td>
<td>152</td>
<td>ų</td>
</tr>
<tr>
<td>99</td>
<td>153</td>
<td>ó</td>
</tr>
<tr>
<td>9A</td>
<td>154</td>
<td>ũ</td>
</tr>
<tr>
<td>9B</td>
<td>155</td>
<td>ç</td>
</tr>
<tr>
<td>9C</td>
<td>156</td>
<td>£</td>
</tr>
<tr>
<td>9D</td>
<td>157</td>
<td>¥</td>
</tr>
<tr>
<td>9E</td>
<td>158</td>
<td>Pt</td>
</tr>
<tr>
<td>9F</td>
<td>159</td>
<td>ŋ</td>
</tr>
<tr>
<td>A0</td>
<td>160</td>
<td>á</td>
</tr>
<tr>
<td>A1</td>
<td>161</td>
<td>í</td>
</tr>
<tr>
<td>A2</td>
<td>162</td>
<td>ó</td>
</tr>
<tr>
<td>A3</td>
<td>163</td>
<td>ú</td>
</tr>
<tr>
<td>A4</td>
<td>164</td>
<td>ñ</td>
</tr>
<tr>
<td>A5</td>
<td>165</td>
<td>Ñ</td>
</tr>
<tr>
<td>A6</td>
<td>166</td>
<td>a</td>
</tr>
<tr>
<td>A7</td>
<td>167</td>
<td>o</td>
</tr>
<tr>
<td>A8</td>
<td>168</td>
<td>ò</td>
</tr>
<tr>
<td>A9</td>
<td>169</td>
<td>—</td>
</tr>
<tr>
<td>AA</td>
<td>170</td>
<td>—</td>
</tr>
<tr>
<td>AB</td>
<td>171</td>
<td>%</td>
</tr>
<tr>
<td>AC</td>
<td>172</td>
<td>$</td>
</tr>
<tr>
<td>AD</td>
<td>173</td>
<td>i</td>
</tr>
<tr>
<td>AE</td>
<td>174</td>
<td>&lt;&lt;</td>
</tr>
<tr>
<td>AF</td>
<td>175</td>
<td>&gt;&gt;</td>
</tr>
<tr>
<td>B0</td>
<td>176</td>
<td>Â</td>
</tr>
<tr>
<td>B1</td>
<td>177</td>
<td>£</td>
</tr>
<tr>
<td>B2</td>
<td>178</td>
<td>¥</td>
</tr>
<tr>
<td>B3</td>
<td>179</td>
<td>£</td>
</tr>
<tr>
<td>B4</td>
<td>180</td>
<td>¥</td>
</tr>
<tr>
<td>B5</td>
<td>181</td>
<td>£</td>
</tr>
<tr>
<td>B6</td>
<td>182</td>
<td>£</td>
</tr>
</tbody>
</table>

C-6 Characters, Keystrokes, and Color
<table>
<thead>
<tr>
<th>Value</th>
<th>As Characters</th>
<th>Color/Graphics Text Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hex Dec</td>
<td>Symbol</td>
<td>Keystrokes</td>
</tr>
<tr>
<td>B7 183</td>
<td>Alt 183</td>
<td>Note 6</td>
</tr>
<tr>
<td>B8 184</td>
<td>Alt 184</td>
<td>Note 6</td>
</tr>
<tr>
<td>B9 185</td>
<td>Alt 185</td>
<td>Note 6</td>
</tr>
<tr>
<td>BA 186</td>
<td>Alt 186</td>
<td>Note 6</td>
</tr>
<tr>
<td>BB 187</td>
<td>Alt 187</td>
<td>Note 6</td>
</tr>
<tr>
<td>BC 188</td>
<td>Alt 188</td>
<td>Note 6</td>
</tr>
<tr>
<td>BD 189</td>
<td>Alt 189</td>
<td>Note 6</td>
</tr>
<tr>
<td>BE 190</td>
<td>Alt 190</td>
<td>Note 6</td>
</tr>
<tr>
<td>BF 191</td>
<td>Alt 191</td>
<td>Note 6</td>
</tr>
<tr>
<td>C0 192</td>
<td>Alt 192</td>
<td>Note 6</td>
</tr>
<tr>
<td>C1 193</td>
<td>Alt 193</td>
<td>Note 6</td>
</tr>
<tr>
<td>C2 194</td>
<td>Alt 194</td>
<td>Note 6</td>
</tr>
<tr>
<td>C3 195</td>
<td>Alt 195</td>
<td>Note 6</td>
</tr>
<tr>
<td>C4 196</td>
<td>Alt 196</td>
<td>Note 6</td>
</tr>
<tr>
<td>C5 197</td>
<td>Alt 197</td>
<td>Note 6</td>
</tr>
<tr>
<td>C6 198</td>
<td>Alt 198</td>
<td>Note 6</td>
</tr>
<tr>
<td>C7 199</td>
<td>Alt 199</td>
<td>Note 6</td>
</tr>
<tr>
<td>C8 200</td>
<td>Alt 200</td>
<td>Note 6</td>
</tr>
<tr>
<td>C9 201</td>
<td>Alt 201</td>
<td>Note 6</td>
</tr>
<tr>
<td>CA 202</td>
<td>Alt 202</td>
<td>Note 6</td>
</tr>
<tr>
<td>CB 203</td>
<td>Alt 203</td>
<td>Note 6</td>
</tr>
<tr>
<td>CC 204</td>
<td>Alt 204</td>
<td>Note 6</td>
</tr>
<tr>
<td>CD 205</td>
<td>Alt 205</td>
<td>Note 6</td>
</tr>
<tr>
<td>CE 206</td>
<td>Alt 206</td>
<td>Note 6</td>
</tr>
<tr>
<td>CF 207</td>
<td>Alt 207</td>
<td>Note 6</td>
</tr>
<tr>
<td>D0 208</td>
<td>Alt 208</td>
<td>Note 6</td>
</tr>
<tr>
<td>D1 209</td>
<td>Alt 209</td>
<td>Note 6</td>
</tr>
<tr>
<td>D2 210</td>
<td>Alt 210</td>
<td>Note 6</td>
</tr>
<tr>
<td>D3 211</td>
<td>Alt 211</td>
<td>Note 6</td>
</tr>
<tr>
<td>D4 212</td>
<td>Alt 212</td>
<td>Note 6</td>
</tr>
<tr>
<td>D5 213</td>
<td>Alt 213</td>
<td>Note 6</td>
</tr>
<tr>
<td>D6 214</td>
<td>Alt 214</td>
<td>Note 6</td>
</tr>
<tr>
<td>D7 215</td>
<td>Alt 215</td>
<td>Note 6</td>
</tr>
<tr>
<td>Value</td>
<td>As Characters</td>
<td>Color/Graphics Text Attributes</td>
</tr>
<tr>
<td>-------</td>
<td>---------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td><strong>Hex</strong></td>
<td><strong>Dec</strong></td>
<td><strong>Symbol</strong></td>
</tr>
<tr>
<td>D8</td>
<td>216</td>
<td></td>
</tr>
<tr>
<td>D9</td>
<td>217</td>
<td></td>
</tr>
<tr>
<td>DA</td>
<td>218</td>
<td></td>
</tr>
<tr>
<td>DB</td>
<td>219</td>
<td></td>
</tr>
<tr>
<td>DC</td>
<td>220</td>
<td></td>
</tr>
<tr>
<td>DD</td>
<td>221</td>
<td></td>
</tr>
<tr>
<td>DE</td>
<td>222</td>
<td></td>
</tr>
<tr>
<td>DF</td>
<td>223</td>
<td></td>
</tr>
<tr>
<td>E0</td>
<td>224</td>
<td>α</td>
</tr>
<tr>
<td>E1</td>
<td>225</td>
<td>β</td>
</tr>
<tr>
<td>E2</td>
<td>226</td>
<td>Γ</td>
</tr>
<tr>
<td>E3</td>
<td>227</td>
<td>π</td>
</tr>
<tr>
<td>E4</td>
<td>228</td>
<td>Σ</td>
</tr>
<tr>
<td>E5</td>
<td>229</td>
<td>σ</td>
</tr>
<tr>
<td>E6</td>
<td>230</td>
<td>μ</td>
</tr>
<tr>
<td>E7</td>
<td>231</td>
<td>τ</td>
</tr>
<tr>
<td>E8</td>
<td>232</td>
<td>Φ</td>
</tr>
<tr>
<td>E9</td>
<td>233</td>
<td>θ</td>
</tr>
<tr>
<td>EA</td>
<td>234</td>
<td>Ω</td>
</tr>
<tr>
<td>EB</td>
<td>235</td>
<td>δ</td>
</tr>
<tr>
<td>EC</td>
<td>236</td>
<td>∞</td>
</tr>
<tr>
<td>ED</td>
<td>237</td>
<td>φ</td>
</tr>
<tr>
<td>EE</td>
<td>238</td>
<td>ε</td>
</tr>
<tr>
<td>EF</td>
<td>239</td>
<td>∩</td>
</tr>
<tr>
<td>F0</td>
<td>240</td>
<td>≡</td>
</tr>
<tr>
<td>F1</td>
<td>241</td>
<td>±</td>
</tr>
<tr>
<td>F2</td>
<td>242</td>
<td>≥</td>
</tr>
<tr>
<td>F3</td>
<td>243</td>
<td>≤</td>
</tr>
<tr>
<td>F4</td>
<td>244</td>
<td>∫</td>
</tr>
<tr>
<td>F5</td>
<td>245</td>
<td>∫</td>
</tr>
<tr>
<td>F6</td>
<td>246</td>
<td>†</td>
</tr>
<tr>
<td>F7</td>
<td>247</td>
<td>≈</td>
</tr>
<tr>
<td>F8</td>
<td>248</td>
<td>O</td>
</tr>
</tbody>
</table>

**C-8 Characters, Keystrokes, and Color**
<table>
<thead>
<tr>
<th>Value</th>
<th>As Characters</th>
<th>Color/Graphics Text Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Hex</strong></td>
<td><strong>Dec</strong></td>
<td><strong>Symbol</strong></td>
</tr>
<tr>
<td>F9</td>
<td>249</td>
<td>⚫</td>
</tr>
<tr>
<td>FA</td>
<td>250</td>
<td>⋅</td>
</tr>
<tr>
<td>FB</td>
<td>251</td>
<td>√</td>
</tr>
<tr>
<td>FC</td>
<td>252</td>
<td>η</td>
</tr>
<tr>
<td>FD</td>
<td>253</td>
<td>2</td>
</tr>
<tr>
<td>FE</td>
<td>254</td>
<td>■</td>
</tr>
<tr>
<td>FF</td>
<td>255</td>
<td>BLANK</td>
</tr>
</tbody>
</table>
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 Prt Sc * 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 . Del 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).
<table>
<thead>
<tr>
<th>DECIMAL VALUE</th>
<th>0</th>
<th>16</th>
<th>32</th>
<th>48</th>
<th>64</th>
<th>80</th>
<th>96</th>
<th>112</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>!</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>#</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>&amp;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>(</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>,</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>=</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>/</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This is a character set (00-7F) quick reference table.
## Character Set (80-FF) Quick Reference

<table>
<thead>
<tr>
<th>DECIMAL VALUE</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
</tr>
<tr>
<td>128</td>
<td>Ç</td>
<td>É</td>
<td>á</td>
<td>Å</td>
<td>ô</td>
<td>à</td>
<td>ò</td>
<td>ç</td>
<td>é</td>
<td>à</td>
<td>Ô</td>
<td>à</td>
<td>ò</td>
<td>ç</td>
<td>é</td>
<td>à</td>
</tr>
<tr>
<td>144</td>
<td>Ü</td>
<td>ã</td>
<td>ï</td>
<td>Ò</td>
<td>Ó</td>
<td>Ò</td>
<td>Ó</td>
<td>Ü</td>
<td>ã</td>
<td>ï</td>
<td>Ò</td>
<td>Ó</td>
<td>Ò</td>
<td>Ó</td>
<td>Ü</td>
<td>ã</td>
</tr>
<tr>
<td>160</td>
<td>Ũ</td>
<td>Ñ</td>
<td>Ñ</td>
<td>Ñ</td>
<td>Ñ</td>
<td>Ñ</td>
<td>Ñ</td>
<td>Ũ</td>
<td>Ñ</td>
<td>Ñ</td>
<td>Ñ</td>
<td>Ñ</td>
<td>Ñ</td>
<td>Ñ</td>
<td>Ũ</td>
<td>Ñ</td>
</tr>
<tr>
<td>176</td>
<td>Å</td>
<td>Í</td>
<td>Í</td>
<td>Í</td>
<td>Í</td>
<td>Í</td>
<td>Í</td>
<td>Å</td>
<td>Í</td>
<td>Í</td>
<td>Í</td>
<td>Í</td>
<td>Í</td>
<td>Í</td>
<td>Å</td>
<td>Í</td>
</tr>
<tr>
<td>192</td>
<td>Ð</td>
<td>Ï</td>
<td>Ï</td>
<td>Ï</td>
<td>Ï</td>
<td>Ï</td>
<td>Ï</td>
<td>Ð</td>
<td>Ï</td>
<td>Ï</td>
<td>Ï</td>
<td>Ï</td>
<td>Ï</td>
<td>Ï</td>
<td>Ð</td>
<td>Ï</td>
</tr>
<tr>
<td>208</td>
<td>ß</td>
<td>Ý</td>
<td>Ý</td>
<td>Ý</td>
<td>Ý</td>
<td>Ý</td>
<td>Ý</td>
<td>ß</td>
<td>Ý</td>
<td>Ý</td>
<td>Ý</td>
<td>Ý</td>
<td>Ý</td>
<td>Ý</td>
<td>ß</td>
<td>Ý</td>
</tr>
<tr>
<td>224</td>
<td>Ç</td>
<td>É</td>
<td>á</td>
<td>Å</td>
<td>ô</td>
<td>à</td>
<td>ò</td>
<td>ç</td>
<td>é</td>
<td>à</td>
<td>Ô</td>
<td>à</td>
<td>ò</td>
<td>ç</td>
<td>é</td>
<td>à</td>
</tr>
<tr>
<td>240</td>
<td>Ü</td>
<td>ã</td>
<td>ï</td>
<td>Ò</td>
<td>Ó</td>
<td>Ò</td>
<td>Ó</td>
<td>Ü</td>
<td>ã</td>
<td>ï</td>
<td>Ò</td>
<td>Ó</td>
<td>Ò</td>
<td>Ó</td>
<td>Ü</td>
<td>ã</td>
</tr>
</tbody>
</table>

### C-12 Characters, Keystrokes, and Colors
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**

<table>
<thead>
<tr>
<th>Voltage</th>
<th>+5 Vdc Input</th>
<th>+12 Vdc Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nominal</td>
<td>+5 Vdc</td>
<td>+12 Vdc</td>
</tr>
</tbody>
</table>

**Ripple**

<table>
<thead>
<tr>
<th>Voltage</th>
<th>+5 Vdc Input</th>
<th>+12 Vdc Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ripple</td>
<td>100 mV</td>
<td>100 mV</td>
</tr>
<tr>
<td>0 to 50 kHz</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Tolerance**

<table>
<thead>
<tr>
<th>Voltage</th>
<th>+5 Vdc Input</th>
<th>+12 Vdc Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>Including Ripple</td>
<td>+/- 5%</td>
<td>+/- 5%</td>
</tr>
</tbody>
</table>
Standby Current

<table>
<thead>
<tr>
<th></th>
<th>+5 Vdc Input</th>
<th>+12 Vdc Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nominal</td>
<td>600 mA</td>
<td>400 mA</td>
</tr>
<tr>
<td>Worst Case</td>
<td>700 mA</td>
<td>500 mA</td>
</tr>
</tbody>
</table>

Operating Current

<table>
<thead>
<tr>
<th></th>
<th>+5 Vdc Input</th>
<th>+12 Vdc Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nominal</td>
<td>600 mA</td>
<td>900 mA</td>
</tr>
<tr>
<td>Worst Case</td>
<td>700 mA</td>
<td>2400 mA</td>
</tr>
</tbody>
</table>

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 (MFM)

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

D-4 Unit Specifications
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

D-6 Unit Specifications
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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>+ 5 Vdc Voltage</th>
<th>+ 12 Vdc Voltage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tolerance</td>
<td>+/- 5%</td>
<td>+/- 10%</td>
</tr>
<tr>
<td>Ripple</td>
<td>50 mV, P-P</td>
<td>50 mV, P-P</td>
</tr>
<tr>
<td>Maximum Current</td>
<td>300 mA</td>
<td>50 mA</td>
</tr>
<tr>
<td>Current Nominal</td>
<td>150 mA</td>
<td>25 mA</td>
</tr>
</tbody>
</table>

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 mm (6 ft)
Size     28 AWG

Signal Cable:

Length  1.89 m (6 ft)
Size     3 by 18 AWG

Electrical:

Voltage  110 Vac 60 Hz

D-8  Unit Specifications
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 alphanemic.

**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.

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, \(2^{20}\); \(1,048,576\) 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). \(10^6\); \(1,000,000\) in decimal notation. When referring to storage capacity, \(2^{20}\); \(1,048,576\) 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. (\(\mu s\)) 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 $P, Q, R, \ldots$ 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 $P, Q, R, \ldots$ 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.

programable read-only memory (PROM). Non-erasable programable 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 1.0. (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 test 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.


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.
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

Index-2
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
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
caracter codes, cordless keyboard  5-27
caracter set, compact printer  3-148, 3-149
caracter set 1  3-128
    description  3-109
caracter 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

4-Index
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

8-Index
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

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

10-Index
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 I 3-23

HLDA, I/O signal 2-27
horizontal drive frequency, color display 3-82

-HREQ, 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

12-Index
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 8048
  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

14-Index
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
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

16-Index
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
-R1, 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

20-Index
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

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-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
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: