LIMITED WARRANTY

The International Business Machines Corporation warrants this IBM Personal Computer Product to be in good working order for a period of 90 days from the date of purchase from IBM or an authorized IBM Personal Computer dealer. Should this Product fail to be in good working order at any time during this 90-day warranty period, IBM will, at its option, repair or replace this Product at no additional charge except as set forth below. Repair parts and replacement Products will be furnished on an exchange basis and will be either reconditioned or new. All replaced parts and Products become the property of IBM. This limited warranty does not include service to repair damage to the Product resulting from accident, disaster, misuse, abuse, or non-IBM modification of the Product.

Limited Warranty service may be obtained by delivering the Product during the 90-day warranty period to an authorized IBM Personal Computer dealer or IBM Service Center and providing proof of purchase date. If this Product is delivered by mail, you agree to insure the Product or assume the risk of loss or damage in transit, to prepay shipping charges to the warranty service location and to use the original shipping container or equivalent. Contact an authorized IBM Personal Computer dealer or write to IBM Personal Computer, Sales and Service, P.O. Box 1328-W, Boca Raton, Florida 33432, for further information.

ALL EXPRESS AND IMPLIED WARRANTIES FOR THIS PRODUCT INCLUDING THE WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE LIMITED IN DURATION TO A PERIOD OF 90 DAYS FROM THE DATE OF PURCHASE, AND NO WARRANTIES, WHETHER EXPRESS OR IMPLIED, WILL APPLY AFTER THIS PERIOD. SOME STATES DO NOT ALLOW LIMITATIONS ON HOW LONG AN IMPLIED WARRANTY LASTS, SO THE ABOVE LIMITATIONS MAY NOT APPLY TO YOU.

IF THIS PRODUCT IS NOT IN GOOD WORKING ORDER AS WARRANTED ABOVE, YOUR SOLE REMEDY SHALL BE REPAIR OR REPLACEMENT AS PROVIDED ABOVE. IN NO EVENT WILL IBM BE LIABLE TO YOU FOR ANY DAMAGES, INCLUDING ANY LOST PROFITS, LOST SAVINGS OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OF OR INABILITY TO USE SUCH PRODUCT, EVEN IF IBM OR AN AUTHORIZED IBM PERSONAL COMPUTER DEALER HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, OR FOR ANY CLAIM BY ANY OTHER PARTY.

SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES FOR CONSUMER PRODUCTS, SO THE ABOVE LIMITATIONS OR EXCLUSIONS MAY NOT APPLY TO YOU.

THIS WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS, AND YOU MAY ALSO HAVE OTHER RIGHTS WHICH MAY VARY FROM STATE TO STATE.
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.

Notice: As sold by the manufacturer, the IBM Prototype Card does not require certification under the FCC’s rules for Class B devices. The user is responsible for any interference to radio or TV reception which may be caused by a user-modified prototype card.

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.

Revised Edition (April 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 Corp., Personal Computer, P.O. Box 1328-C, Boca Raton, Florida 33432. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligations whatever.

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

The information in this publication is both introductory and for reference, and is intended for hardware and software designers, programmers, engineers, and interested persons who need to understand the design and operation of the computer.

You should be familiar with the use of the Personal Computer XT, and you should understand the concepts of computer architecture and programming.

This manual has two sections:

“Section 1: Hardware” describes each functional part of the system. This section also has specifications for power, timing, and interface. Programming considerations are supported by coding tables, command codes, and registers.

“Section 2: ROM BIOS and System Usage” describes the basic input/output system and its use. This section also contains the software interrupt listing, a BIOS memory map, descriptions of vectors with special meanings, and a set of low memory maps. In addition, keyboard encoding and usage is discussed.

The publication has seven appendixes:

Appendix A: ROM BIOS Listings
Appendix B: 8088 Assembly Instruction Set Reference
Appendix C: Of Characters, Keystrokes, and Color
Appendix D: Logic Diagrams
Appendix E: Specifications
Appendix F: Communications
Appendix G: Switch Settings

A glossary and bibliography are included.
Prerequisite Publication:

*Guide to Operations* for the IBM Personal Computer XT
Part Number 6936810

Suggested Reading:

*BASIC* for the IBM Personal Computer
Part Number 6025010

*Disk Operating System (DOS)* for the IBM Personal Computer
Part Number 6024061

*Hardware Maintenance and Service* for the IBM Personal
Computer XT
Part Number 6936809

*MACRO Assembler* for the IBM Personal Computer
Part Number 6024002

Related publications are listed in the bibliography.
# TABLE OF CONTENTS

## Section 1: Hardware

- IBM Personal Computer XT System Unit ........................................ 1-3
- IBM Personal Computer Math Coprocessor .................................... 1-25
- IBM Keyboard .............................................................................. 1-65
- IBM Expansion Unit ..................................................................... 1-71
- IBM 80 CPS Printers ................................................................... 1-81
- IBM Printer Adapter .................................................................... 1-107
- IBM Monochrome Display and Printer Adapter .............................. 1-113
- IBM Monochrome Display .......................................................... 1-121
- IBM Color/Graphics Display Adapter ........................................... 1-123
- IBM Color Display ...................................................................... 1-149
- IBM 5-¼” Diskette Drive Adapter ................................................ 1-151
- IBM 5-¼” Diskette Drive ............................................................... 1-175
- Diskettes ...................................................................................... 1-177
- IBM Fixed Disk Drive Adapter ...................................................... 1-179
- IBM 10MB Fixed Disk Drive ........................................................ 1-195
- IBM Memory Expansion Options .................................................. 1-197
- IBM Game Control Adapter ......................................................... 1-203
- IBM Prototype Card ................................................................... 1-209
- IBM Asynchronous Communications Adapter .............................. 1-215
- IBM Binary Synchronous Communications Adapter ..................... 1-245
- IBM Synchronous Data Link Control (SDLC) Communication Adapter .................................................. 1-265
- IBM Communications Adapter Cable .......................................... 1-295

## Section 2: ROM BIOS and System Usage

- ROM BIOS .................................................................................. 2-2
- Keyboard Encoding and Usage ..................................................... 2-11

## Appendix A: ROM BIOS Listings

- System BIOS ................................................................................ A-2
- Fixed Disk BIOS ......................................................................... A-85

## Appendix B: 8088 Assembly Instruction Set Reference

- ..................................................................................................... B-1
Appendix C: Of Characters, Keystrokes, and Colors .............................................. C-1

Appendix D: Logic Diagrams ................................................................. D-1

System Board ......................................................................................... D-2
Type 1 Keyboard .................................................................................... D-12
Type 2 Keyboard .................................................................................... D-14
Expansion Board ................................................................................... D-15
Extender Card ......................................................................................... D-16
Receiver Card ......................................................................................... D-19
Printer .................................................................................................... D-22
Printer Adapter ....................................................................................... D-25
Monochrome Display Adapter ............................................................... D-26
Color/Graphics Monitor Adapter ............................................................ D-36
Color Display .......................................................................................... D-42
Monochrome Display ............................................................................ D-44
5-⅜ Inch Diskette Drive Adapter ........................................................... D-45
5-⅜ Inch Diskette Drive – Type 1 ............................................................. D-49
5-⅜ Inch Diskette Drive – Type 2 ............................................................. D-52
Fixed Disk Drive Adapter ...................................................................... D-54
Fixed Disk Drive – Type 1 ....................................................................... D-60
Fixed Disk Drive – Type 2 ....................................................................... D-63
32K Memory Expansion Option ............................................................... D-66
64K Memory Expansion Option ............................................................... D-69
64/256K Memory Expansion Option ...................................................... D-72
Game Control Adapter .......................................................................... D-76
Prototype Card ....................................................................................... D-77
Asynchronous Communications Adapter ............................................... D-78
Binary Synchronous Communications Adapter ...................................... D-79
SDLC Communications Adapter ............................................................. D-81

Appendix E: Specifications ................................................................. E-1

Appendix F: Communications ............................................................. F-1

Appendix G: Switch Settings ................................................................. G-1

Glossary .................................................................................................. H-1

Index ...................................................................................................... I-1
INDEX TAB LISTING

Section 1: Hardware .............................................

Section 2: ROM BIOS and System Usage .................

Appendix A: ROM BIOS Listings .........................

Appendix B: 8088 Assembly Instruction ..................
Set Reference

Appendix C: Of Characters, Keystrokes, ..............
and Color

Appendix D: Logic Diagrams ..............................
Appendix E: Specifications

Appendix F: Communications

Appendix G: Switch Settings

Glossary

Bibliography

Index
Notes:
# SECTION 1: HARDWARE

<table>
<thead>
<tr>
<th>Product Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM Personal Computer XT System Unit</td>
<td>1-3</td>
</tr>
<tr>
<td>IBM Personal Computer Math Coprocessor</td>
<td>1-25</td>
</tr>
<tr>
<td>IBM Keyboard</td>
<td>1-65</td>
</tr>
<tr>
<td>IBM Expansion Unit</td>
<td>1-71</td>
</tr>
<tr>
<td>IBM 80 CPS Printers</td>
<td>1-81</td>
</tr>
<tr>
<td>IBM Printer Adapter</td>
<td>1-107</td>
</tr>
<tr>
<td>IBM Monochrome Display and Printer Adapter</td>
<td>1-113</td>
</tr>
<tr>
<td>IBM Monochrome Display</td>
<td>1-121</td>
</tr>
<tr>
<td>IBM Color/Graphics Display Adapter</td>
<td>1-123</td>
</tr>
<tr>
<td>IBM Color Display</td>
<td>1-149</td>
</tr>
<tr>
<td>IBM 5-¼” Diskette Drive Adapter</td>
<td>1-151</td>
</tr>
<tr>
<td>IBM 5-¼” Diskette Drive</td>
<td>1-175</td>
</tr>
<tr>
<td>Diskettes</td>
<td>1-177</td>
</tr>
<tr>
<td>IBM Fixed Disk Drive Adapter</td>
<td>1-179</td>
</tr>
<tr>
<td>IBM 10MB Fixed Disk Drive</td>
<td>1-195</td>
</tr>
<tr>
<td>IBM Memory Expansion Options</td>
<td>1-197</td>
</tr>
<tr>
<td>IBM Game Control Adapter</td>
<td>1-203</td>
</tr>
<tr>
<td>IBM Prototype Card</td>
<td>1-209</td>
</tr>
<tr>
<td>IBM Asynchronous Communications Adapter</td>
<td>1-215</td>
</tr>
<tr>
<td>IBM Binary Synchronous Communications Adapter</td>
<td>1-245</td>
</tr>
<tr>
<td>IBM Synchronous Data Link Control (SDLC) Communication Adapter</td>
<td>1-265</td>
</tr>
<tr>
<td>IBM Communications Adapter Cable</td>
<td>1-295</td>
</tr>
</tbody>
</table>
**System Block Diagram**

1-2 **System Unit**
The system unit is the center of your IBM Personal Computer XT system. The system unit contains the system board, which features eight expansion slots, the 8088 microprocessor, 40K of ROM (includes BASIC), 128K of base R/W memory, and an audio speaker. A power supply is located in the system unit to supply dc voltages to the system board and internal drives.

System Board

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

A dual-in-line package (DIP) switch (one eight-switch pack) is mounted on the board and can be read under program control. The DIP switch provides the system software with information about the installed options, how much storage the system board has, what type of display adapter is installed, what operation modes are desired when power is switched on (color or black-and-white, 80- or 40-character lines), and the number of diskette drives attached.

The system board consists of five functional areas: the processor subsystem and its support elements, the read-only memory (ROM) subsystem, the read/write (R/W) memory subsystem, integrated I/O adapters, and the I/O channel. All are described in this section.
The heart 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. Thus, the 8088 supports 16-bit operations, including multiply and divide, and supports 20 bits of addressing (1 megabyte of storage). It also operates in maximum mode, so a co-processor can be added as a feature. The processor operates 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.

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

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

Three of the four DMA channels are available on the I/O bus and support high-speed data transfers between I/O devices and memory without processor intervention. The fourth DMA channel is programmed to refresh the system dynamic memory. This is done by programming a channel of the timer-counter device to periodically request a dummy DMA transfer. This action creates a memory-read cycle, which is available to refresh dynamic storage both on the system board and in the system expansion slots. All DMA data transfers, except the refresh channel, take five processor clocks of 210 ns, or 1.05 μs if the processor-ready line is not deactivated. Refresh DMA cycles take four clocks or 840 ns.

The three programmable timer/counters are used by the system as follows: Channel 0 is used as a general-purpose timer providing a constant time base for implementing a time-of-day clock; Channel 1 is used to time and request refresh cycles from the DMA channel; and Channel 2 is used to support the tone generation for the audio speaker. Each channel has a minimum timing resolution of 1.05 μs.

Of the eight prioritized levels of interrupt, six are bussed to the system expansion slots for use by feature cards. Two levels are used on the system board. Level 0, the highest priority, is attached to Channel 0 of the timer/counter and provides a periodic
interrupt for the time-of-day clock. Level 1 is attached to the keyboard adapter circuits and receives an interrupt for each scan code sent by the keyboard. The non-maskable interrupt (NMI) of the 8088 is used to report memory parity errors.

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

The system board also has from 128K by 9 to 256K by 9 of R/W memory. A minimum system would have 128K of memory, with module sockets for an additional 128K. Memory greater than the system board’s maximum of 256K is obtained by adding memory cards in the expansion slots. The memory consists of dynamic 64K by 1 chips with an access time of 200 ns and a cycle time of 345 ns. All R/W memory is parity checked.

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

The keyboard interface is a 5-pin DIN connector on the system board that extends through the rear panel of the system unit.

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

The speaker drive circuit is capable of approximately 1/2 watt of power. The control circuits allow the speaker to be driven three different ways: 1.) a direct program control register bit may be toggled to generate a pulse train; 2.) the output from Channel 2 of the timer counter may be programmed to generate a waveform to the speaker; 3.) the clock input to the timer counter can be modulated with a program-controlled I/O register bit. All three methods may be performed simultaneously.
System Board Data Flow (Part 1 of 2)
System Board Data Flow (Part 2 of 2)
<table>
<thead>
<tr>
<th>Hex Range</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>000-00F</td>
<td>DMA Chip 8237A-5</td>
</tr>
<tr>
<td>020-021</td>
<td>Interrupt 8259A</td>
</tr>
<tr>
<td>040-043</td>
<td>Timer 8253-5</td>
</tr>
<tr>
<td>060-063</td>
<td>PPI 8255A-5</td>
</tr>
<tr>
<td>080-083</td>
<td>DMA Page Registers</td>
</tr>
<tr>
<td>0AX*</td>
<td>NMI Mask Register</td>
</tr>
<tr>
<td>0CX</td>
<td>Reserved</td>
</tr>
<tr>
<td>0EX</td>
<td>Reserved</td>
</tr>
<tr>
<td>200-20F</td>
<td>Game Control</td>
</tr>
<tr>
<td>210-217</td>
<td>Expansion Unit</td>
</tr>
<tr>
<td>220-24F</td>
<td>Reserved</td>
</tr>
<tr>
<td>278-27F</td>
<td>Reserved</td>
</tr>
<tr>
<td>2F0-2F7</td>
<td>Reserved</td>
</tr>
<tr>
<td>2F8-2FF</td>
<td>Asynchronous Communications (Secondary)</td>
</tr>
<tr>
<td>300-31F</td>
<td>Prototype Card</td>
</tr>
<tr>
<td>320-32F</td>
<td>Fixed Disk</td>
</tr>
<tr>
<td>378-37F</td>
<td>Printer</td>
</tr>
<tr>
<td>380-38C**</td>
<td>SDLC Communications</td>
</tr>
<tr>
<td>380-389**</td>
<td>Binary Synchronous Communications (Secondary)</td>
</tr>
<tr>
<td>3A0-3A9</td>
<td>Binary Synchronous Communications (Primary)</td>
</tr>
<tr>
<td>3B0-3BF</td>
<td>IBM Monochrome Display/Printer</td>
</tr>
<tr>
<td>3C0-3CF</td>
<td>Reserved</td>
</tr>
<tr>
<td>3D0-3DF</td>
<td>Color/Graphics</td>
</tr>
<tr>
<td>3E0-3E7</td>
<td>Reserved</td>
</tr>
<tr>
<td>3F0-3F7</td>
<td>Diskette</td>
</tr>
<tr>
<td>3F8-3FF</td>
<td>Asynchronous Communications (Primary)</td>
</tr>
</tbody>
</table>

* At power-on time, the Non Mask Interrupt into the 8088 is masked off. This mask bit can be set and reset through system software as follows:

  - Set mask: Write hex 80 to I/O Address hex A0 (enable NMI)
  - Clear mask: Write hex 00 to I/O Address hex A0 (disable NMI)

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

I/O Address Map

1-8 System Unit
<table>
<thead>
<tr>
<th>Number</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>NMI</td>
<td>Parity</td>
</tr>
<tr>
<td>0</td>
<td>Timer</td>
</tr>
<tr>
<td>1</td>
<td>Keyboard</td>
</tr>
<tr>
<td>2</td>
<td>Reserved</td>
</tr>
<tr>
<td>3</td>
<td>Asynchronous Communications (Secondary)</td>
</tr>
<tr>
<td></td>
<td>SDLC Communications</td>
</tr>
<tr>
<td></td>
<td>BSC (Secondary)</td>
</tr>
<tr>
<td>4</td>
<td>Asynchronous Communications (Primary)</td>
</tr>
<tr>
<td></td>
<td>SDLC Communications</td>
</tr>
<tr>
<td></td>
<td>BSC (Primary)</td>
</tr>
<tr>
<td>5</td>
<td>Fixed Disk</td>
</tr>
<tr>
<td>6</td>
<td>Diskette</td>
</tr>
<tr>
<td>7</td>
<td>Printer</td>
</tr>
</tbody>
</table>

8088 Hardware Interrupt Listing
### Hex Port Number

<table>
<thead>
<tr>
<th>Hex Port</th>
<th>PA0</th>
<th>Keyboard Scan Code</th>
<th>0</th>
<th>Diagnostic Outputs</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>2</td>
<td></td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>N</td>
<td>3</td>
<td></td>
<td>2</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>P</td>
<td>4</td>
<td></td>
<td>3</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>U</td>
<td>5</td>
<td></td>
<td>4</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>T</td>
<td>6</td>
<td></td>
<td>5</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>0060</td>
<td></td>
<td></td>
<td>6</td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>0061</td>
<td></td>
<td></td>
<td>7</td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>PBO</td>
<td></td>
<td>Timer 2 Gate</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Speaker</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Spare</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0062</td>
<td></td>
<td>Loop on POST</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Co-Processor</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Installed</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Planar RAM Size 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Planar RAM Size 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Loop on POST</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Co-Processor</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Installed</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Planar RAM Size 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Planar RAM Size 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0063</td>
<td>Command/Mode Register</td>
<td>Hex 99</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mode Register Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>7</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

### Command/Mode Register

<table>
<thead>
<tr>
<th>Mode Register Value</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>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sw—4</th>
<th>Sw—3</th>
</tr>
</thead>
<tbody>
<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>

<table>
<thead>
<tr>
<th>Sw—6</th>
<th>Sw—5</th>
</tr>
</thead>
<tbody>
<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>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sw—8</th>
<th>Sw—7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</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>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

8255A I/O Bit Map

1-10 System Unit
<table>
<thead>
<tr>
<th>Start Address</th>
<th>Hex</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00000</td>
<td>128-256K Read/Write Memory on System Board</td>
</tr>
<tr>
<td>16K</td>
<td>04000</td>
<td></td>
</tr>
<tr>
<td>32K</td>
<td>08000</td>
<td></td>
</tr>
<tr>
<td>48K</td>
<td>0C000</td>
<td></td>
</tr>
<tr>
<td>64K</td>
<td>10000</td>
<td></td>
</tr>
<tr>
<td>80K</td>
<td>14000</td>
<td></td>
</tr>
<tr>
<td>96K</td>
<td>18000</td>
<td></td>
</tr>
<tr>
<td>112K</td>
<td>1C000</td>
<td></td>
</tr>
<tr>
<td>128K</td>
<td>20000</td>
<td></td>
</tr>
<tr>
<td>144K</td>
<td>24000</td>
<td></td>
</tr>
<tr>
<td>160K</td>
<td>28000</td>
<td></td>
</tr>
<tr>
<td>176K</td>
<td>2C000</td>
<td></td>
</tr>
<tr>
<td>192K</td>
<td>30000</td>
<td></td>
</tr>
<tr>
<td>208K</td>
<td>34000</td>
<td></td>
</tr>
<tr>
<td>224K</td>
<td>38000</td>
<td></td>
</tr>
<tr>
<td>240K</td>
<td>3C000</td>
<td></td>
</tr>
<tr>
<td>256K</td>
<td>40000</td>
<td></td>
</tr>
<tr>
<td>272K</td>
<td>44000</td>
<td></td>
</tr>
<tr>
<td>288K</td>
<td>48000</td>
<td></td>
</tr>
<tr>
<td>304K</td>
<td>4C000</td>
<td></td>
</tr>
<tr>
<td>320K</td>
<td>50000</td>
<td></td>
</tr>
<tr>
<td>336K</td>
<td>54000</td>
<td></td>
</tr>
<tr>
<td>352K</td>
<td>58000</td>
<td></td>
</tr>
<tr>
<td>368K</td>
<td>5C000</td>
<td></td>
</tr>
<tr>
<td>384K</td>
<td>60000</td>
<td>384K R/W Memory Expansion in I/O Channel</td>
</tr>
<tr>
<td>400K</td>
<td>64000</td>
<td></td>
</tr>
<tr>
<td>416K</td>
<td>68000</td>
<td></td>
</tr>
<tr>
<td>432K</td>
<td>6C000</td>
<td></td>
</tr>
<tr>
<td>448K</td>
<td>70000</td>
<td></td>
</tr>
<tr>
<td>464K</td>
<td>74000</td>
<td></td>
</tr>
<tr>
<td>480K</td>
<td>78000</td>
<td></td>
</tr>
<tr>
<td>496K</td>
<td>7C000</td>
<td></td>
</tr>
<tr>
<td>512K</td>
<td>80000</td>
<td></td>
</tr>
<tr>
<td>528K</td>
<td>84000</td>
<td></td>
</tr>
<tr>
<td>544K</td>
<td>88000</td>
<td></td>
</tr>
<tr>
<td>560K</td>
<td>8C000</td>
<td></td>
</tr>
<tr>
<td>576K</td>
<td>90000</td>
<td></td>
</tr>
<tr>
<td>592K</td>
<td>94000</td>
<td></td>
</tr>
<tr>
<td>608K</td>
<td>98000</td>
<td></td>
</tr>
<tr>
<td>624K</td>
<td>9C000</td>
<td></td>
</tr>
</tbody>
</table>

System Memory Map (Part 1 of 2)
<table>
<thead>
<tr>
<th>Start Address</th>
<th>Hex</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>640K</td>
<td>A0000</td>
<td></td>
</tr>
<tr>
<td>656K</td>
<td>A4000</td>
<td>128K Reserved</td>
</tr>
<tr>
<td>672K</td>
<td>A8000</td>
<td></td>
</tr>
<tr>
<td>688K</td>
<td>AC000</td>
<td></td>
</tr>
<tr>
<td>704K</td>
<td>B0000</td>
<td>Monochrome</td>
</tr>
<tr>
<td>720K</td>
<td>B4000</td>
<td></td>
</tr>
<tr>
<td>736K</td>
<td>B8000</td>
<td>Color/Graphics</td>
</tr>
<tr>
<td>752K</td>
<td>BC000</td>
<td></td>
</tr>
<tr>
<td>768K</td>
<td>C0000</td>
<td></td>
</tr>
<tr>
<td>784K</td>
<td>C4000</td>
<td></td>
</tr>
<tr>
<td>800K</td>
<td>C8000</td>
<td>Fixed Disk Control</td>
</tr>
<tr>
<td>816K</td>
<td>CC000</td>
<td></td>
</tr>
<tr>
<td>832K</td>
<td>D0000</td>
<td>192K Read Only Memory</td>
</tr>
<tr>
<td>848K</td>
<td>D4000</td>
<td>Expansion and Control</td>
</tr>
<tr>
<td>864K</td>
<td>D8000</td>
<td></td>
</tr>
<tr>
<td>880K</td>
<td>DC000</td>
<td></td>
</tr>
<tr>
<td>896K</td>
<td>E0000</td>
<td></td>
</tr>
<tr>
<td>912K</td>
<td>E4000</td>
<td></td>
</tr>
<tr>
<td>928K</td>
<td>E8000</td>
<td></td>
</tr>
<tr>
<td>944K</td>
<td>EC000</td>
<td></td>
</tr>
<tr>
<td>960K</td>
<td>F0000</td>
<td>64K Base System ROM</td>
</tr>
<tr>
<td>976K</td>
<td>F4000</td>
<td>BIOS and BASIC</td>
</tr>
<tr>
<td>992K</td>
<td>F8000</td>
<td></td>
</tr>
<tr>
<td>1008K</td>
<td>FC000</td>
<td></td>
</tr>
</tbody>
</table>

System Memory Map (Part 2 of 2)
System Board Switch Settings

All system board switch settings for total system memory, number of diskette drives, and type of display are located in “Appendix G: Switch Settings.”
I/O Channel

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

The I/O channel contains an 8-bit, bidirectional data bus, 20 address lines, 6 levels of interrupt, control lines for memory and I/O read or write, clock and timing lines, 3 channels of DMA control lines, memory refresh timing control lines, a channel-check line, and power and ground for the adapters. Four voltage levels are provided for I/O cards: +5 Vdc, -5 Vdc, +12 Vdc, and -12 Vdc. These functions are provided in a 62-pin connector with 100-mil card tab spacing.

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

I/O devices are addressed using I/O mapped address space. The channel is designed so that 768 I/O device addresses are available to the I/O channel cards.

A ‘channel check’ line exists for reporting error conditions to the processor. Activating this line results in a Non-Maskable Interrupt (NMI) to the 8088 processor. Memory expansion options use this line to report parity errors.

The I/O channel is repowered to provide sufficient drive to power all eight (J1 through J8) expansion slots, assuming two low-power Schottky (LS) loads per slot. The IBM I/O adapters typically use only one load.

Timing requirements on slot J8 are much stricter than those on slots J1 through J7. Slot J8 also requires the card to provide a signal designating when the card is selected. The following pages describe the system board’s I/O channel.
I/O Channel Description

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

<table>
<thead>
<tr>
<th>Signal</th>
<th>I/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSC</td>
<td>O</td>
<td>Oscillator: High-speed clock with a 70-ns period (14.31818 MHz). It has a 50% duty cycle.</td>
</tr>
<tr>
<td>CLK</td>
<td>O</td>
<td>System clock: It is a divide-by-three of the oscillator and has a period of 210 ns (4.77 MHz). The clock has a 33% duty cycle.</td>
</tr>
<tr>
<td>RESET DRV</td>
<td>O</td>
<td>This line is used to reset or initialize system logic upon power-up or during a low line voltage outage. This signal is synchronized to the falling edge of clock and is active high.</td>
</tr>
<tr>
<td>A0-A19</td>
<td>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) and A19 is the most significant bit (MSB). These lines are generated by either the processor or DMA controller. They are active high.</td>
</tr>
<tr>
<td>D0-D7</td>
<td>I/O</td>
<td>Data Bits 0 to 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 are active high.</td>
</tr>
<tr>
<td>ALE</td>
<td>O</td>
<td>Address Latch Enable: This line is provided by the 8288 Bus Controller and is used on the system board to latch valid addresses from the processor. It is available to the I/O channel as an indicator of a valid processor address (when used with AEN). Processor addresses are latched with the failing edge of ALE.</td>
</tr>
<tr>
<td>Signal</td>
<td>I/O</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-----</td>
<td>-------------</td>
</tr>
<tr>
<td>I/O CH CK</td>
<td>I</td>
<td>-I/O Channel Check: This line provides the processor with parity (error) information on memory or devices in the I/O channel. When this signal is active low, a parity error is indicated.</td>
</tr>
<tr>
<td>I/O CH RDY</td>
<td>I</td>
<td>I/O Channel Ready: This line, normally high (ready), is pulled low (not ready) by a memory or I/O device to lengthen I/O or memory cycles. It allows slower devices to attach to the I/O channel with a minimum of difficulty. Any slow device using this line should drive it low immediately upon detecting a valid address and a read or write command. This line should never be held low longer than 10 clock cycles. Machine cycles (I/O or memory) are extended by an integral number of CLK cycles (210 ns).</td>
</tr>
<tr>
<td>IRQ2-IRQ7</td>
<td>I</td>
<td>Interrupt Request 2 to 7: These lines are used to signal the processor that an I/O device requires attention. They are prioritized with IRQ2 as the highest priority and IRQ7 as the lowest. An Interrupt Request is generated by raising an IRQ line (low to high) and holding it high until it is acknowledged by the processor (interrupt service routine).</td>
</tr>
<tr>
<td>IOR</td>
<td>O</td>
<td>-I/O Read Command: This command line instructs an I/O device to drive its data onto the data bus. It may be driven by the processor or the DMA controller. This signal is active low.</td>
</tr>
<tr>
<td>IOW</td>
<td>O</td>
<td>-I/O Write Command: This command line instructs an I/O device to read the data on the data bus. It may be driven by the processor or the DMA controller. This signal is active low.</td>
</tr>
<tr>
<td>Signal</td>
<td>I/O</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-----</td>
<td>-------------</td>
</tr>
<tr>
<td>MEMR</td>
<td>O</td>
<td>Memory Read Command: This command line instructs the memory to drive its data onto the data bus. It may be driven by the processor or the DMA controller. This signal is active low.</td>
</tr>
<tr>
<td>MEMW</td>
<td>O</td>
<td>Memory Write Command: This command line instructs the memory to store the data present on the data bus. It may be driven by the processor or the DMA controller. This signal is active low.</td>
</tr>
<tr>
<td>DRQ1-DRQ3</td>
<td>I</td>
<td>DMA Request 1 to 3: These lines are asynchronous channel requests used by peripheral devices to gain DMA service. They are prioritized with DRQ3 being the lowest and DRQ1 being the highest. A request is generated by bringing a DRQ line to an active level (high). A DRQ line must be held high until the corresponding DACK line goes active.</td>
</tr>
<tr>
<td>DACK0-DACK3</td>
<td>O</td>
<td>DMA Acknowledge 0 to 3: These lines are used to acknowledge DMA requests (DRQ1-DRQ3) and to refresh system dynamic memory (DACK0). They are active low.</td>
</tr>
<tr>
<td>AEN</td>
<td>O</td>
<td>Address Enable: This line is used to de-gate the processor and other devices from the I/O channel to allow DMA transfers to take place. When this line is active (high), the DMA controller has control of the address bus, data bus, read command lines (memory and I/O), and the write command lines (memory and I/O).</td>
</tr>
<tr>
<td>T/C</td>
<td>O</td>
<td>Terminal Count: This line provides a pulse when the terminal count for any DMA channel is reached. This signal is active high.</td>
</tr>
</tbody>
</table>
Signal | I/O Description
--- | ---
CARD SLCTD I | -Card Selected: This line is activated by cards in expansion slot J8. It signals the system board that the card has been selected and that appropriate drivers on the system board should be directed to either read from, or write to, expansion slot J8. Connectors J1 through J8 are tied together at this pin, but the system board does not use their signal. This line should be driven by an open collector device.

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

\[ +5 \text{ Vdc } \pm 5\%, \text{ located on 2 connector pins} \]
\[ -5 \text{ Vdc } \pm 10\%, \text{ located on 1 connector pin} \]
\[ +12 \text{ Vdc } \pm 5\%, \text{ located on 1 connector pin} \]
\[ -12 \text{ Vdc } \pm 10\%, \text{ located on 1 connector pin} \]
\[ \text{GND (Ground), located on 3 connector pins} \]
Speaker Interface

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

- An 8255A-5 PPI output bit. The address and bit are defined in the “I/O Address Map.”

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

Speaker Drive System Block Diagram

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

Speaker Tone Generation

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

<table>
<thead>
<tr>
<th>Pin</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Data</td>
</tr>
<tr>
<td>2</td>
<td>Key</td>
</tr>
<tr>
<td>3</td>
<td>Ground</td>
</tr>
<tr>
<td>4</td>
<td>+5 Volts</td>
</tr>
</tbody>
</table>

Speaker Connector

1-20 System Unit
Power Supply

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

The system board takes approximately 2 to 4 A of +5 Vdc, thus allowing approximately 11 A of +5 Vdc for the adapters in the system expansion slots. The +12 Vdc power level is designed to power the internal 5-1/4 inch diskette drive and the 10 M fixed disk drive. The −5 Vdc level is used for analog circuits in the diskette adapter phase lock loop. The +12 Vdc and −12 Vdc are used for powering the EIA drivers for the communications adapters. All four power levels are bussed across the eight system expansion slots.

The IBM Monochrome Display has its own power supply, receiving its ac power from the system unit power system. The ac output for the display is switched on and off with the power switch and is a nonstandard connector, so only the IBM Monochrome Display can be connected.
Operating Characteristics

The power supply is located at the right rear area of the system unit. It supplies operating voltages to the system board, and IBM Monochrome Display, and provides two separate connections for power to the 5-1/4 inch diskette drive and the fixed disk drive. The nominal power requirements and output voltages are listed in the following tables:

<table>
<thead>
<tr>
<th>Nominal Vac</th>
<th>Minimum Vac</th>
<th>Maximum Vac</th>
</tr>
</thead>
<tbody>
<tr>
<td>110</td>
<td>90</td>
<td>137</td>
</tr>
</tbody>
</table>

Input Requirements

Frequency: 50/60 Hz +/- 3 Hz

Current: 4.1 A max @ 90 Vac

<table>
<thead>
<tr>
<th>Voltage (Vdc)</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>+5.0</td>
<td>2.3</td>
<td>15.0</td>
</tr>
<tr>
<td>-5.0</td>
<td>0.0</td>
<td>0.3</td>
</tr>
<tr>
<td>+12.0</td>
<td>0.4</td>
<td>4.2</td>
</tr>
<tr>
<td>-12.0</td>
<td>0.0</td>
<td>0.25</td>
</tr>
</tbody>
</table>

Vdc Output

<table>
<thead>
<tr>
<th>Voltage (Vac)</th>
<th>Current (Amps)</th>
<th>Voltage Limits (Vac)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nominal</td>
<td>Minimum</td>
<td>Maximum</td>
</tr>
<tr>
<td>120</td>
<td>0.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Vac Output
Power Supply Connectors and Pin Assignments

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

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

 assignments

power supply connectors and pin
Over-Voltage/Over-Current Protection

<table>
<thead>
<tr>
<th>Voltage Nominal Vac</th>
<th>Type Protection</th>
<th>Rating Amps</th>
</tr>
</thead>
<tbody>
<tr>
<td>110</td>
<td>Fuse</td>
<td>5</td>
</tr>
</tbody>
</table>

Power On/Off Cycle: When the supply is turned off for a minimum of 1.0 second, and then turned on, the power-good signal will be regenerated.

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

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

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

The sense levels of the dc outputs are:

<table>
<thead>
<tr>
<th>Output (Vdc)</th>
<th>Minimum (Vdc)</th>
<th>Sense Voltage Nominal (Vdc)</th>
<th>Maximum (Vdc)</th>
</tr>
</thead>
<tbody>
<tr>
<td>+5</td>
<td>+4.5</td>
<td>+5.0</td>
<td>+5.5</td>
</tr>
<tr>
<td>-5</td>
<td>-4.3</td>
<td>-5.0</td>
<td>-5.5</td>
</tr>
<tr>
<td>+12</td>
<td>+10.8</td>
<td>+12.0</td>
<td>+13.2</td>
</tr>
<tr>
<td>-12</td>
<td>-10.2</td>
<td>-12.0</td>
<td>-13.2</td>
</tr>
</tbody>
</table>
The IBM Personal Computer Math Coprocessor enables the IBM Personal Computer to perform high speed arithmetic, logarithmic functions, and trigonometric operations with extreme accuracy.

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

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

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

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

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

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

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Bits</th>
<th>Significant Digits (Decimal)</th>
<th>Approximate Range (decimal)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word Integer</td>
<td>16</td>
<td>4</td>
<td>-32,768 ≤ X ≤ +32,767</td>
</tr>
<tr>
<td>Short Integer</td>
<td>32</td>
<td>9</td>
<td>-2x10^9 ≤ X ≤ +2x10^9</td>
</tr>
<tr>
<td>Long Integer</td>
<td>64</td>
<td>18</td>
<td>-9x10^18 ≤ X ≤ +9x10^18</td>
</tr>
<tr>
<td>Packed Decimal</td>
<td>80</td>
<td>18</td>
<td>-99...99 ≤ X ≤ +99...99 (18 digits)</td>
</tr>
<tr>
<td>Short Real*</td>
<td>32</td>
<td>6-7</td>
<td>8.43x10^-37 ≤</td>
</tr>
<tr>
<td>Long Real*</td>
<td>64</td>
<td>15-16</td>
<td>4.19x10^-307 ≤</td>
</tr>
<tr>
<td>Temporary Real</td>
<td>80</td>
<td>19</td>
<td>3.4x10^-4932 ≤</td>
</tr>
</tbody>
</table>

*The short and long real data types correspond to the single and double precision data types.

Data Types
Hardware Interface

The coprocessor utilizes the same clock generator and system bus interface components as the processor. The coprocessor is wired directly into the processor, as shown in the coprocessor interconnection diagram. The processor’s queue status lines (QS0 and QS1) enable the coprocessor to obtain and decode instructions simultaneously with the processor. The coprocessor’s busy signal informs the processor that it is executing; the processor’s WAIT instruction forces the processor to wait until the coprocessor is finished executing (WAIT for NOT BUSY).

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

1. Exception and Interrupt Enable bits of the control word are set to 1’s.
2. System board switch block 1 switch 2 set in the On position.
3. NMI Mask REG is set to zero.

At power-on time the NMI Mask REG is cleared to disable the NMI. Any software using the coprocessor’s interrupt capability must ensure that conditions 2 and 3 are never met during the operation of the software or an “Endless Wait” will occur. An “Endless Wait” will have the processor waiting for the “Not Busy” signal from the coprocessor while the coprocessor is waiting for the processor to interrupt.

Because a memory parity error may also cause an interrupt to the 8088 NMI line, the program should check that a parity error did not occur (by reading the 8255 port), then clear exceptions by executing the FNSAVE or the FNCLEX instruction. In most cases, the status word would be looked at, and the exception would be identified and acted upon.
The NMI Mask REG and the coprocessors interrupt are tied to the NMI line through the NMI interrupt logic. Minor conversions of software designed for use with an 8087 must be made before existing software will be compatible with the IBM Personal Computer Math Coprocessor.

Coprocessor Interconnection
Control Unit

The control unit (CU) of the coprocessor and the processor fetch all instructions at the same time, as well as every byte of the instruction stream at the same time. The simultaneous fetching allows the coprocessor to know what the processor is doing at all times. This is necessary to keep a coprocessor instruction from going unnoticed. Coprocessor instructions are mixed with processor instructions in a single data stream. To aid the coprocessor in tracking the processor, nine status lines are interconnected (QS0, QS1, and S0 through S6).

Coprocessor Block Diagram
Register Stack

Each of the eight registers in the coprocessor’s register stack is 80 bits wide, and each is divided into the “fields” shown in the figure below. The format in the figure below corresponds to the coprocessor’s temporary real data type that is used for all calculations.

The ST field in the status word identifies the current top-of-stack register. A load (“push”) operation decreases ST by 1 and loads a new value into the top register. A store operation stores the value from the current top register and then increases ST by 1. Thus, the coprocessor’s register stack grows “down” toward lower-addressed registers.

Instructions may address registers either implicitly or explicitly. Instructions that operate at the top of the stack, implicitly address the register pointed to by ST. The instruction, FSQRT, replaces the number at the top with its square root; this instruction takes no operands, because the top-of-stack register is implied as the operand. Other instructions specify the register that is to be used. Explicit register addressing is “top-relative.” The expression, ST, denotes the current stack top, and ST(i) refers to the ith register from the ST in the stack. If ST contains “binary 011” (register 3 is the top of the stack), the instruction, FADD ST,ST(2), would add registers 3 and 5.

Passing subroutine parameters to the register stack eliminates the need for the subroutine to know which registers actually contain the parameters. This allows different routines to call the same subroutine without having to observe a convention for passing parameters in dedicated registers. As long as the stack is not full, each routine simply loads the parameters to the stack and calls the subroutine.

Instructions may address registers either implicitly or explicitly. Instructions that operate at the top of the stack, implicitly address the register pointed to by ST. The instruction, FSQRT, replaces the number at the top with its square root; this instruction takes no operands, because the top-of-stack register is implied as the operand. Other instructions specify the register that is to be used. Explicit register addressing is “top-relative.” The expression, ST, denotes the current stack top, and ST(i) refers to the ith register from the ST in the stack. If ST contains “binary 011” (register 3 is the top of the stack), the instruction, FADD ST,ST(2), would add registers 3 and 5.

Passing subroutine parameters to the register stack eliminates the need for the subroutine to know which registers actually contain the parameters. This allows different routines to call the same subroutine without having to observe a convention for passing parameters in dedicated registers. As long as the stack is not full, each routine simply loads the parameters to the stack and calls the subroutine.

Instructions may address registers either implicitly or explicitly. Instructions that operate at the top of the stack, implicitly address the register pointed to by ST. The instruction, FSQRT, replaces the number at the top with its square root; this instruction takes no operands, because the top-of-stack register is implied as the operand. Other instructions specify the register that is to be used. Explicit register addressing is “top-relative.” The expression, ST, denotes the current stack top, and ST(i) refers to the ith register from the ST in the stack. If ST contains “binary 011” (register 3 is the top of the stack), the instruction, FADD ST,ST(2), would add registers 3 and 5.

Passing subroutine parameters to the register stack eliminates the need for the subroutine to know which registers actually contain the parameters. This allows different routines to call the same subroutine without having to observe a convention for passing parameters in dedicated registers. As long as the stack is not full, each routine simply loads the parameters to the stack and calls the subroutine.
Status Word

The status word reflects the overall condition of the coprocessor. It may be stored in memory with a coprocessor instruction then inspected with a processor code. The status word is divided into the fields shown in the figure below. Bit 15 (BUSY) indicates when the coprocessor is executing an instruction (B=1) or when it is idle (B=0).

Several instructions (for example, the comparison instructions) post their results to the condition code (bits 14 and 10 through 8 of the status word). The main use of the condition code is for conditional branching. This may be accomplished by first executing an instruction that sets the condition code, then storing the status word in memory, and then examining the condition code with processor instructions.

Bits 13 through 11 of the status word point to the coprocessor register that is the current stack top (ST). Bit 7 is the interrupt request field, and bits 5 through 0 are set to indicate that the numeric execution unit has detected an exception while executing the instruction.

```
 15  7  0
B C3 ST C2 C1 C0 IR PE UE OE ZE DE IE

Exception Flags (1 = Exception Has Occurred)
- Invalid Operation
- Denormalized Operand
- Zerodivide
- Overflow
- Underflow
- Precision (Reserved)
- Interrupt Request
- Condition Code
- Stack Top Pointer (1)
- Busy
```

(1) ST values:
- 000 = register 0 is stack top
- 001 = register 1 is stack top
- ...
- 111 = register 7 is stack top

Status Word Format
Control Word

The coprocessor provides several options that are selected by loading a control word register.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>IC</td>
</tr>
<tr>
<td>14</td>
<td>RC</td>
</tr>
<tr>
<td>13</td>
<td>PC</td>
</tr>
<tr>
<td>12</td>
<td>IEM</td>
</tr>
<tr>
<td>11</td>
<td>PM</td>
</tr>
<tr>
<td>10</td>
<td>UM</td>
</tr>
<tr>
<td>9</td>
<td>OM</td>
</tr>
<tr>
<td>8</td>
<td>ZM</td>
</tr>
<tr>
<td>7</td>
<td>DM</td>
</tr>
<tr>
<td>6</td>
<td>IM</td>
</tr>
</tbody>
</table>

Exception Masks (1 = Exception is Masked)
- Invalid Operation
- Denormalized Operand
- Zero
- divide
- Overflow
- Underflow
- Precision
- (Reserved)
- Interrupt-Enable Mask (1)
- Precision Control(2)
- Rounding Control(3)
- Infinity Control(4)
- (Reserved)

(1) Interrupt-Enable Mask:
0 = Interrupts Enabled
1 = Interrupts Disabled (Masked)

(2) Precision Control:
00 = 24 bits
01 = (reserved)
10 = 53 bits
11 = 64 bits

(3) Rounding Control:
00 = Round to Nearest or Even
01 = Round Down (toward 00)
10 = Round Up (toward 00)
11 = Chop (Truncate Toward Zero)

(4) Infinity Control:
0 = Projective
1 = Affine

Control Word Format
Tag Word

The tag word marks the content of each register, as shown in the Figure below. The main function of the tag word is to optimize the coprocessor’s performance under certain circumstances, and programmers ordinarily need not be concerned with it.

<table>
<thead>
<tr>
<th>15</th>
<th>7</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAG(7)</td>
<td>TAG(6)</td>
<td>TAG(5)</td>
</tr>
</tbody>
</table>

Tag values:
- 00 = Valid (Normal or Unnormal)
- 01 = Zero (True)
- 10 = Special (Not-A-Number, \( \infty \), or Denormal)
- 11 = Empty

Tag Word Format

Exception Pointers

The exception pointers in the figure below are provided for user-written exception handlers. When the coprocessor executes an instruction, the control unit saves the instruction address and the instruction opcode in the exception pointer registers. An exception handler subroutine can store these pointers in memory and determine which instruction caused the exception.

<table>
<thead>
<tr>
<th>OPERAND ADDRESS(^{(1)})</th>
</tr>
</thead>
<tbody>
<tr>
<td>INSTRUCTION OPCODE(^{(2)})</td>
</tr>
<tr>
<td>INSTRUCTION ADDRESS(^{(1)})</td>
</tr>
</tbody>
</table>

\(^{(1)}\) 20-bit physical address

\(^{(2)}\) 11 least significant bits of opcode: 5 most significant bits are always COPROCESSOR HOOK (11011B)

Exception Pointers Format
Number System

The figure below shows the basic coprocessor real number system on a real number line (decimal numbers are shown for clarity, although the coprocessor actually represents numbers in binary). The dots indicate the subset of real numbers the coprocessor can represent as data and final results of calculations. The coprocessor’s range is approximately $\pm 4.19 \times 10^{-307}$ to $\pm 1.67 \times 10^{308}$.

The coprocessor can represent a great many of, but not all, the real numbers in its range. There is always a "gap" between two adjacent coprocessor numbers, and the result of a calculation may fall within this space. When this occurs, the coprocessor rounds the true result to a number it can represent.

The coprocessor actually uses a number system that is a superset of that shown in the figure below. The internal format (called temporary real) extends the coprocessor’s range to about $\pm 3.4 \times 10^{-4932}$ to $\pm 1.2 \times 10^{4932}$, and its precision to about 19 (equivalent decimal) digits. This format is designed to provide extra range and precision for constants and intermediate results, and is not normally intended for data or final results.
Instruction Set

On the following pages are descriptions of the operation for the coprocessor’s 69 instructions.

An instruction has two basic types of operands – sources and destinations. A source operand simply supplies one of the “inputs” to an instruction; it is not altered by the instruction. A destination operand may also provide an input to an instruction. It is distinguished from a source operand, however, because its content can be altered when it receives the result produced by that operation; that is the destination is replaced by the result.

The operands of any instructions can be coded in more than one way. For example, FADD (add real) may be written without operands, with only a source, or with a destination and a source operand. The instruction descriptions use the simple convention of separating alternative operand forms with slashes; the slashes, however, are not coded. Consecutive slashes indicate there are no explicit operands. The operands for FADD are thus described as:

    // source/destination, source

This means that FADD may be written in any of three ways:

FADD

FADD source

FADD destination,source

It is important to bear in mind that memory operands may be coded with any of the processor’s memory addressing modes.
FABS

FABS (absolute value) changes the top stack element to its absolute value by making its sign positive.

<table>
<thead>
<tr>
<th>FABS (no operands)</th>
<th>Exceptions: 1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operands</strong></td>
<td><strong>Execution Clocks</strong></td>
</tr>
<tr>
<td>(no operands)</td>
<td>Typical 14</td>
</tr>
</tbody>
</table>

FADD

Addition

FADD // source/destination,source

FADDP // destination,source

FIADD // source

The addition instructions (add real, add real and pop, integer add) add the source and destination operands and return the sum to the destination. The operand at the stack top may be doubled by coding FADD ST,ST(0).

<table>
<thead>
<tr>
<th>FADD</th>
<th>Exceptions: I, D, O, U, P</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operands</strong></td>
<td><strong>Execution Clocks</strong></td>
</tr>
<tr>
<td>//ST,ST(i)</td>
<td>Typical 85</td>
</tr>
<tr>
<td>short-real</td>
<td>105+EA</td>
</tr>
<tr>
<td>long-real</td>
<td>110+EA</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FADDP</th>
<th>Exceptions: I, D, O, U, P</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operands</strong></td>
<td><strong>Execution Clocks</strong></td>
</tr>
<tr>
<td>ST(i),ST</td>
<td>Typical 90</td>
</tr>
</tbody>
</table>
FIADD

**Operands**

<table>
<thead>
<tr>
<th></th>
<th>Execution Clocks</th>
<th>Transfers 8088</th>
<th>Bytes</th>
<th>Coding Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Typical</td>
<td>Range</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>word-integer</td>
<td>120+EA</td>
<td>102-137+EA</td>
<td>2</td>
<td>FIADD DISTANCE_TRAVELLED</td>
</tr>
<tr>
<td>short-integer</td>
<td>125+EA</td>
<td>108-143+EA</td>
<td>4</td>
<td>FIADD PULSE_COUNT[SI]</td>
</tr>
</tbody>
</table>

**FBLD**

**FBLD Source**

FBLD (packed decimal BCD load) converts the content of the source operand from packed decimal to temporary real and loads (pushes) the result onto the stack. The packed decimal digits of the source are assumed to be in the range X '0-9H'.

**FBSTP**

**FBSTP destination**

FBSTP (packed decimal (BCD) store and pop) performs the inverse of FBLD, where the stack top is stored to the destination in the packed-decimal data type.
FCHS

FCHS (change sign) complements (reverses) the sign of the top stack element.

<table>
<thead>
<tr>
<th>FCHS (no operands)</th>
<th>Exceptions: I</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands</td>
<td>Execution Clocks</td>
</tr>
<tr>
<td></td>
<td>Typical</td>
</tr>
<tr>
<td>(no operands)</td>
<td>15</td>
</tr>
</tbody>
</table>

FCLEX/FNCLEX

FCLEX/FNCLEX (clear exceptions) clears all exception flags, the interrupt request flag, and the busy flag in the status word.

<table>
<thead>
<tr>
<th>FCLEX/FNCLEX (no operands)</th>
<th>Exceptions: None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands</td>
<td>Execution Clocks</td>
</tr>
<tr>
<td></td>
<td>Typical</td>
</tr>
<tr>
<td>(no operands)</td>
<td>5</td>
</tr>
</tbody>
</table>

FCOM

FCOM/ /source

FCOM (compare real) compares the stack top to the source operand. This results in the setting of the condition code bits.

<table>
<thead>
<tr>
<th>FCOM</th>
<th>Exceptions: I, D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands</td>
<td>Execution Clocks</td>
</tr>
<tr>
<td></td>
<td>Typical</td>
</tr>
<tr>
<td>//ST(i)</td>
<td>45</td>
</tr>
<tr>
<td>short-real</td>
<td>65+EA</td>
</tr>
<tr>
<td>long-real</td>
<td>70+EA</td>
</tr>
<tr>
<td>C3</td>
<td>C0</td>
</tr>
<tr>
<td>----</td>
<td>----</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>

NANS and ∞ (projective) cannot be compared and return C3=C0=1 as shown above.

**FCOMP**

**FCOMP/ /source**

FCOMP (compare real and pop) operates like FCOM, and in addition pops the stack.

<table>
<thead>
<tr>
<th>FCOMP</th>
<th>Exceptions: I, D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands</td>
<td>Execution Clocks</td>
</tr>
<tr>
<td></td>
<td>Typical</td>
</tr>
<tr>
<td>//ST(i)</td>
<td>47</td>
</tr>
<tr>
<td>short-real</td>
<td>68+EA</td>
</tr>
<tr>
<td>long-real</td>
<td>72+EA</td>
</tr>
</tbody>
</table>

**FCOMPP**

**FCOMPP/ /source**

FCOMPP (compare real and pop twice) operates like FCOM and, additionally, pops the stack twice, discarding both operands. The comparison is of the stack top to ST(1); no operands may be explicitly coded.

<table>
<thead>
<tr>
<th>FCOMPP (no operands)</th>
<th>Exceptions: I, D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands</td>
<td>Execution Clocks</td>
</tr>
<tr>
<td></td>
<td>Typical</td>
</tr>
<tr>
<td>(no operands)</td>
<td>50</td>
</tr>
</tbody>
</table>

Coprocessor 1-39
FDECSTP

FDECSTP (decrement stack pointer) subtracts 1 from ST, the stack top pointer in the status word.

<table>
<thead>
<tr>
<th>Operands</th>
<th>Execution Clocks</th>
<th>Transfers 8088</th>
<th>Bytes</th>
<th>Coding Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>(no operands)</td>
<td>Typical: 9</td>
<td>Range: 6-12</td>
<td>0</td>
<td>2 FDECSTP</td>
</tr>
</tbody>
</table>

FDISI/FNDISI

FDISI/FNDISI (disable interrupts) sets the interrupt enable mask in the control word.

<table>
<thead>
<tr>
<th>Operands</th>
<th>Execution Clocks</th>
<th>Transfers 8088</th>
<th>Bytes</th>
<th>Coding Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>(no operands)</td>
<td>Typical: 5</td>
<td>Range: 2-8</td>
<td>0</td>
<td>2 FDISI</td>
</tr>
</tbody>
</table>
FDIV

Normal division

FDIV / /source/ destination,source

FDIVP destination,source

FIDIV source

The normal division instructions (divide real, divide real and pop, integer divide) divide the destination by the source and return the quotient to the destination.

<table>
<thead>
<tr>
<th>Operands</th>
<th>Execution Clocks</th>
<th>Transfers 8088</th>
<th>Bytes</th>
<th>Coding Example</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Typical</td>
<td>Range</td>
<td></td>
<td></td>
</tr>
<tr>
<td>//ST(i),ST</td>
<td>198</td>
<td>193-203</td>
<td>0</td>
<td>FDIV</td>
</tr>
<tr>
<td>short-real</td>
<td>220+EA</td>
<td>215-225+EA</td>
<td>4</td>
<td>FDIV DISTANCE</td>
</tr>
<tr>
<td>long-real</td>
<td>225+EA</td>
<td>220-230+EA</td>
<td>8</td>
<td>FDIV ARC[DI]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Operands</th>
<th>Execution Clocks</th>
<th>Transfers 8088</th>
<th>Bytes</th>
<th>Coding Example</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Typical</td>
<td>Range</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ST(i),ST</td>
<td>202</td>
<td>197-207</td>
<td>0</td>
<td>FDIVP ST(4), ST</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Operands</th>
<th>Execution Clocks</th>
<th>Transfers 8088</th>
<th>Bytes</th>
<th>Coding Example</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Typical</td>
<td>Range</td>
<td></td>
<td></td>
</tr>
<tr>
<td>word-integer</td>
<td>230+EA</td>
<td>224-238+EA</td>
<td>2</td>
<td>FIDIV SURVEY.OBSERVATIONS</td>
</tr>
<tr>
<td>short-integer</td>
<td>236+EA</td>
<td>230-243+EA</td>
<td>4</td>
<td>FIDIV RELATIVE_ANGLE[DI]</td>
</tr>
</tbody>
</table>
FDIVR
Reversed Division

FDIVR / /source/ destination,source

FDIVRP destination,source

FIDIVR source

The reversed division instructions (divide real reversed, divide real reversed and pop, integer divide reversed) divide the source operand by the destination and return the quotient to the destination.

<table>
<thead>
<tr>
<th>FDIVR</th>
<th>Exceptions: I, D, Z, O, U, P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands</td>
<td>Execution Clocks</td>
</tr>
<tr>
<td></td>
<td>Typical</td>
</tr>
<tr>
<td>//ST,ST(i)/ST(i),ST</td>
<td>199</td>
</tr>
<tr>
<td>short-real</td>
<td>221+EA</td>
</tr>
<tr>
<td>long-real</td>
<td>226+EA</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FDIVRP</th>
<th>Exceptions: I, D, Z, O, U, P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands</td>
<td>Execution Clocks</td>
</tr>
<tr>
<td></td>
<td>Typical</td>
</tr>
<tr>
<td>ST(i),ST</td>
<td>203</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FIDIVR</th>
<th>Exceptions: I, D, Z, O, U, P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands</td>
<td>Execution Clocks</td>
</tr>
<tr>
<td></td>
<td>Typical</td>
</tr>
<tr>
<td>word-integer</td>
<td>230+EA</td>
</tr>
<tr>
<td>short-integer</td>
<td>237+EA</td>
</tr>
</tbody>
</table>

1-42 Coprocessor
FENI/FNENI

FENI/FNENI (enable interrupts) clear the interrupt enable mask in the control word.

<table>
<thead>
<tr>
<th>FENI/FNENI (no operands)</th>
<th>Exceptions: None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands</td>
<td>Execution Clocks</td>
</tr>
<tr>
<td>Typical</td>
<td>Range</td>
</tr>
<tr>
<td>(no operands)</td>
<td>5</td>
</tr>
</tbody>
</table>

FFREE

FFREE destination

FFREE (free register) changes the destination register’s tag to empty; the content of the register is not affected.

<table>
<thead>
<tr>
<th>FFREE</th>
<th>Exceptions: None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands</td>
<td>Execution Clocks</td>
</tr>
<tr>
<td>Typical</td>
<td>Range</td>
</tr>
<tr>
<td>ST(i)</td>
<td>11</td>
</tr>
</tbody>
</table>

FICOM

FICOM source

FICOM (integer compare) compares the source to the stack top.

<table>
<thead>
<tr>
<th>FICOM</th>
<th>Exceptions: I, D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands</td>
<td>Execution Clocks</td>
</tr>
<tr>
<td>Typical</td>
<td>Range</td>
</tr>
<tr>
<td>word-integer</td>
<td>80+EA</td>
</tr>
<tr>
<td>short-integer</td>
<td>85+EA</td>
</tr>
</tbody>
</table>
FICOMP

FICOMP source

FICOMP (integer compare and pop) operates the same as FICOM and additionally pops the stack.

<table>
<thead>
<tr>
<th>FICOMP</th>
<th>Exceptions: I, D</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operands</strong></td>
<td><strong>Execution Clocks</strong></td>
</tr>
<tr>
<td></td>
<td>Typical</td>
</tr>
<tr>
<td>word-integer</td>
<td>82+EA</td>
</tr>
<tr>
<td>short-integer</td>
<td>87+EA</td>
</tr>
</tbody>
</table>

FILD

FILD source

FILD (integer load) loads (pushes) the source onto the stack.

<table>
<thead>
<tr>
<th>FILD</th>
<th>Exceptions: I</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operands</strong></td>
<td><strong>Execution Clocks</strong></td>
</tr>
<tr>
<td></td>
<td>Typical</td>
</tr>
<tr>
<td>word-integer</td>
<td>50+EA</td>
</tr>
<tr>
<td>short-integer</td>
<td>56+EA</td>
</tr>
<tr>
<td>long-integer</td>
<td>64+EA</td>
</tr>
</tbody>
</table>

FINCSTP

FINCSTP (increment stack pointer) adds 1 to the stack top pointer (ST) in the status word.

<table>
<thead>
<tr>
<th>FINCSTP (no operands)</th>
<th>Exceptions: None</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operands</strong></td>
<td><strong>Execution Clocks</strong></td>
</tr>
<tr>
<td></td>
<td>Typical</td>
</tr>
<tr>
<td>(no operands)</td>
<td>9</td>
</tr>
</tbody>
</table>

1-44 Coprocessor
FINIT/FNINIT

FINIT/FNINIT (initialize processor) performs the functional equivalent of a hardware RESET.

### FINIT/FNINIT (no operands) Exceptions: None

<table>
<thead>
<tr>
<th>Operands</th>
<th>Execution Clocks</th>
<th>Transfers 8088</th>
<th>Bytes</th>
<th>Coding Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>(no operands)</td>
<td>5</td>
<td>2-8</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control Word</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Infinity Control</td>
<td>0</td>
<td>Projective</td>
</tr>
<tr>
<td>Rounding Control</td>
<td>00</td>
<td>Round to nearest</td>
</tr>
<tr>
<td>Precision Control</td>
<td>11</td>
<td>64 bits</td>
</tr>
<tr>
<td>Interrupt-enable Mask</td>
<td>1</td>
<td>Interrupts disabled</td>
</tr>
<tr>
<td>Exception Masks</td>
<td>111111</td>
<td>All exceptions masked</td>
</tr>
<tr>
<td>Status Word</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Busy</td>
<td>0</td>
<td>Not Busy</td>
</tr>
<tr>
<td>Condition Code</td>
<td>????</td>
<td>(Indeterminate)</td>
</tr>
<tr>
<td>Stack Top</td>
<td>000</td>
<td>Empty stack</td>
</tr>
<tr>
<td>Interrupt Request</td>
<td>0</td>
<td>No interrupt</td>
</tr>
<tr>
<td>Exception Flags</td>
<td>000000</td>
<td>No exceptions</td>
</tr>
<tr>
<td>Tag Word</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tags</td>
<td>11</td>
<td>Empty</td>
</tr>
<tr>
<td>Registers</td>
<td>N.C.</td>
<td>Not changed</td>
</tr>
<tr>
<td>Exception Pointers</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Instruction Code</td>
<td>N.C.</td>
<td>Not changed</td>
</tr>
<tr>
<td>Instruction Address</td>
<td>N.C.</td>
<td>Not changed</td>
</tr>
<tr>
<td>Operand Address</td>
<td>N.C.</td>
<td>Not changed</td>
</tr>
</tbody>
</table>
FIST

FIST destination

FIST (integer store) stores the stack top to the destination in the integer format.

<table>
<thead>
<tr>
<th>FIST</th>
<th>Exceptions: I, P</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operands</strong></td>
<td><strong>Execution Clocks</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Typical</strong></td>
</tr>
<tr>
<td>word-integer</td>
<td>86+EA</td>
</tr>
<tr>
<td>short-integer</td>
<td>88+EA</td>
</tr>
</tbody>
</table>

FISTP

FISTP destination

FISTP (integer store and pop) operates like FIST and also pops the stack following the transfer. The destination may be any of the binary integer data types.

<table>
<thead>
<tr>
<th>FISTP</th>
<th>Exceptions: I, P</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operands</strong></td>
<td><strong>Execution Clocks</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Typical</strong></td>
</tr>
<tr>
<td>word-integer</td>
<td>88+EA</td>
</tr>
<tr>
<td>short-integer</td>
<td>90+EA</td>
</tr>
<tr>
<td>long-integer</td>
<td>100+EA</td>
</tr>
</tbody>
</table>
**FLD**

**FLD source**

FLD (load real) loads (pushes) the source operand onto the top of the register stack.

<table>
<thead>
<tr>
<th>FLD</th>
<th>Exceptions: I, D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands</td>
<td>Execution Clocks</td>
</tr>
<tr>
<td>Typical</td>
<td>Range</td>
</tr>
<tr>
<td>ST(i) short-real</td>
<td>20</td>
</tr>
<tr>
<td>long-real</td>
<td>43+EA</td>
</tr>
<tr>
<td>temp-real</td>
<td>46+EA</td>
</tr>
<tr>
<td></td>
<td>57+EA</td>
</tr>
</tbody>
</table>

**FLDCW**

**FLDCW source**

FLDCW (load control word) replaces the current processor control word with the word defined by the source operand.

<table>
<thead>
<tr>
<th>FLDCW</th>
<th>Exceptions: None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands</td>
<td>Execution Clocks</td>
</tr>
<tr>
<td>Typical</td>
<td>Range</td>
</tr>
<tr>
<td>2-bytes</td>
<td>10+EA</td>
</tr>
</tbody>
</table>
**FLDENV**

**FLDENV source**

FLDENV (load environment) reloads the coprocessor environment from the memory area defined by the source operand.

<table>
<thead>
<tr>
<th>Operands</th>
<th>Execution Clocks</th>
<th>Transfers 8088</th>
<th>Bytes</th>
<th>Coding Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>14-bytes</td>
<td>40+EA</td>
<td>35-45+EA</td>
<td>14</td>
<td>FLDENV [BP+6]</td>
</tr>
</tbody>
</table>

**FLDLG2**

**FLDLG2** (load log base 10 of 2) loads (pushes) the value of LOG\(_{10}\)2 onto the stack.

<table>
<thead>
<tr>
<th>Operands</th>
<th>Execution Clocks</th>
<th>Transfers 8088</th>
<th>Bytes</th>
<th>Coding Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>(no operands)</td>
<td>21</td>
<td>18-24</td>
<td>0</td>
<td>FLDLG2</td>
</tr>
</tbody>
</table>

**FLDLN2**

**FLDLN2** (load log base e of 2) loads (pushes) the value of LOG\(_{e}\)2 onto the stack.

<table>
<thead>
<tr>
<th>Operands</th>
<th>Execution Clocks</th>
<th>Transfers 8088</th>
<th>Bytes</th>
<th>Coding Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>(no operands)</td>
<td>20</td>
<td>17-23</td>
<td>0</td>
<td>FLDLN2</td>
</tr>
</tbody>
</table>
FLDL2E

FLDL2E (load log base 2 of e) loads (pushes) the value $\log_2 e$ onto the stack.

<table>
<thead>
<tr>
<th>FLDL2E (no operands)</th>
<th>Exceptions: 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands</td>
<td>Execution Clocks</td>
</tr>
<tr>
<td>(no operands)</td>
<td>Typical</td>
</tr>
<tr>
<td></td>
<td>18</td>
</tr>
</tbody>
</table>

FLDL2T

FLDL2T (load log base 2 of 10) loads (pushes) the value of $\log_2 10$ onto the stack.

<table>
<thead>
<tr>
<th>FLDL2T (no operands)</th>
<th>Exceptions: 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands</td>
<td>Execution Clocks</td>
</tr>
<tr>
<td>(no operands)</td>
<td>Typical</td>
</tr>
<tr>
<td></td>
<td>19</td>
</tr>
</tbody>
</table>

FLDPI

FLDPI (load $\pi$) loads (pushes) $\pi$ onto the stack.

<table>
<thead>
<tr>
<th>FLDPI (no operands)</th>
<th>Exceptions: 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands</td>
<td>Execution Clocks</td>
</tr>
<tr>
<td>(no operands)</td>
<td>Typical</td>
</tr>
<tr>
<td></td>
<td>19</td>
</tr>
</tbody>
</table>
FLDZ

FLDZ (load zero) loads (pushes) +0.0 onto the stack.

<table>
<thead>
<tr>
<th>Operands</th>
<th>Execution Clocks</th>
<th>Transfers 8088</th>
<th>Bytes</th>
<th>Coding Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>(no operands)</td>
<td>14</td>
<td>11-17</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

FLD1

FLD1 (load one) loads (pushes) +1.0 onto the stack.

<table>
<thead>
<tr>
<th>Operands</th>
<th>Execution Clocks</th>
<th>Transfers 8088</th>
<th>Bytes</th>
<th>Coding Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>(no operands)</td>
<td>18</td>
<td>15-21</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

1-50 Coprocessor
**FMUL**

Multiplication

**FMUL / /source/destination,source**

**FMULP destination,source**

**FIMUL source**

The multiplication instructions (multiply real, multiply real and pop, integer multiply) multiply the source and destination operands and return the product to the destination. Coding FMUL ST,ST(0) square the content of the stack top.

<table>
<thead>
<tr>
<th>FMUL</th>
<th>Execution Clocks</th>
<th>Transfers</th>
<th>Bytes</th>
<th>Coding Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operands</strong></td>
<td><strong>Typical</strong></td>
<td><strong>Range</strong></td>
<td><strong>Coding</strong></td>
<td><strong>Example</strong></td>
</tr>
<tr>
<td>//ST(i),ST/ST,ST(0)</td>
<td>97</td>
<td>90-105</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>//ST(i),ST/ST,ST(0)</td>
<td>138</td>
<td>130-145</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>short-real</td>
<td>118+EA</td>
<td>110-125+EA</td>
<td>4</td>
<td>2-4</td>
</tr>
<tr>
<td>long-real</td>
<td>120+EA</td>
<td>112-126+EA</td>
<td>8</td>
<td>2-4</td>
</tr>
<tr>
<td>long-real</td>
<td>161+EA</td>
<td>154-168+EA</td>
<td>8</td>
<td>2-4</td>
</tr>
</tbody>
</table>

1 occurs when one or both operands is “short” - it has 40 trailing zeros in its fraction.

<table>
<thead>
<tr>
<th>FMULP</th>
<th>Execution Clocks</th>
<th>Transfers</th>
<th>Bytes</th>
<th>Coding Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operands</strong></td>
<td><strong>Typical</strong></td>
<td><strong>Range</strong></td>
<td><strong>Coding</strong></td>
<td><strong>Example</strong></td>
</tr>
<tr>
<td>ST(i),ST</td>
<td>100</td>
<td>94-108</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>ST(i),ST</td>
<td>142</td>
<td>134-148</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

1 occurs when one or both operands is “short” - it has 40 trailing zeros in its fraction.

<table>
<thead>
<tr>
<th>FIMUL</th>
<th>Execution Clocks</th>
<th>Transfers</th>
<th>Bytes</th>
<th>Coding Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operands</strong></td>
<td><strong>Typical</strong></td>
<td><strong>Range</strong></td>
<td><strong>Coding</strong></td>
<td><strong>Example</strong></td>
</tr>
<tr>
<td>word-integer</td>
<td>130+EA</td>
<td>124-138+EA</td>
<td>2</td>
<td>2-4</td>
</tr>
<tr>
<td>short-integer</td>
<td>136+EA</td>
<td>130-144+EA</td>
<td>4</td>
<td>2-4</td>
</tr>
</tbody>
</table>

Coprocessor 1-51
FNOP

FNOP (no operation) stores the stack to the stack top (FST ST,ST((0))) and thus effectively performs no operation.

<table>
<thead>
<tr>
<th>FNOP (no operands)</th>
<th>Exceptions: None</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operands</strong></td>
<td><strong>Execution Clocks</strong></td>
</tr>
<tr>
<td></td>
<td>Typical</td>
</tr>
<tr>
<td>(no operands)</td>
<td>13</td>
</tr>
</tbody>
</table>

FPATAN

FPATAN (partial arctangent) computes the function $\theta = \arctan (Y/X)$. $X$ is taken from the top stack element and $Y$ from ST(1). $Y$ and $X$ must observe the inequality $0 < Y < X < \infty$. The instruction pops the stack and returns $\theta$ to the (new) stack top, overwriting the $Y$ operand.

<table>
<thead>
<tr>
<th>FPATAN (no operands)</th>
<th>Exceptions: U, P (operands not checked)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operands</strong></td>
<td><strong>Execution Clocks</strong></td>
</tr>
<tr>
<td></td>
<td>Typical</td>
</tr>
<tr>
<td>(no operands)</td>
<td>650</td>
</tr>
</tbody>
</table>

FPREM

FPREM (partial remainder) performs modulo division on the top stack element by the next stack element, that is, ST(1) is the modulus.

<table>
<thead>
<tr>
<th>FPREM (no operands)</th>
<th>Exceptions: I, D, U</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operands</strong></td>
<td><strong>Execution Clocks</strong></td>
</tr>
<tr>
<td></td>
<td>Typical</td>
</tr>
<tr>
<td>(no operands)</td>
<td>125</td>
</tr>
</tbody>
</table>
FPTAN

FPTAN (partial tangent) computes the function $Y/X = \tan(\theta)$. $\theta$ is taken from the top stack element; it must lie in the range $0 < \theta < \pi/4$. The result of the operation is a ratio; $Y$ replaces $\theta$ in the stack and $X$ is pushed, becoming the new stack top.

<table>
<thead>
<tr>
<th>FPTAN</th>
<th>Exceptions: I, P (operands not checked)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operands</strong></td>
<td><strong>Execution Clocks</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Typical</strong></td>
</tr>
<tr>
<td>(no operands)</td>
<td>450</td>
</tr>
</tbody>
</table>

FRNDINT

FRNDINT (round to integer) rounds the top stack element to an integer.

<table>
<thead>
<tr>
<th>FRNDINT (no operands)</th>
<th>Exceptions: I, P</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operands</strong></td>
<td><strong>Execution Clocks</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Typical</strong></td>
</tr>
<tr>
<td>(no operands)</td>
<td>45</td>
</tr>
</tbody>
</table>

FRSTOR

FRSTOR source

FRSTOR (restore state) reloads the coprocessor from the 94-byte memory area defined by the source operand.

<table>
<thead>
<tr>
<th>FRSTOR</th>
<th>Exceptions: None</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operands</strong></td>
<td><strong>Execution Clocks</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Typical</strong></td>
</tr>
<tr>
<td>94-bytes</td>
<td>210+EA</td>
</tr>
</tbody>
</table>
FSAVE/FNSAVE

FSAVE/FNSAVE destination

FSAVE/FNSAVE (save state) writes the full coprocessor state – environment plus register stack – to the memory location defined by the destination operand.

<table>
<thead>
<tr>
<th>FSAVE/FNSAVE</th>
<th>Exceptions: None</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operands</strong></td>
<td><strong>Execution Clocks</strong></td>
</tr>
<tr>
<td>94-bytes</td>
<td>210+EA</td>
</tr>
</tbody>
</table>

FScale

FScale (scale) interprets the value contained in ST(1) as an integer, and adds this value to the exponent of the number in ST. This is equivalent to:

\[ ST \leftarrow ST \cdot 2^{ST(1)} \]

Thus, FSscale provides rapid multiplication or division by integral powers of 2.

<table>
<thead>
<tr>
<th>FSscale (no operands)</th>
<th>Exceptions: I, O, U</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operands</strong></td>
<td><strong>Execution Clocks</strong></td>
</tr>
<tr>
<td>(no operands)</td>
<td>35</td>
</tr>
</tbody>
</table>
FSQRT

FSQRT (square root) replaces the content of the top stack element with its square root.

Note: the square root of \(-0\) is defined to be \(-0\).

<table>
<thead>
<tr>
<th>FSQRT (no operands)</th>
<th>Exceptions: I, D, P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands</td>
<td>Execution Clocks</td>
</tr>
<tr>
<td></td>
<td>Typical</td>
</tr>
<tr>
<td>(no operands)</td>
<td>183</td>
</tr>
</tbody>
</table>

FST

FST destination

FST (store real) transfers the stack top to the destination, which may be another register on the stack or long real memory operand.

<table>
<thead>
<tr>
<th>FST</th>
<th>Exceptions: I, O, U, P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands</td>
<td>Execution Clocks</td>
</tr>
<tr>
<td></td>
<td>Typical</td>
</tr>
<tr>
<td>ST(i)</td>
<td>18</td>
</tr>
<tr>
<td>short-real</td>
<td>87+EA</td>
</tr>
<tr>
<td>long-real</td>
<td>100+EA</td>
</tr>
</tbody>
</table>
FSTCW/FNSTCW

FSTCW/FNSTCW destination

FSTCW/FNSTCW (store control word) writes the current processor control word to the memory location defined by the destination.

<table>
<thead>
<tr>
<th>FSTCW/FNSTCW</th>
<th>Exceptions: None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands</td>
<td>Execution Clocks</td>
</tr>
<tr>
<td></td>
<td>Typical</td>
</tr>
<tr>
<td>2-bytes</td>
<td>15+EA</td>
</tr>
</tbody>
</table>

FSTENV/FNSTENV

FSTENV/FNSTENV destination

FSTENV/FNSTENV (store environment) writes the coprocessor's basic status – control, status and tag words, and exception pointers – to the memory location defined by the destination operand.

<table>
<thead>
<tr>
<th>FSTENV/FNSTENV</th>
<th>Exceptions: None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands</td>
<td>Execution Clocks</td>
</tr>
<tr>
<td></td>
<td>Typical</td>
</tr>
<tr>
<td>14-bytes</td>
<td>45+EA</td>
</tr>
</tbody>
</table>

1-56   Coprocessor
FSTP

FSTP destination

FSTP (store real and pop) operates the same as FST, except that the stack is popped following the transfer.

<table>
<thead>
<tr>
<th>FSTP</th>
<th>Exceptions: I, O, U, P</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operands</strong></td>
<td><strong>Execution Clocks</strong></td>
</tr>
<tr>
<td></td>
<td>Typical</td>
</tr>
<tr>
<td>ST(i)</td>
<td>20</td>
</tr>
<tr>
<td>short-real</td>
<td>89+EA</td>
</tr>
<tr>
<td>long-real</td>
<td>102+EA</td>
</tr>
<tr>
<td>temp-real</td>
<td>55+EA</td>
</tr>
</tbody>
</table>

FSTSW/FNSTSW

FSTSW/FNSTSW destination

FSTSW/FNSTSW (store status word) writes the current value of the coprocessor status word to the destination operand in memory.

<table>
<thead>
<tr>
<th>FSTSW/FNSTSW</th>
<th>Exceptions: None</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operands</strong></td>
<td><strong>Execution Clocks</strong></td>
</tr>
<tr>
<td></td>
<td>Typical</td>
</tr>
<tr>
<td>2-bytes</td>
<td>14+EA</td>
</tr>
</tbody>
</table>
FSUB

Subtraction

FSUB / /source/destination,source

FSUBP destination,source

FISUB source

The normal subtraction instructions (subtract real, subtract real and pop, integer subtract) subtract the source operand from the destination and return the difference to the destination.

<table>
<thead>
<tr>
<th>FSUB</th>
<th>Exception: I, D, O, U, P</th>
<th>Operands</th>
<th>Execution Clocks</th>
<th>Transfers 8088</th>
<th>Bytes</th>
<th>Coding Example</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Typical</td>
<td>Range</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>/ST,ST(i)/ST(i),ST</td>
<td>85</td>
<td>70-100</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>short-real</td>
<td>105+EA</td>
<td>90-120+EA</td>
<td>4</td>
<td>2-4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>long-real</td>
<td>110+EA</td>
<td>95-125+EA</td>
<td>8</td>
<td>2-4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>FSUB ST,ST(2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>FSUB BASE_VALUE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>FSUB COORDINATE.X</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FSUBP</th>
<th>Exception: I, D, O, U, P</th>
<th>Operands</th>
<th>Execution Clocks</th>
<th>Transfers 8088</th>
<th>Bytes</th>
<th>Coding Example</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>ST(i),ST</td>
<td>90</td>
<td>75-105</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>FSUBP ST(2),ST</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FISUB</th>
<th>Exception: I, D, O, P</th>
<th>Operands</th>
<th>Execution Clocks</th>
<th>Transfers 8088</th>
<th>Bytes</th>
<th>Coding Example</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Typical</td>
<td>Range</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>word-integer</td>
<td>120+EA</td>
<td>102-137+EA</td>
<td>2</td>
<td>2-4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>short-integer</td>
<td>125+EA</td>
<td>108-143+EA</td>
<td>4</td>
<td>2-4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>FISUB BASE_FREQUENCY</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>FISUB TRAIN_SIZE[DI]</td>
</tr>
</tbody>
</table>
FSUBR

Reversed Subtraction

FSUBR / /source/destination,source

FSUBRP destination,source

FISUBR source

The reversed subtraction instructions (subtract real reversed, subtract real reversed and pop, integer subtract reversed) subtract the destination from the source and return the difference to the destination.

<table>
<thead>
<tr>
<th>FSUBR</th>
<th>Exceptions: I, D, O, U, P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands</td>
<td>Execution Clocks</td>
</tr>
<tr>
<td></td>
<td>Typical</td>
</tr>
<tr>
<td>//ST,ST(i)/ST(i),ST</td>
<td>87</td>
</tr>
<tr>
<td>short-real</td>
<td>105+EA</td>
</tr>
<tr>
<td>long-real</td>
<td>110+EA</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FSUBRP</th>
<th>Exceptions: I, D, O, U, P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands</td>
<td>Execution Clocks</td>
</tr>
<tr>
<td></td>
<td>Typical</td>
</tr>
<tr>
<td>ST(i),ST</td>
<td>90</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FISUBR</th>
<th>Exceptions: I, D, O, P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands</td>
<td>Execution Clocks</td>
</tr>
<tr>
<td></td>
<td>Typical</td>
</tr>
<tr>
<td>word-integer</td>
<td>120+EA</td>
</tr>
<tr>
<td>short-integer</td>
<td>125+EA</td>
</tr>
</tbody>
</table>
FTST

FTST (test) tests the top stack element by comparing it to zero. The result is posted to the condition codes.

<table>
<thead>
<tr>
<th>FTST (no operands)</th>
<th>Exceptions: I, D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands</td>
<td>Execution Clocks</td>
</tr>
<tr>
<td>(no operands)</td>
<td>Typical</td>
</tr>
<tr>
<td></td>
<td>42</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C3</th>
<th>C0</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>ST is positive and nonzero</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>ST is negative and nonzero</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>ST is zero (+ or -)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>ST is not comparable (that is, it is a NAN or projective ∞)</td>
</tr>
</tbody>
</table>

FWAIT

FWAIT (processor instruction)

FWAIT is not actually a coprocessor instruction, but an alternate mnemonic for the processor WAIT instruction. The FWAIT mnemonic should be coded whenever the programmer wants to synchronize the processor to the coprocessor, that is, to suspend further instruction decoding until the coprocessor has completed the current instruction.

<table>
<thead>
<tr>
<th>FWAIT (no operands)</th>
<th>Exceptions: Non (CPU instruction)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands</td>
<td>Execution Clocks</td>
</tr>
<tr>
<td>(no operands)</td>
<td>Typical</td>
</tr>
<tr>
<td></td>
<td>3+5n</td>
</tr>
</tbody>
</table>

1-60 Coprocessor
FXAM

FXAM (examine) reports the content of the top stack element as positive/negative and NAN/unnormal/denormal/normal/zero, or empty.

<table>
<thead>
<tr>
<th>FXAM</th>
<th>Exceptions: None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands</td>
<td>Execution Clocks</td>
</tr>
<tr>
<td>(no operands)</td>
<td>Typical</td>
</tr>
<tr>
<td></td>
<td>17</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Condition Code</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>C3 C2 C1 C0</td>
<td></td>
</tr>
<tr>
<td>0 0 0 0</td>
<td>+ Unnormal</td>
</tr>
<tr>
<td>0 0 0 1</td>
<td>+ NAN</td>
</tr>
<tr>
<td>0 0 1 0</td>
<td>- Unnormal</td>
</tr>
<tr>
<td>0 0 1 1</td>
<td>- NAN</td>
</tr>
<tr>
<td>0 1 0 0</td>
<td>+ Normal</td>
</tr>
<tr>
<td>0 1 0 1</td>
<td>+∞</td>
</tr>
<tr>
<td>0 1 1 0</td>
<td>- Normal</td>
</tr>
<tr>
<td>0 1 1 1</td>
<td>-∞</td>
</tr>
<tr>
<td>1 0 0 0</td>
<td>+0</td>
</tr>
<tr>
<td>1 0 0 1</td>
<td>Empty</td>
</tr>
<tr>
<td>1 0 1 0</td>
<td>-0</td>
</tr>
<tr>
<td>1 0 1 1</td>
<td>Empty</td>
</tr>
<tr>
<td>1 1 0 0</td>
<td>+ Denormal</td>
</tr>
<tr>
<td>1 1 0 1</td>
<td>Empty</td>
</tr>
<tr>
<td>1 1 1 0</td>
<td>- Denormal</td>
</tr>
<tr>
<td>1 1 1 1</td>
<td>Empty</td>
</tr>
</tbody>
</table>
**FXCH**

FXCH/ /destination

FXCH (exchange registers) swaps the contents of the destination and the stack top registers. If the destination is not coded explicitly, ST(1) is used.

<table>
<thead>
<tr>
<th>FXCH</th>
<th>Exceptions: I</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operands</strong></td>
<td><strong>Execution Clocks</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Typical</strong></td>
</tr>
<tr>
<td>//ST(i)</td>
<td>12</td>
</tr>
</tbody>
</table>

**FXTRACT**

FXTRACT (extract exponent and significant) "decomposes" the number in the stack top into two numbers that represent the actual value of the operand's exponent and significand fields contained in the stack top and ST(1).

<table>
<thead>
<tr>
<th>FXTRACT</th>
<th>Exceptions: I</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operands</strong></td>
<td><strong>Execution Clocks</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Typical</strong></td>
</tr>
<tr>
<td>(no operands)</td>
<td>50</td>
</tr>
</tbody>
</table>
FYL2X

FYL2X (Y log base 2 of X) calculates the function $Z = Y \cdot \log_2 X$. X is taken from the stack top and Y from ST(1). The operands must be in the ranges $0 < X < \infty$ and $-\infty < Y < +\infty$. The instruction pops the stack and returns Z at the (new) stack top, replacing the Y operand.

$\log_2 \cdot \log_2 X$

<table>
<thead>
<tr>
<th>FYL2X</th>
<th>Exceptions: P (operands not checked)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands</td>
<td>Execution Clocks</td>
</tr>
<tr>
<td>Typical</td>
<td>Range 8088</td>
</tr>
<tr>
<td>(no operands)</td>
<td>950</td>
</tr>
</tbody>
</table>

FYL2XP1

FYL2XP1 (Y log base 2 of (X + 1)) calculates the function $Z = Y \cdot \log_2 (X+1)$. X is taken from the stack top and must be in the range $0 < |X| < (1 - \sqrt{2}/2))$. Y is taken from ST(1) and must be in the range $-\infty < Y < \infty$. FYL2XP1 pops the stack and returns Z at the (new) stack top, replacing Y.

<table>
<thead>
<tr>
<th>FYL2XP1</th>
<th>Exceptions: P (operands not checked)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands</td>
<td>Execution Clocks</td>
</tr>
<tr>
<td>Typical</td>
<td>Range 8088</td>
</tr>
<tr>
<td>(no operands)</td>
<td>850</td>
</tr>
</tbody>
</table>
F2XM1

F2XM1 (2 to the X minus 1) calculates the function \( Y = 2^x - 1 \). X is taken from the stack top and must be in the range \( 0 < X < 0.5 \). The result Y replaces the stack top.

This instruction is designed to produce a very accurate result even when X is close to zero. To obtain \( Y = 2^x \), add 1 to the result delivered by F2XM1.

<table>
<thead>
<tr>
<th>F2XM1</th>
<th>Exceptions: U, P (operands not checked)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operands</strong></td>
<td><strong>Execution Clocks</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Typical</strong></td>
</tr>
<tr>
<td>(no operands)</td>
<td>500</td>
</tr>
</tbody>
</table>

I-64 Coprocessor
IBM Keyboard

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

The keyboard uses a capacitive technology with a microcomputer (Intel 8048) performing the keyboard scan function. The keyboard has three tilt positions for operator comfort (5-, 7-, or 15-degree tilt orientations).

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

The keyboard interface is defined so that system software has maximum flexibility in defining certain keyboard operations. This is accomplished by having the keyboard return scan codes rather than American Standard Code for Information Interchange (ASCII) codes. In addition, all keys are typematic and generate both a make and a break scan code. For example, key 1 produces scan code hex 01 on make and code hex 81 on break. Break codes are formed by adding hex 80 to make codes. The keyboard I/O driver can define keyboard keys as shift keys or typematic, as required by the application.
The microcomputer (Intel 8048) in the keyboard performs several functions, including a power-on self-test when requested by the system unit. This test checks the microcomputer ROM, tests memory, and checks for stuck keys. Additional functions are: keyboard scanning, buffering of up to 16 key scan codes, maintaining bidirectional serial communications with the system unit, and executing the hand-shake protocol required by each scan-code transfer.

The following pages have figures that show the keyboard, the scan codes, and the keyboard interface connector specifications.
Keyboard Interface Block Diagram

Keyboard Clock
Keyboard Data
Reset
GND
+5 V

PCLK

Reset
Note: Nomenclature is on both the top and front face of the keybutton as shown. The number to the upper left designates the button position.
<table>
<thead>
<tr>
<th>Key Position</th>
<th>Scan Code in Hex</th>
<th>Key Position</th>
<th>Scan Code in Hex</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>01</td>
<td>43</td>
<td>2B</td>
</tr>
<tr>
<td>2</td>
<td>02</td>
<td>44</td>
<td>2C</td>
</tr>
<tr>
<td>3</td>
<td>03</td>
<td>45</td>
<td>2D</td>
</tr>
<tr>
<td>4</td>
<td>04</td>
<td>46</td>
<td>2E</td>
</tr>
<tr>
<td>5</td>
<td>05</td>
<td>47</td>
<td>2F</td>
</tr>
<tr>
<td>6</td>
<td>06</td>
<td>48</td>
<td>30</td>
</tr>
<tr>
<td>7</td>
<td>07</td>
<td>49</td>
<td>31</td>
</tr>
<tr>
<td>8</td>
<td>08</td>
<td>50</td>
<td>32</td>
</tr>
<tr>
<td>9</td>
<td>09</td>
<td>51</td>
<td>33</td>
</tr>
<tr>
<td>10</td>
<td>0A</td>
<td>52</td>
<td>34</td>
</tr>
<tr>
<td>11</td>
<td>0B</td>
<td>53</td>
<td>35</td>
</tr>
<tr>
<td>12</td>
<td>0C</td>
<td>54</td>
<td>36</td>
</tr>
<tr>
<td>13</td>
<td>0D</td>
<td>55</td>
<td>37</td>
</tr>
<tr>
<td>14</td>
<td>0E</td>
<td>56</td>
<td>38</td>
</tr>
<tr>
<td>15</td>
<td>0F</td>
<td>57</td>
<td>39</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>58</td>
<td>3A</td>
</tr>
<tr>
<td>17</td>
<td>11</td>
<td>59</td>
<td>3B</td>
</tr>
<tr>
<td>18</td>
<td>12</td>
<td>60</td>
<td>3C</td>
</tr>
<tr>
<td>19</td>
<td>13</td>
<td>61</td>
<td>3D</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>62</td>
<td>3E</td>
</tr>
<tr>
<td>21</td>
<td>15</td>
<td>63</td>
<td>3F</td>
</tr>
<tr>
<td>22</td>
<td>16</td>
<td>64</td>
<td>40</td>
</tr>
<tr>
<td>23</td>
<td>17</td>
<td>65</td>
<td>41</td>
</tr>
<tr>
<td>24</td>
<td>18</td>
<td>66</td>
<td>42</td>
</tr>
<tr>
<td>25</td>
<td>19</td>
<td>67</td>
<td>43</td>
</tr>
<tr>
<td>26</td>
<td>1A</td>
<td>68</td>
<td>44</td>
</tr>
<tr>
<td>27</td>
<td>1B</td>
<td>69</td>
<td>45</td>
</tr>
<tr>
<td>28</td>
<td>1C</td>
<td>70</td>
<td>46</td>
</tr>
<tr>
<td>29</td>
<td>1D</td>
<td>71</td>
<td>47</td>
</tr>
<tr>
<td>30</td>
<td>1E</td>
<td>72</td>
<td>48</td>
</tr>
<tr>
<td>31</td>
<td>1F</td>
<td>73</td>
<td>49</td>
</tr>
<tr>
<td>32</td>
<td>20</td>
<td>74</td>
<td>4A</td>
</tr>
<tr>
<td>33</td>
<td>21</td>
<td>75</td>
<td>4B</td>
</tr>
<tr>
<td>34</td>
<td>22</td>
<td>76</td>
<td>4C</td>
</tr>
<tr>
<td>35</td>
<td>23</td>
<td>77</td>
<td>4D</td>
</tr>
<tr>
<td>36</td>
<td>24</td>
<td>78</td>
<td>4E</td>
</tr>
<tr>
<td>37</td>
<td>25</td>
<td>79</td>
<td>4F</td>
</tr>
<tr>
<td>38</td>
<td>26</td>
<td>80</td>
<td>50</td>
</tr>
<tr>
<td>39</td>
<td>27</td>
<td>81</td>
<td>51</td>
</tr>
<tr>
<td>40</td>
<td>28</td>
<td>82</td>
<td>52</td>
</tr>
<tr>
<td>41</td>
<td>29</td>
<td>83</td>
<td>53</td>
</tr>
<tr>
<td>42</td>
<td>2A</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Keyboard Scan Codes
<table>
<thead>
<tr>
<th>Pin</th>
<th>TTL Signal</th>
<th>Signal Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>+ Keyboard Clock</td>
<td>+5 Vdc</td>
</tr>
<tr>
<td>2</td>
<td>+ Keyboard Data</td>
<td>+5 Vdc</td>
</tr>
<tr>
<td>3</td>
<td>- Keyboard Reset (Not used by keyboard)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Ground</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>+5 Volts</td>
<td>+5 Vdc</td>
</tr>
</tbody>
</table>

**Keyboard Interface Connector Specifications**
Expansion Unit

The expansion unit option upgrades the IBM Personal Computer XT by adding expansion slots in a separate unit. This option consists of an extender card, an expansion cable, and the expansion unit. The expansion unit contains a power supply, an expansion board, and a receiver card. This option utilizes one expansion slot in the system unit to provide seven additional expansion slots in the expansion unit.

Expansion Unit Cable

The expansion unit cable consists of a 56-wire, foil-shielded cable terminated on each end with a 62-pin D-shell male connector. Either end of the expansion unit cable can be plugged into the extender card or the receiver card.

Power Supply

The expansion unit power supply provides $+5$, $-5$, $+12$, and $-12\, \text{Vdc}$ to the expansion board. The expansion unit power supply has the same specifications as the system unit power supply.

Expansion Board

The expansion board is a support board that carries the I/O channel signals from the option adapters and receiver card. These signals, except 'osc,' are carried over the expansion cable. Because 'osc' is not sent over the expansion cable, a 14.31818-MHz signal is generated on the expansion board. This signal may not be in phase with the 'osc' signal in the system unit.

Decoupling capacitors provided on the expansion board aid in noise filtering.
Expansion Board Block Diagram

1-72 Expansion Unit
Expansion Channel

All signals found on the system unit’s I/O channel will be provided to expansion slots in the expansion unit, with the exception of the ‘osc’ signal and the voltages mentioned previously.

A ‘ready’ line on the expansion channel makes it possible to operate with slow I/O or memory devices. If the channel’s ‘I/O ch rdy’ line is not activated by an addressed device, all processor-generated memory cycles take five processor clock cycles per byte for memory in the expansion unit.

The following table contains a list of all the signals that are redriven by the extender and receiver cards, and their associated time delays. The delay times include the delay due to signal propagation in the expansion cable. Assume a nominal cable delay of 3 ns. As such, device access will be less than 260 ns.

<table>
<thead>
<tr>
<th>Signal</th>
<th>Nominal Delay (ns)</th>
<th>Maximum Delay (ns)</th>
<th>Direction (*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AO - A19</td>
<td>27</td>
<td>39</td>
<td>Output</td>
</tr>
<tr>
<td>AEN</td>
<td>27</td>
<td>39</td>
<td>Output</td>
</tr>
<tr>
<td>DACK0 - DACK3</td>
<td>27</td>
<td>39</td>
<td>Output</td>
</tr>
<tr>
<td>MEMR</td>
<td>27</td>
<td>39</td>
<td>Output</td>
</tr>
<tr>
<td>MEMW</td>
<td>51</td>
<td>75</td>
<td>Output</td>
</tr>
<tr>
<td>IOR</td>
<td>51</td>
<td>75</td>
<td>Output</td>
</tr>
<tr>
<td>IOW</td>
<td>27</td>
<td>39</td>
<td>Output</td>
</tr>
<tr>
<td>ALE</td>
<td>27</td>
<td>39</td>
<td>Output</td>
</tr>
<tr>
<td>CLK</td>
<td>27</td>
<td>39</td>
<td>Output</td>
</tr>
<tr>
<td>T/C</td>
<td>27</td>
<td>39</td>
<td>Output</td>
</tr>
<tr>
<td>RESET</td>
<td>27</td>
<td>39</td>
<td>Output</td>
</tr>
<tr>
<td>IRQ2 - IRQ7</td>
<td>36</td>
<td>(***)</td>
<td>Input</td>
</tr>
<tr>
<td>DRQ1 - DRQ3</td>
<td>36</td>
<td>(***)</td>
<td>Input</td>
</tr>
<tr>
<td>I/O CH RDY</td>
<td>36</td>
<td>51</td>
<td>Input</td>
</tr>
<tr>
<td>I/O CH CK</td>
<td>36</td>
<td>51</td>
<td>Input</td>
</tr>
<tr>
<td>DO - D7 (Read)</td>
<td>84</td>
<td>133</td>
<td>Input</td>
</tr>
<tr>
<td>DO - D7 (Write)</td>
<td>19</td>
<td>27</td>
<td>Output</td>
</tr>
</tbody>
</table>

(*) With respect to the system unit.

(**) Asynchronous nature of interrupts and other requests are more dependent on processor recognition than electrical signal propagation through expansion logic.
Extender Card

The extender card is a four-plane card. The extender card redrives the I/O channel to provide sufficient power to avoid capacitive effects of the cable. The extender card presents only one load per line of the I/O channel.

The extender card has a wait-state generator that inserts a wait-state on 'memory read' and 'memory write' operations (except refreshing) for all memory contained in the expansion unit. The address range for wait-state generation is controlled by switch settings on the extender card.

The DIP switch on the extender card should be set to indicate the maximum contiguous read/write memory housed in the system unit. The extender card switch settings are located in "Appendix G: Switch Settings." Switch positions 1 through 4 correspond to address bits hex A19 to hex A16, respectively.

The switch settings determine which address segments have a wait state inserted during 'memory read' and 'memory write' operations. Wait states are required for any memory, including ROM on option adapters, in the expansion unit. Wait states are not inserted in the highest segment, hex addresses F0000 to FFFFFF (segment F).
Extender Card Programming Considerations

Several registers associated with the expansion option are programmable and readable for diagnostic purposes. The following figure indicates the locations and functions of the registers on the extender card.

<table>
<thead>
<tr>
<th>Location</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory FXXXX(*)</td>
<td>Write to memory to latch address bits</td>
</tr>
<tr>
<td>Port 210</td>
<td>Write to latch expansion bus data (ED0-ED7)</td>
</tr>
<tr>
<td>Port 210</td>
<td>Read to verify expansion bus data (ED0-ED7)</td>
</tr>
<tr>
<td>Port 211</td>
<td>Read high-order address bits (A8 - A15)</td>
</tr>
<tr>
<td>Port 211</td>
<td>Write to clear wait test latch</td>
</tr>
<tr>
<td>Port 212</td>
<td>Read low-order address bits (A0 - A7)</td>
</tr>
<tr>
<td>Port 213</td>
<td>Write 00 to disable expansion unit</td>
</tr>
<tr>
<td>Port 213</td>
<td>Write 01 to enable expansion unit</td>
</tr>
<tr>
<td>Port 213</td>
<td>Read status of expansion unit</td>
</tr>
<tr>
<td></td>
<td>D0 = enable/disable</td>
</tr>
<tr>
<td></td>
<td>D1 = wait-state request flag</td>
</tr>
<tr>
<td></td>
<td>D2-D3 = not used</td>
</tr>
<tr>
<td></td>
<td>D4-D7 = switch position</td>
</tr>
<tr>
<td></td>
<td>1 = Off</td>
</tr>
<tr>
<td></td>
<td>0 = On</td>
</tr>
</tbody>
</table>

(*) Example: Write to memory location F123:4=00
Read Port 211 = 12
Read Port 212 = 34

(All values in hex)

The expansion unit is automatically enabled upon power-up. The extender card and receiver card will both be written to, if the expansion unit is not disabled when writing to FXXXX. However, the system unit and the expansion unit are read back separately.
Extender Card Block Diagram

1-76 Expansion Unit
Receiver Card

The receiver card is a four-plane card that fits in expansion slot 8 of the expansion unit. The receiver card redrives the I/O channel to provide sufficient power for additional options and to avoid capacitive effects. Directional control logic is contained on the receiver card to resolve contention and direct data flow on the I/O channel. Steering signals are transmitted back over the expansion cable for use on the extender card.

Receiver Card Programming Considerations

Several registers associated with the expansion option are programmable and readable for diagnostic purposes. The following figure indicates the locations and functions of the registers on the receiver card.

<table>
<thead>
<tr>
<th>Location</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory FXXXX(*)</td>
<td>Write to memory to latch address bits</td>
</tr>
<tr>
<td>Port 214</td>
<td>Write to latch data bus bits (D0 - D7)</td>
</tr>
<tr>
<td>Port 214</td>
<td>Read data bus bits (D0 - D7)</td>
</tr>
<tr>
<td>Port 215</td>
<td>Read high-order address bits (A8 - A15)</td>
</tr>
<tr>
<td>Port 215</td>
<td>Read low-order address bits (A0 - A7)</td>
</tr>
</tbody>
</table>

(*) Example:
- Write to memory location F123:4=00
- Read Port 215 =12
- Read Port 216 =34

(All values in hex)

The expansion unit is automatically enabled upon power-up. The expansion unit and the system unit will be written to, if the expansion unit is not disabled when writing to FXXXX. However, the system unit and the expansion unit are read back separately.
Receiver Card Block Diagram
Expansion Unit Interface Information

The extender card and receiver card rear-panel connectors are the same. Pin and signal assignments for the extender and receiver cards are shown below.

![Diagram showing pin connections]

<table>
<thead>
<tr>
<th>Pin</th>
<th>Signal</th>
<th>Pin</th>
<th>Signal</th>
<th>Pin</th>
<th>Signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>+E IRQ6</td>
<td>22</td>
<td>+E D5</td>
<td>43</td>
<td>+E IRQ7</td>
</tr>
<tr>
<td>2</td>
<td>+E DRQ2</td>
<td>23</td>
<td>+E DRQ1</td>
<td>44</td>
<td>+E D6</td>
</tr>
<tr>
<td>3</td>
<td>+E DIR</td>
<td>24</td>
<td>+E DRQ3</td>
<td>45</td>
<td>+E I/O CH RDY</td>
</tr>
<tr>
<td>4</td>
<td>+E ENABLE</td>
<td>25</td>
<td>RESERVED</td>
<td>46</td>
<td>+E IRQ3</td>
</tr>
<tr>
<td>5</td>
<td>+E CLK</td>
<td>26</td>
<td>+E ALE</td>
<td>47</td>
<td>+E D7</td>
</tr>
<tr>
<td>6</td>
<td>-E MEM IN EXP</td>
<td>27</td>
<td>+E T/C</td>
<td>48</td>
<td>+E D1</td>
</tr>
<tr>
<td>7</td>
<td>+E A17</td>
<td>28</td>
<td>+E RESET</td>
<td>49</td>
<td>-E I/O CH CK</td>
</tr>
<tr>
<td>8</td>
<td>+E A16</td>
<td>29</td>
<td>+E AEN</td>
<td>50</td>
<td>+E IRQ2</td>
</tr>
<tr>
<td>9</td>
<td>+E A5</td>
<td>30</td>
<td>+E A19</td>
<td>51</td>
<td>+E D0</td>
</tr>
<tr>
<td>10</td>
<td>-E DACK0</td>
<td>31</td>
<td>+E A14</td>
<td>52</td>
<td>+E D2</td>
</tr>
<tr>
<td>11</td>
<td>+E A15</td>
<td>32</td>
<td>+E A12</td>
<td>53</td>
<td>+E D4</td>
</tr>
<tr>
<td>12</td>
<td>+E A11</td>
<td>33</td>
<td>+E A18</td>
<td>54</td>
<td>+E IRQ5</td>
</tr>
<tr>
<td>13</td>
<td>+E A10</td>
<td>34</td>
<td>-E MEMR</td>
<td>55</td>
<td>+E IRQ4</td>
</tr>
<tr>
<td>14</td>
<td>+E A9</td>
<td>35</td>
<td>-E MEMW</td>
<td>56</td>
<td>+E D3</td>
</tr>
<tr>
<td>15</td>
<td>+E A1</td>
<td>36</td>
<td>+E A0</td>
<td>57</td>
<td>GND</td>
</tr>
<tr>
<td>16</td>
<td>+E A3</td>
<td>37</td>
<td>-E DACK3</td>
<td>58</td>
<td>GND</td>
</tr>
<tr>
<td>17</td>
<td>-E DACK1</td>
<td>38</td>
<td>+E A6</td>
<td>59</td>
<td>GND</td>
</tr>
<tr>
<td>18</td>
<td>+E A4</td>
<td>39</td>
<td>-E IOR</td>
<td>60</td>
<td>GND</td>
</tr>
<tr>
<td>19</td>
<td>-E DACK2</td>
<td>40</td>
<td>+E A8</td>
<td>61</td>
<td>GND</td>
</tr>
<tr>
<td>20</td>
<td>-E IOW</td>
<td>41</td>
<td>+E A2</td>
<td>62</td>
<td>GND</td>
</tr>
<tr>
<td>21</td>
<td>+E A13</td>
<td>42</td>
<td>+E A7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

E = Extended

Connector Specifications
IBM 80 CPS Printers

The IBM 80 CPS (characters-per-second) Printers are self-powered, stand-alone, tabletop units. They attach to the system unit through a parallel signal cable, 6 feet in length. The units obtain ac power from a standard wall outlet (120 Vac). The printers are 80 cps, bidirectional, wire-matrix devices. They print characters in a 9 by 9 dot matrix with a 9-wire head. They 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. The printers can print double-size characters and double-strike characters. The printers print the standard ASCII, 96-character, uppercase and lowercase character sets. A printer without an extended character set also has a set of 64 special block graphic characters.

The IBM 80 CPS Graphics Printer has additional capabilities including: an extended character set for international languages, subscript, superscript, an underline mode, and programmable graphics.

The printers can also accept commands setting the line-feed control desired for the application. They attach to the system unit through the printer adapter or the combination monochrome display and printer adapter. The cable is a 25-lead shielded cable with a 25-pin D-shell connector at the system unit end, and a 36-pin connector at the printer end.
<table>
<thead>
<tr>
<th>(1) Print Method:</th>
<th>Serial-impact dot matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>(2) Print Speed:</td>
<td>80 cps</td>
</tr>
<tr>
<td>(3) Print Direction:</td>
<td>Bidirectional with logical seeking</td>
</tr>
<tr>
<td>(4) Number of Pins in Head:</td>
<td>9</td>
</tr>
<tr>
<td>(5) Line Spacing:</td>
<td>1/16 inch (4.23 mm) or programmable</td>
</tr>
<tr>
<td>(6) Printing Characteristics Matrix:</td>
<td>9 x 9</td>
</tr>
<tr>
<td>Character Set:</td>
<td>Full 96-character ASCII with descenders plus 9 international characters/symbols.</td>
</tr>
<tr>
<td>Graphic Character:</td>
<td>See “Additional Printer Specifications”</td>
</tr>
<tr>
<td>(7) Printing Sizes</td>
<td></td>
</tr>
<tr>
<td>Characters per inch</td>
<td>Maximum characters per inch</td>
</tr>
<tr>
<td>Normal:</td>
<td>10</td>
</tr>
<tr>
<td>Double Width:</td>
<td>5</td>
</tr>
<tr>
<td>Compressed:</td>
<td>16.5</td>
</tr>
<tr>
<td>Double Width-Compressed:</td>
<td>8.25</td>
</tr>
<tr>
<td>(8) Media Handling:</td>
<td></td>
</tr>
<tr>
<td>Paper Feed:</td>
<td>Adjustable sprocket pin feed</td>
</tr>
<tr>
<td>Paper Width Range:</td>
<td>4 inch (101.6 mm) to 10 inch (254 mm)</td>
</tr>
<tr>
<td>Copies:</td>
<td>One original plus two carbon copies (total thickness not to exceed 0.012 inch (0.3 mm)). Minimum paper thickness is 0.0025 inch (0.064 mm).</td>
</tr>
<tr>
<td>Paper Path:</td>
<td>Rear</td>
</tr>
<tr>
<td>(9) Interfaces:</td>
<td></td>
</tr>
<tr>
<td>Standard:</td>
<td>Parallel 8-bit Data and Control Lines</td>
</tr>
<tr>
<td>(10) Inked Ribbon:</td>
<td></td>
</tr>
<tr>
<td>Color:</td>
<td>Black</td>
</tr>
<tr>
<td>Type:</td>
<td>Cartridge</td>
</tr>
<tr>
<td>Life Expectancy:</td>
<td>3 million characters</td>
</tr>
<tr>
<td>(11) Environmental Conditions</td>
<td></td>
</tr>
<tr>
<td>Operating Temperature Range:</td>
<td>41 to 95°F (5 to 35°C)</td>
</tr>
<tr>
<td>Operating Humidity:</td>
<td>10 to 80% non-condensing</td>
</tr>
<tr>
<td>(12) Power Requirement:</td>
<td></td>
</tr>
<tr>
<td>Voltage:</td>
<td>120 Vac, 60 Hz</td>
</tr>
<tr>
<td>Current:</td>
<td>1 A maximum</td>
</tr>
<tr>
<td>Power Consumption:</td>
<td>100 VA maximum</td>
</tr>
<tr>
<td>(13) Physical Characteristics:</td>
<td></td>
</tr>
<tr>
<td>Height:</td>
<td>4.2 inches (107 mm)</td>
</tr>
<tr>
<td>Width:</td>
<td>14.7 inches (374 mm)</td>
</tr>
<tr>
<td>Depth:</td>
<td>12.0 inches (305 mm)</td>
</tr>
<tr>
<td>Weight:</td>
<td>12 pounds (5.5 kg)</td>
</tr>
</tbody>
</table>

**Printer Specifications**
(6) Printing Characteristics:
IBM 80 CPS Matrix Printer
Graphics
IBM 80 CPS Graphics Printer
64 block characters.

(6) 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 difference 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.

(7) Printing Sizes:

<table>
<thead>
<tr>
<th></th>
<th>Characters per inch</th>
<th>Maximum characters per line</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subscript</td>
<td>10</td>
<td>80</td>
</tr>
<tr>
<td>Superscript</td>
<td>10</td>
<td>80</td>
</tr>
</tbody>
</table>

Additional Printer Specifications
Setting the DIP Switches

There are two DIP switches on the control circuit board. In order to satisfy the user's specific requirements, desired control modes are selectable by the DIP switches. The functions of the switches and their preset conditions at the time of shipment are as shown in the following figures.

**Location of Printer DIP Switches**

<table>
<thead>
<tr>
<th>Switch Number</th>
<th>Function</th>
<th>On</th>
<th>Off</th>
<th>Factory-Set Condition</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 &amp; Line Feed</td>
<td>On</td>
</tr>
<tr>
<td>1-3</td>
<td>Buffer Full</td>
<td>Print Only</td>
<td>Print &amp; 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>Delete Code</td>
<td>Invalid</td>
<td>Valid</td>
<td>On</td>
</tr>
<tr>
<td>1-6</td>
<td>Error</td>
<td>Sounds</td>
<td>Does Not Sound</td>
<td>On</td>
</tr>
<tr>
<td>1-7</td>
<td>Character Generator</td>
<td>N.A.</td>
<td>Graphic Patterns Select</td>
<td>Off</td>
</tr>
<tr>
<td></td>
<td>(Graphic Pattern Select)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-8</td>
<td>SLCT IN Signal Fixed</td>
<td>Fixed</td>
<td>Not Fixed</td>
<td>On</td>
</tr>
<tr>
<td></td>
<td>Internally</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Functions and Conditions of DIP Switch 1 (Matrix)**

1-84 Printers
### Functions and Conditions of DIP Switch 2 (Matrix)

<table>
<thead>
<tr>
<th>Switch Number</th>
<th>Function</th>
<th>On</th>
<th>Off</th>
<th>Factory-Set Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-1</td>
<td>Not Applicable</td>
<td>—</td>
<td>—</td>
<td>On</td>
</tr>
<tr>
<td>2-2</td>
<td>Not Applicable</td>
<td>—</td>
<td>—</td>
<td>On</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>Coding Table Select</td>
<td>N.A.</td>
<td>Standard</td>
<td>Off</td>
</tr>
</tbody>
</table>

### Functions and Conditions of DIP Switch 1 (Graphics)

<table>
<thead>
<tr>
<th>Switch Number</th>
<th>Function</th>
<th>On</th>
<th>Off</th>
<th>Factory-Set Condition</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>Not Valid</td>
<td>Off</td>
</tr>
</tbody>
</table>

### Functions and Conditions of DIP Switch 2 (Graphics)
Parallel Interface Description

Specifications:

- Data transfer rate: 1000 cps (maximum)
- Synchronization: By externally-supplied STROBE pulses.
- Handshaking ACKNLG or BUSY signals.
- Logic level: Input data and all interface control signals are compatible with the TTL level.

Connector: Plug: 57-30360 (Amphenol)

Connector pin assignment and descriptions of respective interface signals are provided on the following pages.

Data transfer sequence:

![Parallel Interface Timing Diagram](image-url)

Parallel Interface Timing Diagram
<table>
<thead>
<tr>
<th>Signal Pin No.</th>
<th>Return Pin No.</th>
<th>Signal</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>19</td>
<td>STROBE</td>
<td>In</td>
<td>STROBE pulse to read data in. Pulse width must be more than 0.5 μs at receiving terminal. The signal level is normally &quot;high&quot;; read-in of data is performed at the &quot;low&quot; level of this signal.</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>DATA 1</td>
<td>In</td>
<td>These signals represent information of the 1st to 8th bits of parallel data respectively. Each signal is at &quot;high&quot; level when data is logical &quot;1&quot; and &quot;low&quot; when logical &quot;0.&quot;</td>
</tr>
<tr>
<td>3</td>
<td>21</td>
<td>DATA 2</td>
<td>In</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>22</td>
<td>DATA 3</td>
<td>In</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>23</td>
<td>DATA 4</td>
<td>In</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>24</td>
<td>DATA 5</td>
<td>In</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>25</td>
<td>DATA 6</td>
<td>In</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>26</td>
<td>DATA 7</td>
<td>In</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>27</td>
<td>DATA 8</td>
<td>In</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>28</td>
<td>ACKNLG</td>
<td>Out</td>
<td>Approximately 5 μs pulse; &quot;low&quot; indicates that data has been received and the printer is ready to accept other data.</td>
</tr>
<tr>
<td>11</td>
<td>29</td>
<td>BUSY</td>
<td>Out</td>
<td>A &quot;high&quot; signal indicates that the printer cannot receive data. The signal becomes &quot;high&quot; in the following cases: 1. During data entry. 2. During printing operation. 3. In &quot;offline&quot; state. 4. During printer error status.</td>
</tr>
<tr>
<td>Signal Pin No.</td>
<td>Return Pin No.</td>
<td>Signal</td>
<td>Direction</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>----------------</td>
<td>--------</td>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>12</td>
<td>30</td>
<td>PE</td>
<td>Out</td>
<td>A “high” signal indicates that the printer is out of paper.</td>
</tr>
<tr>
<td>13</td>
<td></td>
<td>SLCT</td>
<td>Out</td>
<td>This signal indicates that the printer is in the selected state.</td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>AUTO</td>
<td>In</td>
<td>With this signal being at “low” level, the paper is automatically fed one line after printing. (The signal level can be fixed to “low” with DIP SW pin 2-3 provided on the control circuit board.)</td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>NC</td>
<td></td>
<td>Not used.</td>
</tr>
<tr>
<td>16</td>
<td></td>
<td>OV</td>
<td></td>
<td>Logic GND level.</td>
</tr>
<tr>
<td>17</td>
<td></td>
<td>CHASSIS-GND</td>
<td>—</td>
<td>Printer chassis GND. In the printer, the chassis GND and the logic GND are isolated from each other.</td>
</tr>
<tr>
<td>18</td>
<td></td>
<td>NC</td>
<td>—</td>
<td>Not used.</td>
</tr>
<tr>
<td>19-30</td>
<td></td>
<td>GND</td>
<td>—</td>
<td>“Twisted-Pair Return” signal; GND level.</td>
</tr>
<tr>
<td>31</td>
<td></td>
<td>INT</td>
<td>In</td>
<td>When the level of this signal becomes “low” the printer controller is reset to its initial state and the print buffer is cleared. This signal is normally at “high” level, and its pulse width must be more than 50 μs at the receiving terminal.</td>
</tr>
</tbody>
</table>

Connector Pin Assignment and Descriptions of Interface Signals (Part 2 of 3)
### Connector Pin Assignment and Descriptions of Interface Signals (Part 3 of 3)

<table>
<thead>
<tr>
<th>Signal Pin No.</th>
<th>Return Pin No.</th>
<th>Signal</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td></td>
<td>ERROR</td>
<td>Out</td>
<td>The level of this signal becomes &quot;low&quot; when the printer is in &quot;Paper End&quot; state, &quot;Offline&quot; state and &quot;Error&quot; state.</td>
</tr>
<tr>
<td>33</td>
<td>—</td>
<td>GND</td>
<td>—</td>
<td>Same as with pin numbers 19 to 30.</td>
</tr>
<tr>
<td>34</td>
<td>—</td>
<td>NC</td>
<td>—</td>
<td>Not used.</td>
</tr>
<tr>
<td>35</td>
<td>—</td>
<td></td>
<td></td>
<td>Pulled up to +5 Vdc through 4.7 k-ohms resistance.</td>
</tr>
<tr>
<td>36</td>
<td>—</td>
<td>SLCT IN</td>
<td>In</td>
<td>Data entry to the printer is possible only when the level of this signal is &quot;low&quot;. (Internal fixing can be carried out with DIP SW 1-8. The condition at the time of shipment is set &quot;low&quot; for this signal.)</td>
</tr>
</tbody>
</table>

**Notes:**
1. "Direction" refers to the direction of signal flow as viewed from the printer.
2. "Return" denotes "Twisted-Pair Return" and is to be connected at signal-ground level. When wiring the interface, be sure to use a twisted-pair cable for each signal and never fail to complete connection on the return side. To prevent noise effectively, these cables should be shielded and connected to the chassis of the system unit and printer, respectively.
3. All interface conditions are based on TTL level. Both the rise and fall times of each signal must be less than 0.2 μs.
4. Data transfer must not be carried out by ignoring the ACKNLG or BUSY signal. (Data transfer to this printer can be carried out only after confirming the ACKNLG signal or when the level of the BUSY signal is "low.")
Printer Modes for the IBM 80 CPS Printers

The IBM 80 CPS Graphics Printer can use any of the combinations listed below, and the print mode can be changed at any place within a line.

The IBM 80 CPS Matrix Printer cannot use the Subscript, Superscript, or Underline print modes. The Double Width print mode will affect the entire line with the matrix printer.

The allowed combinations of print modes that can be selected are listed in the following table. Modes can be selected and combined if they are in the same vertical column.

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

1-90 Printers
Printer Control Codes

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

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

The descriptions that follow assume that the printer DIP switches have not been changed from their factory settings.
<table>
<thead>
<tr>
<th>Printer Code</th>
<th>Printer Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUL</td>
<td>Null Used with ESC B and ESC D as a list terminator. NUL is also used with other printer control codes to select options (for example, ESC S). Example: LPRINT CHR$ (0);</td>
</tr>
<tr>
<td>BEL</td>
<td>Bell Sounds the printer buzzer for 1 second. Example: LPRINT CHR$ (7);</td>
</tr>
<tr>
<td>HT</td>
<td>Horizontal Tab Tabs to the next horizontal tap stop. Tab stops are set with ESC D. No tab stops are set when the printer is powered on. (Graphics Printer sets a tab stop every 8 columns when powered on.) Example: LPRINT CHR$ (9);</td>
</tr>
<tr>
<td>LF</td>
<td>Line Feed Spaces the paper up one line. Line spacing is 1/6-inch unless reset by ESC A, ESC 0, ESC 1, ESC 2 or ESC 3. Example: LPRINT CHR$ (10);</td>
</tr>
<tr>
<td>VT</td>
<td>Vertical TabSpaces the paper to the next vertical tab position. (Graphics Printer does not allow vertical tabs to be set; therefore, the VT code is treated as LF.) Example: LPRINT CHR$ (11);</td>
</tr>
<tr>
<td>FF</td>
<td>Form FeedAdvances 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);</td>
</tr>
<tr>
<td>CR</td>
<td>Carriage ReturnEnds 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);</td>
</tr>
<tr>
<td>Printer Code</td>
<td>Printer Function</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------</td>
</tr>
</tbody>
</table>
| SO           | **Shift Out (Double Width)**<br>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$(14); |
| SI           | **Shift In (Compressed)**<br>Changes the printer to the Compressed Character print mode.  
Example:  
LPRINT CHR$(15); |
| DC1          | **Device Control 1 (Printer Selected)**<br>(Graphics Printer ignores DC1)  
Printer accepts data from the system unit. Printer DIP switch 1-8 must be set to the Off position.  
Example:  
LPRINT CHR$(17); |
| DC2          | **Device Control 2 (Compressed Off)**<br>Stops printing in the Compressed print mode.  
Example:  
LPRINT CHR$(18); |
| DC3          | **Device Control 3 (Printer Deselected)**<br>(Graphics Printer ignores DC3)  
Printer does not accept data from the system unit. The system unit must have the printer select line low, and DIP switch 1-8 must be in the Off position.  
Example:  
LPRINT CHR$(19); |
| DC4          | **Device Control 4 (Double Width Off)**<br>Stops printing in the Double Width print mode.  
Example:  
LPRINT CHR$(20); |
| CAN          | **Cancel**<br>Clears the printer buffer. Control codes, except SO, remain in effect.  
Example:  
LPRINT CHR$(24); |
| ESC          | **Escape**<br>Lets the printer know that the next data sent is a printer command.  
(See the following list of commands.)  
Example:  
LPRINT CHR$(27); |
<table>
<thead>
<tr>
<th>Printer Code</th>
<th>Printer Function</th>
</tr>
</thead>
</table>
| ESC -        | Escape Minus (Underline)  
Format: ESC -;n;  
(Graphics Printer only)  
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$(48); |
| ESC 1        | Escape One (7/72-Inch Line Feeding)  
Changes paper feed to 7/72 inch.  
Example:  
LPRINT CHR$(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$(50); |
| ESC 3        | Escape Three (Variable Line Feeding)  
Format: ESC 3;n;  
(Graphics Printer only)  
Changes the paper feeding to n/216-inch. The example below sets the paper feeding to 54/216 (1/4) inch. The value of n must be between 1 and 255.  
Example:  
LPRINT CHR$(27);CHR$(51);CHR$(54); |
| ESC 6        | Escape Six (Select Character Set 2)  
(Graphics Printer only)  
Selects character set 2. (See "Printer Character Set 2.")  
Example:  
LPRINT CHR$(27);CHR$(54); |
| ESC 7        | Escape Seven (Select Character Set 1.)  
(Graphics Printer only)  
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); |
<table>
<thead>
<tr>
<th>Printer Code</th>
<th>Printer Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESC 9</td>
<td><strong>Escape Nine (Cancel Ignore Paper End)</strong>&lt;br&gt;Cancels the Ignore Paper End command. ESC 9 is selected when the printer is powered on or reset.&lt;br&gt;Example:&lt;br&gt;LPRINT CHR$(27);CHR$(57);</td>
</tr>
<tr>
<td>ESC &lt;</td>
<td><strong>Escape Less Than (Home Head)</strong>&lt;br&gt;(Graphics Printer only)&lt;br&gt;The print head will return to the left margin to print the line following ESC &lt;. This will occur for one line only.&lt;br&gt;Example:&lt;br&gt;LPRINT CHR$(27);CHR$(60);</td>
</tr>
<tr>
<td>ESC A</td>
<td><strong>Escape A (Sets Variable Line Feeding)</strong>&lt;br&gt;Format: ESC A;n;&lt;br&gt;Escape A sets the line-feed to (\frac{n}{72})-inch. The example below tells the printer to set line feeding to (\frac{24}{72})-inch. ESC 2 must be sent to the printer before the line feeding will change. For example, ESC A;24 (text) ESC 2 (text). The text following ESC A;24 will space at the previously set line-feed increments. The text following ESC 2 will be printed with new line-feed increments of (\frac{24}{72})-inch. Any increment between (\frac{1}{72}) and (\frac{85}{72}) may be used.&lt;br&gt;Example:&lt;br&gt;LPRINT CHR$(27);CHR$(65);CHR$(24);CHR$(27);CHR$(50);</td>
</tr>
<tr>
<td>ESC B</td>
<td><strong>Escape B (Set Vertical Tabs)</strong>&lt;br&gt;Format: ESC B;(\frac{n_1}{72};\frac{n_2}{72};\ldots;\frac{n_k}{72});NUL;&lt;br&gt;(Graphics Printer ignores ESC B)&lt;br&gt;Sets vertical tab stop positions. Up to 64 vertical tab stop positions are recognized by the printer. The (n)'s, in the format above, are used to indicate tab stop positions. Tab stop numbers must be received in ascending numeric order. The tab stop numbers will not become valid until the NUL code is entered. Once vertical tab stops are established, they will be valid until new tab stops are specified. (If the printer is reset or powered Off, set tab stops are cleared.) If no tab stop is set, the Vertical Tab command behaves as a Line Feed command. ESC B followed only by NUL will cancel tab stops. The form length must be set by the ESC C command prior to setting tabs.&lt;br&gt;Example:&lt;br&gt;LPRINT CHR$(27);CHR$(66);CHR$(10);CHR$(20);CHR$(40);CHR$(0);</td>
</tr>
<tr>
<td>Printer Code</td>
<td>Printer Function</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------</td>
</tr>
<tr>
<td><strong>ESC C</strong></td>
<td><strong>Escape C (Set Lines per Page)</strong></td>
</tr>
<tr>
<td></td>
<td>Format: ESC C;n;</td>
</tr>
<tr>
<td></td>
<td>Sets the page length. The ESC C command must have a value</td>
</tr>
<tr>
<td></td>
<td>following it to specify the length of page desired. (Maximum form</td>
</tr>
<tr>
<td></td>
<td>length for the printer is 127 lines.)</td>
</tr>
<tr>
<td></td>
<td>The example below sets the page length to 55 lines. The printer</td>
</tr>
<tr>
<td></td>
<td>defaults to 66 lines per page when powered on or reset.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>LPRINT CHR$(27);CHR$(67);CHR$(55);</td>
</tr>
<tr>
<td><strong>Escape C (Set Inches per Page)</strong></td>
<td></td>
</tr>
<tr>
<td>Format: ESC C;n;m;</td>
<td></td>
</tr>
<tr>
<td>(Graphics Printer only)</td>
<td></td>
</tr>
<tr>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>LPRINT CHR$(27);CHR$(67);CHR$(0);CHR$(12);</td>
<td></td>
</tr>
<tr>
<td><strong>ESC D</strong></td>
<td><strong>Escape D (Set Horizontal Tab Stops)</strong></td>
</tr>
<tr>
<td></td>
<td>Format: ESC D;n_1;n_2;...n_k;NUL;</td>
</tr>
<tr>
<td></td>
<td>Sets the horizontal tab stop positions. The example below shows the horizontal tab stop positions set at printer column positions of 10, 20, and 40. They are followed by CHR$(0), the NUL code. They must also be in ascending numeric order as shown. Tab stops can be set between 1 and 80. When in the Compressed print mode, tab stops can be set up to 132.</td>
</tr>
<tr>
<td></td>
<td>The maximum number of tabs that can be set is 112. The Graphics Printer can have a maximum of 28 tab stops. The HT (CHR$(9)) is used to execute a tab operation.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>LPRINT CHR$(27);CHR$(68);CHR$(10)CHR$(20)CHR$(40);CHR$(0);</td>
</tr>
<tr>
<td><strong>ESC E</strong></td>
<td><strong>Escape E (Emphasized)</strong></td>
</tr>
<tr>
<td></td>
<td>Changes the printer to the Emphasized print mode. The speed of the printer is reduced to half speed during the Emphasized print mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>LPRINT CHR$(27);CHR$(69);</td>
</tr>
<tr>
<td><strong>ESC F</strong></td>
<td><strong>Escape F (Emphasized Off)</strong></td>
</tr>
<tr>
<td></td>
<td>Stops printing in the Emphasized print mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>LPRINT CHR$(27);CHR$(70);</td>
</tr>
<tr>
<td><strong>ESC G</strong></td>
<td><strong>Escape G (Double Strike)</strong></td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>LPRINT CHR$(27);CHR$(71);</td>
</tr>
<tr>
<td>Printer Code</td>
<td>Printer Function</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------</td>
</tr>
</tbody>
</table>
| **ESC H**    | **Escape H (Double Strike Off)**  
|              | Stops printing in the Double Strike mode. |  
|              | Example: LPRINT CHR$(27);CHR$(72); |  |
| **ESC J**    | **Escape J (Set Variable Line Feeding)**  
|              | Format: ESC J;n; (Graphics Printer only)  
|              | When ESC J is sent to the printer, the paper will feed in increments of n/216 of an inch. The value of n must be between 1 and 255.  
|              | The example below 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; (Graphics Printer only)  
|              | 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. |

<table>
<thead>
<tr>
<th>n₂</th>
<th>MSB</th>
<th>LSB</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
</tr>
<tr>
<td>12</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>n₁</th>
<th>MSB</th>
<th>LSB</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>
Data sent to the printer.

<table>
<thead>
<tr>
<th>Text (20 characters)</th>
<th>ESC</th>
<th>K</th>
<th>n=360</th>
<th>Bit-image data</th>
<th>Next data</th>
</tr>
</thead>
</table>

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>

Example:

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

This example will give you a row of slashes printed in the 480 Bit-Image mode.
<table>
<thead>
<tr>
<th>Printer Code</th>
<th>Printer Function</th>
</tr>
</thead>
</table>
| ESC L        | **Escape L (960 Bit-Image Graphics Mode)**  
Format: ESC L;n₁;n₂;v₁;v₂;…vₖ;  
(Graphics Printer only)  
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;  
(Graphics Printer only)  
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 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.  
Example:  
CHR$(27);CHR$(7B);CHR$(12);  |
| ESC O        | **Escape O (Cancel Skip Perforation)**  
(Graphics Printer only)  
Cancels the Skip Perforation function.  
Example:  
LPRINT CHR$(27);CHR$(79);  |
| ESC S        | **Escape S (Subscript/Superscript)**  
Format: ESC S;n;  
(Graphics Printer only)  
Changes the printer to the Subscript print mode when ESC S is followed by a 1, as in the example below. When ESC S is followed by a 0 (zero), the printer will print in the Superscript print mode.  
Example:  
LPRINT CHR$(27);CHR$(83);CHR$(1);  |
| ESC T        | **Escape T (Subscript/Superscript Off)**  
(Graphics Printer only)  
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;  
(Graphics Printer only)  
The printer will print 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$(1);  |
<table>
<thead>
<tr>
<th>Printer Code</th>
<th>Printer Function</th>
</tr>
</thead>
</table>
| ESC W        | **Escape W (Double Width)**  
Format: ESC W;n;  
(Graphics Printer only)  
Changes the printer to the Double Width print mode when ESC W is followed by a 1. This mode is not canceled by a line-feed operation and must be canceled with ESC W followed by a 0 (zero).  
Example:  
LPRINT CHR$(27);CHR$(87);CHR$(1); |
| ESC Y        | **Escape Y (960 Bit-Image Graphics Mode Normal Speed)**  
Format: ESC Y n₁;n₂;v₁;v₂;...vₖ;  
(Graphics Printer only)  
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 positions. The input of data is similar to ESC L. |
| ESC Z        | **Escape Z (1920 Bit-Image Graphics Mode)**  
Format: ESC Z;n₁;n₂;v₁;v₂;...vₖ;  
(Graphics Printer only)  
Changes from the Text mode to the 1920 Bit-Image Graphics mode. The input is similar to the other Bit-Image Graphics modes. ESC Z can print only every third dot position. |
| DEL          | **Delete (Clear Printer Buffer)**  
(Graphics Printer ignores DEL)  
Clears the printer buffer. Control codes, except SO, still remain in effect. DIP switch 1-5 must be in the Off position.  
Example:  
LPRINT CHR$(127); |

1-100 Printers
Matrix Printer Character Set (Part 2 of 2)
Graphics Printer Character Set 1 (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>BEI</td>
<td>HT</td>
<td>LF</td>
<td>VT</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>FF</td>
<td>CR</td>
<td>SO</td>
<td>SI</td>
<td>DC2</td>
<td>DC4</td>
<td></td>
<td></td>
<td></td>
<td></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>CAN</td>
<td>ESC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>ú</td>
<td>ñ</td>
<td>Ñ</td>
<td>à</td>
<td>ò</td>
<td></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>1/2</td>
<td>1/4</td>
<td>i</td>
<td>&lt;&lt;</td>
<td>&gt;&gt;</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>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>φ</td>
<td>ε</td>
<td>Ψ</td>
<td>Π</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>
</tbody>
</table>

Graphics Printer Character Set 1 (Part 2 of 2)

1-104 Printers
<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>BEL</td>
<td>HT</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>DC2</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>DC4</td>
<td>¥</td>
<td>CAN</td>
<td>ESC</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>SP</td>
<td>!</td>
<td>&quot;</td>
<td>#</td>
<td>$</td>
<td>%</td>
<td>&amp;</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>0</td>
<td>1</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>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>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>&lt;</td>
<td>=</td>
<td>&gt;</td>
<td>?</td>
<td>☝️</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</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>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>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>P</td>
<td>Q</td>
<td>R</td>
<td>S</td>
<td>T</td>
<td>U</td>
<td>V</td>
<td>W</td>
<td>X</td>
<td>Y</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>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>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>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>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>

Graphics Printer Character Set 2 (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>éãäàâåçêëèêîïîÀAEæÆôöò</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ûùûũûüç£¥Rp</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>áíóúñÑaoć</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1/2 1/4 i &lt;&lt; &gt;&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>μτφθΩδ∞øε∩</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>-√π²SP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Graphics Printer Character Set 2 (Part 2 of 2)
IBM Printer Adapter

The printer adapter is specifically designed to attach printers with a parallel port interface, but it can be used as a general input/output port for any device or application that matches its input/output capabilities. It has 12 TTL-buffer output points, which are latched and can be written and read under program control using the processor In or Out instruction. The adapter also has five steady-state input points that may be read using the processor’s 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.

The input/output signals are made available at the back of the adapter through a right-angled, PCB-mounted, 25-pin, D-shell connector. This connector protrudes through the rear panel of the system or expansion unit, where a cable may be attached.

When this adapter is used to attach a printer, data or printer commands are loaded into an 8-bit, latched, output port, and the strobe line is activated, writing data to the printer. The program then may 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 may also be read at the card’s interface for diagnostic loop functions. This allows faults to be isolated between the adapter and the attaching device.

This same function is also part of the combination IBM Monochrome Display and Printer Adapter. A block diagram of the printer adapter is on the next page.
8
25-Pin D-Shell Bus Buffer Data Latch Connector

8
8
Enable
Clock

Enable

Command Decoder

DIR
Read Data
Write Data
Write Control
Read Status
Read Control

Bus Buffers
Enable

Control Latch
Clock
Clear
O.C. Drivers

SLCT IN
STROBE
AUTO
FD XT
INIT
ERROR
SLCT
PE
ACK
BUSY

Reset

Printer Adapter Block Diagram

1-108 Printer Adapter
Programming Considerations

The printer adapter responds to five I/O instructions: two output and three input. The output instructions transfer data into 2 latches whose outputs are presented on pins of a 25-pin D-shell connector.

Two of the three input instructions allow the processor to read back the contents of the two latches. The third allows the processor to read the real time status of a group of pins on the connector.

A description of each instruction follows.

<table>
<thead>
<tr>
<th>IBM Monochrome Display &amp; Printer Adapter</th>
<th>Printer Adapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output to address hex 3BC</td>
<td>Output to address hex 378</td>
</tr>
<tr>
<td>Bit 7</td>
<td>Bit 3</td>
</tr>
<tr>
<td>Pin 9</td>
<td>Pin 5</td>
</tr>
<tr>
<td>Bit 6</td>
<td>Bit 2</td>
</tr>
<tr>
<td>Pin 8</td>
<td>Pin 4</td>
</tr>
<tr>
<td>Bit 5</td>
<td>Bit 1</td>
</tr>
<tr>
<td>Pin 7</td>
<td>Pin 3</td>
</tr>
<tr>
<td>Bit 4</td>
<td>Bit 0</td>
</tr>
<tr>
<td>Pin 6</td>
<td>Pin 2</td>
</tr>
</tbody>
</table>

The instruction captures data from the data bus and is present on the respective pins. These pins are each capable of sourcing 2.6 mA and sinking 24 mA.

It is essential that the external device not try to pull these lines to ground.
This instruction causes the latch to capture the five least significant bits of the data bus. The four least significant bits present their outputs, or inverted versions of their outputs, to the respective pins shown above. If bit 4 is written as 1, the card will interrupt the processor on the condition that pin 10 transitions high to low.

These pins are driven by open collector drivers pulled to +5 Vdc through 4.7 k-ohm resistors. They can each sink approximately 7 mA and maintain 0.8 volts down-level.

This command presents the processor with data present on the pins associated with the out to hex 3BC. This should normally reflect the exact value that was last written to hex 3BC. If an external device should be driving data on these pins (in violation of usage groundrules) at the time of an input, this data will be ORed with the latch contents.
This command presents realtime status to the processor from the pins as follows.

<table>
<thead>
<tr>
<th>IBM Monochrome Display &amp; Printer Adapter</th>
<th>Printer Adapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input from address hex 3BD</td>
<td>Input from address hex 379</td>
</tr>
</tbody>
</table>

This instruction causes the data present on pins 1, 14, 15, 17, and the IRQ bit to read by the processor. In the absence of external drive applied to these pins, data read by the processor will exactly match data last written to hex 3BE in the same bit positions. Note that data bits 0-2 are not included. If external drivers are dotted to these pins, that data will be ORed with data applied to the pins by the hex 3BE latch.

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pin 11</td>
<td>Pin 10</td>
<td>Pin 12</td>
<td>Pin 13</td>
<td>Pin 15</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IBM Monochrome Display &amp; Printer Adapter</th>
<th>Printer Adapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input from address hex 3BE</td>
<td>Input from address hex 37A</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>IRQ</td>
<td>POR=0</td>
<td>POR=1</td>
<td>POR=0</td>
<td>POR=1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P17</td>
<td>Pin 16</td>
<td>Pin 17</td>
<td>Pin 14</td>
<td>Pin 1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

These pins assume the states shown after a reset from the processor.
Note: All outputs are software-generated, and all inputs are real-time signals (not latched).

At Standard TTL Levels

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Adapter Pin Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Strobe</td>
<td>1</td>
</tr>
<tr>
<td>+Data Bit 0</td>
<td>2</td>
</tr>
<tr>
<td>+Data Bit 1</td>
<td>3</td>
</tr>
<tr>
<td>+Data Bit 2</td>
<td>4</td>
</tr>
<tr>
<td>+Data Bit 3</td>
<td>5</td>
</tr>
<tr>
<td>+Data Bit 4</td>
<td>6</td>
</tr>
<tr>
<td>+Data Bit 5</td>
<td>7</td>
</tr>
<tr>
<td>+Data Bit 6</td>
<td>8</td>
</tr>
<tr>
<td>+Data Bit 7</td>
<td>9</td>
</tr>
<tr>
<td>- Acknowledge</td>
<td>10</td>
</tr>
<tr>
<td>+Busy</td>
<td>11</td>
</tr>
<tr>
<td>+P.End (out of paper)</td>
<td>12</td>
</tr>
<tr>
<td>+Select</td>
<td>13</td>
</tr>
<tr>
<td>- Auto Feed</td>
<td>14</td>
</tr>
<tr>
<td>- Error</td>
<td>15</td>
</tr>
<tr>
<td>- Initialize Printer</td>
<td>16</td>
</tr>
<tr>
<td>- Select Input</td>
<td>17</td>
</tr>
<tr>
<td>Ground</td>
<td>18-25</td>
</tr>
</tbody>
</table>

Connector Specifications
IBM Monochrome Display and Printer Adapter

This chapter has two functions. The first is to provide the interface to the IBM Monochrome Display. The second provides a parallel interface for the IBM CPS Printer. This second function is fully discussed in the “IBM Printer Adapter” section.

The monitor adapter is designed around the Motorola 6845 CRT controller module. There are 4K bytes of static memory on the adapter which is used for the display buffer. This buffer has two ports and may be accessed directly by the processor. No parity is provided on the display buffer.

Two bytes are fetched from the display buffer in 553 ns, providing a data rate of 1.8M bytes/second.

The monitor adapter supports 256 different character codes. An 8K-byte character generator contains the fonts for the character codes. The characters, values, and screen characteristics are given in “Appendix C: Of Characters, Keystrokes, and Color.”

This monitor adapter, when used with a display containing P39 phosphor, will not support a light pen.

Where possible, only one low-power Schottky (LS) load is present on any I/O slot. Some of the address bus lines have two LS loads. No signal has more than two LS loads.

Characteristics of the monitor adapter are listed below:

- 80 by 25 screen
- Direct-drive output
- 9 by 14 character box
- 7 by 9 character
- 18 kHz monitor
- Character attributes
IBM Monochrome Adapter Block Diagram

1-114  Monochrome Adapter
Programming Considerations

The following table summarizes the 6845 internal data registers, their functions, and their parameters. For the IBM Monochrome Display, the values must be programmed into the 6845 to ensure proper initialization of the device.

<table>
<thead>
<tr>
<th>Register Number</th>
<th>Register File</th>
<th>Program Unit</th>
<th>IBM Monochrome Display (Address in hex)</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0</td>
<td>Horizontal Total</td>
<td>Characters</td>
<td>61</td>
</tr>
<tr>
<td>R1</td>
<td>Horizontal Displayed</td>
<td>Characters</td>
<td>50</td>
</tr>
<tr>
<td>R2</td>
<td>Horizontal Sync Position</td>
<td>Characters</td>
<td>52</td>
</tr>
<tr>
<td>R3</td>
<td>Horizontal Sync Width</td>
<td>Characters</td>
<td>F</td>
</tr>
<tr>
<td>R4</td>
<td>Vertical Total</td>
<td>Character Rows</td>
<td>19</td>
</tr>
<tr>
<td>R5</td>
<td>Vertical Total Adjust</td>
<td>Scan Line</td>
<td>6</td>
</tr>
<tr>
<td>R6</td>
<td>Vertical Displayed</td>
<td>Character Row</td>
<td>19</td>
</tr>
<tr>
<td>R7</td>
<td>Vertical Sync Position</td>
<td>Character Row</td>
<td>19</td>
</tr>
<tr>
<td>R8</td>
<td>Interlace Mode</td>
<td>--------------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>R9</td>
<td>Maximum Scan Line Address</td>
<td>Scan Line</td>
<td>D</td>
</tr>
<tr>
<td>R10</td>
<td>Cursor Start</td>
<td>Scan Line</td>
<td>B</td>
</tr>
<tr>
<td>R11</td>
<td>Cursor End</td>
<td>Scan Line</td>
<td>C</td>
</tr>
<tr>
<td>R12</td>
<td>Start Address (H)</td>
<td>--------------------</td>
<td>00</td>
</tr>
<tr>
<td>R13</td>
<td>Start Address (L)</td>
<td>--------------------</td>
<td>00</td>
</tr>
<tr>
<td>R14</td>
<td>Cursor (H)</td>
<td>--------------------</td>
<td>00</td>
</tr>
<tr>
<td>R15</td>
<td>Cursor (L)</td>
<td>--------------------</td>
<td>00</td>
</tr>
<tr>
<td>R16</td>
<td>Reserved</td>
<td>--------------------</td>
<td>--</td>
</tr>
<tr>
<td>R17</td>
<td>Reserved</td>
<td>--------------------</td>
<td>--</td>
</tr>
</tbody>
</table>

To ensure proper initialization, the first command issued to the attachment must be to send to CRT control port 1 (hex 3B8), a hex 01, to set the high-resolution mode. If this bit is not set, then the processor access to the monochrome adapter must never occur. If the high-resolution bit is not set, the processor will stop running.

System configurations that have both an IBM Monochrome Display Adapter and Printer Adapter, and an IBM Color/Graphics Monitor Adapter, must ensure that both adapters are properly initialized after a power-on reset. Damage to either display may occur if not properly initialized.
The IBM Monochrome Display and Printer Adapter supports 256 different character codes. In the character set are alphanumerics and block graphics. Each character in the display buffer has a corresponding character attribute. The character code must be an even address, and the attribute code must be an odd address in the display buffer.

<table>
<thead>
<tr>
<th>Character Code</th>
<th>Even Address (M)</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 6 5 4 3 2 1 0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attribute Code</th>
<th>Odd Address (M+1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 6 5 4 3 2 1 0</td>
<td>B L R G B I R G B</td>
</tr>
</tbody>
</table>

- Foreground
- Intensity
- Background
- Blink

The adapter decodes the character attribute byte as defined above. The blink and intensity bits may be combined with the foreground and background bits to further enhance the character attribute functions listed below.

<table>
<thead>
<tr>
<th>Background R G B</th>
<th>Foreground R G B</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td>0 0 0</td>
<td>Non-Display</td>
</tr>
<tr>
<td>0 0 0</td>
<td>0 0 1</td>
<td>Underline</td>
</tr>
<tr>
<td>0 0 0</td>
<td>1 1 1</td>
<td>White Character/Black Background</td>
</tr>
<tr>
<td>1 1 1</td>
<td>0 0 0</td>
<td>Reverse Video</td>
</tr>
</tbody>
</table>
The 4K display buffer supports one screen of 25 rows of 80 characters, plus a character attribute for each display character. The starting address of the buffer is hex B0000. The display buffer can be read from using DMA; however, at least one wait-state will be inserted by the processor. The duration of the wait-state will vary, because the processor/monitor access is synchronized with the character clock on this adapter.

Interrupt level 7 is used on the parallel interface. Interrupts can be enabled or disabled through the printer control port. The interrupt is a high-level active signal.

The figure below breaks down the functions of the I/O address decode for the adapter. The I/O address decode is from hex 3B0 through hex 3BF. The bit assignment for each I/O address follows:

<table>
<thead>
<tr>
<th>I/O Register Address</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>3B0</td>
<td>Not Used</td>
</tr>
<tr>
<td>3B1</td>
<td>Not Used</td>
</tr>
<tr>
<td>3B2</td>
<td>Not Used</td>
</tr>
<tr>
<td>3B3</td>
<td>Not Used</td>
</tr>
<tr>
<td>3B4*</td>
<td>6845 Index Register</td>
</tr>
<tr>
<td>3B5*</td>
<td>6845 Data Register</td>
</tr>
<tr>
<td>3B6</td>
<td>Not Used</td>
</tr>
<tr>
<td>3B7</td>
<td>Not Used</td>
</tr>
<tr>
<td>3B8</td>
<td>CRT Control Port 1</td>
</tr>
<tr>
<td>3B9</td>
<td>Reserved</td>
</tr>
<tr>
<td>3BA</td>
<td>CRT Status Port</td>
</tr>
<tr>
<td>3BB</td>
<td>Reserved</td>
</tr>
<tr>
<td>3BC</td>
<td>Parallel Data Port</td>
</tr>
<tr>
<td>3BD</td>
<td>Printer Status Port</td>
</tr>
<tr>
<td>3BE</td>
<td>Printer Control Port</td>
</tr>
<tr>
<td>3BF</td>
<td>Not Used</td>
</tr>
</tbody>
</table>

*The 6845 Index and Data Registers are used to program the CRT controller to interface the high-resolution IBM Monochrome Display.
<table>
<thead>
<tr>
<th>Bit Number</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>+High Resolution Mode</td>
</tr>
<tr>
<td>1</td>
<td>Not Used</td>
</tr>
<tr>
<td>2</td>
<td>Not Used</td>
</tr>
<tr>
<td>3</td>
<td>+Video Enable</td>
</tr>
<tr>
<td>4</td>
<td>Not Used</td>
</tr>
<tr>
<td>5</td>
<td>+Enable Blink</td>
</tr>
<tr>
<td>6,7</td>
<td>Not Used</td>
</tr>
</tbody>
</table>

6845 CRT Control Port 1 (Hex 3B8)

<table>
<thead>
<tr>
<th>Bit Number</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>+Horizontal Drive</td>
</tr>
<tr>
<td>1</td>
<td>Reserved</td>
</tr>
<tr>
<td>2</td>
<td>Reserved</td>
</tr>
<tr>
<td>3</td>
<td>+Black/White Video</td>
</tr>
</tbody>
</table>

6845 CRT Status Port (Hex 3BA)
At Standard TTL Levels

<table>
<thead>
<tr>
<th>IBM Monochrome Display</th>
<th>IBM Monochrome Display and Printer Adapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>+Intensity</td>
<td>+Video</td>
</tr>
<tr>
<td>+Horizontal</td>
<td>-Vertical</td>
</tr>
<tr>
<td>Ground</td>
<td>Ground</td>
</tr>
<tr>
<td>Not Used</td>
<td>Not Used</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
</tr>
</tbody>
</table>

Note: Signal voltages are 0.0 to 0.6 Vdc at down level and +2.4 to 3.5 Vdc at high level.
Notes:

1-120  Monochrome Adapter
IBM Monochrome Display

The high-resolution IBM Monochrome Display attaches to the system unit through two cables approximately 3 feet (914 millimeters) in length. One cable is a signal cable that contains the direct drive interface from the IBM Monochrome Display and Printer Adapter.

The second cable provides ac power to the display from the system unit. This allows the system-unit power switch to also control the display unit. An additional benefit is a reduction in the requirements for wall outlets to power the system. The display contains an 11-½ inch (283 millimeters), diagonal 90° deflection CRT. The CRT and analog circuits are packaged in an enclosure so the display may either sit on top of the system unit or on a nearby tabletop or desk. The unit has both brightness and contrast adjustment controls on the front surface that are easily accessible to the operator.
Operating Characteristics

Screen

- High-persistence green phosphor (P39).
- Etched surface to reduce glare.
- Size is 80 characters by 25 lines.
- Character box is 9 dots wide by 14 dots high.

Video Signal

- Maximum bandwidth of 16.257 MHz.

Vertical Drive

- Screen refreshed at 50 Hz with 350 lines of vertical resolution and 720 lines of horizontal resolution.

Horizontal Drive

- Positive-level, TTL-compatibility at a frequency of 18.432 kHz.
IBM Color/Graphics Monitor Adapter

The IBM Color/Graphics Monitor Adapter is designed to attach to the IBM Color Display, to a variety of television-frequency monitors, or to home television sets (user-supplied RF modulator is required for home television sets). The adapter is capable of operating in black-and-white or color. It provides three video interfaces: a composite-video port, a direct-drive port, and a connection interface for driving a user-supplied RF modulator. In addition, a light pen interface is provided.

The adapter 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 monitor or home television, or in 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 for lowercase characters. Both uppercase and lowercase characters are supported in all modes.

The character attributes of reverse video, blinking, and highlighting are available in the black-and-white mode. In the color mode, sixteen foreground and eight background colors are available for each character. In addition, blinking on a per-character basis is available.

The monitor adapter contains 16K bytes of storage. As an example, a 40-column by 25-row display screen uses 1000 bytes to store character information, and 1000 bytes to store attribute/color information. This would mean that up to eight display screens can be stored in the adapter memory. Similarly, in an 80-column by 25-row mode, four display screens may be stored in the adapter. The entire 16K bytes of storage on the display adapter are directly addressable by the processor, which allows maximum software flexibility in managing the screen.
In A/N color modes, it is also possible to select the color of the screen’s border. One of sixteen colors can be selected.

In the APA mode, there are two resolutions available: a medium-resolution color graphics mode (320 PELs by 200 rows) and a high-resolution black-and-white graphics mode (640 PELs by 200 rows). In the medium-resolution mode, each picture element (PEL) may have one of four colors. The background color (color 0) may be any of the 16 possible colors. The remaining three colors come from one of the two software-selectable palettes. One palette contains green/red/brown; the other contains cyan/magenta/white.

The high-resolution mode is available only in black-and-white because the entire 16K bytes of storage in the adapter is used to define the on or off state of the PELs.

The adapter operates in noninterlace mode at either 7 or 14 MHz, depending on the mode of operation selected.

In the A/N mode, 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 support (allowing drawing of charts, boxes, and tables using single and double lines)
- 16 selected Greek characters
- 15 selected scientific-notation characters
The color/graphics monitor adapter function is packaged on a single card. The direct-drive and composite-video ports are right-angle mounted connectors on the adapter, and extend through the rear panel of the unit. The direct-drive video port is a 9-pin D-shell female connector. The composite-video port is a standard female phono-jack.

The display adapter is implemented using a Motorola 6845 CRT controller device. This adapter is highly programmable with respect to raster and character parameters. Therefore, many additional modes are possible with clever programming of the adapter.

A block diagram of the color/graphics adapter is on the following page.
Color/Graphics Monitor Adapter Block Diagram
Descriptions of Major Components

Motorola 6845 CRT Controller

This device provides the necessary interface to drive a raster-scan CRT.

Mode Set Register

This is a general-purpose, programmable, I/O register. It has I/O ports that may be individually programmed. Its function in this attachment is to provide mode selection and color selection in the medium-resolution color-graphics mode.

Display Buffer

The display buffer resides in the processor-address space, starting at address hex B8000. It provides 16K bytes of dynamic read/write memory. A dual-ported implementation allows the processor and the graphics control unit to access the buffer. The processor and the CRT control unit have equal access to this buffer during all modes of operation, except in the high-resolution alphanumeric mode. In this mode, only the processor should access this buffer during the horizontal-retrace intervals. While the processor may write to the required buffer at any time, a small amount of display interference will result if this does not occur during the horizontal-retrace intervals.

Character Generator

This attachment utilizes a ROM character generator. It consists of 8K bytes of storage that cannot be read from or written to under software control. This is a general-purpose ROM character generator with three different character fonts. Two character fonts are used on the color/graphics adapter: a 7-high by 7-wide double-dot font and a 5-wide by 7-high single-dot font. The font is selected by a jumper (P3). The single-dot font is selected by inserting the jumper; the double-dot font is selected by removing the jumper.
Timing Generator

This generator produces the timing signals used by the 6845 CRT controller and by the dynamic memory. It also resolves the processor/graphic controller contentions for accessing the display buffer.

Composite Color Generator

This generator produces base band video color information.

Alphanumeric Mode

Every display-character position in the alphanumeric mode is defined by two bytes in the regen buffer (a part of the monitor adapter), not the system memory. Both the color/graphics and the monochrome display adapter use the following 2-byte character/attribute 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>

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

<table>
<thead>
<tr>
<th>Attribute Function</th>
<th>Attribute Byte</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>7 6 5 4 3 2 1 0</td>
</tr>
<tr>
<td></td>
<td>B R G B I R G B</td>
</tr>
<tr>
<td>FG Background</td>
<td>FG Background</td>
</tr>
<tr>
<td>Normal</td>
<td>B 0 0 0 I 1 1 1</td>
</tr>
<tr>
<td>Reverse Video</td>
<td>B 1 1 1 I 0 0 0</td>
</tr>
<tr>
<td>Nondisplay (Black)</td>
<td>B 0 0 0 I 0 0 0</td>
</tr>
<tr>
<td>Nondisplay (White)</td>
<td>B 1 1 1 I 1 1 1</td>
</tr>
</tbody>
</table>

I = Highlighted Foreground (Character)
B = Blinking Foreground (Character)
The attribute byte definitions are:

```
7 6 5 4 3 2 1 0
B R G B | R G B
```

Foreground Color
Intensity
Background Color
Blinking

In the alphanumeric mode, the display mode can be operated in either a low-resolution mode or a high-resolution mode.

The low-resolution alphanumeric mode has the following features:

- Supports home color televisions or low-resolution monitors
- Displays up to 25 rows of 40 characters each
- ROM character generator that contains dot patterns for a maximum of 256 different characters
- Requires 2,000 bytes of read/write memory (on the adapter)
- Character box is 8-high by 8-wide
- Two jumper-controlled character fonts are available:
  5-wide by 7-high single-dot character font with one descender
  7-wide by 7-high double-dot character font with one descender
- One character attribute for each character
The high-resolution alphanumeric mode has the following features:

- Supports the IBM Color Display or other color monitor with direct-drive input capability
- Supports a black-and-white composite-video monitor
- Displays up to 25 rows of 80 characters each
- ROM displays generator that contains dot patterns for a maximum of 256 different characters
- Requires 4,000 bytes of read/write memory (on the adapter)
- Character box is 8-high by 8-wide
- Two jumper-controlled character fonts are available: 5-wide by 7-high single-dot character font with one descender, 7-wide by 7-high double-dot character font with one descender
- One character attribute for each character

### Monochrome vs Color/Graphics Character Attributes

Foreground and background colors are defined by the attribute byte of each character, whether using the IBM Monochrome Display and Printer Adapter or the IBM Color/Graphics Monitor Adapter. The following table describes the colors for each adapter:

<table>
<thead>
<tr>
<th>Attribute Byte</th>
<th>Monochrome Display Adapter</th>
<th>Color/Graphics Monitor Adapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>FG Background</td>
<td>Foreground</td>
<td>Background Color</td>
</tr>
<tr>
<td>B 0 0 0</td>
<td>I 1 1 1</td>
<td>Black</td>
</tr>
<tr>
<td>B 1 1 1</td>
<td>I 0 0 0</td>
<td>White</td>
</tr>
<tr>
<td>B 0 0 0</td>
<td>I 0 0 0</td>
<td>Black</td>
</tr>
<tr>
<td>B 1 1 1</td>
<td>I 1 1 1</td>
<td>White</td>
</tr>
</tbody>
</table>
The monochrome display adapter will produce white characters on a white background with any other code. The color/graphics adapter will change foreground and background colors according to the color value selected. The color values for the various red, green, blue, and intensity bit settings are given in the table below.

<table>
<thead>
<tr>
<th>R</th>
<th>G</th>
<th>B</th>
<th>I</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>1</td>
<td>0</td>
<td>Blue</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Green</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Cyan</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Red</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Magenta</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Brown</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>White</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Gray</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Light Blue</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Light Green</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Light Cyan</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Light Red</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Light Magenta</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Yellow</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>White (High Intensity)</td>
</tr>
</tbody>
</table>

Code written with an underline attribute for the IBM Monochrome Display, when executed on a color/graphics monitor adapter, will result in a blue character where the underline attribute is encountered. Also, code written on a color/graphics monitor adapter with blue characters will be displayed as white characters on a black background, with a white underline on the IBM Monochrome Display.

Remember that not all monitors recognize the intensity (I) bit.
Graphics Mode

The IBM Color/Graphics Monitor Adapter has three modes available within the graphics mode. They are low-resolution color graphics, medium-resolution color graphics, and high-resolution color graphics. However, only medium- and high-resolution graphics are supported in ROM. The following table summarizes the three modes.

<table>
<thead>
<tr>
<th></th>
<th>Horizontal (PELs)</th>
<th>Vertical (Rows)</th>
<th>Number of Colors Available (Includes Background Color)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low Resolution</td>
<td>160</td>
<td>100</td>
<td>16 (Includes black-and-white)</td>
</tr>
<tr>
<td>Medium Resolution</td>
<td>320</td>
<td>200</td>
<td>4 Colors Total</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 of 16 for Background and</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 of Green, Red, or Brown or</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 of Cyan, Magenta, or White</td>
</tr>
<tr>
<td>High Resolution</td>
<td>640</td>
<td>200</td>
<td>Black-and-white only</td>
</tr>
</tbody>
</table>

Low-Resolution Color-Graphics Mode

The low-resolution mode supports home television or color monitors. This mode is not supported in ROM. It has the following features:

- Contains a maximum of 100 rows of 160 PELs, with each PEL being 2-high by 2-wide
- Specifies 1 of 16 colors for each PEL by the I, R, G, and B bits
- Requires 16,000 bytes of read/write memory (on the adapter)
- Uses memory-mapped graphics

1-132 Color Graphics Adapter
Medium-Resolution Color-Graphics Mode

The medium-resolution mode supports home televisions or color monitors. It has the following features:

- Contains a maximum of 200 rows of 320 PELs, with each PEL being 1-high by 1-wide
- Preselects one of four colors for each PEL
- Requires 16,000 bytes of read/write memory (on the adapter)
- Uses memory-mapped graphics
- Formats 4 PELs per byte in the following manner:

<table>
<thead>
<tr>
<th>7 6</th>
<th>5 4</th>
<th>3 2</th>
<th>1 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>C0</td>
<td>C1</td>
<td>C0</td>
</tr>
<tr>
<td>First Display PEL</td>
<td>Second Display PEL</td>
<td>Third Display PEL</td>
<td>Fourth Display PEL</td>
</tr>
</tbody>
</table>

- Organizes graphics storage in two banks of 8,000 bytes, using the following format:

<table>
<thead>
<tr>
<th>Memory Address (in hex)</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>B8000</td>
<td>Even Scans (0,2,4,...198) 8,000 bytes</td>
</tr>
<tr>
<td>B9F3F</td>
<td>Not Used</td>
</tr>
<tr>
<td>BA000</td>
<td>Odd Scans (1,3,5...199) 8,000 Bytes</td>
</tr>
<tr>
<td>BBF3F</td>
<td>Not Used</td>
</tr>
<tr>
<td>BBFFF</td>
<td></td>
</tr>
</tbody>
</table>

Address hex B8000 contains PEL instruction for the upper-left corner of the display area.
• Color selection is determined by the following logic:

<table>
<thead>
<tr>
<th>C1</th>
<th>C0</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Dot takes on the color of 1 of 16 preselected background colors</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Selects first color of preselected Color Set 1 or Color Set 2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Selects second color of preselected Color Set 1 or Color Set 2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Selects third color of preselected Color Set 1 or Color Set 2</td>
</tr>
</tbody>
</table>

C1 and C0 will select 4 of 16 preselected colors. This color selection (palette) is preloaded in an I/O port.

The two colors are:

<table>
<thead>
<tr>
<th>Color Set 1</th>
<th>Color Set 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color 1 is Green</td>
<td>Color 1 is Cyan</td>
</tr>
<tr>
<td>Color 2 is Red</td>
<td>Color 2 is Magenta</td>
</tr>
<tr>
<td>Color 3 is Brown</td>
<td>Color 3 is White</td>
</tr>
</tbody>
</table>

The background colors are the same basic 8 colors as defined for low-resolution graphics, plus 8 alternate intensities defined by the intensity bit, for a total of 16 colors, including black and white.
High-Resolution Black-and-White Graphics Mode

The high-resolution mode supports color monitors. This mode has the following features:

- Contains a maximum of 200 rows of 640 PELs, with each PEL being 1-high by 1-wide.
- Supports black-and-white mode only.
- Requires 16,000 bytes of read/write memory (on the adapter).
- Addressing and mapping procedures are the same as medium-resolution color graphics, but the data format is different. In this mode, each bit in memory is mapped to a PEL on the screen.
- Formats 8 PELs per byte in the following manner:
Description of Basic Operations

In the alphanumeric mode, the adapter fetches character and attribute information from its display buffer. The starting address of the display buffer is programmable through the 6845, but it must be an even address. The character codes and attributes are then displayed according to their relative positions in the buffer.

<table>
<thead>
<tr>
<th>Memory Address (in hex)</th>
<th>Display Buffer</th>
</tr>
</thead>
<tbody>
<tr>
<td>B8000</td>
<td>Character Code A</td>
</tr>
<tr>
<td>B8001</td>
<td>Attribute A</td>
</tr>
<tr>
<td>B8002</td>
<td>Character Code B</td>
</tr>
<tr>
<td>B8003</td>
<td>Attribute B</td>
</tr>
<tr>
<td>B87CE</td>
<td>Character Code X</td>
</tr>
<tr>
<td>B87CF</td>
<td>Attribute X</td>
</tr>
</tbody>
</table>

(Example of a 40 by 25 Screen)

The processor and the display control unit have equal access to the display buffer during all the operating modes, except the high-resolution alphanumeric mode. During this mode, the processor should access the display buffer during the vertical retrace time. If it does not, the display will be affected with random patterns as the processor is using the display buffer. In the alphanumeric mode, the characters are displayed from a prestored ROM character generator that contains the dot patterns of all the displayable characters.

In the graphics mode, the displayed dots and colors (up to 16K bytes) are also fetched from the display buffer. The bit configuration for each graphics mode is explained in “Graphics Mode.”
Summary of Available Colors

Programming Considerations

Programming the 6845 CRT Controller

The 6845 has 19 accessible internal registers, which are used to define and control a raster-scan 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>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>White</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>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>Light Red</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>High Intensity White</td>
</tr>
</tbody>
</table>

Note: ‘I’ 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.
<table>
<thead>
<tr>
<th>Address Register</th>
<th>Register Number</th>
<th>Register Type</th>
<th>Units</th>
<th>I/O</th>
<th>40 by 25 Alpha-numeric</th>
<th>80 by 25 Alpha-numeric</th>
<th>Graphic Modes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>R0</td>
<td>Horizontal Total</td>
<td>Character</td>
<td>Write</td>
<td>38</td>
<td>71</td>
<td>38</td>
</tr>
<tr>
<td>1</td>
<td>R1</td>
<td>Horizontal Displayed</td>
<td>Character</td>
<td>Write</td>
<td>28</td>
<td>50</td>
<td>28</td>
</tr>
<tr>
<td>2</td>
<td>R2</td>
<td>Horizontal Sync Position</td>
<td>Character</td>
<td>Write</td>
<td>2D</td>
<td>5A</td>
<td>2D</td>
</tr>
<tr>
<td>3</td>
<td>R3</td>
<td>Horizontal Sync Width</td>
<td>Character</td>
<td>Write</td>
<td>0A</td>
<td>0A</td>
<td>0A</td>
</tr>
<tr>
<td>4</td>
<td>R4</td>
<td>Vertical Total</td>
<td>Character Row</td>
<td>Write</td>
<td>1F</td>
<td>1F</td>
<td>7F</td>
</tr>
<tr>
<td>5</td>
<td>R5</td>
<td>Vertical Total Adjust</td>
<td>Scan Line</td>
<td>Write</td>
<td>06</td>
<td>06</td>
<td>06</td>
</tr>
<tr>
<td>6</td>
<td>R6</td>
<td>Vertical Displayed</td>
<td>Character Row</td>
<td>Write</td>
<td>19</td>
<td>19</td>
<td>64</td>
</tr>
<tr>
<td>7</td>
<td>R7</td>
<td>Vertical Sync Position</td>
<td>Character Row</td>
<td>Write</td>
<td>1C</td>
<td>1C</td>
<td>70</td>
</tr>
<tr>
<td>8</td>
<td>R8</td>
<td>Interlace Mode</td>
<td>-</td>
<td>Write</td>
<td>02</td>
<td>02</td>
<td>02</td>
</tr>
<tr>
<td>9</td>
<td>R9</td>
<td>Maximum Scan Line Address</td>
<td>Scan Line</td>
<td>Write</td>
<td>07</td>
<td>07</td>
<td>01</td>
</tr>
<tr>
<td>A</td>
<td>R10</td>
<td>Cursor Start</td>
<td>Scan Line</td>
<td>Write</td>
<td>06</td>
<td>06</td>
<td>06</td>
</tr>
<tr>
<td>B</td>
<td>R11</td>
<td>Cursor End</td>
<td>Scan Line</td>
<td>Write</td>
<td>07</td>
<td>07</td>
<td>07</td>
</tr>
<tr>
<td>C</td>
<td>R12</td>
<td>Start Address (H)</td>
<td>-</td>
<td>Write</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>D</td>
<td>R13</td>
<td>Start Address (L)</td>
<td>-</td>
<td>Write</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>E</td>
<td>R14</td>
<td>Cursor Address (H)</td>
<td>-</td>
<td>Read/Write</td>
<td>XX</td>
<td>XX</td>
<td>XX</td>
</tr>
<tr>
<td>F</td>
<td>R15</td>
<td>Cursor Address (L)</td>
<td>-</td>
<td>Read/Write</td>
<td>XX</td>
<td>XX</td>
<td>XX</td>
</tr>
<tr>
<td>10</td>
<td>R16</td>
<td>Light Pen (H)</td>
<td>-</td>
<td>Read Only</td>
<td>XX</td>
<td>XX</td>
<td>XX</td>
</tr>
<tr>
<td>11</td>
<td>R17</td>
<td>Light Pen (L)</td>
<td>-</td>
<td>Read Only</td>
<td>XX</td>
<td>XX</td>
<td>XX</td>
</tr>
</tbody>
</table>

Note: All register values are given in hexadecimal

6845 Register Description

1-138  Color Graphics Adapter
Programming the Mode Control and Status Register

The following I/O devices are defined on the color/graphics adapter.

<table>
<thead>
<tr>
<th>Hex Address</th>
<th>A9 A8 A7 A6 A5 A4 A3 A2 A1 AO</th>
<th>Function of Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>3D8</td>
<td>1 1 1 1 0 1 1 0 0 0</td>
<td>Mode Control Register (D0)</td>
</tr>
<tr>
<td>3D9</td>
<td>1 1 1 1 0 1 1 0 0 1</td>
<td>Color Select Register (D0)</td>
</tr>
<tr>
<td>3DA</td>
<td>1 1 1 1 0 1 1 0 1 0</td>
<td>Status Register (D1)</td>
</tr>
<tr>
<td>3DB</td>
<td>1 1 1 1 0 1 1 0 1 1</td>
<td>Clear Light Pen Latch</td>
</tr>
<tr>
<td>3DC</td>
<td>1 1 1 1 0 1 1 1 0 0</td>
<td>Preset Light Pen Latch</td>
</tr>
<tr>
<td>3D4</td>
<td>1 1 1 1 0 1 0 Z Z 0</td>
<td>6845 Index Register</td>
</tr>
<tr>
<td>3D5</td>
<td>1 1 1 1 0 1 0 Z Z 1</td>
<td>6845 Data Register</td>
</tr>
<tr>
<td>3D0</td>
<td>1 1 1 1 0 1 0 Z Z 0</td>
<td>6845 Registers</td>
</tr>
<tr>
<td>3D1</td>
<td>1 1 1 1 0 1 0 Z Z 1</td>
<td>6845 Registers</td>
</tr>
</tbody>
</table>

Z = don't care condition
Color-Select Register

This is a 6-bit output-only register (cannot be read). Its I/O address is hex 3D9, and it can be written to by using the 8088 I/O Out command.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Selects B (Blue) Border Color in 40 x 25 Alphanumeric Mode</th>
<th>Selects B (Blue) Background Color in 320 x 200 Graphics Mode</th>
<th>Selects B (Blue) Foreground Color in 640 x 200 Graphics Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 0</td>
<td>Selects G (Green) Border Color in 40 x 25 Alphanumeric Mode</td>
<td>Selects G (Green) Background Color in 320 x 200 Graphics Mode</td>
<td>Selects G (Green) Foreground Color in 640 x 200 Graphics Mode</td>
</tr>
<tr>
<td>Bit 1</td>
<td>Selects R (Red) Border Color in 40 x 25 Alphanumeric Mode</td>
<td>Selects R (Red) Background Color in 320 x 200 Graphics Mode</td>
<td>Selects R (Red) Foreground Color in 640 x 200 Graphics Mode</td>
</tr>
<tr>
<td>Bit 2</td>
<td>Selects I (Intensified) Border Color in 40 x 25 Alphanumeric Mode</td>
<td>Selects I (Intensified) Background Color in 320 x 200 Graphics Mode</td>
<td>Selects I (Intensified) Foreground Color in 640 x 200 Graphics Mode</td>
</tr>
<tr>
<td>Bit 3</td>
<td>Selects Alternate, Intensified Set of Colors in Graphics Mode</td>
<td>Selects Background Colors in the Alphanumeric Mode</td>
<td></td>
</tr>
<tr>
<td>Bit 4</td>
<td>Selects Active Color Set in 320 x 200 Graphics Mode</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 5</td>
<td>Not Used</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 6</td>
<td>Not Used</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 0, 1, 2, 3 These bits select the screen’s border color in the 40 x 25 alphanumeric mode. They select the screen’s background color (C0-C1) in the medium-resolution (320 by 200) color-graphics mode.

Bits 4 This bit, when set, will select an alternate, intensified set of colors. Selects background colors in the alphanumeric mode.

Bit 5 This bit is only used in the medium-resolution (320 by 200) color-graphics mode. It is used to select the active set of screen colors for the display.
When bit 5 is set to 1, colors are determined as follows:

<table>
<thead>
<tr>
<th>C1</th>
<th>C0</th>
<th>Set Selected</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Background (Defined by bits 0-3 of port hex 3D9)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Cyan</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Magenta</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>White</td>
</tr>
</tbody>
</table>

When bit 5 is set to 0, colors are determined as follows:

<table>
<thead>
<tr>
<th>C1</th>
<th>C0</th>
<th>Set Selected</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Background (Defined by bits 0-3 of port hex 3D9)</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>Brown</td>
</tr>
</tbody>
</table>

Mode-Select Register

This is a 6-bit output-only register (cannot be read). Its I/O address is hex 3D8, and it can be written to using the 8088 I/O Out command.

The following is a description of the register’s functions:

<table>
<thead>
<tr>
<th>Bit 0</th>
<th>80 x 25 Alphanumeric Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 1</td>
<td>Graphics Select</td>
</tr>
<tr>
<td>Bit 2</td>
<td>Black/White Select</td>
</tr>
<tr>
<td>Bit 3</td>
<td>Enable Video Signal</td>
</tr>
<tr>
<td>Bit 4</td>
<td>High-Resolution (640 x 200) Black/White Mode</td>
</tr>
<tr>
<td>Bit 5</td>
<td>Change Background Intensity to Blink Bit</td>
</tr>
<tr>
<td>Bit 6</td>
<td>Not Used</td>
</tr>
<tr>
<td>Bit 7</td>
<td>Not Used</td>
</tr>
</tbody>
</table>
Bit 0   A 1 selects 80 by 25 alphanumeric mode
        A 0 selects 40 by 25 alphanumeric mode

Bit 1   A 1 selects 320 by 200 graphics mode
        A 0 selects alphanumeric mode

Bit 2   A 1 selects black-and-white mode
        A 0 selects color mode

Bit 3   A 1 enables the video signal at certain times when modes are being changed. The video signal should be disabled when changing modes.

Bit 4   A 1 selects the high-resolution (640 by 200) black-and-white graphics mode. One color of 8 can be selected on direct-drive sets in this mode by using register hex 3D9.

Bit 5   When on, this bit will change the character background intensity to the blinking attribute function for alphanumeric modes. When the high-order attribute bit is not selected, 16 background colors (or intensified colors) are available. For normal operation, this bit should be set to 1 to allow the blinking function.
Mode Register Summary

<table>
<thead>
<tr>
<th>Bits</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>z</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>z</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>z</td>
</tr>
</tbody>
</table>

- **0 x 25 Alphanumeric Black-and-White**
- **40 x 25 Alphanumeric Color**
- **80 x 25 Alphanumeric Black-and-White**
- **80 x 25 Alphanumeric Color**
- **320 x 200 Black-and-White Graphics**
- **320 x 200 Color Graphics**
- **640 x 200 Black-and-White Graphics**

\[ z = \text{don't care condition} \]

**Note:** The low-resolution (160 by 100) mode requires special programming and is set up as the 40 by 25 alphanumeric mode.

Status Register

The status register is a 4-bit read-only register. Its I/O address is hex 3DA, and it can be read using the 8088 I/O In instruction. The following is a description of the register functions:

<table>
<thead>
<tr>
<th>Bit 0</th>
<th>Display Enable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 1</td>
<td>Light-Pen Trigger Set</td>
</tr>
<tr>
<td>Bit 2</td>
<td>Light-Pen Switch Made</td>
</tr>
<tr>
<td>Bit 3</td>
<td>Vertical Sync</td>
</tr>
<tr>
<td>Bit 4</td>
<td>Not Used</td>
</tr>
<tr>
<td>Bit 5</td>
<td>Not Used</td>
</tr>
<tr>
<td>Bit 6</td>
<td>Not Used</td>
</tr>
<tr>
<td>Bit 7</td>
<td>Not Used</td>
</tr>
</tbody>
</table>
Bit 0  This bit, when active, indicates that a regen buffer memory access can be made without interfering with the display.

Bit 1  This bit, when active, indicates that a positive-going edge from the light-pen has set the light pen's trigger. This trigger is reset upon power-on and may also be cleared by performing an I/O Out command to hex address 3DB. No specific data setting is required; the action is address-activated.

Bit 2  The light-pen switch status is reflected in this status bit. The switch is not latched or debounced. A 0 indicates that the switch is on.

Bit 3  This bit, when active, indicates that the raster is in a vertical retrace mode. This is a good time to perform screen-buffer updating.

Sequence of Events for Changing Modes

1. Determine the mode of operation.

2. Reset 'video enable' bit in mode-select register.

3. Program 6845 to select mode.

4. Program mode/color select registers including re-enabling video.
### Memory Requirements

The memory used by this adapter is self-contained. It consists of 16K bytes of memory without parity. This memory is used as both a display buffer for alphanumeric data and as a bit map for graphics data. The regen buffer’s address starts at hex B8000.

<table>
<thead>
<tr>
<th>Read/Write Memory Address Space (in hex)</th>
<th>128K Reserved Regen Area</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System Read/Write Memory</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Display Buffer (16K Bytes)</strong></td>
<td></td>
</tr>
</tbody>
</table>

- **01000**
- **A0000**
- **B8000**
- **BC000**
- **C0000**
### IBM Color Display or other Direct-Drive Monitor

At Standard TTL Levels

<table>
<thead>
<tr>
<th>Ground</th>
<th>1</th>
</tr>
</thead>
<tbody>
<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>Reserved</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>

### Composite Phono Jack Hookup to Monitor

- **1**: Composite Video Signal of Approximately 1.5 Volts Peak to Peak Amplitude
- **2**: Chassis Ground

### Connector Specifications (Part 1 of 2)
P1 (4-Pin Berg Strip) for RF Modulator

P2 (6-Pin Berg Strip) for Light-Pen Connector

Color/Graphics Adapter

RF Modulator Interface

+12 Volts 1
(key) Not Used 2
Composite Video Output 3
Logic Ground 4

Light Pen Interface

-light Pen Input 1
(key) Not Used 2
-light Pen Switch 3
Chassis Ground 4
+5 Volts 5
+12 Volts 6

Color/Graphics Adapter

Connector Specifications (Part 2 of 2)
Notes:
IBM Color Display

The IBM Color Display attaches to the system unit by a signal cable that is approximately 5 feet (1.5 meters) in length. This signal cable provides a direct-drive interface from the IBM Color/Graphics Monitor Adapter.

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, or 220-volt 50-Hz power. The power supply in the display automatically switches to match the applied power.

The display has a 13-inch (340 millimeters) CRT. The CRT and analog circuits are packaged in an enclosure so the display may sit either on top of the system unit or on a nearby tabletop or desk. 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, when used with the IBM Color/Graphics Monitor Adapter.
- Characters defined in an 8-high by 8-wide matrix.

Video Signal

- Maximum video bandwidth of 14 MHz.
- Red, green, and blue video signals and intensity are all independent.

Vertical Drive

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

Horizontal Drive

- Positive-level, TTL-compatibility, at a frequency of 15.75 kHz.
IBM 5-1/4" Diskette Drive Adapter

The 5-1/4 inch diskette drive adapter fits into one of the expansion slots in the system unit. It attaches to one or two diskette drives through an internal, daisy-chained flat cable that connects to one end of the drive adapter. The adapter has a connector at the other end that extends through the rear panel of the system unit. This connector has signals for two additional external diskette drives; thus, the 5-1/4 inch diskette drive adapter can attach four 5-1/4 inch drives – two internal and two external.

The adapter is designed for double-density, MFM-coded, diskette drives and uses write precompensation with an analog phase-lock loop for clock and data recovery. The adapter is a general-purpose device using the NEC μPD765 compatible controller. Therefore, the diskette drive parameters are programmable. 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 board’s direct memory access (DMA) for record data transfers. An interrupt level is also used to indicate when an operation is complete and that a status condition requires processor attention.

In general, the 5-1/4 inch diskette drive adapter presents a high-level command interface to software I/O drivers. A block diagram of the 5-1/4 inch diskette drive adapter is on the following page.
5-1/4 Inch Diskette Drive Adapter Block Diagram
Functional Description

From a programming point of view, this attachment consists of an 8-bit digital-output register in parallel with an NEC μPD765 or equivalent floppy disk controller (FDC).

In the following description, drive numbers 0, 1, 2, and 3 are equivalent to drives A, B, C, and D.

Digital-Output Register

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

Bits 0 and 1 These bits are decoded by the hardware to select one drive if its motor is on:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Drive</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0</td>
<td>0 (A)</td>
</tr>
<tr>
<td>0 1</td>
<td>1 (B)</td>
</tr>
<tr>
<td>1 0</td>
<td>2 (C)</td>
</tr>
<tr>
<td>1 1</td>
<td>3 (D)</td>
</tr>
</tbody>
</table>

Bit 2 The FDC is held reset when this bit is clear. It must be set by the program to enable the FDC.

Bit 3 This bit allows the FDC interrupt and DMA requests to be gated onto the I/O interface. If this bit is cleared, the interrupt and DMA request I/O interface drivers are disabled.

Bits 4, 5, 6, and 7 These bits control, respectively, the motors of drives 0, 1, 2 (A, B, C), and 3 (D). If a bit is clear, the associated motor is off, and the drive cannot be selected.
Floppy Disk Controller

The floppy disk controller (FDC) contains two registers that may be accessed by the main system processor: a status register and a data register. The 8-bit main status register contains the status information of the FDC and may be accessed at any time. The 8-bit data register (actually consisting of several registers in a stack with only one register presented to the data bus at a time) 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 may only be read and is used to facilitate the transfer of data between the processor and FDC.

The bits in the main status register (hex 34F) are defined as follows:

<table>
<thead>
<tr>
<th>Bit Number</th>
<th>Name</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB0</td>
<td>FDD A Busy</td>
<td>DAB</td>
<td>FDD number 0 is in the Seek mode.</td>
</tr>
<tr>
<td>DB1</td>
<td>FDD B Busy</td>
<td>DBB</td>
<td>FDD number 1 is in the Seek mode.</td>
</tr>
<tr>
<td>DB2</td>
<td>FDD C Busy</td>
<td>DCB</td>
<td>FDD number 2 is in the Seek mode.</td>
</tr>
<tr>
<td>DB3</td>
<td>FDD D Busy</td>
<td>DDB</td>
<td>FDD number 3 is in the Seek mode.</td>
</tr>
<tr>
<td>DB4</td>
<td>FDC Busy</td>
<td>CB</td>
<td>A read or write command is in process.</td>
</tr>
<tr>
<td>DB5</td>
<td>Non-DMA Mode</td>
<td>NDM</td>
<td>The FDC is in the non-DMA mode.</td>
</tr>
<tr>
<td>DB6</td>
<td>Data Input/Output</td>
<td>DIO</td>
<td>Indicates direction of data transfer between FDC and processor. If DIO = “1”, then transfer is from FDC data register to the processor. If DIO = “0”, then transfer is from the processor to the FDC data register.</td>
</tr>
<tr>
<td>DB7</td>
<td>Request for Master</td>
<td>RQM</td>
<td>Indicates data register is ready to send or receive data to or from the processor. Both bits DIO and RQM should be used to perform the handshaking functions of “ready” and “direction” to the processor.</td>
</tr>
</tbody>
</table>

1-154  Diskette Adapter
The FDC is capable of performing 15 different commands. Each command is initiated by a multi-byte transfer from the processor, and the result after execution of the command may also be a multi-byte transfer back to the processor. Because of this multi-byte interchange of information between the FDC and the processor, it is convenient to consider each command as consisting of three phases:

<table>
<thead>
<tr>
<th>Command Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>The FDC receives all information required to perform a particular operation from the processor.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Execution Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>The FDC performs the operation it was instructed to do.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Result Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>After completion of the operation, status and other housekeeping information is made available to the processor.</td>
</tr>
</tbody>
</table>
Programming Considerations

The following tables define the symbols used in the command summary, which follows.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AO</td>
<td>Address Line 0</td>
<td>AO controls selection of main status register (AO = 0) or data register (AO = 1).</td>
</tr>
<tr>
<td>C</td>
<td>Cylinder Number</td>
<td>C stands for the current/selected cylinder (track) number of the medium.</td>
</tr>
<tr>
<td>D</td>
<td>Data</td>
<td>D stands for the data pattern that is going to be written into a sector.</td>
</tr>
<tr>
<td>D7-D0</td>
<td>Data Bus</td>
<td>8-bit data bus, where D7 stands for a most significant bit, and D0 stands for a least significant bit.</td>
</tr>
<tr>
<td>DTL</td>
<td>Data Length</td>
<td>When N is defined as 00, DTL stands for the data length that users are going to read from or write to the sector.</td>
</tr>
<tr>
<td>EOT</td>
<td>End of Track</td>
<td>EOT stands for the final sector number on a cylinder.</td>
</tr>
<tr>
<td>GPL</td>
<td>Gap Length</td>
<td>GPL stands for the length of gap 3 (spacing between sectors excluding VCO sync field).</td>
</tr>
<tr>
<td>H</td>
<td>Head Address</td>
<td>H stands for head number 0 or 1, as specified in ID field.</td>
</tr>
<tr>
<td>HD</td>
<td>Head</td>
<td>HD stands for a selected head number 0 or 1. (H = HD in all command words.)</td>
</tr>
<tr>
<td>HLT</td>
<td>Head Load Time</td>
<td>HLT stands for the head load time in the FDD (4 to 512 ms in 4-ms increments).</td>
</tr>
<tr>
<td>HUT</td>
<td>Head Unload Time</td>
<td>HUT stands for the head unload time after a read or write operation has occurred (0 to 480 ms in 32-ms increments).</td>
</tr>
<tr>
<td>MF</td>
<td>FM or MFM Mode</td>
<td>If MF is low, FM mode is selected; if it is high, MFM mode is selected only if MFM is implemented.</td>
</tr>
<tr>
<td>MT</td>
<td>Multi-Track</td>
<td>If MT is high, a multi-track operation is to be performed. (A cylinder under both HD0 and HD1 will be read or written.)</td>
</tr>
<tr>
<td>N</td>
<td>Number</td>
<td>N stands for the number of data bytes written in a sector.</td>
</tr>
</tbody>
</table>

Symbol Descriptions (Part 1 of 2)

1-156 Diskette Adapter
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NCN</td>
<td>New Cylinder Number</td>
<td>NCN stands for a new cylinder number, which is going to be reached as a result of the seek operation. (Desired position of the head.)</td>
</tr>
<tr>
<td>ND</td>
<td>Non-DMA Mode</td>
<td>ND stands for operation in the non-DMA mode.</td>
</tr>
<tr>
<td>PCN</td>
<td>Present Cylinder Number</td>
<td>PCN stands for cylinder number at the completion of sense-interrupt-status command indicating the position of the head at present time.</td>
</tr>
<tr>
<td>R</td>
<td>Record</td>
<td>R stands for the sector number, which will be read or written.</td>
</tr>
<tr>
<td>R/W</td>
<td>Read/Write</td>
<td>R/W stands for either read (R) or write (W) signal.</td>
</tr>
<tr>
<td>SC</td>
<td>Sector</td>
<td>SC indicates the number of sectors per cylinder.</td>
</tr>
<tr>
<td>SK</td>
<td>Skip</td>
<td>SK stands for skip deleted-data address mark.</td>
</tr>
<tr>
<td>SRT</td>
<td>Step Rate Time</td>
<td>SRT stands for the stepping rate for the FDD (2 to 32 ms in 2-ms increments).</td>
</tr>
<tr>
<td>ST 0</td>
<td>Status 0</td>
<td>ST 0-3 stand for one of four registers that store the status information after a command has been executed. This information is available during the result phase after command execution. These registers should not be confused with the main status register (selected by AO = 0). ST 0-3 may be read only after a command has been executed and contain information relevant to that particular command.</td>
</tr>
<tr>
<td>ST 1</td>
<td>Status 1</td>
<td></td>
</tr>
<tr>
<td>ST 2</td>
<td>Status 2</td>
<td></td>
</tr>
<tr>
<td>ST 3</td>
<td>Status 3</td>
<td></td>
</tr>
<tr>
<td>STP</td>
<td>Scan Test</td>
<td>During a scan operation, if STP = 1, the data in contiguous sectors is compared byte-by-byte with data sent from the processor (or DMA), and if STP = 2, then alternate sectors are read and compared.</td>
</tr>
<tr>
<td>US0,</td>
<td>Unit Select</td>
<td>US stands for a selected drive number encoded the same as bits 0 and 1 of the digital output register (DOR).</td>
</tr>
<tr>
<td>US1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Symbol Descriptions (Part 2 of 2)**

*Diskette Adapter* 1-157
## Command Summary

In the following table, 0 indicates "logical 0" for that bit, 1 means "logical 1," and X means "don’t care."

<table>
<thead>
<tr>
<th>Phase</th>
<th>R/W</th>
<th>D7 D6 D5 D4 D3 D2 D1 D0</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command</td>
<td>W</td>
<td>MT MF SK 0 0 1 1 0</td>
<td>Command Codes</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>X X X X X HD US1 US0</td>
<td>Sector ID information prior to command</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>C</td>
<td>execution.</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>EOT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>GPL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>DTL</td>
<td></td>
</tr>
<tr>
<td>Execution</td>
<td>R</td>
<td>ST 0</td>
<td>Data transfer between the FDD and main system.</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>ST 1</td>
<td>Status information after command execution.</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>ST 2</td>
<td>Sector ID information after command execution.</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>Result</td>
<td>R</td>
<td>ST 0</td>
<td>Command Codes</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>ST 1</td>
<td>Sector ID information prior to command</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>ST 2</td>
<td>execution.</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>N</td>
<td></td>
</tr>
</tbody>
</table>

### Read Deleted Data

| Command    | W   | MT MF SK 0 1 1 0 0      | Command Codes                                |
|            | W   | X X X X HD US1 US0      | Sector ID information prior to command       |
|            | W   | C                       | execution.                                   |
|            | W   | H                       |                                              |
|            | W   | R                       |                                              |
|            | W   | N                       |                                              |
|            | W   | EOT                     |                                              |
|            | W   | GPL                     |                                              |
|            | W   | DTL                     |                                              |
| Execution  | R   | ST 0                    | Data transfer between the FDD and main system.|
|            | R   | ST 1                    | Status information after command execution.  |
|            | R   | ST 2                    | Sector ID information after command execution.|
|            | R   | C                       |                                              |
|            | R   | H                       |                                              |
|            | R   | R                       |                                              |
|            | R   | N                       |                                              |

---

1-158  Diskette Adapter
<table>
<thead>
<tr>
<th>Phase</th>
<th>R/W</th>
<th>Data Bus</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Command</strong></td>
<td>W</td>
<td>D7 D6 D5 D4 D3 D2 D1 D0</td>
<td>Command Codes</td>
</tr>
<tr>
<td>Write Data</td>
<td>W</td>
<td>MT MF 0 0 0 1 0 1 0 1</td>
<td>Sector ID information to command execution.</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>X X X X X X HD US1 US0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>EOT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>GPL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>DTL</td>
<td></td>
</tr>
</tbody>
</table>

| Execution     |     |          | Data transfer between the main system and FDD. |
|               | R   | ST 0     | Status information after command execution. |
|               | R   | ST 1     |                              |
|               | R   | ST 2     |                              |
|               | R   | C         | Sector ID information after command execution. |
|               | R   | H         |                              |
|               | R   | R         |                              |
|               | R   | N         |                              |

| Result        | R   | ST 0     | Status ID information after command execution. |
|               | R   | ST 1     |                              |
|               | R   | ST 2     |                              |
|               | R   | C         | Sector ID information after command execution. |
|               | R   | H         |                              |
|               | R   | R         |                              |
|               | R   | N         |                              |

| **Command**   | W   | MT MF 0 0 0 1 0 0 1 0 0 1 0 1 | Command Codes                |
| Write Deleted Data | W   | X X X X X X X HD US1 US0 | Sector ID information prior to command execution. |
|               | W   | C         |                              |
|               | W   | H         |                              |
|               | W   | R         |                              |
|               | W   | N         |                              |
|               | W   | EOT       |                              |
|               | W   | GPL       |                              |
|               | W   | DTL       |                              |

<p>| Execution     |     |          | Data transfer between the FDD and main system. |
|               | R   | ST 0     | Status ID information after command execution. |
|               | R   | ST 1     |                              |
|               | R   | ST 2     |                              |
|               | R   | C         |                              |
|               | R   | H         |                              |
|               | R   | R         |                              |
|               | R   | N         |                              |</p>
<table>
<thead>
<tr>
<th>Phase</th>
<th>R/W</th>
<th>Data Bus</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command</td>
<td>W</td>
<td>0 MF SK 0 0 0 1 0</td>
<td>Command Codes</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>X X X X X HD US1 US0</td>
<td>Sector ID information prior to command execution.</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>EOT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>GPL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>DTL</td>
<td></td>
</tr>
<tr>
<td>Execution</td>
<td></td>
<td></td>
<td>Data transfer between the FDD and main system. FDC reads all of cylinder’s contents from index hole to EOT.</td>
</tr>
<tr>
<td>Result</td>
<td>R</td>
<td>ST 0</td>
<td>Status information after command execution.</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>ST 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>ST 2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>Command</td>
<td>W</td>
<td>0 MF 0 0 1 0 1 0</td>
<td>Command Codes</td>
</tr>
<tr>
<td>Execution</td>
<td></td>
<td>X X X X X HD US1 US0</td>
<td>The first correct ID information on the cylinder is stored in data register.</td>
</tr>
<tr>
<td>Result</td>
<td>R</td>
<td>ST 0</td>
<td>Status information after command execution.</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>ST 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>ST 2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Sector ID information during execution phase.</td>
</tr>
</tbody>
</table>

1-160 Diskette Adapter
<table>
<thead>
<tr>
<th>Phase</th>
<th>R/W</th>
<th>Data Bus</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command</td>
<td>W</td>
<td>D7 D6 D5 D4 D3 D2 D1 D0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Format a Track</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>W 0 MF 0 0 1 1 0 0</td>
<td>Command Codes</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>X X X X X HD US1 US0</td>
<td>Bytes/Sector</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>N</td>
<td>Sector/Track</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>SC</td>
<td>Gap 3</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>GPL</td>
<td>FDC formats an entire cylinder.</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>D</td>
<td>fill byte.</td>
</tr>
<tr>
<td>Execution</td>
<td>R</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Result</td>
<td>R</td>
<td>ST 0</td>
<td>Status information</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>ST 1</td>
<td>after command execution.</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>ST 2</td>
<td>In this case, the ID</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>C</td>
<td>information has no meaning.</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>Scan Equal</td>
<td>W</td>
<td>MT MF SK 1 0 0 0 1</td>
<td>Command Codes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X X X X X HD US1 US0</td>
<td>Sector ID information</td>
</tr>
<tr>
<td></td>
<td></td>
<td>C</td>
<td>prior to command execution.</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>EOT</td>
<td>Data compared between the FDD</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>GPL</td>
<td>and the main system.</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>STP</td>
<td>Status information</td>
</tr>
<tr>
<td>Execution</td>
<td>R</td>
<td></td>
<td>after command execution.</td>
</tr>
<tr>
<td>Result</td>
<td>R</td>
<td>ST 0</td>
<td>Sector ID information</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>ST 1</td>
<td>after Command</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>ST 2</td>
<td>execution.</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>Phase</td>
<td>R/W</td>
<td>D7 D6 D5 D4 D3 D2 D1 D0</td>
<td>Remarks</td>
</tr>
<tr>
<td>-------</td>
<td>-----</td>
<td>-------------------------</td>
<td>---------</td>
</tr>
<tr>
<td><strong>Scan Low or Equal</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Command</td>
<td>W</td>
<td>MT MF SK 1 1 0 0 1</td>
<td><strong>Command Codes</strong></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>X X X X X HD US1 US0</td>
<td><strong>Sector ID information prior to command execution.</strong></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>EOT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>GPL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>STP</td>
<td></td>
</tr>
<tr>
<td>Execution</td>
<td></td>
<td></td>
<td><strong>Data compared between the FDD and main system.</strong></td>
</tr>
<tr>
<td>Result</td>
<td>R</td>
<td>ST 0</td>
<td><strong>Status information after command execution.</strong></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>ST 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>ST 2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>C</td>
<td><strong>Sector ID information after command execution.</strong></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td><strong>Scan High or Equal</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Command</td>
<td>W</td>
<td>MT MF SK 1 1 1 0 1</td>
<td><strong>Command Codes</strong></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>X X X X X HD US1 US0</td>
<td><strong>Sector ID information prior to command execution.</strong></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>EOT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>GPL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>STP</td>
<td></td>
</tr>
<tr>
<td>Execution</td>
<td></td>
<td></td>
<td><strong>Data compared between the FDD and main system.</strong></td>
</tr>
<tr>
<td>Result</td>
<td>R</td>
<td>ST 0</td>
<td><strong>Status information after command execution.</strong></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>ST 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>ST 2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>C</td>
<td><strong>Sector ID information after command execution.</strong></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>N</td>
<td></td>
</tr>
</tbody>
</table>

1-162 Diskette Adapter
<table>
<thead>
<tr>
<th>Phase</th>
<th>R/W</th>
<th>Data Bus</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command Execution</td>
<td>W</td>
<td>W</td>
<td>Recalibrate</td>
</tr>
<tr>
<td>Command Phase</td>
<td>W</td>
<td>0 0 0 0 0 1 1 1</td>
<td>Command Codes</td>
</tr>
<tr>
<td>Recalibrate</td>
<td></td>
<td>W</td>
<td>Head retracted to track 0</td>
</tr>
<tr>
<td>Result</td>
<td>R</td>
<td>W</td>
<td>Sense Interrupt Status</td>
</tr>
<tr>
<td>Command Phase</td>
<td>W</td>
<td>0 0 0 0 1 0 0 0</td>
<td>Command Codes</td>
</tr>
<tr>
<td>Command Phase</td>
<td>R</td>
<td>ST 0</td>
<td>Status information at the end of seek operation about the FDC</td>
</tr>
<tr>
<td>Result</td>
<td>R</td>
<td>PCN</td>
<td></td>
</tr>
<tr>
<td>Specify Phase</td>
<td>W</td>
<td>W</td>
<td>Specify</td>
</tr>
<tr>
<td>Command Phase</td>
<td>W</td>
<td>0 0 0 0 0 0 1 1</td>
<td>Command Codes</td>
</tr>
<tr>
<td>Result</td>
<td>W</td>
<td>SRT</td>
<td></td>
</tr>
<tr>
<td>Command Phase</td>
<td>W</td>
<td>HUT</td>
<td></td>
</tr>
<tr>
<td>Result</td>
<td>W</td>
<td>HLT</td>
<td></td>
</tr>
<tr>
<td>Command Phase</td>
<td>W</td>
<td>ND</td>
<td></td>
</tr>
<tr>
<td>Sense Drive Status</td>
<td>W</td>
<td>W</td>
<td>Sense Drive Status</td>
</tr>
<tr>
<td>Command Phase</td>
<td>W</td>
<td>0 0 0 0 0 1 0 0</td>
<td>Command Codes</td>
</tr>
<tr>
<td>Result</td>
<td>R</td>
<td>X X X X X</td>
<td>Status information about FDD.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HD US1 US0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ST 3</td>
<td></td>
</tr>
<tr>
<td>Seek</td>
<td>W</td>
<td>W</td>
<td>Seek</td>
</tr>
<tr>
<td>Command Phase</td>
<td>W</td>
<td>0 0 0 0 1 1 1 1</td>
<td>Command Codes</td>
</tr>
<tr>
<td>Execution Phase</td>
<td>W</td>
<td>X X X X X</td>
<td>Head is positioned over proper cylinder on diskette.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HD US1 US0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>NCN</td>
<td></td>
</tr>
<tr>
<td>Result</td>
<td>W</td>
<td>W</td>
<td>Invalid</td>
</tr>
<tr>
<td>Command Phase</td>
<td>W</td>
<td>Invalid Codes</td>
<td>Invalid command codes (NoOp - FDC goes into standy state).</td>
</tr>
<tr>
<td>Result</td>
<td>R</td>
<td>ST 0</td>
<td>ST 0 = 80.</td>
</tr>
<tr>
<td>No.</td>
<td>Name</td>
<td>Symbol</td>
<td>Description</td>
</tr>
<tr>
<td>-----</td>
<td>---------------------------</td>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>D7</td>
<td>Interrupt Code</td>
<td>IC</td>
<td>D7 = 0 and D6 = 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Normal termination of command (NT). Command was completed and properly</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>executed.</td>
</tr>
<tr>
<td>D6</td>
<td></td>
<td>IC</td>
<td>D7 = 0 and D6 = 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Abnormal termination of command (AT). Execution of command was started, but</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>was not successfully completed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IC</td>
<td>D7 = 1 and D6 = 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Invalid command issue (IC). Command that was issued was never started.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IC</td>
<td>D7 = 1 and D6 = 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Abnormal termination because, during command execution, the ready signal</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>from FDD changed state.</td>
</tr>
<tr>
<td>D5</td>
<td>Seek End</td>
<td>SE</td>
<td>When the FDC completes the seek command, this flag is set to 1 (high).</td>
</tr>
<tr>
<td>D4</td>
<td>Equipment Check</td>
<td>EC</td>
<td>If a fault signal is received from the FDD, or if the track 0 signal fails</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>to occur after 77 step pulses (recalibrate command), then this flag is set.</td>
</tr>
<tr>
<td>D3</td>
<td>Not Ready</td>
<td>NR</td>
<td>When the FDD is in the not-ready state and a read or write command is</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>issued, this flag is set. If a read or write command is issued to side 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>of a single-sided drive, then this flag is set.</td>
</tr>
<tr>
<td>D2</td>
<td>Head Address</td>
<td>HD</td>
<td>This flag is used to indicate the state of the head at interrupt.</td>
</tr>
<tr>
<td>D1</td>
<td>Unit Select 1</td>
<td>US 1</td>
<td>These flags are used to indicate a drive unit number at interrupt.</td>
</tr>
<tr>
<td>D0</td>
<td>Unit Select 0</td>
<td>US 0</td>
<td></td>
</tr>
</tbody>
</table>

**Command Status Register 0**
<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7</td>
<td>End of Cylinder</td>
<td>EN</td>
<td>When the FDC tries to access a sector beyond the final sector of a cylinder, this flag is set.</td>
</tr>
<tr>
<td>D6</td>
<td>—</td>
<td>—</td>
<td>Not used. This bit is always 0 (low).</td>
</tr>
<tr>
<td>D5</td>
<td>Data Error</td>
<td>DE</td>
<td>When the FDC detects a CRC error in either the ID field or the data field, this flag is set.</td>
</tr>
<tr>
<td>D4</td>
<td>Over Run</td>
<td>OR</td>
<td>If the FDC is not serviced by the main system during data transfers within a certain time interval, this flag is set.</td>
</tr>
<tr>
<td>D3</td>
<td>—</td>
<td>—</td>
<td>Not used. This bit is always 0 (low).</td>
</tr>
<tr>
<td>D2</td>
<td>No Data</td>
<td>ND</td>
<td>During execution of a read data, write deleted data, or scan command, if the FDC cannot find the sector specified in the ID register, this flag is set.</td>
</tr>
<tr>
<td>D1</td>
<td>Not Writable</td>
<td>NW</td>
<td>During execution of a write data, write deleted data, or format-a-cylinder command, if the FDC detects a write-protect signal from the FDD, then this flag is set.</td>
</tr>
<tr>
<td>D0</td>
<td>Missing Address Mark</td>
<td>MA</td>
<td>If the FDC cannot detect the ID address mark, this flag is set. Also, at the same time, the MD (missing address mark in the data field) of status register 2 is set.</td>
</tr>
</tbody>
</table>

**Command Status Register 1**
<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7</td>
<td>—</td>
<td>—</td>
<td>Not used. This bit is always 0 (low).</td>
</tr>
<tr>
<td>D6</td>
<td>Control Mark</td>
<td>CM</td>
<td>During execution of the read data or scan command, if the FDC encounters a sector that contains a deleted data address mark, this flag is set.</td>
</tr>
<tr>
<td>D5</td>
<td>Data Error in Data Field</td>
<td>DD</td>
<td>If the FDC detects a CRC error in the data, then this flag is set.</td>
</tr>
<tr>
<td>D4</td>
<td>Wrong Cylinder</td>
<td>WC</td>
<td>This bit is related to the ND bit, and when the contents of C on the medium are different from that stored in the ID register, this flag is set.</td>
</tr>
<tr>
<td>D3</td>
<td>Scan Equal Hit</td>
<td>SH</td>
<td>During execution of the scan command, if the condition of &quot;equal&quot; is satisfied, this flag is set.</td>
</tr>
<tr>
<td>D2</td>
<td>Scan Not Satisfied</td>
<td>SN</td>
<td>During execution of the scan command, if the FDC cannot find a sector on the cylinder that meets the condition, then this flag is set.</td>
</tr>
<tr>
<td>D1</td>
<td>Bad Cylinder</td>
<td>BC</td>
<td>This bit is related to the ND bit, and when the contents of C on the medium are different from that stored in the ID register, and the contents of C is FF, then this flag is set.</td>
</tr>
<tr>
<td>D0</td>
<td>Missing Address Mark in Data Field</td>
<td>MD</td>
<td>When data is read from the medium, if the FDC cannot find a data address mark or deleted data address mark, then this flag is set.</td>
</tr>
</tbody>
</table>

**Command Status Register 2**
<table>
<thead>
<tr>
<th>Bit No.</th>
<th>Name</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7</td>
<td>Fault</td>
<td>FT</td>
<td>This bit is the status of the fault signal from the FDD.</td>
</tr>
<tr>
<td>D6</td>
<td>Write Protected</td>
<td>WP</td>
<td>This bit is the status of the write-protected signal from the FDD.</td>
</tr>
<tr>
<td>D5</td>
<td>Ready</td>
<td>RY</td>
<td>This bit is the status of the ready signal from the FDD.</td>
</tr>
<tr>
<td>D4</td>
<td>Track 0</td>
<td>T0</td>
<td>This bit is the status of the track 0 signal from the FDD.</td>
</tr>
<tr>
<td>D3</td>
<td>Two Side</td>
<td>TS</td>
<td>This bit is the status of the two-side signal from the FDD.</td>
</tr>
<tr>
<td>D2</td>
<td>Head Address</td>
<td>HD</td>
<td>This bit is the status of the side-select signal from the FDD.</td>
</tr>
<tr>
<td>D1</td>
<td>Unit Select 1</td>
<td>US 1</td>
<td>This bit is the status of the unit-select-1 signal from the FDD.</td>
</tr>
<tr>
<td>D0</td>
<td>Unit Select 0</td>
<td>US 0</td>
<td>This bit is the status of the unit-select-0 signal from the FDD.</td>
</tr>
</tbody>
</table>

Command Status Register 3

Programming Summary

FDC Data Register I/O Address Hex 3F5
FDC Main Status Register I/O Address Hex 3F4
Digital Output Register I/O Address Hex 3F2

<table>
<thead>
<tr>
<th>Bit</th>
<th>Drive</th>
<th>Select</th>
<th>Not FDC Reset</th>
<th>Enable INT &amp; DMA Requests</th>
<th>Drive A Motor Enable</th>
<th>Drive B Motor Enable</th>
<th>Drive C Motor Enable</th>
<th>Drive D Motor Enable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>01</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

All bits cleared with channel reset.

DPC Registers
FDC Constants (in hex)

N: 02  GPL Format: 05
SC: 08  GPL R/W: 2A
HUT: F  HLT: 01
SRT: C  (6 ms track-to-track)

Drive Constants

Head Load  35 ms
Head Settle  15 ms
Motor Start  250 ms

Comments

- Head loads with drive select, wait HD load before R/W.

- Following access, wait HD settle time before R/W.

- Drive motors should be off when not in use. Only A or B and C or D may run simultaneously. Wait motor start time before R/W.

- Motor must be on for drive to be selected.

- Data errors can occur while using a home television as the system display. Locating the TV too close to the diskette area can cause this to occur. To correct the problem, move the TV away from, or to the opposite side of the system unit.

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.

+D0-7 (Bidirectional, load: 1 74LS, driver: 74LS 3-state). These eight lines form a bus by which all commands, status, and data are transferred. Bit 0 is the low-order bit.

+A0-9 (Adapter input, load: 1 74LS) These ten 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.

+AEN (Adapter input, load: 1 74LS) The content of lines A0-9 is ignored if this line is active.

−IOW (Adapter input, load: 1 74LS) The content of lines D0-7 is stored in the register addressed by lines A0-9 or DACK2 at the trailing edge of this signal.

−IOR (Adapter input, load: 1 74LS) The content of the register addressed by lines A0-9 or DACK2 is gated onto lines D0-7 when this line is active.

−DACK2 (Adapter input, load: 2 74LS) This line being active degates output DRQ2, selects the FDC data register as the source/destination of bus D0-7, and indirectly gates T/C to IRQ6.

+T/C (Adapter input, load: 4 74LS) This line and DACK2 being active indicates that the byte of data for which the DMA count was initialized is now being transferred.

+RESET (Adapter input, load: 1 74LS) An up level aborts any operation in process and clears the digital output register (DOR).
+DRQ2 (Adapter output, driver: 74LS 3-state)  
This line is made active when the attachment is ready to transfer a byte of data to or from main storage. The line is made inactive by DACK2 becoming active or an I/O read of the FDC data register.

+IRQ6 (Adapter output, driver: 74LS 3-state)  
This line is made active when the FDC has completed an operation. It results in an interrupt to a routine which should examine the FDC result bytes to reset the line and determine the ending condition.

Drive A and B Interface

All signals are TTL-compatible:

<table>
<thead>
<tr>
<th>Signal</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 open-collector gates. The drive(s) must provide termination networks to Vcc (except motor enable, which has a 2000-ohm resistor to Vcc).

Each adapter input is terminated with a 150-ohm resistor to Vcc.

Adapter Outputs

-Drive Select A and B (Driver: 7438)  
These two lines are used by drives A and B to degate all drivers to the adapter and receivers from the attachment (except motor enable) when the line associated with a drive is inactive.
Motor Enable A and B (Driver: 7438)
The drive associated with each of these lines must control its spindle motor such that it starts when the line becomes active and stops when the line becomes inactive.

Step (Driver: 7438)
The selected drive moves the read/write head one cylinder in or out per the direction line for each pulse present on this line.

Direction (Driver: 7438)
For each recognized pulse of the step line, the read/write head moves one cylinder toward the spindle if this line is active, and away from the spindle if inactive.

Head Select (Driver: 7438)
Head 1 (upper head) will be selected when this line is active (low).

Write Data (Driver: 7438)
For each inactive to active transition of this line while write enable is active, the selected drive causes a flux change to be stored on the diskette.

Write Enable (Driver: 7438)
The drive disables write current in the head unless this line is active.
**Adapter Inputs**

<table>
<thead>
<tr>
<th>Input</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Index</td>
<td>The selected drive supplies one pulse per diskette revolution on this line.</td>
</tr>
<tr>
<td>Write Protect</td>
<td>The selected drive makes this line active if a write-protected diskette is mounted in the drive.</td>
</tr>
<tr>
<td>Track 0</td>
<td>The selected drive makes this line active if the read/write head is over track 0.</td>
</tr>
<tr>
<td>Read Data</td>
<td>The selected drive supplies a pulse on this line for each flux change encountered on the diskette.</td>
</tr>
</tbody>
</table>
34-Pin Keyed Edge Connector

Component Side

Note: Lands 1-33 (odd numbers) are on the back of the board. Lands 2-34 (even numbers) are on the front, or component side.

<table>
<thead>
<tr>
<th>At Standard TTL Levels</th>
<th>Land Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ground-Odd Numbers</td>
<td>1-33</td>
</tr>
<tr>
<td>Unused</td>
<td>2, 4, 6</td>
</tr>
<tr>
<td>Index</td>
<td>8</td>
</tr>
<tr>
<td>Motor Enable A</td>
<td>10</td>
</tr>
<tr>
<td>Drive Select B</td>
<td>12</td>
</tr>
<tr>
<td>Drive Select A</td>
<td>14</td>
</tr>
<tr>
<td>Motor Enable B</td>
<td>16</td>
</tr>
<tr>
<td>Direction (Stepper Motor)</td>
<td>18</td>
</tr>
<tr>
<td>Step Pulse</td>
<td>20</td>
</tr>
<tr>
<td>Write Data</td>
<td>22</td>
</tr>
<tr>
<td>Write Enable</td>
<td>24</td>
</tr>
<tr>
<td>Track 0</td>
<td>26</td>
</tr>
<tr>
<td>Write Protect</td>
<td>28</td>
</tr>
<tr>
<td>Read Data</td>
<td>30</td>
</tr>
<tr>
<td>Select Head 1</td>
<td>32</td>
</tr>
<tr>
<td>Unused</td>
<td>34</td>
</tr>
</tbody>
</table>

Connector Specifications (Part 1 of 2)

Diskette Drives

Drive Adapter
<table>
<thead>
<tr>
<th>At Standard TTL Levels</th>
<th>Pin Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unused</td>
<td>1-5</td>
</tr>
<tr>
<td>Index</td>
<td>6</td>
</tr>
<tr>
<td>Motor Enable C</td>
<td>7</td>
</tr>
<tr>
<td>Drive Select D</td>
<td>8</td>
</tr>
<tr>
<td>Drive Select C</td>
<td>9</td>
</tr>
<tr>
<td>Motor Enable D</td>
<td>10</td>
</tr>
<tr>
<td>Direction (Stepper Motor)</td>
<td>11</td>
</tr>
<tr>
<td>Step Pulse</td>
<td>12</td>
</tr>
<tr>
<td>Write Data</td>
<td>13</td>
</tr>
<tr>
<td>Write Enable</td>
<td>14</td>
</tr>
<tr>
<td>Track 0</td>
<td>15</td>
</tr>
<tr>
<td>Write Protect</td>
<td>16</td>
</tr>
<tr>
<td>Read Data</td>
<td>17</td>
</tr>
<tr>
<td>Select Head 1</td>
<td>18</td>
</tr>
<tr>
<td>Ground</td>
<td>20-37</td>
</tr>
</tbody>
</table>

Connector Specifications (Part 2 of 2)
IBM 5-1/4" Diskette Drive

The system unit has space and power for one or two 5-1/4 inch diskette drives. A drive can be single-sided or 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.

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 raises the latch at the front of the diskette drive and inserts the diskette into the slot. Plastic guides in the slot ensure the diskette is in the correct position. Closing the latch centers the diskette and clamps it to the drive hub. After 250 milliseconds, the servo-controlled dc drive motor starts and drives the hub at a constant speed of 300 rpm. The head positioning system, which consists of a 4-phase stepper-motor and band assembly with its associated electronics, moves the magnetic head so it comes in contact with the desired track of the diskette. The stepper-motor and band assembly uses one-step rotation to cause a one-track linear movement of the magnetic head. No operator intervention is required during normal operation. During a write operation, a 0.013-inch (0.33 millimeter) data track is recorded, then tunnel-erased to 0.012 inch (0.030 millimeter). 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 an adapter card.

The diskette drive also has the following sensor systems:

1. The track 00 switch, which senses when the head/carriage assembly is at track 00.
2. 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.

3. The write-protect sensor disables the diskette drive’s electronics whenever a write-protect tab is applied to the diskette.

For interface information, refer to “IBM 5-1/4” Diskette Drive Adapter” earlier in this section.

<table>
<thead>
<tr>
<th>Media</th>
<th>Industry-compatible 5-1/4 inch diskette</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracks per inch</td>
<td>48</td>
</tr>
<tr>
<td>Number of tracks</td>
<td>40</td>
</tr>
<tr>
<td>Dimensions</td>
<td></td>
</tr>
<tr>
<td>Height</td>
<td>3.38 inches (85.85 mm)</td>
</tr>
<tr>
<td>Width</td>
<td>5.87 inches (149.10 mm)</td>
</tr>
<tr>
<td>Depth</td>
<td>8.00 inches (203.2 mm)</td>
</tr>
<tr>
<td>Weight</td>
<td>4.50 pounds (2.04 kg)</td>
</tr>
<tr>
<td>Temperature</td>
<td></td>
</tr>
<tr>
<td>(Exclusive of media)</td>
<td></td>
</tr>
<tr>
<td>Operating</td>
<td>50°F to 112°F (10°C to 44°C)</td>
</tr>
<tr>
<td>Non operating</td>
<td>−40°F to 140°F (−40°C to 60°C)</td>
</tr>
<tr>
<td>Relative humidity</td>
<td></td>
</tr>
<tr>
<td>(Exclusive of media)</td>
<td></td>
</tr>
<tr>
<td>Operating</td>
<td>20% to 80% (non condensing)</td>
</tr>
<tr>
<td>Non operating</td>
<td>5% to 95% (non condensing)</td>
</tr>
<tr>
<td>Seek Time</td>
<td>6 ms track-to-track</td>
</tr>
<tr>
<td>Head Settling Time</td>
<td>15 ms (last track addressed)</td>
</tr>
<tr>
<td>Error Rate</td>
<td>1 per 10^9 (recoverable)</td>
</tr>
<tr>
<td></td>
<td>1 per 10^12 (non recoverable)</td>
</tr>
<tr>
<td></td>
<td>1 per 10^6 (seeks)</td>
</tr>
<tr>
<td>Head Life</td>
<td>20,000 hours (normal use)</td>
</tr>
<tr>
<td>Media Life</td>
<td>3.0 x 10^6 passes per track</td>
</tr>
<tr>
<td>Disk Speed</td>
<td>300 rpm +/- 1.5% (long term)</td>
</tr>
<tr>
<td>Instantaneous Speed Variation</td>
<td>+/- 3.0%</td>
</tr>
<tr>
<td>Start/Stop Time</td>
<td>250 ms (maximum)</td>
</tr>
<tr>
<td>Transfer Rate</td>
<td>250K bits/sec</td>
</tr>
<tr>
<td>Recording Mode</td>
<td>MFM</td>
</tr>
<tr>
<td>Power</td>
<td>+12 Vdc +/- 0.6 V, 900 mA average</td>
</tr>
<tr>
<td></td>
<td>+5 Vdc +/- 0.25 V, 600 mA average</td>
</tr>
</tbody>
</table>

**Mechanical and Electrical Specifications**

1-176 Diskette Drive
Diskettes

The IBM 5-1/4" Diskette Drive uses a standard 5.25-inch (133.4-millimeter) 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 made through an opening in the jacket. Openings for the drive hub and diskette index hole are also provided.
Notes:

1-178 Diskettes
IBM Fixed Disk Drive Adapter

The fixed disk drive adapter attaches to one or two fixed disk drive units, through an internal daisy-chained flat cable (data/control cable). Each system supports a maximum of one fixed disk drive adapter and two fixed disk drives.

The adapter is buffered on the I/O bus and uses the system board direct memory access (DMA) for record data transfers. An interrupt level also is used to indicate operation completion and status conditions that require processor attention.

The fixed disk drive adapter provides automatic 11-bit burst error detection and correction in the form of 32-bit error checking and correction (ECC).

The device level control for the fixed disk drive adapter is contained on a ROM module on the adapter. A listing of this device level control can be found in “Appendix A: ROM BIOS Listings.”

WARNING: The last cylinder on the fixed disk drive is reserved for diagnostic use. Diagnostic write tests will destroy any data on this cylinder.

Fixed Disk Controller

The disk controller has two registers that may be accessed by the main system processor: a status register and a data register. The 8-bit status register contains the status information of the disk controller, and can be accessed at any time. The 8-bit data register (actually consisting of several registers in a stack with only one register presented to the data bus) stores data, commands, parameters, and provides the disk controller’s status information. Data bytes are read from, or written to the data register in order to program or obtain the results after a particular command. The status register is a read-only register, and is used to help the transfer of data between the processor and the disk controller. The controller-select pulse is generated by writing to port address hex 322.
Fixed Disk Drive Adapter Block Diagram
Programming Considerations

Status Register

At the end of all commands from the system board, the disk controller returns a completion status byte back to the system board. This byte informs the system unit if an error occurred during the execution of the command. The following shows the format of this byte.

<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>0</td>
<td>0</td>
<td>d</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>e</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 0, 1, 2, 3, 4, 6, 7  These bits are set to zero.

Bit 1  When set, this bit shows an error has occurred during command execution.

Bit 5  This bit shows the logical unit number of the drive.

If the interrupts are enabled, the controller sends an interrupt when it is ready to transfer the status byte. Busy from the disk controller is unasserted when the byte is transferred to complete the command.

Sense Bytes

If the status register receives an error (bit 1 is set), then the disk controller requests four bytes of sense data. The format for the four bytes is as follows:

<table>
<thead>
<tr>
<th>Bits</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>Byte 0</td>
<td>Address Valid</td>
<td>0</td>
<td>Error Type</td>
<td>Error Code</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Byte 1</td>
<td>0</td>
<td>0</td>
<td>d</td>
<td>Head Number</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Byte 2</td>
<td>Cylinder High</td>
<td>Sector Number</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Byte 3</td>
<td>Cylinder Low</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Remarks

d = drive
Byte 0  Bits 0, 1, 2, 3  Error code.

Byte 0  Bits 4, 5  Error type.

Byte 0  Bit 6  Set to 0 (spare).

Byte 0  Bit 7  The address valid bit. Set only when the previous command required a disk address, in which case it is returned as a 1; otherwise, it is a 0.

The following disk controller tables list the error types and error codes found in byte 0:

<table>
<thead>
<tr>
<th>Error Type</th>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 5 4 3 2 1 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0 0 0 0 0 0</td>
<td>The controller did not detect any error during the execution of the previous operation.</td>
<td></td>
</tr>
<tr>
<td>0 0 0 0 0 1</td>
<td>The controller did not detect an index signal from the drive.</td>
<td></td>
</tr>
<tr>
<td>0 0 0 1 0 0</td>
<td>The controller did not get a seek-complete signal from the drive after a seek operation (for all non-buffered step seeks).</td>
<td></td>
</tr>
<tr>
<td>0 0 0 1 1 0</td>
<td>The controller detected a write fault from the drive during the last operation.</td>
<td></td>
</tr>
<tr>
<td>0 0 0 1 0 0</td>
<td>After the controller selected the drive, the drive did not respond with a ready signal.</td>
<td></td>
</tr>
<tr>
<td>0 0 0 1 1 0</td>
<td>Not used.</td>
<td></td>
</tr>
<tr>
<td>0 0 0 1 1 1</td>
<td>Not used.</td>
<td></td>
</tr>
<tr>
<td>0 0 1 0 0 0</td>
<td>The drive is still seeking. This status is reported by the Test Drive Ready command for an overlap seek condition when the drive has not completed the seek. No time-out is measured by the controller for the seek to complete.</td>
<td></td>
</tr>
<tr>
<td>Error Type</td>
<td>Error Code</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td>0 1</td>
<td>0 0 0 0</td>
<td>ID Read Error: The controller detected an ECC error in the target ID field on the disk.</td>
</tr>
<tr>
<td>0 1</td>
<td>0 0 0 1</td>
<td>Data Error: The controller detected an uncorrectable ECC error in the target sector during a read operation.</td>
</tr>
<tr>
<td>0 1</td>
<td>0 0 1 0</td>
<td>Address Mark: The controller did not detect the target address mark (AM) on the disk.</td>
</tr>
<tr>
<td>0 1</td>
<td>0 0 1 1</td>
<td>Not used.</td>
</tr>
<tr>
<td>0 1</td>
<td>0 1 0 0</td>
<td>Sector Not Found: The controller found the correct cylinder and head, but not the target sector.</td>
</tr>
<tr>
<td>0 1</td>
<td>0 1 0 1</td>
<td>Seek Error: The cylinder or head address (either or both) did not compare with the expected target address as a result of a seek.</td>
</tr>
<tr>
<td>0 1</td>
<td>0 1 1 0</td>
<td>Not used.</td>
</tr>
<tr>
<td>0 1</td>
<td>0 1 1 1</td>
<td>Not used.</td>
</tr>
<tr>
<td>0 1</td>
<td>1 0 0 0</td>
<td>Correctable Data Error: The controller detected a correctable ECC error in the target field.</td>
</tr>
<tr>
<td>0 1</td>
<td>1 0 0 1</td>
<td>Bad Track: The controller detected a bad track flag during the last operation. No retries are attempted on this error.</td>
</tr>
<tr>
<td>Error Type</td>
<td>Error Code</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Bits</td>
<td>5 4 3 2 1 0</td>
<td></td>
</tr>
<tr>
<td>1 0</td>
<td>0 0 0 0</td>
<td>Invalid Command: The controller has received an invalid command from the system unit.</td>
</tr>
<tr>
<td>1 0</td>
<td>0 0 0 1</td>
<td>Illegal Disk Address: The controller detected an address that is beyond the maximum range.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Error Type</th>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>5 4 3 2 1 0</td>
<td></td>
</tr>
<tr>
<td>1 1</td>
<td>0 0 0 0</td>
<td>RAM Error: The controller detected a data error during the RAM sector-buffer diagnostic test.</td>
</tr>
<tr>
<td>1 1</td>
<td>0 0 0 1</td>
<td>Program Memory Checksum Error: During this internal diagnostic test, the controller detected a program-memory checksum error.</td>
</tr>
<tr>
<td>1 1</td>
<td>0 0 1 0</td>
<td>ECC Polynominal Error: During the controller's internal diagnostic tests, the hardware ECC generator failed its test.</td>
</tr>
</tbody>
</table>
Data Register

The processor specifies the operation by sending the 6-byte device control block (DCB) to the controller. The figure below shows the composition of the DCB, and defines the bytes that make up the DCB.

<table>
<thead>
<tr>
<th>Bits</th>
<th>7 6 5 4 3 2 1 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte 0</td>
<td>Command Class</td>
</tr>
<tr>
<td>Byte 1</td>
<td>0</td>
</tr>
<tr>
<td>Byte 2</td>
<td>Cylinder High</td>
</tr>
<tr>
<td>Byte 3</td>
<td>Cylinder Low</td>
</tr>
<tr>
<td>Byte 4</td>
<td>Interleave or Block Count</td>
</tr>
<tr>
<td>Byte 5</td>
<td>Control Field</td>
</tr>
</tbody>
</table>

Byte 0 – Bits 7, 6, and 5 identify the class of the command. Bits 4 through 0 contain the Opcode command.

Byte 1 – Bit 5 identifies the drive number. Bits 4 through 0 contain the disk head number to be selected. Bits 6 and 7 are not used.

Byte 2 – Bits 6 and 7 contain the two most significant bits of the cylinder number. Bits 0 through 5 contain the sector number.

Byte 3 – Bits 0 through 7 are the eight least significant bits of the cylinder number.

Byte 4 – Bits 0 through 7 specify the interleave or block count.

Byte 5 – Bits 0 through 7 contain the control field.
Control Byte

Byte 5 is the control field of the DCB and allows the user to select options for several types of disk drives. The format of this byte is as follows:

<table>
<thead>
<tr>
<th>Bits</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>r</td>
<td>a</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>s</td>
<td>s</td>
<td>s</td>
<td></td>
</tr>
</tbody>
</table>

Remarks:
- \( r \) = retries
- \( s \) = step option
- \( a \) = retry option on data ECC error

Bit 7
Disables the four retries by the controller on all disk-access commands. Set this bit only during the evaluation of the performance of a disk drive.

Bit 6
If set to 0 during read commands, a reread is attempted when an ECC error occurs. If no error occurs during reread, the command will complete with no error status. If this bit is set to 1, no reread is attempted.

Bits 5, 4, 3
Set to 0.

Bits 2, 1, 0
These bits define the type of drive and select the step option. See the following figure.

<table>
<thead>
<tr>
<th>Bits</th>
<th>2, 1, 0</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td>This drive is not specified and defaults to 3 milliseconds per step.</td>
<td></td>
</tr>
<tr>
<td>0 0 1</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>0 1 0</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>0 1 1</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>1 0 0</td>
<td>200 microseconds per step.</td>
<td></td>
</tr>
<tr>
<td>1 0 1</td>
<td>70 microseconds per step (specified by BIOS).</td>
<td></td>
</tr>
<tr>
<td>1 1 0</td>
<td>3 milliseconds per step.</td>
<td></td>
</tr>
<tr>
<td>1 1 1</td>
<td>3 milliseconds per step.</td>
<td></td>
</tr>
</tbody>
</table>
## Command Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Data Control Block</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Drive</td>
<td>Bit 7 6 5 4 3 2 1 0 d = drive (0 or 1)</td>
<td></td>
</tr>
<tr>
<td>Ready (Class 0, Opcode 00)</td>
<td>Byte 0 0 0 0 0 0 0 x = don't care</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 1 0 0 d x x x x x Bytes 2, 3, 4, 5 = don't care</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 5 r 0 0 0 0 s s s</td>
<td></td>
</tr>
<tr>
<td>Recalibrate</td>
<td>Bit 7 6 5 4 3 2 1 0 d = drive (0 or 1)</td>
<td></td>
</tr>
<tr>
<td>(Class 0, Opcode 01)</td>
<td>Byte 0 0 0 0 0 0 0 0 1 x = don't care</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 1 0 0 d x x x x x Bytes 2, 3, 4, 5 = don't care</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 5 r 0 0 0 0 s s s</td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td>Bit 7 6 5 4 3 2 1 0</td>
<td></td>
</tr>
<tr>
<td>(Class 0, Opcode 02)</td>
<td>Byte 0 0 0 0 0 0 0 0 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 1 0 0 d x x x x x Bytes 2, 3, 4, 5 = don't care</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 5 r 0 0 0 0 s s s</td>
<td></td>
</tr>
<tr>
<td>Request Sense Status</td>
<td>Bit 7 6 5 4 3 2 1 0 d = drive (0 or 1)</td>
<td></td>
</tr>
<tr>
<td>(Class 0, Opcode 03)</td>
<td>Byte 0 0 0 0 0 0 0 0 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 1 0 0 d x x x x x Bytes 2, 3, 4, 5 = don't care</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 5 r 0 0 0 0 s s s</td>
<td></td>
</tr>
<tr>
<td>Format Drive</td>
<td>Bit 7 6 5 4 3 2 1 0 d = drive (0 or 1)</td>
<td></td>
</tr>
<tr>
<td>(Class 0, Opcode 04)</td>
<td>Byte 0 0 0 0 0 1 0 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 1 0 0 d Head Number s</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 2 ch 0 0 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 3 Cylinder Low</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 4 0 0 Interleave</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 5 r 0 0 0 0 s s s</td>
<td></td>
</tr>
<tr>
<td>Ready Verify</td>
<td>Bit 7 6 5 4 3 2 1 0 d = drive (0 or 1)</td>
<td></td>
</tr>
<tr>
<td>(Class 0, Opcode 05)</td>
<td>Byte 0 0 0 0 0 1 0 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 1 0 0 d Head Number s</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 2 ch 0 0 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 3 Cylinder Low</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 4 Block Count</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 5 r a 0 0 0 s s s</td>
<td></td>
</tr>
<tr>
<td>Command</td>
<td>Data Control Block</td>
<td>Remarks</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------------------------------------------------------------------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>Format Track</td>
<td>Bit 7 6 5 4 3 2 1 0</td>
<td>d = drive (0 or 1)</td>
</tr>
<tr>
<td>(Class 0,</td>
<td>Byte 0 0 0 0 0 1 1 0</td>
<td>r = retries</td>
</tr>
<tr>
<td>Opcode 06)</td>
<td>Byte 1 0 0 d Head Number</td>
<td>s = step option</td>
</tr>
<tr>
<td></td>
<td>Byte 2 ch 0 0 0 0 0 0 0</td>
<td>ch = cylinder high</td>
</tr>
<tr>
<td></td>
<td>Byte 3 Cylinder Low</td>
<td>Interleave: 1 to 16</td>
</tr>
<tr>
<td></td>
<td>Byte 4 0 0 Interleave</td>
<td>for 512-byte sectors</td>
</tr>
<tr>
<td></td>
<td>Byte 5 r 0 0 0 0 s s s</td>
<td></td>
</tr>
<tr>
<td>Format Bad Track</td>
<td>Bit 7 6 5 4 3 2 1 0</td>
<td>d = drive (0 or 1)</td>
</tr>
<tr>
<td>(Class 0,</td>
<td>Byte 0 0 0 0 0 1 1 1</td>
<td>r = retries</td>
</tr>
<tr>
<td>Opcode 07)</td>
<td>Byte 1 0 0 d Head Number</td>
<td>s = step option</td>
</tr>
<tr>
<td></td>
<td>Byte 2 ch 0 0 0 0 0 0 0</td>
<td>ch = cylinder high</td>
</tr>
<tr>
<td></td>
<td>Byte 3 Cylinder Low</td>
<td>Interleave: 1 to 16</td>
</tr>
<tr>
<td></td>
<td>Byte 4 0 0 Interleave</td>
<td>for 512-byte sectors</td>
</tr>
<tr>
<td></td>
<td>Byte 5 r 0 0 0 0 s s s</td>
<td></td>
</tr>
<tr>
<td>Read</td>
<td>Bit 7 6 5 4 3 2 1 0</td>
<td>d = drive (0 or 1)</td>
</tr>
<tr>
<td>(Class 0,</td>
<td>Byte 0 0 0 0 1 0 0 0</td>
<td>r = retries</td>
</tr>
<tr>
<td>Opcode 08)</td>
<td>Byte 1 0 0 d Head Number</td>
<td>a = retry option on</td>
</tr>
<tr>
<td></td>
<td>Byte 2 ch Sector Number</td>
<td>data ECC error</td>
</tr>
<tr>
<td></td>
<td>Byte 3 Cylinder Low</td>
<td>s = step option</td>
</tr>
<tr>
<td></td>
<td>Byte 5 r a 0 0 0 s s s</td>
<td>ch = cylinder high</td>
</tr>
<tr>
<td>Reserved</td>
<td></td>
<td>This Opcode is not used</td>
</tr>
<tr>
<td>(Class 0,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Opcode 09)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Write</td>
<td>Bit 7 6 5 4 3 2 1 0</td>
<td>d = drive (0 or 1)</td>
</tr>
<tr>
<td>(Class 0,</td>
<td>Byte 0 0 0 0 1 0 1 0</td>
<td>r = retries</td>
</tr>
<tr>
<td>Opcode 0A)</td>
<td>Byte 1 0 0 d Head Number</td>
<td>s = step option</td>
</tr>
<tr>
<td></td>
<td>Byte 2 ch Sector Number</td>
<td>ch = cylinder high</td>
</tr>
<tr>
<td></td>
<td>Byte 3 Cylinder Low</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 4 Block Count</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 5 r 0 0 0 0 s s s</td>
<td></td>
</tr>
<tr>
<td>Seek</td>
<td>Bit 7 6 5 4 3 2 1 0</td>
<td>d = drive (0 or 1)</td>
</tr>
<tr>
<td>(Class 0,</td>
<td>Byte 0 0 0 0 1 0 1 1</td>
<td>r = retries</td>
</tr>
<tr>
<td>Opcode 0B)</td>
<td>Byte 1 0 0 d Head Number</td>
<td>s = step option</td>
</tr>
<tr>
<td></td>
<td>Byte 2 ch 0 0 0 0 0 0 0</td>
<td>x = don't care</td>
</tr>
<tr>
<td></td>
<td>Byte 3 Cylinder Low</td>
<td>ch = cylinder high</td>
</tr>
<tr>
<td></td>
<td>Byte 4 x x x x x x x</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 5 r 0 0 0 0 s s s</td>
<td></td>
</tr>
<tr>
<td>Command</td>
<td>Data Control Block</td>
<td>Remarks</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>--------------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>Initialize Drive</td>
<td>Bit 7 6 5 4 3 2 1 0 Byte 0 0 0 0 1 1 0 0</td>
<td>Bytes 1, 2, 3, 4, 5 = don't care</td>
</tr>
<tr>
<td>Characteristics*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Class 0, Opcode 0C)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Read ECC Burst Error Length</td>
<td>Bit 7 6 5 4 3 2 1 0 Byte 0 0 0 0 1 1 0 1</td>
<td>Bytes 1, 2, 3, 4, 5 = don't care</td>
</tr>
<tr>
<td>(Class 0, Opcode 0D)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Read Data from Sector Buffer</td>
<td>Bit 7 6 5 4 3 2 1 0 Byte 0 0 0 0 1 1 1 1</td>
<td>Bytes 1, 2, 3, 4, 5 = don't care</td>
</tr>
<tr>
<td>(Class 0, Opcode 0E)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Write Data to Sector Buffer</td>
<td>Bit 7 6 5 4 3 2 1 0 Byte 0 0 0 0 1 1 1 1</td>
<td>Bytes 1, 2, 3, 4, 5 = don't care</td>
</tr>
<tr>
<td>(Class 0, Opcode 0F)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RAM Diagnostic</td>
<td>Bit 7 6 5 4 3 2 1 0 Byte 0 1 1 1 0 0 0 0</td>
<td>Bytes 1, 2, 3, 4, 5 = don't care</td>
</tr>
<tr>
<td>(Class 7, Opcode 00)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Class 7, Opcode 01)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Class 7, Opcode 02)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Initialize Drive Characteristics: The DCB must be followed by eight additional bytes.

Maximum number of cylinders (2 bytes)
Maximum number of heads (1 byte)
Start reduced write current cylinder (2 bytes)
Start write precompensation cylinder (2 bytes)
Maximum ECC data burst length (1 byte)
<table>
<thead>
<tr>
<th>Command</th>
<th>Data Control Block</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drive</td>
<td>Bit 7 6 5 4 3 2 1 0</td>
<td>d = drive (0 or 1)</td>
</tr>
<tr>
<td>Diagnostic (Class 7,</td>
<td>Byte 0 1 1 1 0 0 0 1 1</td>
<td>s = step option</td>
</tr>
<tr>
<td>Opcode 03)</td>
<td>Byte 1 0 0 d x x x x x</td>
<td>r = retries</td>
</tr>
<tr>
<td></td>
<td>Byte 2 x x x x x x x x</td>
<td>x = don't care</td>
</tr>
<tr>
<td></td>
<td>Byte 3 x x x x x x x x</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 4 x x x x x x x x</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 5 r 0 0 0 0 s s s</td>
<td></td>
</tr>
<tr>
<td>Controller Internal</td>
<td>Bit 7 6 5 4 3 2 1 0</td>
<td>first byte = 12</td>
</tr>
<tr>
<td>Diagnostics</td>
<td>Byte 0 1 1 1 0 0 1 0 0</td>
<td>Bytes 1, 2, 3, 4, 5 = don't care</td>
</tr>
<tr>
<td>(Class 7, Opcode 04)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Read Long* (Class 7,</td>
<td>Bit 7 6 5 4 3 2 1 0</td>
<td>d = (0 or 1)</td>
</tr>
<tr>
<td>Opcode 05)</td>
<td>Byte 0 1 1 1 0 0 1 0 1</td>
<td>s = step option</td>
</tr>
<tr>
<td></td>
<td>Byte 1 0 0 d Head Number</td>
<td>r = retries</td>
</tr>
<tr>
<td></td>
<td>Byte 2 ch Sector Number</td>
<td>ch = cylinder high</td>
</tr>
<tr>
<td></td>
<td>Byte 3 Cylinder Low</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 4 Block Count</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 5 r 0 0 0 0 s s s</td>
<td></td>
</tr>
<tr>
<td>Write Long** (Class 7,</td>
<td>Bit 7 6 5 4 3 2 1 0</td>
<td>d = drive (0 or 1)</td>
</tr>
<tr>
<td>Opcode 06)</td>
<td>Byte 0 1 1 1 0 0 1 1 0</td>
<td>s = step option</td>
</tr>
<tr>
<td></td>
<td>Byte 1 0 0 d Head Number</td>
<td>r = retries</td>
</tr>
<tr>
<td></td>
<td>Byte 2 ch Sector Number</td>
<td>ch = cylinder high</td>
</tr>
<tr>
<td></td>
<td>Byte 3 Cylinder Low</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 4 Block Count</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 5 r 0 0 0 0 s s s</td>
<td></td>
</tr>
</tbody>
</table>

*Returns 512 bytes plus 4 bytes of ECC data per sector.

**Requires 512 bytes plus 4 bytes of ECC data per sector.
Programming Summary

The two least-significant bits of the address bus are sent to the system board’s I/O port decoder, which has two sections. One section is enabled by the I/O read signal (−IOR) and the other by the I/O write signal (−IOW). The result is a total of four read/write ports assigned to the disk controller board.

The address enable signal (AEN) is asserted by the system board when DMA is controlling data transfer. When AEN is asserted, the I/O port decoder is disabled.

The following figure is a table of the four read/write ports:

<table>
<thead>
<tr>
<th>R/W</th>
<th>Port Address</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read</td>
<td>320</td>
<td>Read data (from controller to system unit).</td>
</tr>
<tr>
<td>Write</td>
<td>320</td>
<td>Write data (from system unit to controller).</td>
</tr>
<tr>
<td>Read</td>
<td>321</td>
<td>Read controller hardware status.</td>
</tr>
<tr>
<td>Write</td>
<td>321</td>
<td>Controller reset.</td>
</tr>
<tr>
<td>Read</td>
<td>322</td>
<td>Reserved.</td>
</tr>
<tr>
<td>Write</td>
<td>322</td>
<td>Generate controller-select pulse.</td>
</tr>
<tr>
<td>Read</td>
<td>323</td>
<td>Not used.</td>
</tr>
<tr>
<td>Write</td>
<td>323</td>
<td>Write pattern to DMA and interrupt mask register.</td>
</tr>
</tbody>
</table>
System I/O Channel Interface

The following lines are used by the disk controller:

A0-A19 Positive true 20-bit address. The least-significant 10 bits contain the I/O address within the range of hex 320 to hex 323 when an I/O read or write is executed by the system unit. The full 20 bits are decoded to address the read-only storage (ROS) between the addresses of hex C8000 and C9FFF.

D0-D7 Positive 8-bit data bus over which data and status information is passed between the system board and the controller.

IOR Negative true signal that is asserted when the system board reads status or data from the controller under either programmed I/O or DMA control.

IOW Negative true signal that is asserted when the system board sends a command or data to the controller under either programmed I/O or DMA control.

AEN Positive true signal that is asserted when the DMA in the system board is generating the I/O Read (–IOR) or I/O Write (–IOW) signals and has control of the address and data buses.

RESET Positive true signal that forces the disk controller to its initial power-up condition.

IRQ 5 Positive true interrupt request signal that is asserted by the controller when enabled to interrupt the system board on the return ending status byte from the controller.

DRQ 3 Positive true DMA-request signal that is asserted by the controller when data is available for transfer to or from the controller under DMA control. This signal remains active until the system board’s DMA channel activates the DMA-acknowledge signal (–DACK 3) in response.

DACK 3 This signal is true when negative, and is generated by the system board DMA channel in response to a DMA request (DRQ 3).
### Fixed Disk Adapter Interface Specifications

#### Disk Drive Connector J1

<table>
<thead>
<tr>
<th>Signal</th>
<th>Pin Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ground</td>
<td>1-33</td>
</tr>
<tr>
<td>Odd Numbers</td>
<td>1-33</td>
</tr>
<tr>
<td>Reserved</td>
<td>4, 16, 30, 32</td>
</tr>
<tr>
<td>Reduced Write Current</td>
<td>2</td>
</tr>
<tr>
<td>Write Gate</td>
<td>6</td>
</tr>
<tr>
<td>Seek Complete</td>
<td>8</td>
</tr>
<tr>
<td>Track 00</td>
<td>10</td>
</tr>
<tr>
<td>Write Fault</td>
<td>12</td>
</tr>
<tr>
<td>Head Select 2⁰</td>
<td>14</td>
</tr>
<tr>
<td>Head Select 2¹</td>
<td>18</td>
</tr>
<tr>
<td>Index</td>
<td>20</td>
</tr>
<tr>
<td>Ready</td>
<td>22</td>
</tr>
<tr>
<td>Step</td>
<td>24</td>
</tr>
<tr>
<td>Drive Select 1</td>
<td>26</td>
</tr>
<tr>
<td>Drive Select 2</td>
<td>28</td>
</tr>
<tr>
<td>Direction In</td>
<td>34</td>
</tr>
</tbody>
</table>

Position 5 has No Pin (for Cable Orientation)

#### Disk Drive Connector J2 or J3

<table>
<thead>
<tr>
<th>Signal</th>
<th>Pin Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ground</td>
<td>2, 4, 6, 8, 12, 16, 20</td>
</tr>
<tr>
<td>Drive Select</td>
<td>1</td>
</tr>
<tr>
<td>Reserved</td>
<td>3, 7</td>
</tr>
<tr>
<td>Spare</td>
<td>9, 10, 5 (No Pin)</td>
</tr>
<tr>
<td>Ground</td>
<td>11</td>
</tr>
<tr>
<td>MFM Write Data</td>
<td>13</td>
</tr>
<tr>
<td>MFM Write Data</td>
<td>14</td>
</tr>
<tr>
<td>MFM Ground</td>
<td>15</td>
</tr>
<tr>
<td>MFM Read Data</td>
<td>17</td>
</tr>
<tr>
<td>MFM Read Data</td>
<td>18</td>
</tr>
<tr>
<td>Ground</td>
<td>19</td>
</tr>
</tbody>
</table>

Fixed Disk Adapter 1-193
Notes:
IBM 10MB Fixed Disk Drive

The disk drive is a random-access storage device that uses two non-removable 5-1/4 inch disks for storage. Each disk surface employs one movable head to service 306 cylinders. The total formatted capacity of the four heads and surfaces is 10 megabytes (17 sectors per track with 512 bytes per sector and a total of 1224 tracks).

An impact-resistant enclosure provides mechanical and contamination protection for the heads, actuator, and disks. A self-contained recirculating system supplies clean air through a 0.3-micron filter. Thermal isolation of the stepper and spindle motor assemblies from the disk enclosure results in a very low temperature rise within the enclosure. This isolation provides a greater off-track margin and the ability to perform read and write operations immediately after power-up with no thermal stabilization delay.
<table>
<thead>
<tr>
<th>Media</th>
<th>Rigid media disk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Tracks</td>
<td>1224</td>
</tr>
<tr>
<td>Track Density</td>
<td>345 tracks per inch</td>
</tr>
<tr>
<td>Dimensions</td>
<td></td>
</tr>
<tr>
<td>Height</td>
<td>3.25 inches (82.55 mm)</td>
</tr>
<tr>
<td>Width</td>
<td>5.75 inches (146.05 mm)</td>
</tr>
<tr>
<td>Depth</td>
<td>8.0 inches (203.2 mm)</td>
</tr>
<tr>
<td>Weight</td>
<td>4.6 lb (2.08 kg)</td>
</tr>
<tr>
<td>Temperature</td>
<td></td>
</tr>
<tr>
<td>Operating</td>
<td>40°F to 122°F (4°C to 50°C)</td>
</tr>
<tr>
<td>Non operating</td>
<td>-40°F to 140°F (-40°C to 60°C)</td>
</tr>
<tr>
<td>Relative Humidity</td>
<td></td>
</tr>
<tr>
<td>Operating</td>
<td>8% to 80% (non condensing)</td>
</tr>
<tr>
<td>Maximum Wet Bulb</td>
<td>78°F (26°C)</td>
</tr>
<tr>
<td>Shock</td>
<td></td>
</tr>
<tr>
<td>Operating</td>
<td>10 Gs</td>
</tr>
<tr>
<td>Non operating</td>
<td>20 Gs</td>
</tr>
<tr>
<td>Access Time</td>
<td>3 ms track-to-track</td>
</tr>
<tr>
<td>Average Latency</td>
<td>8.33 ms</td>
</tr>
<tr>
<td>Error Rates</td>
<td></td>
</tr>
<tr>
<td>Soft Read Errors</td>
<td>1 per $10^{10}$ bits read</td>
</tr>
<tr>
<td>Hard Read Errors</td>
<td>1 per $10^{12}$ bits read</td>
</tr>
<tr>
<td>Seek Errors</td>
<td>1 per $10^6$ seeks</td>
</tr>
<tr>
<td>Design Life</td>
<td>5 years (8,000 hours MTF)</td>
</tr>
<tr>
<td>Disk Speed</td>
<td>3600 rpm ±1%</td>
</tr>
<tr>
<td>Transfer Rate</td>
<td>5.0 M bits/sec</td>
</tr>
<tr>
<td>Recording Mode</td>
<td>MFM</td>
</tr>
<tr>
<td>Power</td>
<td>$+12$ Vdc $\pm 5%$ 1.8 A (4.5 A maximum)</td>
</tr>
<tr>
<td></td>
<td>$+5$ Vdc $\pm 5%$ 0.7 A (1.0 A maximum)</td>
</tr>
<tr>
<td>Maximum Ripple</td>
<td>1% with equivalent resistive load</td>
</tr>
</tbody>
</table>

**Mechanical and Electrical Specifications**
Three memory expansion options and a memory module kit are available for the IBM Personal Computer XT. They are the 32KB, 64KB, and 64/256KB Memory Expansion Options and the 64KB Memory Module Kit. The base system has a standard 128K of RAM on the system board. One or two memory module kits can be added, providing the system board with 192K or 256K of RAM. The base 64/256K option has a standard 64K of RAM. One, two, or three 64K memory module kits may be added, providing the 64/256K option with 128K, 192K, or 256K of RAM. A maximum of 256K of RAM can be installed on the system board as modules without using any of the system unit expansion slots or expansion options. The system board must be populated to the maximum 256K of RAM before any memory expansion options can be installed.

An expansion option must be configured to reside at a sequential 32K or 64K memory address boundary within the system address space. This is done by setting DIP switches on the option.

The 32K and 64K options both use 16K by 1 bit memory modules, while the 64/256K option uses 64K by 1 bit memory modules. On the 32K and 64/256K options, 16-pin industry-standard parts are used. On the 64K option, stacked modules are used resulting in a 32K by 1 bit, 18-pin module. This allows the 32K and 64K options to have approximately the same physical size.

All memory expansion options are parity checked. If a parity error is detected, a latch is set and an I/O channel check line is activated, indicating an error to the processor.

In addition to the memory modules, the memory expansion options contain the following circuits: bus buffering, dynamic memory timing generation, address multiplexing, and card-select decode logic.

Dynamic-memory refresh timing and address generation are functions that are performed on the system board and made available in the I/O channel for all devices.
To allow the system to address 32K, 64K, or 64/256K memory expansion options, refer to “Appendix G: Switch Settings” for the proper memory expansion option switch settings.

**Operating Characteristics**

The system board operates at a frequency of 4.77 MHz, which results in a clock cycle of 210 ns.

Normally four clock cycles are required for a bus cycle so that an 840-ns memory cycle time is achieved. Memory-write and memory-read cycles both take four clock cycles, or 840 ns.

General specifications for memory used on all cards are:

<table>
<thead>
<tr>
<th></th>
<th>16K by 1 Bit</th>
<th>32K by 1 Bit</th>
<th>64K by 1 Bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access</td>
<td>250 ns</td>
<td>250 ns</td>
<td>200 ns</td>
</tr>
<tr>
<td>Cycle</td>
<td>410 ns</td>
<td>410 ns</td>
<td>345 ns</td>
</tr>
</tbody>
</table>

**Memory Module Description**

Both the 32K and the 64K options contain 18 dynamic memory modules. The 32K memory expansion option utilizes 16K by 1 bit modules, and the 64K memory expansion option utilizes 32K by 1 bit modules.

The 64/256K option has four banks of 9 pluggable sockets. Each bank will accept a 64K memory module kit, consisting of 9 (64K by 1) modules. The kits must be installed sequentially into banks 1, 2, and 3. The base 64/256K option comes with modules installed in bank 0, providing 64K of memory. One, two, or three 64K bits may be added, upgrading the option to 128K, 192K, or 256K of memory.
The 16K by 1 and the 32K by 1 modules require three voltage levels: +5 Vdc, −5 Vdc, and +12 Vdc. The 64K by 1 modules require only one voltage level of +5 Vdc. All three memory modules require 128 refresh cycles every 2 ns. Absolute maximum access times are:

<table>
<thead>
<tr>
<th></th>
<th>16K by 1 Bit</th>
<th>32K by 1 Bit</th>
<th>64K by 1 Bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>From RAS</td>
<td>250 ns</td>
<td>250 ns</td>
<td>200 ns</td>
</tr>
<tr>
<td>From CAS</td>
<td>165 ns</td>
<td>165 ns</td>
<td>115 ns</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pin</th>
<th>16K by 1 Bit Module (used on 32K option)</th>
<th>32K by 1 Bit Module (used on 64K option)</th>
<th>64K by 1 Bit Module (used on 64/256K option)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>−5 Vdc</td>
<td>−5 Vdc</td>
<td>N/C</td>
</tr>
<tr>
<td>2</td>
<td>Data In**</td>
<td>Data In**</td>
<td>Data In***</td>
</tr>
<tr>
<td>3</td>
<td>−Write</td>
<td>−Write</td>
<td>−Write</td>
</tr>
<tr>
<td>4</td>
<td>−RAS</td>
<td>−RAS 0</td>
<td>−RAS</td>
</tr>
<tr>
<td>5</td>
<td>A0</td>
<td>−RAS 1</td>
<td>A0</td>
</tr>
<tr>
<td>6</td>
<td>A2</td>
<td>A0</td>
<td>A2</td>
</tr>
<tr>
<td>7</td>
<td>A1</td>
<td>A2</td>
<td>A1</td>
</tr>
<tr>
<td>8</td>
<td>+12 Vdc</td>
<td>A1</td>
<td>+5 Vdc</td>
</tr>
<tr>
<td>9</td>
<td>+5 Vdc</td>
<td>+12 Vdc</td>
<td>A7</td>
</tr>
<tr>
<td>10</td>
<td>A5</td>
<td>+5 Vdc</td>
<td>A5</td>
</tr>
<tr>
<td>11</td>
<td>A4</td>
<td>A5</td>
<td>A4</td>
</tr>
<tr>
<td>12</td>
<td>A3</td>
<td>A4</td>
<td>A3</td>
</tr>
<tr>
<td>13</td>
<td>A6</td>
<td>A3</td>
<td>A6</td>
</tr>
<tr>
<td>14</td>
<td>Data Out**</td>
<td>A6</td>
<td>Data Out***</td>
</tr>
<tr>
<td>15</td>
<td>−CAS</td>
<td>Data Out**</td>
<td>−CAS</td>
</tr>
<tr>
<td>16</td>
<td>GND</td>
<td>−CAS 1</td>
<td>GND</td>
</tr>
<tr>
<td>17</td>
<td>*</td>
<td>−CAS 0</td>
<td>*</td>
</tr>
<tr>
<td>18</td>
<td>*</td>
<td>GND</td>
<td>*</td>
</tr>
</tbody>
</table>

*16K by 1 and 64K by 1 bit modules have 16 pins.
**Data In and Data Out are tied together (three-state bus).
***Data In and Data Out are tied together on Data Bits 0-7 (three-state bus).

Memory Module Pin Configuration
Switch-Configurable Start Address

Each card has a small DIP module, that contains eight switches. The switches are used to set the card start address as follows:

<table>
<thead>
<tr>
<th>Number</th>
<th>32K and 64K Options</th>
<th>64/256K Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ON: A19=0; OFF: A19=1</td>
<td>ON: A19=0; OFF: A19=1</td>
</tr>
<tr>
<td>2</td>
<td>ON: A18=0; OFF: A18=1</td>
<td>ON: A18=0; OFF: A18=1</td>
</tr>
<tr>
<td>3</td>
<td>ON: A17=0; OFF: A17=1</td>
<td>ON: A17=0; OFF: A17=1</td>
</tr>
<tr>
<td>4</td>
<td>ON: A16=0; OFF: A16=1</td>
<td>ON: A16=0; OFF: A16=1</td>
</tr>
<tr>
<td>5</td>
<td>ON: A15=0; OFF: A15=1*</td>
<td>ON: Select 64K</td>
</tr>
<tr>
<td>6</td>
<td>Not used</td>
<td>ON: Select 128K</td>
</tr>
<tr>
<td>7</td>
<td>Not used</td>
<td>ON: Select 192K</td>
</tr>
<tr>
<td>8</td>
<td>Used only in 64K RAM Card*</td>
<td>ON: Select 256K</td>
</tr>
</tbody>
</table>

*Switch 8 may be set on the 64K memory expansion option to use only half the memory on the card (that is, 32K). If switch 8 is on, all 64K is accessible. If switch 8 is off, address bit A15 (as set by switch 5) is used to determine which 32K are accessible, and the 64K option behaves as a 32K option.

DIP Module Start Address

Memory Option Switch Settings

Switch settings for all memory expansion options are located in “Appendix G: Switch Settings.”

1-200 Memory Expansion Options
The following method can be used to determine the switch settings for the 32K memory expansion option.

Starting Address = xxxK

32K = Decimal value

Convert decimal value to binary

Bit... 4 3 2 1 0
Bit value... 16 8 4 2 1

Switch

The following method can be used to determine the switch settings for the 64K memory expansion option.

Starting Address = xxxK

64K = Decimal value

Convert decimal value to binary

Bit... 3 2 1 0
Bit value... 8 4 2 1

Switch
The following method can be used to determine the switch settings for the 64/256K memory expansion option.

Starting Address = \( xxxK \) =Decimal value

\[ \frac{64K}{xxxK} \]

Convert decimal value to binary

Bit . . . . . . . . . . 3 2 1 0
Bit value . . . . 8 4 2 1

Switch

Amount of memory installed on option

- 256K
- 192K (on = logical 1)
- 128K
- 64K

bit
- 0
- 1
- 2 (off = logical 1)
- 3
IBM Game Control Adapter

The game control adapter allows up to four paddles or two joy sticks to be attached to the system. This card fits into one of the system board’s or expansion board’s expansion slots. The game control interface cable attaches to the rear of the adapter. In addition, four inputs for switches are provided. Paddle and joystick positions are determined by changing resistive values sent to the adapter. The adapter 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 position can be determined. This adapter could be used as a general purpose I/O card with four analog (resistive) inputs plus four digital input points.

Game Control Adapter Block Diagram
Functional Description

Address Decode
The select on the game control adapter is generated by two 74LS138s as an address decoder. AEN must be inactive while the address is hex 201 in order to generate the select. The select allows a write to fire the one-shots or read to give the values of the trigger buttons and one-shot outputs.

Data Bus Buffer/Driver
The data bus is buffered by a 74LS244 buffer/driver. For an In from address hex 201, the game control adapter will drive the data bus; at all other times, the buffer is left in the high impedance state.

Trigger Buttons
The trigger button inputs are read by an In from address hex 201. A trigger button is on each joy stick or paddle. 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. Software should be aware that these buttons are not debounced in hardware.

Joy Stick Positions
The joy stick position is indicated by a potentiometer for each coordinate. Each potentiometer has a range from 0 to 100 k-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 at once 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.
I/O Channel Description

A9-A0: Address lines 9 through 0 are used to address the game control adapter.

D7-D0: Data lines 7 through 0 are the data bus.

IOR, IOW: I/O read and I/O write are used when reading from or writing to an adapter (In, Out).

AEN: When active, the adapter must be inactive and the data bus driver inactive.

+5 Vdc: Power for the game control adapter.

GND: Common ground.

A19-A10: Unused.

MEMR, MEMW: Unused.

DACK0-DACK3: Unused.

IRQ7-IRQ2: Unused.

DRQ3-DRQ1: Unused.

ALE, T/C: Unused.

CLK, OSC: Unused.

I/O CHCK: Unused.

I/O CH RDY: Unused.

RESET DRV: Unused.

−5 Vdc, +12 Vdc, −12 Vdc: Unused.
Interface Description

The game control adapter has eight input lines, four of which are digital inputs and 4 of which are resistive inputs. The inputs are read with one In from address hex 201.

The four digital inputs each have a 1 k-ohm pullup resistor to +5 Vdc. With no drives on these inputs, a 1 is read. For a 0 reading, the inputs must be pulled to ground.

The four resistive pullups, measured to +5 Vdc, will be converted to a digital pulse with a duration proportional to the resistive load, according to the following equation:

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

The user must first begin the conversation 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-bit 0) function in the same manner; their digital pulse will all go high simultaneously and will reset independently according to the input resistance value.

The typical input to the game control adapter is a set of joy sticks or game paddles.

The joy sticks will typically be a set of two (A and B). These will have one or two buttons each with two variable resistances each, with a range from 0 to 100 k-ohms. One variable resistance will indicate the X-coordinate and the other variable resistance will indicate the Y-coordinate. This should be attached to give the following input data:

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>B-#2</td>
<td>B-#1</td>
<td>A-#2</td>
<td>A-#1</td>
<td>B-Y</td>
<td>B-X</td>
<td>A-Y</td>
<td>A-X</td>
</tr>
<tr>
<td>Button</td>
<td>Button</td>
<td>Button</td>
<td>Button</td>
<td>Coordinate</td>
<td>Coordinate</td>
<td>Coordinate</td>
<td>Coordinate</td>
</tr>
</tbody>
</table>
The game paddles will have a set of two (A and B) or four (A, B, C, and D) paddles. These will have one button each and one variable resistance each, with a range of 0 to 100 k-ohms. This should be attached to give the following input data:

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>D Button</td>
<td>C Button</td>
<td>B Button</td>
<td>A Button</td>
<td>D Coordinate</td>
<td>C Coordinate</td>
<td>B Coordinate</td>
<td>A Coordinate</td>
</tr>
</tbody>
</table>

Refer to “Joy Stick Schematic Diagram” for attaching game controllers.

Note: Potentiometer for X- and Y-Coordinates has a range of 0 to 100 k-ohms. Button is normally open; closed when pressed.
At Standard TTL Levels

<table>
<thead>
<tr>
<th>Voltage</th>
<th>Adapter Pin No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>+5 Vdc</td>
<td>1</td>
</tr>
<tr>
<td>Button 4</td>
<td>2</td>
</tr>
<tr>
<td>Position 0</td>
<td>3</td>
</tr>
<tr>
<td>Ground</td>
<td>4</td>
</tr>
<tr>
<td>Ground</td>
<td>5</td>
</tr>
<tr>
<td>Position 1</td>
<td>6</td>
</tr>
<tr>
<td>Button 5</td>
<td>7</td>
</tr>
<tr>
<td>+5 Vdc</td>
<td>8</td>
</tr>
<tr>
<td>Button 6</td>
<td>10</td>
</tr>
<tr>
<td>Position 2</td>
<td>11</td>
</tr>
<tr>
<td>Ground</td>
<td>12</td>
</tr>
<tr>
<td>Position 3</td>
<td>13</td>
</tr>
<tr>
<td>Button 7</td>
<td>14</td>
</tr>
<tr>
<td>+5 Vdc</td>
<td>15</td>
</tr>
</tbody>
</table>

Connector Specifications
IBM Prototype Card

The prototype card is 4.2 inches (106.7 millimeters) high by 13.2 inches (335.3 millimeters) long and plugs into an expansion unit or system unit expansion slot. All system control signals and voltage requirements are provided through a 2 by 31 position card-edge tab.

The card contains a voltage bus (+5 Vdc) and a ground bus (0 Vdc). Each bus borders the card, with the voltage bus on the back (pin side) and the ground bus on the front (component side). A system interface design is also provided on the prototype card.

The prototype card can also accommodate a D-shell connector if it is needed. The connector size can range from a 9 to a 37 position connector.

Note: Install all components on the component side of the prototype card. The total width of the card including components should not exceed 0.500 inch (12.7 millimeters). If these specifications are not met, components on the prototype card may touch other cards plugged into adjacent slots.
Prototype Card Block Diagram

1-210 Prototype Card
I/O Channel Interface

The prototype card has two layers screened onto it (one on the front and one on the back). It also has 3,909 plated through-holes that are 0.040 inch (10.1 millimeters) in size and have a 0.060 inch (1.52 millimeters) pad, which is located on a 0.10 inch (2.54 millimeters) grid. There are 37 plated through-holes that are 0.048 inch (1.22 millimeters) in size. These holes are located at the rear of the card (viewed as if installed in the machine). These 37 holes are used for a 9 to 37 position D-shell connector. The card also has 5 holes that are 0.125 inch (3.18 millimeters) in size. One hold is located just above the two rows of D-shell connector holes, and the other four are located in the corners of the board (one in each corner).

Prototype Card Layout

The component side has the ground bus [0.05 inch (1.27 millimeters) wide] screened on it and card-edge tabs that are labeled A1 through A31.
The component side also has a silk screen printed on it that is used as a component guide for the I/O interface.

Component Side

The pin side has a +5 Vdc bus [0.05 inch (1.27 millimeters) wide] screened onto it and card-edge tabs that are labeled B1 through B31.

Pin Side
Each card-edged tab is connected to a plated through-hole by a 0.012-inch (0.3-millimeter) land. There are three ground tabs connected to the ground bus by three 0.012-inch (0.3 millimeter) lands. Also, there are two +5 Vdc tabs connected to the voltage bus by two 0.012-inch (0.3 millimeter) lands.

For additional interfacing information, refer to “I/O Channel Description” and “I/O Channel Diagram” in this manual. Also, the “Prototype Card Interface Logic Diagram” is in Appendix D of this manual. If the recommended interface logic is used, the list of TTL type numbers listed below will help you select the necessary components.

<table>
<thead>
<tr>
<th>Component</th>
<th>TTL Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1</td>
<td>74LS245</td>
<td>Octal Bus Transceiver</td>
</tr>
<tr>
<td>U2, U5</td>
<td>74LS244</td>
<td>Octal Buffers Line Driver/Line Receivers</td>
</tr>
<tr>
<td>U4</td>
<td>74LS04</td>
<td>Hex Inverters</td>
</tr>
<tr>
<td>U3</td>
<td>74LS08</td>
<td>Quadruple 2 - Input Positive - AND Gate</td>
</tr>
<tr>
<td>U6</td>
<td>74LS02</td>
<td>Quadruple 2 - Input Positive - NOR Gate</td>
</tr>
<tr>
<td>U7</td>
<td>74LS21</td>
<td>Dual 4 - Input Positive - AND Gate</td>
</tr>
<tr>
<td>C1</td>
<td></td>
<td>10.0 μF Tantalum Capacitor</td>
</tr>
<tr>
<td>C2, C3, C4</td>
<td></td>
<td>0.047 μF Ceramic Capacitor</td>
</tr>
</tbody>
</table>

**System Loading and Power Limitations**

Because of the number of options that may be installed in the system, the I/O bus loading should be limited to one Schottky TTL load. If the interface circuitry on the card is used, then this requirement is met.

Refer to the power supply information in this manual for the power limitations to be observed.
Prototype Card External Interface

If a connector is required for the card function, then you should purchase one of the recommended connectors (manufactured by Amp) or equivalent listed below:

<table>
<thead>
<tr>
<th>Connector Size</th>
<th>Part Number (Amp)</th>
</tr>
</thead>
<tbody>
<tr>
<td>9-pin D-shell (Male)</td>
<td>205865-1</td>
</tr>
<tr>
<td>9-pin D-shell (Female)</td>
<td>205866-1</td>
</tr>
<tr>
<td>15-pin D-shell (Male)</td>
<td>205867-1</td>
</tr>
<tr>
<td>15-pin D-shell (Female)</td>
<td>205868-1</td>
</tr>
<tr>
<td>25-pin D-shell (Male)</td>
<td>205857-1</td>
</tr>
<tr>
<td>25-pin D-shell (Female)</td>
<td>205858-1</td>
</tr>
<tr>
<td>37-pin D-shell (Male)</td>
<td>205859-1</td>
</tr>
<tr>
<td>37-pin D-shell (Female)</td>
<td>205860-1</td>
</tr>
</tbody>
</table>

The following example shows a 15-pin, D-shell, female connector attached to a prototype card.
IBM Asynchronous Communications Adapter

The asynchronous communications adapter system control signals and voltage requirements are provided through a 2 by 31 position card edge tab. Two jumper modules are provided on the adapter. One jumper module selects either RS-232C or current-loop operation. The other jumper module selects one of two addresses for the adapter, so two adapters may be used in one system.

The adapter 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 9600 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, error, line status and data set interrupts. Diagnostic capabilities provide loopback functions of transmit/receive and input/output signals.

The heart of the adapter is a INS8250 LSI chip or functional equivalent. Features in addition to those listed above are:

- Full double buffering eliminates 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), ring indicator (RI), and carrier detect.
- False-start bit detection.
- Line-break generation and detection.

All communications protocol is a function of the system microcode 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. The following figure is a block diagram of the asynchronous communications adapter.
Modes of Operation

The different modes of operation are selected by programming the 8250 asynchronous communications element. This is done by selecting the I/O address (hex 3F8 to 3FF primary, and hex 2F8 to 2FF secondary) and writing data out to the card. Address bits A0, A1, and A2 select the different registers that define the modes of operation. Also, the divisor latch access bit (bit 7) of the line control register is used to select certain registers.
### I/O Decode (in Hex)

<table>
<thead>
<tr>
<th>Primary Adapter</th>
<th>Alternate Adapter</th>
<th>Register Selected</th>
<th>DLAB State</th>
</tr>
</thead>
<tbody>
<tr>
<td>3F8</td>
<td>2F8</td>
<td>TX Buffer</td>
<td>DLAB=0 (Write)</td>
</tr>
<tr>
<td>3F8</td>
<td>2F8</td>
<td>RX Buffer</td>
<td>DLAB=0 (Read)</td>
</tr>
<tr>
<td>3F8</td>
<td>2F8</td>
<td>Divisor Latch LSB</td>
<td>DLAB=1</td>
</tr>
<tr>
<td>3F9</td>
<td>2F9</td>
<td>Divisor Latch MSB</td>
<td>DLAB=1</td>
</tr>
<tr>
<td>3F9</td>
<td>2F9</td>
<td>Interrupt Enable Register</td>
<td></td>
</tr>
<tr>
<td>3FA</td>
<td>3FA</td>
<td>Interrupt Identification Registers</td>
<td></td>
</tr>
<tr>
<td>3FB</td>
<td>2FB</td>
<td>Line Control Register</td>
<td></td>
</tr>
<tr>
<td>3FC</td>
<td>2FC</td>
<td>Modem Control Register</td>
<td></td>
</tr>
<tr>
<td>3FD</td>
<td>2FD</td>
<td>Line Status Register</td>
<td></td>
</tr>
<tr>
<td>3FE</td>
<td>2FE</td>
<td>Modem Status Register</td>
<td></td>
</tr>
</tbody>
</table>

### I/O Decodes

### Hex Address 3F8 to 3FF and 2F8 to 2FF

<table>
<thead>
<tr>
<th>A9</th>
<th>A8</th>
<th>A7</th>
<th>A6</th>
<th>A5</th>
<th>A4</th>
<th>A3</th>
<th>A2</th>
<th>A1</th>
<th>A0</th>
<th>DLAB</th>
<th>Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1/0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td>0</td>
<td>Receive Buffer (read), Transmit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>Holding Reg. (write)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>Interrupt Identification</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>Line Control</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>Modem Control</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>Line Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>Modem Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>Divisor Latch (LSB)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>Divisor Latch (MSB)</td>
</tr>
</tbody>
</table>

**Note:** Bit 8 will be logical 1 for the adapter designated as primary or a logical 0 for the adapter designated as alternate (as defined by the address jumper module on the adapter).

A2, A1 and A0 bits are “don’t cares” and are used to select the different register of the communications chip.

### Address Bits
Interrupts

One interrupt line is provided to the system. This interrupt is IRQ4 for a primary adapter or IRQ3 for an alternate adapter, and is positive active. To allow the communications card to send interrupts to the system, bit 3 of the modem control register must be set to 1 (high). At this point, any interrupts allowed by the interrupt enable register will cause an interrupt.

The data format will be as follows:

![Data Format Diagram]

Data bit 0 is the first bit to be transmitted or received. The adapter automatically inserts the start bit, the correct parity bit if programmed to do so, and the stop bit (1, 1-1/2, or 2 depending on the command in the line-control register).

Interface Description

The communications adapter provides an EIA RS-232C-like interface. One 25-pin D-shell, male type connector is provided to attach various peripheral devices. In addition, a current loop interface is also located in this same connector. A jumper block is provided to manually select either the voltage interface, or the current loop interface.

The current loop interface is provided to attach certain printers provided by IBM that use this particular type of interface.

Pin 18 + receive current loop data
Pin 25 - receive current loop return
Pin 9 + transmit current loop return
Pin 11 - transmit current loop data
Current Loop Interface

The voltage interface is a serial interface. It supports certain data and control signals, as listed below.

- **Pin 2**: Transmitted Data
- **Pin 3**: Received Data
- **Pin 4**: Request to Send
- **Pin 5**: Clear to Send
- **Pin 6**: Data Set Ready
- **Pin 7**: Signal Ground
- **Pin 8**: Carrier Detect
- **Pin 20**: Data Terminal Ready
- **Pin 22**: Ring Indicator

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.
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>

Invalid Levels

+$15\text{ Vdc}$

On Function

+$3\text{ Vdc}$

$0\text{ Vdc}$

Invalid Levels

$-3\text{ Vdc}$

Off Function

$-15\text{ Vdc}$

Invalid Levels

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 $-3\text{ Vdc}$ with respect to signal ground. The signal will be considered in the “spacing” condition when the voltage is more positive than $+3\text{ Vdc}$ with respect to signal ground. The region between $+3\text{ Vdc}$ and $-3\text{ Vdc}$ is defined as the transition region, and considered an invalid level. The voltage that is more negative than $-15\text{ Vdc}$ or more positive than $+15\text{ Vdc}$ 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 “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 $+3\text{ Vdc}$ with respect to signal ground and is “off” when the voltage is more negative than $-3\text{ Vdc}$ with respect to signal ground.
INS8250 Functional Pin Description

The following describes the function of all INS8250 input/output pins. Some of these descriptions reference internal circuits.

Note: In the following descriptions, a low represents a logical 0 (0 Vdc nominal) and a high represents a logical 1 (+2.4 Vdc nominal).

Input Signals

Chip Select (CS0, CS1, CS2), Pins 12-14: When CS0 and CS1 are high and CS2 is low, the chip is selected. Chip selection is complete when the decoded chip select signal is latched with an active (low) address strobe (ADS) input. This enables communications between the INS8250 and the processor.

Data Input Strobe (DISTR, DISTR) Pins 22 and 21: When DISTR is high or DISTR is low while the chip is selected, allows the processor to read status information or data from a selected register of the INS8250.

Note: Only an active DISTR or DISTR input is required to transfer data from the INS8250 during a read operation. Therefore, tie either the DISTR input permanently low or the DISTR input permanently high, if not used.

Data Output Strobe (DOSTR, DOSTR), Pins 19 and 18: When DOSTR is high or DOSTR is low while the chip is selected, allows the processor to write data or control words into a selected register of the INS8250.

Note: Only an active DOSTR or DOSTR input is required to transfer data to the INS8250 during a write operation. Therefore, tie either the DOSTR input permanently low or the DOSTR input permanently high, if not used.

Address Strobe (ADS), Pin 25: When low, provides latching for the register select (A0, A1, A2) and chip select (CS0, CS1, CS2) signals.
Note: An active ADS input is required when the register select (A0, A1, A2) signals are not stable for the duration of a read or write operation. If not required, tie the ADS input permanently low.

Register Select (A0, A1, A2), Pins 26-28: These three inputs are used during a read or write operation to select an INS8250 register to read from or write to as indicated in the table below. Note that 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 INS8250 registers. The DLAB must be set high by the system software to access the baud generator divisor latches.

<table>
<thead>
<tr>
<th>DLAB</th>
<th>A2</th>
<th>A1</th>
<th>A0</th>
<th>Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Receiver Buffer (Read), Transmitter Holding Register (Write)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Interrupt Enable</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Interrupt Identification (Read Only)</td>
</tr>
<tr>
<td>X</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Line Control</td>
</tr>
<tr>
<td>X</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Modem Control</td>
</tr>
<tr>
<td>X</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Line Status</td>
</tr>
<tr>
<td>X</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Modem Status</td>
</tr>
<tr>
<td>X</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>None</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Divisor Latch (Least Significant Bit)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Divisor Latch (Most Significant Bit)</td>
</tr>
</tbody>
</table>

Master Reset (MR), Pin 35: When high, clears all the registers (except the receiver buffer, transmitter holding, and divisor latches), and the control logic of the INS8250. Also, the state of various output signals (SOUT, INTRPT, OUT1, OUT2, RTS, DTR) are affected by an active MR input. Refer to the “Asynchronous Communications Reset Functions” table.

Receiver Clock (RCLK), Pin 9: This input is the 16 x baud rate clock for the receiver section of the chip.

Serial Input (SIN), Pin 10: Serial data input from the communications link (peripheral device, modem, or data set).
Clear to Send (CTS), Pin 36: The CTS signal is a modem control function input whose condition can be tested by the processor by reading bit 4 (CTS) of the modem status register. Bit 0 (DCTS) of the modem status register indicates whether the CTS input has changed state since the previous reading of the modem status register.

Note: Whenever the CTS bit of the modem status register changes state, an interrupt is generated if the modem status interrupt is enabled.

Data Set Ready (DSR), Pin 37: When low, indicates that the modem or data set is ready to establish the communications link and transfer data with the INS8250. The DSR signal is a modem-control function input whose condition can be tested by the processor by reading bit 5 (DSR) of the modem status register. Bit 1 (DDSR) of the modem status register indicates whether the DSR input has changed since the previous reading of the modem status register.

Note: Whenever the DSR bit of the modem status register changes state, an interrupt is generated if the modem status interrupt is enabled.

Received Line Signal Detect (RLSD), Pin 38: When low, indicates that the data carrier had been detected by the modem or data set. The RLSD signal is a modem-control function input whose condition can be tested by the processor by reading bit 7 (RLSD) of the modem status register. Bit 3 (DRLSD) of the modem status register indicates whether the RLSD input has changed state since the previous reading of the modem status register.

Note: Whenever the RLSD bit of the modem status register changes state, an interrupt is generated if the modem status interrupt is enabled.
Ring Indicator (RI), Pin 39: When low, indicates that a telephone ringing signal has been received by the modem or data set. The RI signal is a modem-control function input whose condition can be tested by the processor by reading bit 6 (RI) of the modem status register. Bit 2 (TERI) of the modem status register indicates whether the RI input has changed from a low to high state since the previous reading of the modem status register.

Note: Whenever the RI bit of the modem status register changes from a high to a low state, an interrupt is generated if the modem status register interrupt is enabled.

VCC, Pin 40: +5 Vdc supply.

VSS, Pin 20: Ground (0 Vdc) reference.

Output Signals

Data Terminal Ready (DTR), Pin 33: When low, informs the modem or data set that the INS8250 is ready to communicate. The DTR output signal can be set to an active low by programming bit 0 (DTR) of the modem control register to a high level. The DTR signal is set high upon a master reset operation.

Request to Send (RTS), Pin 32: When low, informs the modem or data set that the INS8250 is ready to transmit data. The RTS output signal can be set to an active low by programming bit 1 (RTS) of the modem control register. The RTS signal is set high upon a master reset operation.

Output 1 (OUT 1), Pin 34: User-designated output that can be set to an active low by programming bit 2 (OUT 1) of the modem control register to a high level. The OUT 1 signal is set high upon a master reset operation.

Output 2 (OUT 2), Pin 31: User-designated output that can be set to an active low by programming bit 3 (OUT 2) of the modem control register to a high level. The OUT 2 signal is set high upon a master reset operation.
Chip Select Out (CSOUT), Pin 24: When high, indicates that the chip has been selected by active CS0, CS1, and CS2 inputs. No data transfer can be initiated until the CSOUT signal is a logical 1.

Driver Disable (DDIS), Pin 23: Goes low whenever the processor is reading data from the INS8250. A high-level DDIS output can be used to disable an external transceiver (if used between the processor and INS8250 on the D7-D0 data bus) at all times, except when the processor is reading data.

Baud Out (BAUDOUT), Pin 15: 16 x clock signal for the transmitter section of the INS8250. The clock rate is equal to the main reference oscillator frequency divided by the specified divisor in the baud generator divisor latches. The BAUDOUT may also be used for the receiver section by typing this output to the RCLK input of the chip.

Interrupt (INTRPT), Pin 30: Goes high whenever any one of the following interrupt types has an active high condition and is enabled through the IER: receiver error flag, received data available, transmitter holding register empty, or modem status. The INTRPT signal is reset low upon the appropriate interrupt service or a master reset operation.

Serial Output (SOUT), Pin 11: Composite serial data output to the communications link (peripheral, modem, or data set). The SOUT signal is set to the marking (logical 1) state upon a master reset operation.

Input/Output Signals

Data Bus (D7-D0), Pins 1-8: This bus comprises eight tri-state input/output lines. The bus provides bidirectional communications between the INS8250 and the processor. Data, control words, and status information are transferred through the D7-D0 data bus.

External Clock Input/Output (XTAL1, XTAL2), Pins 16 and 17: These two pins connect the main timing reference (crystal or signal clock) to the INS8250.
Programming Considerations

The INS8250 has a number of accessible registers. The system programmer may access or control any of the INS8250 registers through the processor. These registers are used to control INS8250 operations and to transmit and receive data. A table listing and description of the accessible registers follows.

<table>
<thead>
<tr>
<th>Register/Signal</th>
<th>Reset Control</th>
<th>Reset State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interrupt Enable Register</td>
<td>Master Reset</td>
<td>All Bits Low (0-3 Forced and 4-7 Permanent)</td>
</tr>
<tr>
<td>Interrupt Identification Register</td>
<td>Master Reset</td>
<td>Bit 0 is High, Bits 1 and 2 Low</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bits 3-7 are Permanently Low</td>
</tr>
<tr>
<td>Line Control Register</td>
<td>Master Reset</td>
<td>All Bits Low</td>
</tr>
<tr>
<td>Modem Control Register</td>
<td>Master Reset</td>
<td>All Bits Low</td>
</tr>
<tr>
<td>Line Status Register</td>
<td>Master Reset</td>
<td>Except Bits 5 and 6 are High</td>
</tr>
<tr>
<td>Modem Status Register</td>
<td>Master Reset</td>
<td>Bits 0-3 Low</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bits 4-7 - Input Signal</td>
</tr>
<tr>
<td>SOUT</td>
<td>Master Reset</td>
<td>High</td>
</tr>
<tr>
<td>INTRPT (RCVR Errors)</td>
<td>Read LSR/MR</td>
<td>Low</td>
</tr>
<tr>
<td>INTRPT (RCVR Data Ready)</td>
<td>Read RBR/MR</td>
<td>Low</td>
</tr>
<tr>
<td>INTRPT (RCVR Data Ready)</td>
<td>Read IIR/</td>
<td>Low</td>
</tr>
<tr>
<td></td>
<td>Write THR/MR</td>
<td></td>
</tr>
<tr>
<td>INTRPT (Modem Status Changes)</td>
<td>Read MSR/MR</td>
<td>Low</td>
</tr>
<tr>
<td>OUT 2</td>
<td>Master Reset</td>
<td>High</td>
</tr>
<tr>
<td>RTS</td>
<td>Master Reset</td>
<td>High</td>
</tr>
<tr>
<td>DTR</td>
<td>Master Reset</td>
<td>High</td>
</tr>
<tr>
<td>OUT 1</td>
<td>Master Reset</td>
<td>High</td>
</tr>
</tbody>
</table>

**Asynchronous Communications Reset Functions**
Line-Control Register

The system programmer specifies the format of the asynchronous data communications exchange through the line-control register. In addition to controlling the format, the programmer may retrieve the contents of the line-control register for inspection. This feature simplifies system programming and eliminates the need for separate storage in system memory of the line characteristics. The contents of the line-control register are indicated and described below.

<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>Word Length Select Bit 0 (WLS0)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Word Length Select Bit 1 (WLS1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of Stop Bits (STB)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parity Enable (PEN)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Even Parity Select (EPS)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stick Parity</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Set Break</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Divisor Latch Access Bit (DLAB)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Line-Control Register (LCR)

Bits 0 and 1: These two bits specify the number of bits in each transmitted or received serial character. The encoding of bits 0 and 1 is as follows:

<table>
<thead>
<tr>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Word Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>5 Bits</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>6 Bits</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>7 Bits</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>8 Bits</td>
</tr>
</tbody>
</table>
Bit 2: This bit specifies the number of stop bits in each transmitted or received serial character. If bit 2 is a logical 0, one stop bit is generated or checked in the transmit or receive data, respectively. If bit 2 is logical 1 when a 5-bit word length is selected through bits 0 and 1, 1-1/2 stop bits are generated or checked. If bit 2 is logical 1 when either a 6-, 7-, or 8-bit word length is selected, two stop bits are generated or checked.

Bit 3: This bit is the parity enable bit. When bit 3 is a logical 1, a parity bit is generated (transmit data) or checked (receive data) between the last data word bit and stop bit of the serial data. (The parity bit is used to produce an even or odd number of 1’s when the data word bits and the parity bit are summed.)

Bit 4: This bit is the even parity select bit. When bit 3 is a logical 1 and bit 4 is a logical 0, an odd number of logical 1’s is transmitted or checked in the data word bits and parity bit. When bit 3 is a logical 1 and bit 4 is a logical 1, an even number of bits is transmitted or checked.

Bit 5: This bit is the stick parity bit. When bit 3 is a logical 1 and bit 5 is a logical 1, the parity bit is transmitted and then detected by the receiver as a logical 0 if bit 4 is a logical 1, or as a logical 1 if bit 4 is a logical 0.

Bit 6: This bit is the set break control bit. When bit 6 is a logical 1, the serial output (SOUT) is forced to the spacing (logical 0) state and remains there regardless of other transmitter activity. The set break is disabled by setting bit 6 to a logical 0. This feature enables the processor to alert a terminal in a computer communications system.

Bit 7: This bit is the divisor latch access bit (DLAB). It must be set high (logical 1) to access the divisor latches of the baud rate generator during a read or write operation. It must be set low (logical 0) to access the receiver buffer, the transmitter holding register, or the interrupt enable register.
Programmable Baud Rate Generator

The INS8250 contains a programmable baud rate generator that is capable of taking the clock input (1.8432 MHz) and dividing it by any divisor from 1 to \((2^{16} - 1)\). The output frequency of the baud generator is 16 x the baud rate \([\text{divisor } = \frac{\text{frequency input}}{\text{baud rate} \times 16}]\). 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.

<table>
<thead>
<tr>
<th>Hex Address 3F8 DLAB = 1</th>
<th>Bit 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>Bit 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Divisor Latch Least Significant Bit (DLL)
Hex Address 3F9 DLAB = 1

Bit 7 6 5 4 3 2 1 0

Bit 8
Bit 9
Bit 10
Bit 11
Bit 12
Bit 13
Bit 14
Bit 15

Divisor Latch Most Significant Bit (DLM)

The following figure illustrates the use of the baud rate generator with a frequency of 1.8432 MHz. For baud rates of 9600 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 9600 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 Difference Between Desired and Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>2304</td>
<td>900</td>
<td>—</td>
</tr>
<tr>
<td>75</td>
<td>1536</td>
<td>600</td>
<td>—</td>
</tr>
<tr>
<td>110</td>
<td>1047</td>
<td>417</td>
<td>0.026</td>
</tr>
<tr>
<td>134.5</td>
<td>857</td>
<td>359</td>
<td>0.058</td>
</tr>
<tr>
<td>150</td>
<td>768</td>
<td>300</td>
<td>—</td>
</tr>
<tr>
<td>300</td>
<td>384</td>
<td>180</td>
<td>—</td>
</tr>
<tr>
<td>600</td>
<td>192</td>
<td>0C0</td>
<td>—</td>
</tr>
<tr>
<td>1200</td>
<td>96</td>
<td>060</td>
<td>—</td>
</tr>
<tr>
<td>1800</td>
<td>64</td>
<td>040</td>
<td>—</td>
</tr>
<tr>
<td>2000</td>
<td>58</td>
<td>03A</td>
<td>0.69</td>
</tr>
<tr>
<td>2400</td>
<td>48</td>
<td>030</td>
<td>—</td>
</tr>
<tr>
<td>3600</td>
<td>32</td>
<td>020</td>
<td>—</td>
</tr>
<tr>
<td>4800</td>
<td>24</td>
<td>018</td>
<td>—</td>
</tr>
<tr>
<td>7200</td>
<td>16</td>
<td>010</td>
<td>—</td>
</tr>
<tr>
<td>9600</td>
<td>12</td>
<td>00C</td>
<td>—</td>
</tr>
</tbody>
</table>

Baud Rate at 1.843 MHz

1-230 Asynchronous Adapter
Line Status Register

This 8-bit register provides status information on the processor concerning the data transfer. The contents of the line status register are indicated and described below:

### Line Status Register (LSR)

**Bit 0:** This bit is the receiver data ready (DR) indicator. Bit 0 is set to a logical 1 whenever a complete incoming character has been received and transferred into the receiver buffer register. Bit 0 may be reset to a logical 0 either by the processor reading the data in the receiver buffer register or by writing a logical 0 into it from the processor.

**Bit 1:** This bit is the overrun error (OE) indicator. Bit 1 indicates that data in the receiver buffer register was not read by the processor before the next character was transferred into the receiver buffer register, thereby destroying the previous character. The OE indicator is reset whenever the processor reads the contents of the line status register.

**Bit 2:** This bit is the parity error (PE) indicator. Bit 2 indicates that the received data character does not have the correct even or odd parity, as selected by the even parity-select bit. The PE bit is set to a logical 1 upon detection of a parity error and is reset to a logical 0 whenever the processor reads the contents of the line status register.
Bit 3: This bit is the framing error (FE) indicator. Bit 3 indicates that the received character did not have a valid stop bit. Bit 3 is set to a logical 1 whenever the stop bit following the last data bit or parity is detected as a zero bit (spacing level).

Bit 4: This bit is the break interrupt (BI) indicator. Bit 4 is set to a logical 1 whenever the received data input is held in the spacing (logical 0) state for longer than a full word transmission time (that is, the total time of start bit + data bits + parity + stop bits).

Note: Bits 1 through 4 are the error conditions that produce a receiver line status interrupt whenever any of the corresponding conditions are detected.

Bit 5: This bit is the transmitter holding register empty (THRE) indicator. Bit 5 indicates that the INS8250 is ready to accept a new character for transmission. In addition, this bit causes the INS8250 to issue an interrupt to the processor when the transmit holding register empty interrupt enable is set high. The THRE bit is set to a logical 1 when a character is transferred from the transmitter holding register into the transmitter shift register. The bit is reset to logical 0 concurrently with the loading of the transmitter holding register by the processor.

Bit 6: This bit is the transmitter shift register empty (TSRE) indicator. Bit 6 is set to a logical 1 whenever the transmitter shift register is idle. It is reset to logical 0 upon a data transfer from the transmitter holding register to the transmitter shift register. Bit 6 is a read-only bit.

Bit 7: This bit is permanently set to logical 0.

Interrupt Identification Register

The INS8250 has an on-chip interrupt capability that allows for complete flexibility in interfacing to all the popular microprocessors presently available. In order to provide minimum software overhead during data character transfers, the INS8250 prioritizes interrupts into four levels: receiver line status (priority 1), received data ready (priority 2), transmitter holding register empty (priority 3), and modem status (priority 4).
Information indicating that a prioritized interrupt is pending and the type of prioritized interrupt is stored in the interrupt identification register. Refer to the “Interrupt Control Functions” table. The interrupt identification register (IIR), when addressed during chip-select time, freezes the highest priority interrupt pending, and no other interrupts are acknowledged until that particular interrupt is serviced by the processor. The contents of the IIR are indicated and described below.

<table>
<thead>
<tr>
<th>Hex Address 3FA</th>
<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>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 If Interrupt Pending</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interrupt ID Bit (0)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interrupt ID Bit (1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>= 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>= 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>= 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>= 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>= 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Interrupt Identification Register (IIR)**

**Bit 0:** This bit can be used in either a hard-wired prioritized or polled environment to indicate whether an interrupt is pending and the IIR contents may be used as a pointer to the appropriate interrupt service routine. When bit 0 is a logical 1, no interrupt is pending and polling (if used) is continued.

**Bits 1 and 2:** These two bits of the IIR are used to identify the highest priority interrupt pending as indicated in the “Interrupt Control Functions” table.

**Bits 3 through 7:** These five bits of the IIR are always logical 0.
### Interrupt ID Register

<table>
<thead>
<tr>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Priority Level</th>
<th>Interrupt Type</th>
<th>Interrupt Source</th>
<th>Interrupt Reset Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Highest</td>
<td>Receiver Line Status</td>
<td>Overrun Error or Parity Error or Framing Error or Break Interrupt</td>
<td>Reading the Line Status Register</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Second</td>
<td>Received Data Available</td>
<td>Receiver Data Available</td>
<td>Reading the Receiver Buffer Register</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Third</td>
<td>Transmitter Holding Register Empty</td>
<td>Transmitter Holding Register Empty</td>
<td>Reading the IIR Register (if source of interrupt) or Writing into the Transmitter Holding Register</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Fourth</td>
<td>Modem Status</td>
<td>Clear to Send or Data Set Ready or Ring Indicator or Received Line Signal Direct</td>
<td>Reading the Modem Status Register</td>
</tr>
</tbody>
</table>

### Interrupt Control Functions

1-234 Asynchronous Adapter
Interrupt Enable Register

This eight-bit register enables the four types of interrupt of the INS8250 to separately activate the chip interrupt (INTRPT) output signal. It is possible to totally disable the interrupt system by resetting bits 0 through 3 of the interrupt enable register. Similarly, by setting the appropriate bits of this register to a logical 1, selected interrupts can be enabled. Disabling the interrupt system inhibits the interrupt identification register and the active (high) INTRPT output from the chip. All other system functions operate in their normal manner, including the setting of the line status and modem status registers. The contents of the interrupt enable register are indicated and described below:

<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>1 = Enable Data Available Interrupt</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = Enable Tx Holding Register Empty Interrupt</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = Enable Receive Line Status Interrupt</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = Enable Modem Status Interrupt</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>= 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>= 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>= 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>= 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Interrupt Enable Register (IER)

**Bit 0:** This bit enables the received data available interrupt when set to logical 1.

**Bit 1:** This bit enables the transmitter holding register empty interrupt when set to logical 1.

**Bit 2:** This bit enables the receiver line status interrupt when set to logical 1.
Bit 3: This bit enables the modem status interrupt when set to logical 1.

Bits 4 through 7: These four bits are always logical 0.

Modem Control Register

This eight-bit register controls the interface with the modem or data set (or peripheral device emulating a modem). The contents of the modem control register are indicated and described below:

<table>
<thead>
<tr>
<th>Hex Address</th>
<th>3FC</th>
</tr>
</thead>
</table>
| Bit 7 6 5 4 3 2 1 0 | Data Terminal Ready (DTR)  
|               | Request to Send (RTS)  
|               | Out 1  
|               | Out 2  
|               | Loop  
|               | = 0  
|               | = 0  
|               | = 0  

Modem Control Register (MCR)

Bit 0: This bit controls the data terminal ready (DTR) output. When bit 0 is set to logical 1, the DTR output is forced to a logical 0. When bit 0 is reset to a logical 0, the DTR output is forced to a logical 1.

Note: The DIR output of the INS8250 may be applied to an EIA inverting line driver (such as the DS1488) to obtain the proper polarity input at the succeeding modem or data set.

Bit 1: This bit controls the request to send (RTS) output. Bit 1 affects the RTS output in a manner identical to that described above for bit 0.
Bit 2: This bit controls the output 1 (OUT1) signal, which is an auxiliary user-designated output. Bit 2 affects the OUT1 output in a manner identical to that described above for bit 0.

Bit 3: This bit controls the output 2 (OUT2) signal, which is an auxiliary user-designated output. Bit 3 affects the OUT2 output in a manner identical to that described above for bit 0.

Bit 4: This bit provides a loopback feature for diagnostic testing of the INS8250. When bit 4 is set to logical 1, the following occurs: the transmitter serial output (SOUT) is set to the marking (logical 1) state; the receiver serial input (SIN) is disconnected; the output of the transmitter shift register is "looped back" into the receiver shift register input; the four modem control inputs (CTS, DRS, RLSD, and RI) are disconnected; and the four modem control outputs (DTR, RTS, OUT1, and OUT2) are internally connected to the four modem control inputs. In the diagnostic mode, data that is transmitted is immediately received. This feature allows the processor to verify the transmit- and receive-data paths of the INS8250.

In the diagnostic mode, the receiver and transmitter interrupts are fully operational. The modem control interrupts are also operational but the interrupts' sources are now the lower four bits of the modem control register instead of the four modem control inputs. The interrupts are still controlled by the interrupt enable register.

The INS8250 interrupt system can be tested by writing into the lower four bits of the modem status register. Setting any of these bits to a logical 1 generates the appropriate interrupt (if enabled). The resetting of these interrupts is the same as in normal INS8250 operation. To return to normal operation, the registers must be reprogrammed for normal operation and then bit 4 of the modem control register must be reset to logical 0.

Bits 5 through 7: These bits are permanently set to logical 0.
Modem Status Register

This eight-bit register provides the current state of the control lines from the modem (or peripheral device) to the processor. In addition to this current-state information, four bits of the modem status register provide change information. These bits are set to a logical 1 whenever a control input from the modem changes state. They are reset to logical 0 whenever the processor reads the modem status register.

The content of the modem status register are indicated and described below:

<table>
<thead>
<tr>
<th>Hex Address 3FE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit</td>
</tr>
</tbody>
</table>

- Delta Clear to Send (DCTS)
- Delta Data Set Ready (DDSR)
- Trailing Edge Ring Indicator (TERI)
- Delta Rx Line Signal Detect (DRLSD)
- Clear to Send (CTS)
- Data Set Ready (DSR)
- Ring Indicator (RI)
- Receive Line Signal Detect (RLSD)

Modem Status Register (MSR)
Bit 0: This bit is the delta clear to send (DCTS) indicator. Bit 0 indicates that the CTS input to the chip has changed state since the last time it was read by the processor.

Bit 1: This bit is the delta data set ready (DDSR) indicator. Bit 1 indicates that the DRS input to the chip has changed since the last time it was read by the processor.

Bit 2: This bit is the trailing edge of ring indicator (TERI) detector. Bit 2 indicates that the RI input to the chip has changed from an on (logical 1) to an off (logical 0) condition.

Bit 3: This bit is the delta received line signal detector (DRLSD) indicator. Bit 3 indicates that the RLSD input to the chip has changed state.

Note: Whenever bit 0, 1, 2, or 3 is set to a logical 1, a modem status interrupt is generated.

Bit 4: This bit is the complement of the clear to send (CTS) input. If bit 4 (LOOP) of the MCR is set to a logical 1, this is equivalent to RTS in the MCR.

Bit 5: This bit is the complement of the data set ready (DSR) input. If bit 4 of the MCR is set to a logical 1, this bit is equivalent to DTR in the MCR.

Bit 6: This bit is the complement of the ring indicator (RI) input. If bit 4 of the MCR is set to a logical 1, this bit is equivalent to OUT 1 in the MCR.

Bit 7: This bit is the complement of the received line signal detect (RLSD) input. If bit 4 of the MCR is set to a logical 1, this bit is equivalent to OUT 2 of the MCR.
Receiver Buffer Register

The receiver buffer register contains the received character as defined below:

<table>
<thead>
<tr>
<th>Hex Address 3F8</th>
<th>DLAB = 0</th>
<th>Read Only</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 7</td>
<td>Bit 6</td>
<td>Bit 5</td>
</tr>
</tbody>
</table>

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

Receiver Buffer Register (RBR)

Bit 0 is the least significant bit and is the first bit serially received.
Transmitter Holding Register

The transmitter holding register contains the character to be serially transmitted and is defined below:

<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>
</tbody>
</table>

- Data Bit 0
- Data Bit 1
- Data Bit 2
- Data Bit 3
- Data Bit 4
- Data Bit 5
- Data Bit 6
- Data Bit 7

Transmitter Holding Register (THR)

Bit 0 is the least significant bit and is the first bit serially transmitted.
Selecting the Interface Format and Adapter Address

The voltage or current loop interface and adapter address are selected by plugging the programmed shunt modules with the locator dots up or down. See the figure below for the configurations.
At Standard RS-232C Levels
(With Exception of Current Loops)

<table>
<thead>
<tr>
<th>Description</th>
<th>Pin</th>
</tr>
</thead>
<tbody>
<tr>
<td>NC</td>
<td>1</td>
</tr>
<tr>
<td>Transmitted Data</td>
<td>2</td>
</tr>
<tr>
<td>Received Data</td>
<td>3</td>
</tr>
<tr>
<td>Request to Send</td>
<td>4</td>
</tr>
<tr>
<td>Clear to Send</td>
<td>5</td>
</tr>
<tr>
<td>Data Set Ready</td>
<td>6</td>
</tr>
<tr>
<td>Signal Ground</td>
<td>7</td>
</tr>
<tr>
<td>Received Line Signal Detector</td>
<td>8</td>
</tr>
<tr>
<td>+Transmit Current Loop Data</td>
<td>9</td>
</tr>
<tr>
<td>NC</td>
<td>10</td>
</tr>
<tr>
<td>–Transmit Current Loop Data</td>
<td>11</td>
</tr>
<tr>
<td>NC</td>
<td>12</td>
</tr>
<tr>
<td>NC</td>
<td>13</td>
</tr>
<tr>
<td>NC</td>
<td>14</td>
</tr>
<tr>
<td>NC</td>
<td>15</td>
</tr>
<tr>
<td>NC</td>
<td>16</td>
</tr>
<tr>
<td>NC</td>
<td>17</td>
</tr>
<tr>
<td>+Receive Current Loop Data</td>
<td>18</td>
</tr>
<tr>
<td>NC</td>
<td>19</td>
</tr>
<tr>
<td>Data Terminal Ready</td>
<td>20</td>
</tr>
<tr>
<td>NC</td>
<td>21</td>
</tr>
<tr>
<td>Ring Indicator</td>
<td>22</td>
</tr>
<tr>
<td>NC</td>
<td>23</td>
</tr>
<tr>
<td>NC</td>
<td>24</td>
</tr>
<tr>
<td>–Receive Current Loop Return</td>
<td>25</td>
</tr>
</tbody>
</table>

Note: To avoid inducing voltage surges on interchange circuits, signals from interchange circuits shall not be used to drive inductive devices, such as relay coils.

Connector Specifications
Binary Synchronous Communications Adapter

The binary synchronous communications (BSC) adapter is a 4-inch high by 7.5-inch wide card that provides an RS232C-compatible communication interface for the IBM Personal Computer. All system control, voltage, and data signals are provided through a 2- by 31-position card-edge tab. External interface is in the form of EIA drivers and receivers connected to an RS232C, standard 25-pin, D-shell connector.

The adapter is programmed by communication software to operate in binary synchronous mode. Maximum transmission rate is 9600 bits per second (bps). The heart of the adapter is an Intel 8251A Universal Synchronous/Asynchronous Receiver/Transmitter (USART). An Intel 8255A-5 programmable peripheral interface (PPI) is also used for an expanded modem interface, and an Intel 8253-5 programmable interval timer provides time-outs and generates interrupts.

The following is a block diagram of the BSC adapter.
Functional Description

8251A Universal Synchronous/Asynchronous Receiver/Transmitter

The 8251A operational characteristics are programmed by the system unit's software, and it can support virtually any form of synchronous data technique currently in use. In the configuration being described, the 8251A is used for IBM's binary synchronous communications (BSC) protocol in half-duplex mode.

Operation of the 8251A is started by programming the communications format, then entering commands to tell the 8251A what operation is to be performed. In addition, the 8251A can pass device status to the system unit by doing a Status Read operation. The sequence of events to accomplish this are mode instruction, command instruction, and status read. Mode instruction must follow a master reset operation. Commands can be issued in the data block at any time during operation of the 8251A.

A block diagram of the 8251A follows:
Data Bus Buffer

The system unit's data bus interfaces the 8251A through the data bus buffer. Data is transferred or received by the buffer upon execution of input or output instructions from the system unit. Control words, command words, and status information are also transferred through the data bus buffer.

Read/Write Control Logic

The read/write control logic controls the transfer of information between the system unit and the 8251A. It consists of pins designated as RESET, CLK, WR, RD, C/D, and CS.

RESET: The Reset pin is gated by Port B, bit 4 of the 8255, and performs a master reset of the 8251A. The minimum reset pulse width is 6 clock cycles. Clock-cycle duration is determined by the oscillator speed of the processor.

CLK (Clock): The clock generates internal device timing. No external inputs or outputs are referenced to CLK. The input is the system board's bus clock of 4.77 MHz.

WR (Write): An input to WR informs the 8251A that the system unit is writing data or control words to it. The input is the WR signal from the system-unit bus.

RD (Read): An input to RD informs the 8251A that the processing unit is reading data or status information from it. The input is the RD signal from the system-unit bus.

C/D (Control/Data): An input on this pin, in conjunction with the WR and RD inputs, informs the 8251A that the word on the data bus is either a data character, a control word, or status information. The input is the low-order address bit from the system board's address bus.

CS (Chip Select): A low on the input selects the 8251A. No reading or writing will occur unless the device is selected. An input is decoded at the adapter from the address information on the system-unit bus.
Modem Control

The 8251A has the following input and output control signals which are used to interface the transmission equipment selected by the user.

DSR (Data Set Ready): The DSR input port is a general-purpose, 1-bit, inverting input port. The 8251A can test its condition with a Status Read operation.

CTS (Clear to Send): A low on this input enables the 8251A to transfer serial data if the TxEnable bit in the command byte is set to 1. If either a TxEnable off or CTS off condition occurs while the transmitter is in operation, the transmitter will send all the data in the USART that was written prior to the TxDisable command, before shutting down.

DTR (Data Terminal Ready): The DTR output port is a general-purpose, 1-bit, inverting output port. It can be set low by programming the appropriate bit in the command instruction word.

RTS (Request to Send): The RTS output signal is a general-purpose, 1-bit, inverting output port. It can be set low by programming the appropriate bit in the Command Instruction word.

Transmitter Buffer

The transmitter buffer accepts parallel data from the data-bus buffer, converts it to a serial bit stream, and inserts the appropriate characters or bits for the BSC protocol. The output from the transmit buffer is a composite serial stream of data on the falling edge of Transmit Clock. The transmitter will begin transferring data upon being enabled, if CTS = 0 (active). The transmit data (TxD) line will be set in the marking state upon receipt of a master reset, or when transmit enable/CTS is off and the transmitter is empty (TxEmpty).
Transmitter Control

Transmitter Control manages all activities associated with the transfer of serial data. It accepts and issues the following signals, both externally and internally, to accomplish this function:

**TxRDY (Transmitter Ready):** This output signals the system unit that the transmitter is ready to accept a data character. The TxRDY output pin is used as an interrupt to the system unit (Level 4) and is masked by turning off Transmit Enable. TxRDY is automatically reset by the leading edge of a WR input signal when a data character is loaded from the system unit.

**TxE (Transmitter Empty):** This signal is used only as a status register input.

**TxC (Transmit Clock):** The Transmit Clock controls the rate at which the character is to be transmitted. In synchronous mode, the bit-per-second rate is equal to the TxC frequency. The falling edge of TxC shifts the serial data out of the 8251A.

Receiver Buffer

The receiver accepts serial data, converts it to parallel format, checks for bits or characters that are unique to the communication technique, and sends an “assembled” character to the system unit. Serial data input is received on the RxD (Receive Data) pin, and is clocked in on the rising edge of RxC (Receive Clock).

Receiver Control

This control manages all receiver-related activities. The parity-toggle and parity-error flip-flop circuits are used for parity-error detection, and set the corresponding status bit.
RxRDY (Receiver Ready): This output indicates that the 8251A has a character that is ready to be received by the system unit. RxRDY is connected to the interrupt structure of the system unit (Interrupt Level 3). With Receive Enable off, RxRDY is masked and held in the reset mode. To set RxRDY, the receiver must be enabled, and a character must finish assembly and be transferred to the data output register. Failure to read the received character from the RxRDY output register before the assembly of the next Rx Data character will set an overrun-condition error, and the previous character will be lost.

RxC (Receiver Clock): The receiver clock controls the rate at which the character is to be received. The bit rate is equal to the actual frequency of RxC.

SYNDET (Synchronization Detect): This pin is used for synchronization detection and may be used as either input or output, programmable through the control word. It is reset to output-mode-low upon reset. When used as an output (internal synchronization mode), the SYNDET pin will go to 1 to indicate that the 8251A has found the synchronization character in the receive mode. If the 8251A is programmed to use double synchronization characters (bisynchronization, as in this application), the SYNDET pin will go to 1 in the middle of the last bit of the second synchronization character. SYNDET is automatically reset for a Status Read operation.

8255A-5 Programmable Peripheral Interface

The 8255A-5 is used on the BSC adapter to provide an expanded modem interface and for internal gating and control functions. It has three 8-bit ports, which are defined by the system during initialization of the adapter. All levels are considered plus active unless otherwise indicated. A detailed description of the ports is in “Programming Considerations” in this section.
8253-5 Programmable Interval Timer

The 8253-5 is driven by a divided-by-two system-clock signal. Its outputs are used as clocking signals and to generate inactivity timeout interrupts. These level 4 interrupts occur when either of the timers reaches its programmed terminal counts. The 8253-5 has the following outputs:

Timer 0: Not used for synchronous-mode operation.

Timer 1: Connected to port A, bit 7 of the 8255 and Interrupt Level 4.

Timer 2: Connected to port A, bit 6 of the 8255 and Interrupt Level 4.

Operation

The complete functional definition of the BSC adapter is programmed by the system software. Initialization and control words are sent out by the system to initialize the adapter and program the communications format in which it operates. Once programmed, the BSC Adapter is ready to perform its communication functions.

Transmit

In synchronous transmission, the TxD output is continuously at a mark level until the system sends its first character, which is a synchronization character to the 8251A. When the CTS line goes on, the first character is serially transmitted. All bits are shifted out on the falling edge of TxC. When the 8251A is ready to receive another character from the system for transmission, it raises TxRDY, which causes a level-4 interrupt.
Once transmission has started, the data stream at the TxD output must continue at the TxC rate. If the system does not provide the 8251A with a data character before the 8251A transmit buffers become empty, the synchronization characters will be automatically inserted in the TxD data stream. In this case, the TxE bit in the status register is raised high to signal that the 8251A is empty and that synchronization characters are being sent out. (Note that this TxE bit is in the status register, and is not the TxE pin on the 8251A). TxE does not go low when SYNC is being shifted out. The TxE status bit is internally reset by a data character being written to the 8251A.

Receive

In synchronous reception, the 8251A will achieve character synchronization, because the hardware design of the BSC adapter is intended for internal synchronization. Therefore, the SYNDET pin on the 8251A is not connected to the adapter circuits. For internal synchronization, the Enter Hunt command should be included in the first command instruction word written. Data on the RxD pin is then sampled in on the rising edge of RxC. The content of the RxD buffer is compared at every bit boundary with the first SYNC character until a match occurs. Because the 8251A has been programmed for two synchronization characters (bisynchronization), the next received character is also compared. When both SYNC characters have been detected, the 8251A ends the hunt mode and is in character synchronization. The SYNDET bit in the status register (not the SYNDET pin) is then set high, and is reset automatically by a Status Read.

Once synchronization has occurred, the 8251A begins to assemble received data bytes. When a character is assembled and ready to be transferred to memory from the 8251A, it raises RxRDY, causing an interrupt level 3 to the system.

If the system has not fetched a previous character by the time another received character is assembled (and an interrupt-level 3 issued by the adapter), the old character will be overwritten, and the overrun error flag will be raised. All error flags can be reset by an error reset operation.
Programming Considerations

Before starting data transmission or reception, the BSC adapter is programmed by the system unit to define control and gating ports, timer functions and counts, and the communication environment in which it is to operate.

Typical Programming Sequence

The 8255A-5 programmable peripheral interface (PPI) is initialized for the proper mode by selecting address hex 3A3 and writing the control word. This defines port A as an input, port B as an output for modem control and gating, and port C for 4-bit input and 4-bit output. The bit descriptions for the 8255A-5 are shown in the following figures. Using an output to port C, the adapter is then set to wrap mode, disallow interrupts, and gate external clocks (address=3A2H, data=0DH). The adapter is now isolated from the communication interface, and initialization continues.

Through bit 4 of 8255 Port B, the 8251A reset pin is brought high, held, then dropped. This resets the internal registers of the 8251A.
The 8253-5 programmable interval timer is used in the synchronous mode to provide inactivity time-outs to interrupt the system unit after a preselected period of time has elapsed from the start of a communication operation. Counter 0 is not used for synchronous operation. Counters 1 and 2 are connected to interrupt-level 4, and are programmed to terminal-count values, which will provide the desired time delay before a level-4 interrupt is generated. These interrupts will indicate to the system software that a predetermined period of time has elapsed without a TxRDY (level 4) or RxRDY (level 3) interrupt being sent to the system unit.
The modes for each counter are programmed by selecting each timer-register address and writing the correct control word for counter operation to the adapter. The mode for counters 1 and 2 is set to 0. The terminal-count values are loaded using control-word bits D4 and D5 to select “load.” The 8253-5 Control Word format is shown in the following chart.

<table>
<thead>
<tr>
<th>Control Word Format</th>
<th>Address hex 3A7</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7</td>
<td>D6</td>
</tr>
<tr>
<td>SC1</td>
<td>SC0</td>
</tr>
</tbody>
</table>

**Definition of Control**

**SC — Select Counter:**

<table>
<thead>
<tr>
<th>SC1</th>
<th>SC0</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Select Counter 0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Select Counter 1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Select Counter 2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Illegal</td>
</tr>
</tbody>
</table>

**RL — Read/Load:**

<table>
<thead>
<tr>
<th>RL1</th>
<th>RL0</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Counter Latching operation</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Read/Load most significant byte only</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Read/Load least significant byte only</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Read/Load least significant byte first, then most significant byte</td>
</tr>
</tbody>
</table>

**M — Mode:**

<table>
<thead>
<tr>
<th>M2</th>
<th>M1</th>
<th>M0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**BCD:**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>Binary Counter 16-bits</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>Binary Coded Decimal (BCD) Counter (4 Decades)</td>
</tr>
</tbody>
</table>

8253-5 Control Word Format
8251A Programming Procedures

After the support devices on the BSC adapter are programmed, the 8251A is loaded with a set of control words that define the communication environment. The control words are split into two formats, mode instruction, and command instruction.

Both the mode and command instructions must conform to a specified sequence for proper device operation. The mode instruction must be inserted immediately after a reset operation, before using the 8251A for data communications. The required synchronization characters for the defined communication technique are next loaded into the 8251A (usually hex 32 for BSC). All control words written to the 8251A after the mode instruction will load the command instruction. Command instructions can be written to the 8251A at any time in the data block anytime during the operation of the 8251A. To return to the mode instruction format, the master reset bit in the command instruction word can be set to start an internal reset operation which automatically places the 8251A back into the mode instruction format. Command instructions must follow the mode instructions or synchronization characters.

The following diagram is a typical data block, showing the mode instruction and command instruction.

Typical Data Block
Mode Instruction Definition

The mode instruction defines the general operational characteristics of the 8251A. It follows a reset operation (internal or external). Once the mode instruction has been written to the 8251A by the system unit, synchronization characters or command instructions may be written to the device.

The following figure shows the format for the mode instruction.

<table>
<thead>
<tr>
<th>Mode Instruction Format</th>
<th>Address: Hex 3A9 for BSC</th>
<th>Hex 389 for Alternate BSC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 7 6 5 4 3 2 1 0</td>
<td>Not Used (Always 0)</td>
<td>0 0 5 Bits</td>
</tr>
<tr>
<td></td>
<td>Not Used (Always 0)</td>
<td>0 1 6 Bits</td>
</tr>
<tr>
<td></td>
<td>Character Length Bit</td>
<td>1 0 7 Bits</td>
</tr>
<tr>
<td></td>
<td>Character Length Bit</td>
<td>1 1 8 Bits</td>
</tr>
<tr>
<td></td>
<td>1 = Parity Enable</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = Even Parity</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = SYNDET is an Input</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = Double SYNC Character</td>
<td></td>
</tr>
</tbody>
</table>

Bit 0   Not used; always 0

Bit 1   Not used; always 0

Bit 2 and 3 These two bits are used together to define the character length. With 0 and 1 as inputs on bits 2 and 3, character lengths of 5, 6, 7, and 8 bits can be established, as shown in the preceding figure.

Bit 4   In the synchronous mode, parity is enabled from this bit. A 1 on this bit sets parity enable.

Bit 5   The parity generation/check is set from this bit. For BSC, even parity is used by having bit 5 = 1.

Bit 6   External synchronization is set by this bit. A 1 on this bit establishes synchronization detection as an input.

Bit 7   This bit establishes the mode of character synchronization. A 0 is set on this bit to give double character synchronization.
Command-Instruction Format

The command-instruction format defines a status word that is used to control the actual operation of the 8251A. Once the mode instruction has been written to the 8251A, and SYNC characters loaded, all further “Control Writes” to I/O address hex 3A9 or hex 389 will load a command instruction.

Data is transferred by accessing two I/O ports on the 8251A, ports 3A8 and 388. A byte of data can be read from port 3A8 and can be written to port 388.

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Address: Hex 3A9 for BSC Hex 389 for Alternate BSC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transmit Enable</td>
<td>Data Terminal Ready</td>
<td>Receive Enable</td>
<td>Send Break Character</td>
<td>Error Reset</td>
<td>Request to Send</td>
<td>Internal Reset</td>
<td>Enter Hunt Mode</td>
<td></td>
</tr>
</tbody>
</table>

Command Instruction Format

Bit 0  The Transmit Enable bit sets the function of the 8251A to either enabled (1) or disabled (0).

Bit 1  The Data Terminal Ready bit, when set to 1 will force the data terminal output to 0. This is a one-bit inverting output port.

Bit 2  The Receive Enable bit sets the function to either enable the bit (1), or to disable the bit (0).

Bit 3  The Send Break Character bit is set to 0 for normal BSC operation.

Bit 4  The Error Reset bit is set to 1 to reset error flags from the command instruction.

Bit 5  A 1 on the Request to Send bit will set the output to 0. This is a one-bit inverting output port.
Bit 6  The Internal Reset bit when set to 1 returns the 8251A to mode-instruction format.

Bit 7  The Enter Hunt bit is set to 1 for BSC to enable a search for synchronization characters.

**Status Read Definition**

In telecommunication systems, the status of the active device must often be checked to determine if errors or other conditions have occurred that require the processor’s attention. The 8251A has a status read facility that allows the system software to read the status of the device at anytime during the functional operation. A normal read command is issued by the processor with I/O address hex 3A9 for BSC, and hex 389 for Alternate BSC to perform a status read operation.

The format for a status read word is shown in the figure below. Some of the bits in the status read format have the same meanings as external output pins so the 8251A can be used in a completely polled environment or in an interrupt-driven environment.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>TxRDY (See Note Below)</td>
</tr>
<tr>
<td>1</td>
<td>RxRDY</td>
</tr>
<tr>
<td>2</td>
<td>TxEmpty</td>
</tr>
<tr>
<td>3</td>
<td>Parity Error (PE Flag On when a Parity Error Occurs)</td>
</tr>
<tr>
<td>4</td>
<td>Overrun Error (OE Flag On when Overrun Error Occurs)</td>
</tr>
<tr>
<td>5</td>
<td>Framing Error (Not Used for Synchronous Communications)</td>
</tr>
<tr>
<td>6</td>
<td>SYNDET</td>
</tr>
<tr>
<td>7</td>
<td>Data Set Ready (Indicates that DSR is at 0 Level)</td>
</tr>
</tbody>
</table>

**Note:** TxRDY status bit does not have the same meaning as the 8251A TxRDY output pin. The former is not conditioned by CTS and TxEnable. The latter is conditioned by both CTS and TxEnable.
Bit 0  See the Note in the preceding figure.

Bit 1  An output on this bit means a character is ready to be received by the computer’s 8088 microprocessor.

Bit 2  A 1 on this bit indicates the 8251A has no characters to transmit.

Bit 3  The Parity Error bit sets a flag when errors are detected. It is reset by the error reset in the command instruction.

Bit 4  This bit sets a flag when the computers 8088 microprocessor does not read a character before another one is presented. The 8251A operation is not inhibited by this flag, but the overrun character will be lost.

Bit 5  Not used

Bit 6  SYNDET goes to 1 when the synchronization character is found in receive mode. For BSC, SYNDET goes high in the middle of the last bit of the second synchronization character.

Bit 7  The Data Set Ready bit is a one bit inverting input. It is used to check modem conditions, such as data-set ready.

**Interface Signal Information**

The BSC adapter conforms to interface signal levels standardized by the Electronics Industry Association (EIA) RS232C Standard. These levels are shown in the following figure.

Additional lines, not standardized by the EIA, are pins 11, 18, and 25 on the interface connector. These lines are designated as Select Standby, Test, and Test Indicate. Select Standby is used to support the switched network backup facility of a modem that provides this option. Test and Test Indicate support a modem wrap function on modems that are designated for business-machine, controlled-modem wraps.
<table>
<thead>
<tr>
<th>Driver</th>
<th>EIA RS232C/CCITT V24-V28 Signal Levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>+15 Vdc</td>
<td>Active/Data = 0</td>
</tr>
<tr>
<td>+5 Vdc</td>
<td></td>
</tr>
<tr>
<td>+5 Vdc</td>
<td></td>
</tr>
<tr>
<td>-5 Vdc</td>
<td>Invalid Level</td>
</tr>
<tr>
<td>-5 Vdc</td>
<td></td>
</tr>
<tr>
<td>-15 Vdc</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Receiver</th>
<th>EIA RS232C/CCITT V24-V28 Signal Levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>+25 Vdc</td>
<td>Active/Data = 0</td>
</tr>
<tr>
<td>+3 Vdc</td>
<td></td>
</tr>
<tr>
<td>+3 Vdc</td>
<td></td>
</tr>
<tr>
<td>-3 Vdc</td>
<td>Invalid Level</td>
</tr>
<tr>
<td>-3 Vdc</td>
<td></td>
</tr>
<tr>
<td>-25 Vdc</td>
<td></td>
</tr>
</tbody>
</table>

Interface Voltage Levels
Interrupt Information

Interrupt Level 4: Transmitter Ready
Counter 1
Counter 2

Interrupt Level 3: Receiver Ready

The following chart is a device address summary for the primary and alternate modes of the binary synchronous communications adapter.

<table>
<thead>
<tr>
<th>Hex Address</th>
<th>Device</th>
<th>Register Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primary</td>
<td>Alternate</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3A0</td>
<td>380</td>
<td>8255</td>
<td>Port A Data</td>
</tr>
<tr>
<td>3A1</td>
<td>381</td>
<td>8255</td>
<td>Port B Data</td>
</tr>
<tr>
<td>3A2</td>
<td>382</td>
<td>8255</td>
<td>Port C Data</td>
</tr>
<tr>
<td>3A3</td>
<td>383</td>
<td>8255</td>
<td>Mode Set</td>
</tr>
<tr>
<td>3A4</td>
<td>384</td>
<td>8253</td>
<td>Counter 0 LSB</td>
</tr>
<tr>
<td>3A4</td>
<td>384</td>
<td>8253</td>
<td>Counter 0 MSB</td>
</tr>
<tr>
<td>3A5</td>
<td>385</td>
<td>8253</td>
<td>Counter 1 LSB</td>
</tr>
<tr>
<td>3A5</td>
<td>385</td>
<td>8253</td>
<td>Counter 1 MSB</td>
</tr>
<tr>
<td>3A6</td>
<td>386</td>
<td>8253</td>
<td>Counter 2 LSB</td>
</tr>
<tr>
<td>3A6</td>
<td>386</td>
<td>8253</td>
<td>Counter 2 MSB</td>
</tr>
<tr>
<td>3A7</td>
<td>387</td>
<td>8253</td>
<td>Mode Register</td>
</tr>
<tr>
<td>3A8</td>
<td>388</td>
<td>8251</td>
<td>Data Select</td>
</tr>
<tr>
<td>3A9</td>
<td>389</td>
<td>8251</td>
<td>Command/Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Device Address Summary
**Signal Name — Description** | **Pin**
---|---
No Connection | 1
Transmitted Data | 2
Received Data | 3
Request to Send | 4
Clear to Send | 5
Data Set Ready | 6
Signal Ground | 7
Received Line Signal Detector | 8
No Connection | 9
No Connection | 10
Select Standby* | 11
No Connection | 12
No Connection | 13
No Connection | 14
Transmitter Signal Element Timing | 15
No Connection | 16
Receiver Signal Element Timing | 17
Test (IBM Modems Only)* | 18
No Connection | 19
Data Terminal Ready | 20
No Connection | 21
Ring Indicator | 22
Data Signal Rate Selector | 23
No Connection | 24
Test Indicate (IBM Modems Only)* | 25

*Not standardized by EIA (Electronics Industry Association).

**Connector Specifications**
Notes:
IBM Synchronous Data Link Control (SDLC) Communications Adapter

The SDLC communications adapter system control, voltage, and data signals are provided through a 2 by 31 position card edge tab. Modem interface is in the form of EIA drivers and receivers connecting to an RS232C standard 25-pin, D-shell, male connector.

The adapter is programmed by communications software to operate in a half-duplex synchronous mode. Maximum transmission rate is 9600 bits per second, as generated by the attached modem or other data communication equipment.

The SDLC adapter utilizes an Intel 8273 SDLC protocol controller and an Intel 8255A-5 programmable peripheral interface for an expanded external modem interface. An Intel 8253 programmable interval timer is also provided to generate timing and interrupt signals. Internal test loop capability is provided for diagnostic purposes.

The figure below is a block diagram of the SDLC communications adapter.
The 8273 SDLC protocol control module has the following key features:

- Automatic frame check sequence generation and checking.
- Automatic zero bit insertion and deletion.
- TTL compatibility.
- Dual internal processor architecture, allowing frame level command structure and control of data channel with minimal system processor intervention.

The 8273 SDLC protocol controller operations, whether transmission, reception, or port read, are each comprised of three phases:

Command

Command Commands and/or parameters for the required operation are issued by the processor.

Execution

Execution Executes the command, manages the data link, and may transfer data to or from memory utilizing direct memory access (DMA), thus freezing the processor except for minimal interruptions.

Result

Result Returns the outcome of the command by returning interrupt results.

Support of the controller operational phases is through internal registers and control blocks of the 8273 controller.
8273 Protocol Controller Structure

The 8273 module consists of two major interfaces: the processor interface and the modem interface. A block diagram of the 8273 protocol controller module follows.
Processor Interface

The processor interface consists of four major blocks: the control/read/write logic (C/R/W), internal registers, data transfer logic, and data bus buffers.

Control/Read/Write Logic

The control/read/write logic is used by the processor to issue commands to the 8273. Once the 8273 receives and executes a command, it returns the results using the C/R/W logic. The logic is supported by seven registers which are addressed by A0, A1, RD, and WR, in addition to CS. A0 and A1 are the two low-order bits of the adapter address-byte. RD and WR are the processor read and write signals present on the system control bus. CS is the chip select, also decoded by the adapter address logic. The table below shows the address of each register using the C/R/W logic.

<table>
<thead>
<tr>
<th>Address Inputs</th>
<th>Control Inputs</th>
<th>Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>A0</td>
<td>A1</td>
<td>CS</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

8273 SDLC Protocol Controller Register Selection
# 8273 Control/Read/Write Registers

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command</td>
<td>Operations are initialized by writing the appropriate command byte into this register.</td>
</tr>
<tr>
<td>Status</td>
<td>This register provides the general status of the 8273. The status register supplies the processor/adapter handshaking necessary during various phases of the 8273 operation.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Additional information that is required to process the command is written into this register. Some commands require more than one parameter.</td>
</tr>
<tr>
<td>Immediate Result</td>
<td>Commands that execute immediately produce a result byte in this register, to be read by the processor.</td>
</tr>
<tr>
<td>(Result)</td>
<td></td>
</tr>
<tr>
<td>Transmit Interrupt</td>
<td>Results of transmit operations are passed to the processor from this register. This result generates an interrupt to the processor when the result becomes available.</td>
</tr>
<tr>
<td>Results (TxI/R)</td>
<td></td>
</tr>
<tr>
<td>Receiver Interrupt</td>
<td>Results of receive operations are passed to the processor from this register. This result generates an interrupt to the processor when the result becomes available.</td>
</tr>
<tr>
<td>Results (RxI/R)</td>
<td></td>
</tr>
<tr>
<td>Reset</td>
<td>This register provides a software reset function for the 8273.</td>
</tr>
</tbody>
</table>

The other elements of the C/R/W logic are the interrupt lines (RxINT and TxINT). Interrupt priorities are listed in the “Interrupt Information” table in this section. These lines signal the processor that either the transmitter or the receiver requires service (results should be read from the appropriate register), or a data transfer is required. The status of each interrupt line is also reflected by a bit in the status register, so non-interrupt driven operation is also possible by the communication software examining these bits periodically.
Data Interfaces

The 8273 supports two independent data interfaces through the data transfer logic: received data and transmitted data. These interfaces are programmable for either DMA or non-DMA data transfers. Speeds below 9600 bits-per-second may or may not require DMA, depending on the task load and interrupt response time of the processor. The processor DMA controller is used for management of DMA data transfer timing and addressing. The 8273 handles the transfer requests and actual counts of data-block lengths. DMA level 1 is used to transmit and receive data transfers. Dual DMA support is not provided.

Elements of Data Transfer Interface

**TxDRQ/RxDRQ** This line requests a DMA to or from memory and is asserted by the 8273.

**TxDACK/RxDACK** This line notifies the 8273 that a request has been granted and provides access to data regions. This line is returned by the DMA controller (DACK1 on the system unit control bus is connected to TxDACK/RxDACK on the 8273).

**RD (Read)** This line indicates data is to be read from the 8273 and placed in memory. It is controlled by the processor DMA controller.

**WR (Write)** This line indicates if data is to be written to the 8273 from memory and is controlled by the processor DMA controller.

To request a DMA transfer, the 8273 raises the DMA request line. Once the DMA controller obtains control of the system bus, it notifies the 8273 that the DRQ is granted by returning DACK, and WR or RD, for a transmit or receive operation, respectively. The DACK and WR or RD signals transfer data between the 8273 and memory, independent of the 8273 chip-select pin (CS). This “hard select” of data into the transmitter or out of the receiver alleviates the need for the normal transmit and receive data registers, addressed by a combination of address lines, CS, and WR or RD.

1-270   SDLC Adapter
Modem Interface

The modem interface of the 8273 consists of two major blocks: the modem control block and the serial data timing block.

Modem Control Block

The modem control block provides both dedicated and user-defined modem control function. EIA inverting drivers and receivers are used to convert TTL levels to EIA levels.

Port A is a modem control input port. Bits PA0 and PA1 have dedicated functions.

<table>
<thead>
<tr>
<th>8273 Port A (Modem Control Input Port)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit PA  7  6  5  4  3  2  1  0</td>
</tr>
<tr>
<td>PA0 Clear to Send</td>
</tr>
<tr>
<td>PA1 Carrier Detect</td>
</tr>
<tr>
<td>PA2 Data Set Ready</td>
</tr>
<tr>
<td>PA3 CTS Change</td>
</tr>
<tr>
<td>PA4 DSR Change</td>
</tr>
<tr>
<td>Not Used</td>
</tr>
</tbody>
</table>

Bit PA0  This bit reflects the logical state of the clear to send (CTS) pin. The 8273 waits until CTS is active before it starts transmitting a frame. If CTS goes inactive while transmitting, the frame is aborted and the processor is interrupted. A CTS failure will be indicated in the appropriate interrupt-result register.

Bit PA1  This bit reflects the logical state of the carrier detect pin (CD). CD must be active in sufficient time for reception of a frame’s address field. If CD is lost (goes inactive) while receiving a frame, an interrupt is generated with a CD failure result.

Bit PA2  This bit is a sense bit for data set ready (DSR).

Bit PA3  This bit is a sense bit to detect a change in CTS.
Bit PA4  This bit is a sense bit to detect a change in data set ready.

Bits PA5 to PA7  These bits are not used and each is read as a 1 for a read port A command.

Port B is a modem control output port. Bits PB0 and PB5 are dedicated function pins.

<table>
<thead>
<tr>
<th>8273 Port B (Modem Control Output Port)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit PB</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>PB0 - Request to Send</td>
</tr>
<tr>
<td>PB1 - Reserved</td>
</tr>
<tr>
<td>PB2 - Data Terminal Ready</td>
</tr>
<tr>
<td>PB3 - Reserved</td>
</tr>
<tr>
<td>PB4 - Reserved</td>
</tr>
<tr>
<td>PB5 - Flag Detect</td>
</tr>
<tr>
<td>PB6 - Not Used</td>
</tr>
<tr>
<td>PB7 - Not Used</td>
</tr>
</tbody>
</table>

Bit PB0  This bit represents the logical state of request to send (RTS). This function is handled automatically by the 8273.

Bit PB1  Reserved.

Bit PB2  Used for data terminal ready.

Bit PB3  Reserved.

Bit PB4  Reserved.

Bit PB5  This bit reflects the state of the flag detect pin. This pin is activated whenever an active receiver sees a flag character.

Bit PB6  Not used.

Bit PB7  Not used.

1-272  SDLC Adapter
Serial Data Timing Block

The serial data timing block is comprised of two sections: the serial data logic and the digital phase locked loop (DPLL).

Elements of the serial data logic section are the data pins TxD (transmitted data output) and RxD (received data input), and the respective clocks. The leading edge of TxC generates new transmitted data and the trailing edge of RxC is used to capture the received data. The figure below shows the timing for these signals.

8273 SDLC Protocol Controller Transmit/Receive Timing

The digital phase locked loop provided on the 8273 controller module is utilized to capture looped data in proper synchronization during wrap operations performed by diagnostics.
The 8255A-5 contains three eight bit ports. Descriptions of each bit of these ports are as follows:

**8255A-5 Port A Assignments**

<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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Oscillating = Transmit Clock Active</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Oscillating = Receive Clock Active</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Port A is defined as an input port

**8255A-5 Port B Assignments**

<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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Turn On Data Signal Rate Select at Modem Interface</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Turn On Select Standby at Modem Interface</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Turn On Test</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset Modem Status Changed Logic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset 8273</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Enable Level 4 Interrupt</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Port B is defined as an output port
8253-5 Programmable Interval Timer

The 8253-5 is driven by a processor clock signal divided by two. It has the following output:

**Timer 0** Programmed to generate a square wave signal, used as an input to timer 2. Also connected to 8253 port C, bit 5.

**Timer 1** Connected to 8255 port A, bit 7, and interrupt level 4.

**Timer 2** Connected to 8255 port A, bit 6, and interrupt level 4.

**Programming Considerations**

The software aspects of the 8273 involve the communication of both commands from the processor to the 8273 and the return of results of those commands from the 8273 to the processor. Due to the internal processor architecture of the 8273, this system unit/8273 communication is basically a form of interprocessor communication, and must be considered when programming for the SDLC communications adapter.
The protocol for this interprocessor communication is implemented through use of handshaking supplied in the 8273 status register. The bit definitions of this register are shown below.

<table>
<thead>
<tr>
<th>8273 Status Register Format</th>
<th>Hex Address 388</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit  7 6 5 4 3 2 1 0</td>
<td>TxIRA 1 = TxINT Result Available</td>
</tr>
<tr>
<td></td>
<td>RxIRA 1 = RxINT Result Available</td>
</tr>
<tr>
<td></td>
<td>TxINT 1 = Tx Interrupt</td>
</tr>
<tr>
<td></td>
<td>RxINT 1 = Rx Interrupt</td>
</tr>
<tr>
<td></td>
<td>CRBF 1 = Command Result Buffer Full</td>
</tr>
<tr>
<td></td>
<td>CPBF 1 = Command Parameter Buffer Full</td>
</tr>
<tr>
<td></td>
<td>CBF 1 = Command Buffer Full</td>
</tr>
<tr>
<td></td>
<td>CBSY 1 = Command Busy</td>
</tr>
</tbody>
</table>

Bit 0  This bit is the transmitter interrupt result available (TxIRA) bit. This bit is set when the 8273 places an interrupt-result byte in the TxI/R register, and reset when the processor reads the TxI/R register.

Bit 1  This bit is the receiver interrupt result available (RxIRA) bit. It is the corresponding result-available bit for the receiver. It is set when the 8273 places an interrupt-result byte in the RxI/R register and reset when the processor reads the register.

Bit 2  This bit is the transmitter interrupt (TxINT) bit and reflects the state of the TxINT pin. TxINT is set by the 8273 whenever the transmitter needs servicing, and reset when the processor reads the result or performs the data transfer.
Bit 3  This bit is the receiver interrupt (RxINT) bit and is identical to the TxINT, except action is initiated based on receiver interrupt-sources.

Bit 4  This bit is the command result buffer full (CRBF) bit. It is set when the 8273 places a result from an immediate-type command in the result register, and reset when the processor reads the result or performs the data transfer.

Bit 5  This bit is the command parameter buffer full (CPBF) bit and indicates that the parameter register contains a parameter. It is set when the processor deposits a parameter in the parameter register, and reset when the 8273 accepts the parameter.

Bit 6  This bit is the command buffer full (CBF) bit and, when set, it indicates that a byte is present in the command register. This bit is normally not used.

Bit 7  This bit is the command busy (CBSY) bit and indicates when the 8273 is in the command phase. It is set when the processor writes a command into the command register, starting the command phase. It is reset when the last parameter is deposited in the parameter register and accepted by the 8273, completing the command phase.
Initializing the Adapter (Typical Sequence)

Before initialization of the 8273 protocol controller, the support devices on the card must be initialized to the proper modes of operation.

Configuration of the 8255A-5 programmable peripheral interface is accomplished by selecting the mode-set address for the 8255 (see the “SDLC Communications Adapter Device Addresses” table later in this section) and writing the appropriate control word to the device (hex 98) to set ports A, B, and C to the modes described previously in this section.

Next, a bit pattern is output to port C which disallows interrupts, sets wrap mode on, and gates the external clock pins (address = hex 382, data = hex 0D). The adapter is now isolated from the communications interface.

Using bit 4 of port B, the 8273 reset line is brought high, held and then dropped. This resets the internal registers of the 8273.

The 8253-5’s counter 1 and 2 terminal-count values are now set to values which will provide the desired time delay before a level 4 interrupt is generated. These interrupts may be used to indicate to the communication software that a pre-determined period of time has elapsed without a result interrupt (interrupt level 3). The terminal count-values for these counters are set for any time delay which the programmer requires. Counter 0 is also set at this time to mode 3 (generates square wave signal, used to drive counter 2 input).

To setup the counter modes, the address for the 8253 counter mode register is selected (see the “SDLC Communications Adapter Device Addresses” table, later in this section), and the control word for each individual counter is written to the device separately. The control-word format and bit definitions for the 8253 are shown below. Note that the two most-significant bits of the control word select each individual counter, and each counter mode is defined separately.

Once the support devices have been initialized to the proper modes and the 8273 has been reset, the 8273 protocol controller is ready to be configured for the operating mode that defines the communications environment in which it will be used.
### Control Word Format

<table>
<thead>
<tr>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SC1</td>
<td>SCO</td>
<td>RL1</td>
<td>RL0</td>
<td>M2</td>
<td>M1</td>
<td>M0</td>
<td>BCD</td>
<td></td>
</tr>
</tbody>
</table>

### Definitions of Control

**SC** - Select Counter:

<table>
<thead>
<tr>
<th>SC1</th>
<th>SCO</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Select Counter 0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Select Counter 1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Select Counter 2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Illegal</td>
</tr>
</tbody>
</table>

**RL** - Read/Load:

<table>
<thead>
<tr>
<th>RL1</th>
<th>RL0</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Counter Latching operation</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Read/Load most significant byte (MSB)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Read/Load least significant byte (LSB)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Read/Load least significant byte first, then most significant byte.</td>
</tr>
</tbody>
</table>

**M** - Mode:

<table>
<thead>
<tr>
<th>M2</th>
<th>M1</th>
<th>M0</th>
<th>Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Mode 0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Mode 1</td>
</tr>
<tr>
<td>X</td>
<td>1</td>
<td>0</td>
<td>Mode 2</td>
</tr>
<tr>
<td>X</td>
<td>1</td>
<td>1</td>
<td>Mode 3</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Mode 4</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Mode 5</td>
</tr>
</tbody>
</table>

**BCD:**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Binary Counter 16-bits</td>
</tr>
<tr>
<td>1</td>
<td>Binary Coded Decimal (BCD) Counter (4 Decades)</td>
</tr>
</tbody>
</table>

8253-5 Programmable Interval Timer Control Word
Initialization/Configuration Commands

The initialization/configuration commands manipulate internal registers of the 8273, which define operating modes. After chip reset, the 8273 defaults to all 1’s in the mode registers. The initialization/configuration commands either set or reset specified bits in the registers depending on the type of command. One parameter is required with the commands. The parameter is actually the bit pattern (mask) used by the set or reset command to manipulate the register bits.

Set commands perform a logical OR operation of the parameter (mask) of the internal register. This mask contains 1’s where register bits are to be set. Zero (0’s) in the mask cause no change to the corresponding register bit.

Reset commands perform a logical AND operation of the parameter (mask) and internal register. The mask 0 is reset to register bit, and 1 to cause no change.

The following are descriptions of each bit of the operating, serial I/O, one-bit delay, and data transfer mode registers.

Operating Mode Register

<table>
<thead>
<tr>
<th>8273 Operating Mode Register Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit</td>
</tr>
<tr>
<td>-----</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>1 = Flag Stream Mode</td>
</tr>
<tr>
<td>1 = Two Preframe Sync Characters</td>
</tr>
<tr>
<td>1 = Buffered Mode</td>
</tr>
<tr>
<td>1 = Enable Early Tx Interrupt</td>
</tr>
<tr>
<td>1 = EOP Interrupt Enable</td>
</tr>
<tr>
<td>1 = HDLC Abort Enable</td>
</tr>
<tr>
<td>Not Used</td>
</tr>
<tr>
<td>Not Used</td>
</tr>
</tbody>
</table>
Bit 0  If bit 0 is set to a 1, flags are sent immediately if the transmitter was idle when the bit was set. If a transmit or transmit-transparent command was active, flags are sent immediately after transmit completion. This mode is ignored if loop transmit is active or the one-bit-delay mode register is set for one-bit delay. If bit 0 is reset (to 0), the transmitter sends idles on the next character boundary if idle or, after transmission is complete, if the transmitter was active at bit-0 reset time.

Bit 1  If bit 1 is set to a 1, the 8273 sends two characters before the first flag of a frame. These characters are hex 00 if NRZI is set or hex 55 if NRZI is not set. (See “Serial I/O Mode Register,” for NRZI encoding mode format.)

Bit 2  If bit 2 is set to a 1, the 8273 buffers the first two bytes of a received frame (the bytes are not passed to memory). Resetting this bit (to 0) causes these bytes to be passed to and from memory.

Bit 3  This bit indicates to the 8273 when to generate an end-of-frame interrupt. If bit 3 is set, an early interrupt is generated when the last data character has been passed to the 8273. If the processor responds to the early interrupt with another transmit command before the final flag is sent, the final-flag interrupt will not be generated and a new frame will begin when the current frame is complete. Thus, frames may be sent separated by a single flag. A reset condition causes an interrupt to be generated only following a final flag.

Bit 4  This is the EOP-interrupt-mode function and is not used on the SDLC communications-mode adapter. This bit should always be in the reset condition.

Bit 5  This bit is always reset for SDLC operation, which causes the 8273 protocol controller to recognize eight ones (0 1 1 1 1 1 1 1) as an abort character.
Serial I/O Mode Register

<table>
<thead>
<tr>
<th>Bit</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>
</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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>Not Used</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Not Used</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Not Used</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Not Used</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Not Used</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bit 0  Set to 1, this bit specifies NRZI encoding and decoding. Resetting this bit specifies that transmit and receive data be treated as a normal positive-logic bit stream.

Bit 1  When bit 1 is set to 1, the transmit clock is internally routed to the receive-clock circuitry. It is normally used with the loopback bit (bit 2). The reset condition causes the transmit and receive clocks to be routed to their respective 8273 I/O pins.

Bit 2  When bit 2 is set, the transmitted data is internally routed to the received data circuitry. The reset condition causes the transmitted and received data to be routed to their respective 8273 I/O pins.

Data Transfer Mode Register

<table>
<thead>
<tr>
<th>Bit</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>
</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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>Not Used</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1-282 SDLC Adapter
When the data transfer mode register is set, the 8273 protocol controller will interrupt when data bytes are required for transmission, or are available from a reception. If a transmit or receive interrupt occurs and the status register indicates that there is no transmit or receive interrupt result, the interrupt is a transmit or receive data request, respectively. Reset of this register causes DMA requests to be performed with no interrupts to the processor.

One-Bit Delay Mode Register

When one-bit delay is set, the 8273 retransmits the received data stream one-bit delayed. Reset of this bit stops the one-bit delay mode.

The table below is a summary of all set and reset commands associated with the 8273 mode registers. The set or reset mask used to define individual bits is treated as a single parameter. No result or interrupt is generated by the 8273 after execution of these commands.

<table>
<thead>
<tr>
<th>Register</th>
<th>Command</th>
<th>Hex Code</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>One-Bit Delay Mode</td>
<td>Set</td>
<td>A4</td>
<td>Set Mask</td>
</tr>
<tr>
<td></td>
<td>Reset</td>
<td>64</td>
<td>Reset Mask</td>
</tr>
<tr>
<td>Data Transfer Mode</td>
<td>Set</td>
<td>97</td>
<td>Set Mask</td>
</tr>
<tr>
<td></td>
<td>Reset</td>
<td>57</td>
<td>Reset Mask</td>
</tr>
<tr>
<td>Operating Mode</td>
<td>Set</td>
<td>91</td>
<td>Set Mask</td>
</tr>
<tr>
<td></td>
<td>Reset</td>
<td>51</td>
<td>Reset Mask</td>
</tr>
<tr>
<td>Serial I/O Mode</td>
<td>Set</td>
<td>A0</td>
<td>Set Mask</td>
</tr>
<tr>
<td></td>
<td>Reset</td>
<td>60</td>
<td>Reset Mask</td>
</tr>
</tbody>
</table>

8273 SDLC Protocol Controller Mode Register Commands
Command Phase

Although the 8273 is a full duplex device, there is only one command register. Thus, the command register must be used for only one command sequence at a time and the transmitter and receiver may never be simultaneously in a command phase.

The system software starts the command phase by selecting the 8273 command register address and writing a command byte into the register. The following table lists command and parameter information for the 8273 protocol controller. If further information is required by the 8273 prior to execution of the command, the system software must write this information into the parameter register.
<table>
<thead>
<tr>
<th>Command Description</th>
<th>Command (Hex)</th>
<th>Parameter</th>
<th>Results</th>
<th>Result Port</th>
<th>Completion Interrupt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set One-Bit Delay</td>
<td>A4</td>
<td>Set Mask</td>
<td>None</td>
<td></td>
<td>No</td>
</tr>
<tr>
<td>Reset One-Bit Delay</td>
<td>64</td>
<td>Reset Mask</td>
<td>None</td>
<td></td>
<td>No</td>
</tr>
<tr>
<td>Set Data Transfer Mode</td>
<td>97</td>
<td>Set Mask</td>
<td>None</td>
<td></td>
<td>No</td>
</tr>
<tr>
<td>Reset Data Transfer Mode</td>
<td>57</td>
<td>Reset Mask</td>
<td>None</td>
<td></td>
<td>No</td>
</tr>
<tr>
<td>Set Operating Mode</td>
<td>91</td>
<td>Set Mask</td>
<td>None</td>
<td></td>
<td>No</td>
</tr>
<tr>
<td>Reset Operating Mode</td>
<td>51</td>
<td>Reset Mask</td>
<td>None</td>
<td></td>
<td>No</td>
</tr>
<tr>
<td>Set Serial I/O Mode</td>
<td>A0</td>
<td>Set Mask</td>
<td>None</td>
<td></td>
<td>No</td>
</tr>
<tr>
<td>Reset Serial I/O Mode</td>
<td>60</td>
<td>Reset Mask</td>
<td>None</td>
<td></td>
<td>No</td>
</tr>
<tr>
<td>General Receive</td>
<td>C0</td>
<td>B0,B1</td>
<td>RIC,R0,R1,A,C</td>
<td>RXI/R</td>
<td>Yes</td>
</tr>
<tr>
<td>Selective Receive</td>
<td>C1</td>
<td>B0, B1, A1, A2</td>
<td>RIC,R0,R1,A,C</td>
<td>RXI/R</td>
<td>Yes</td>
</tr>
<tr>
<td>Receive Disable</td>
<td>C5</td>
<td>None</td>
<td>None</td>
<td></td>
<td>No</td>
</tr>
<tr>
<td>Transmit Frame</td>
<td>C8</td>
<td>L0,L1,A,C</td>
<td>TIC</td>
<td>TXI/R</td>
<td>Yes</td>
</tr>
<tr>
<td>Transmit Transparent</td>
<td>C9</td>
<td>L0,L1</td>
<td>TIC</td>
<td>TXI/R</td>
<td>Yes</td>
</tr>
<tr>
<td>Abort Transmit Frame</td>
<td>CC</td>
<td>None</td>
<td>TIC</td>
<td>TXI/R</td>
<td>Yes</td>
</tr>
<tr>
<td>Abort Transmit Transparent</td>
<td>CD</td>
<td>None</td>
<td>TIC</td>
<td>TXI/R</td>
<td>Yes</td>
</tr>
<tr>
<td>Read Port A</td>
<td>22</td>
<td>None</td>
<td>Port Value</td>
<td>Result</td>
<td>No</td>
</tr>
<tr>
<td>Read Port B</td>
<td>23</td>
<td>None</td>
<td>Port Value</td>
<td>Result</td>
<td>No</td>
</tr>
<tr>
<td>Set Port B Bit</td>
<td>A3</td>
<td>Set Mask</td>
<td>None</td>
<td></td>
<td>No</td>
</tr>
<tr>
<td>Reset Port B Bit</td>
<td>63</td>
<td>Reset Mask</td>
<td>None</td>
<td></td>
<td>No</td>
</tr>
</tbody>
</table>

**8273 Command Summary Key**

- **B0** — Least significant byte of the receiver buffer length.
- **B1** — Most significant byte of the receiver buffer length.
- **L0** — Least significant byte of the Tx frame length.
- **L1** — Most significant byte of the Tx frame length.
- **A1** — Receive frame address match field one.
- **A2** — Receive frame address match field two.
- **A** — Address field of received frame. If non-buffered mode is specified, this result is not provided.
- **C** — Control field of received frame. If non-buffered mode is specified, this result is not provided.
- **RXI/R** — Receive interrupt result register.
- **TXI/R** — Transmit interrupt result register.
- **R0** — Least significant byte of the length of the frame received.
- **R1** — Most significant byte of the length of the frame received.
- **RIC** — Receiver interrupt result code.
- **TIC** — Transmitter interrupt result code.
A flowchart of the command phase is shown below. Handshaking of the command and parameter bytes is accomplished by the CBSY and CPBF bits of the status register. A command may not be written if the 8273 is busy (CBSY = 1). The original command will be overwritten if a second command is issued while CBSY = 1. The flowchart also indicates a parameter buffer full check. The processor must wait until CPBF = 0 before writing a parameter to the parameter register. Previous parameters are overwritten and lost if a parameter is written while CPBF = 1.

8273 SDLC Protocol Controller Command Phase Flowchart
Execution Phase

During the execution phase, the operation specified by the command phase is performed. If DMA is utilized for data transfers, no processor involvement is required.

For interrupt-driven transfers the 8273 raises the appropriate INT pin (TxINT or RxINT). When the processor responds to the interrupt, it must determine the cause by examining the status register and the associated IRA (interrupt result available) bit of the status register. If IRA = 0, the interrupt is a data transfer request. If IRA = 1, an operation is complete and the associated interrupt result register must be read to determine completion status.

Result Phase

During the result phase, the 8273 notifies the processor of the outcome of a command execution. This phase is initiated by either a successful completion or error detection during execution.

Some commands such as reading or writing the I/O ports provide immediate results. These results are made available to the processor in the 8273 result register. Presence of a valid immediate result is indicated by the CRBF (command result buffer full) bit of the status register.

Non-immediate results deal with the transmitter and receiver. These results are provided in the TxI/R (transmit interrupt result) or RxI/R (receiver interrupt result) registers, respectively. The 8273 notifies the processor that a result is available with the TxIRA and RxIRA bits of the status register. Results consist of one-byte result interrupt code indicating the condition for the interrupt and, if required, one or more bytes supplying additional information. The “Result Code Summary” table later in this section provides information on the format and decode of the transmitter and receiver results.

The following are typical frame transmit and receive sequences. These examples assume DMA is utilized for data transfer operations.
Transmit

Before a frame can be transmitted, the DMA controller is supplied, by the communication software, the starting address for the desired information field. The 8273 is then commanded to transmit a frame (by issuing a transmit frame command).

After a command, but before transmission begins, the 8273 needs some more information (parameters). Four parameters are required for the transmit frame command; the frame address field byte, the frame control field byte, and two bytes which are the least significant and most significant bytes of the information field byte length. Once all four parameters are loaded, the 8273 makes RTS (request to send) active and waits for CTS (clear to send) to go active from the modem interface. Once CTS is active, the 8273 starts the frame transmission. While the 8273 is transmitting the opening flag, address field, and control field, it starts making transmitter DMA requests. These requests continue at character (byte) boundaries until the pre-loaded number of bytes of information field have been transmitted. At this point, the requests stop, the FCS (frame check sequence) and closing flag are transmitted, and the TxINT line is raised, signaling the processor the frame transmission is complete and the result should be read. Note that after the initial command and parameter loading, no processor intervention was required (since DMA is used for data transfers) until the entire frame was transmitted.

General Receive

Receiver operation is very similar. Like the initial transmit sequence, the processor’s DMA controller is loaded with a starting address for a receive data buffer and the 8273 is commanded to receive. Unlike the transmitter, there are two different receive commands; a general receive, where all received frames are transferred to memory, and selective receive, where only frames having an address field matching one of two preprogrammed 8273 address fields are transferred to memory.
(This example covers a general receive operation.) After the receive command, two parameters are required before the receiver becomes active; the least significant and most significant bytes of the receiver buffer length. Once these bytes are loaded, the receiver is active and the processor may return to other tasks. The next frame appearing at the receiver input is transferred to memory using receiver DMA requests. When the closing flag is received, the 8273 checks the FCS and raises its RxINT line. The processor can then read the results, which indicate if the frame was error-free or not. (If the received frame had been longer than the pre-loaded buffer length, the processor would have been notified of that occurrence earlier with a receiver error interrupt). Like the transmit example, after the initial command, the processor is free for other tasks until a frame is completely received.

**Selective Receive**

In selective receive, two parameters (A1 and A2) are required in addition to those for general receive. These parameters are two address match bytes. When commanded to selective receive, the 8273 passes to memory or the processor only those frames having an address field matching either A1 or A2. This command is usually used for secondary stations with A1 designating the secondary address and A2 being the “all parties” address. If only one match byte is needed, A1 and A2 should be equal. As in general receive, the 8273 counts the incoming data bytes and interrupts the processor if the received frame is larger than the preset receive buffer length.
## Result Code Summary

<table>
<thead>
<tr>
<th>Hex Code</th>
<th>Result</th>
<th>Status After Interrupt</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>0C</td>
<td>Early Transmit Interrupt</td>
</tr>
<tr>
<td></td>
<td>0D</td>
<td>Frame Transmit Complete</td>
</tr>
<tr>
<td></td>
<td>0E</td>
<td>DMA Underrun</td>
</tr>
<tr>
<td></td>
<td>0F</td>
<td>Clear to Send Error</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>Abort Complete</td>
</tr>
<tr>
<td>R</td>
<td>X0</td>
<td>A1 Match or General Receive</td>
</tr>
<tr>
<td></td>
<td>X1</td>
<td>A2 Match</td>
</tr>
<tr>
<td></td>
<td>03</td>
<td>CRC Error</td>
</tr>
<tr>
<td></td>
<td>04</td>
<td>Abort Detected</td>
</tr>
<tr>
<td></td>
<td>05</td>
<td>Idle Detected</td>
</tr>
<tr>
<td></td>
<td>06</td>
<td>EOP Detected</td>
</tr>
<tr>
<td></td>
<td>07</td>
<td>Frame Less Than 32 Bits</td>
</tr>
<tr>
<td></td>
<td>08</td>
<td>DMA Overrun</td>
</tr>
<tr>
<td></td>
<td>09</td>
<td>Memory Buffer Overflow</td>
</tr>
<tr>
<td></td>
<td>0A</td>
<td>Carrier Detect Failure</td>
</tr>
<tr>
<td></td>
<td>0B</td>
<td>Receiver Interrupt Overrun</td>
</tr>
</tbody>
</table>

**Note:** X decodes to number of bits in partial byte received.

The first two codes in the receive result code table result from the error free reception of a frame. Since SDLC allows frames of arbitrary length (>32 bits), the high order bits of the receive result report the number of valid received bits in the last received information field byte. The chart below shows the decode of this receive result bit.

<table>
<thead>
<tr>
<th>X</th>
<th>Bits Received in Last Byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>All Eight Bits of Last Byte</td>
</tr>
<tr>
<td>0</td>
<td>Bit0 Only</td>
</tr>
<tr>
<td>8</td>
<td>Bit1-Bit0</td>
</tr>
<tr>
<td>4</td>
<td>Bit2-Bit0</td>
</tr>
<tr>
<td>C</td>
<td>Bit3-Bit0</td>
</tr>
<tr>
<td>2</td>
<td>Bit4-Bit0</td>
</tr>
<tr>
<td>A</td>
<td>Bit5-Bit0</td>
</tr>
<tr>
<td>6</td>
<td>Bit6-Bit0</td>
</tr>
</tbody>
</table>
Address and Interrupt Information

The following tables provide address and interrupt information for the SDLC adapter:

<table>
<thead>
<tr>
<th>Hex Code</th>
<th>Device</th>
<th>Register Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>380</td>
<td>8255</td>
<td>Port A Data</td>
<td>Internal/External Sensing</td>
</tr>
<tr>
<td>381</td>
<td>8255</td>
<td>Port B Data</td>
<td>External Modem Interface</td>
</tr>
<tr>
<td>382</td>
<td>8255</td>
<td>Port C Data</td>
<td>Internal Control</td>
</tr>
<tr>
<td>383</td>
<td>8255</td>
<td>Mode Set</td>
<td>8255 Mode Initialization</td>
</tr>
<tr>
<td>384</td>
<td>8253</td>
<td>Counter 0 LSB</td>
<td>Square Wave Generator</td>
</tr>
<tr>
<td>384</td>
<td>8253</td>
<td>Counter 0 MSB</td>
<td>Square Wave Generator</td>
</tr>
<tr>
<td>385</td>
<td>8253</td>
<td>Counter 1 LSB</td>
<td>Inactivity Time-Outs</td>
</tr>
<tr>
<td>385</td>
<td>8253</td>
<td>Counter 1 MSB</td>
<td>Inactivity Time-Outs</td>
</tr>
<tr>
<td>386</td>
<td>8253</td>
<td>Counter 2 LSB</td>
<td>Inactivity Time-Outs</td>
</tr>
<tr>
<td>386</td>
<td>8253</td>
<td>Counter 2 MSB</td>
<td>Inactivity Time-Outs</td>
</tr>
<tr>
<td>387</td>
<td>8253</td>
<td>Mode Register</td>
<td>8253 Mode Set</td>
</tr>
<tr>
<td>388</td>
<td>8273</td>
<td>Command/Status</td>
<td>Out=Command In=Status</td>
</tr>
<tr>
<td>389</td>
<td>8273</td>
<td>Parameter/Result</td>
<td>Out=Parameter In=Status</td>
</tr>
<tr>
<td>38A</td>
<td>8273</td>
<td>Transmit INT Status</td>
<td>DMA/INT</td>
</tr>
<tr>
<td>38B</td>
<td>8273</td>
<td>Receive INT Status</td>
<td>DMA/INT</td>
</tr>
<tr>
<td>38C</td>
<td>8273</td>
<td>Data</td>
<td>DPC (Direct Program Control)</td>
</tr>
</tbody>
</table>

SDLC Communications Adapter Device Addresses

<table>
<thead>
<tr>
<th>Interrupt Level 3</th>
<th>Transmit/Receive Interrupt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interrupt Level 4</td>
<td>Timer 1 Interrupt</td>
</tr>
<tr>
<td></td>
<td>Timer 2 Interrupt</td>
</tr>
<tr>
<td></td>
<td>Clear to Send Changed</td>
</tr>
<tr>
<td></td>
<td>Data Set Ready Changed</td>
</tr>
</tbody>
</table>

DMA Level One is used for Transmit and Receive

Interrupt Information
Interface Information

The SDLC communications adapter conforms to interface signal levels standardized by the Electronics Industries Association RC-232C Standard. These levels are shown in the figure below.

Additional lines used but not standardized by EIA are pins 11, 18, and 25. These lines are designated as select standby, test and test indicate, respectively. Select Standby is used to support the switched network backup facility of a modem providing this option. Test and test indicate support a modem wrap function on modems which are designed for business machine controlled modem wraps. Two jumpers on the adapter (P1 and P2) are used to connect test and test indicate to the interface, if required (see Appendix D for these jumpers).

![Diagram of interface levels](image-url)
<table>
<thead>
<tr>
<th>Signal Name — Description</th>
<th>Pin</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Connection</td>
<td>1</td>
</tr>
<tr>
<td>Transmitted Data</td>
<td>2</td>
</tr>
<tr>
<td>Received Data</td>
<td>3</td>
</tr>
<tr>
<td>Request to Send</td>
<td>4</td>
</tr>
<tr>
<td>Clear to Send</td>
<td>5</td>
</tr>
<tr>
<td>Data Set Ready</td>
<td>6</td>
</tr>
<tr>
<td>Signal Ground</td>
<td>7</td>
</tr>
<tr>
<td>Received Line Signal Detector</td>
<td>8</td>
</tr>
<tr>
<td>No Connection</td>
<td>9</td>
</tr>
<tr>
<td>No Connection</td>
<td>10</td>
</tr>
<tr>
<td>Select Standby*</td>
<td>11</td>
</tr>
<tr>
<td>No Connection</td>
<td>12</td>
</tr>
<tr>
<td>No Connection</td>
<td>13</td>
</tr>
<tr>
<td>No Connection</td>
<td>14</td>
</tr>
<tr>
<td>Transmitter Signal Element Timing</td>
<td>15</td>
</tr>
<tr>
<td>No Connection</td>
<td>16</td>
</tr>
<tr>
<td>Receiver Signal Element Timing</td>
<td>17</td>
</tr>
<tr>
<td>Test (IBM Modems Only)*</td>
<td>18</td>
</tr>
<tr>
<td>No Connection</td>
<td>19</td>
</tr>
<tr>
<td>Data Terminal Ready</td>
<td>20</td>
</tr>
<tr>
<td>No Connection</td>
<td>21</td>
</tr>
<tr>
<td>Ring Indicator</td>
<td>22</td>
</tr>
<tr>
<td>Data Signal Rate Selector</td>
<td>23</td>
</tr>
<tr>
<td>No Connection</td>
<td>24</td>
</tr>
<tr>
<td>Test Indicate (IBM Modems Only)*</td>
<td>25</td>
</tr>
</tbody>
</table>

*Not standardized by EIA (Electronics Industry Association).

### Connector Specifications

SDLC Adapter 1-293
Notes:
The IBM Communications Adapter Cable is a ten foot cable for connection of an IBM communications adapter to a modem or other RC-232C DCE (data communications equipment). It is fully shielded and provides a high quality, low noise channel for interface between the communications adapter and DCE.

The connector ends are 25-pin D-shell connectors. All pin connections conform with the EIA RS-232C standard. In addition, connection is provided on pins 11, 18 and 25. These pins are designated as select standby, test and test indicate, respectively, on some modems. Select standby is used to support the switched network backup facility, if applicable. Test and test indicate support a modem wrap function on modems designed for business machine controlled modem wraps.
The IBM Communications Adapter Cable connects the following pins on the 25-pin D-shell connectors.

![Diagram of the IBM Communications Adapter Cable](image)

<table>
<thead>
<tr>
<th>Communications Adapter Connector Pin #</th>
<th>Name</th>
<th>Modem Connector Pin #</th>
</tr>
</thead>
<tbody>
<tr>
<td>NC</td>
<td>Outer Cable Shield</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Transmitted Data</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>Received Data</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>Request to Send</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>Clear to Send</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>Data Set Ready</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>Signal Ground (Inner Lead Shields)</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>Received Line Signal Detector</td>
<td>8</td>
</tr>
<tr>
<td>NC</td>
<td></td>
<td>NC</td>
</tr>
<tr>
<td>11</td>
<td>Select Standby</td>
<td>11</td>
</tr>
<tr>
<td>NC</td>
<td></td>
<td>NC</td>
</tr>
<tr>
<td>NC</td>
<td></td>
<td>NC</td>
</tr>
<tr>
<td>NC</td>
<td>Transmitter Signal Element Timing</td>
<td>15</td>
</tr>
<tr>
<td>NC</td>
<td>Receiver Signal Element Timing</td>
<td>NC</td>
</tr>
<tr>
<td>17</td>
<td>Test</td>
<td>17</td>
</tr>
<tr>
<td>18</td>
<td>Data Terminal Ready</td>
<td>NC</td>
</tr>
<tr>
<td>20</td>
<td>Data Signal Rate Selector</td>
<td>23</td>
</tr>
<tr>
<td>22</td>
<td>Ring Indicator</td>
<td>NC</td>
</tr>
<tr>
<td>23</td>
<td>Test</td>
<td>25</td>
</tr>
<tr>
<td>25</td>
<td>Test Indicate</td>
<td>NC</td>
</tr>
</tbody>
</table>

**Connector Specifications**

1-296 Communications Cable
SECTION 2: ROM BIOS AND SYSTEM USAGE

ROM BIOS ........................................ 2-2
Keyboard Encoding and Usage ....................... 2-11
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 (disk and diskette) or character-level I/O operations without concern for device address and operating characteristics. System services, such as time-of-day and memory size determination, are provided by the BIOS.

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

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

Use of BIOS

Access to BIOS is through the 8088 software interrupts. Each BIOS entry point is available through its own interrupt, which can be found in the “8088 Software Interrupt Listing.”

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

```
INT 12H
```

will invoke the BIOS routine for determining memory size and will return the value to the caller.
Parameter Passing

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

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

```
MOV AH,1 ;function is to set time of day.
MOV CX,HIGH_COUNT ;establish the current time.
MOV DX,LOW_COUNT ;set the time.
INT 1AH
```

To read the time of day:

```
MOV AH,0 ;function is to read time of day.
INT 1AH ;read the timer.
```

Generally, the BIOS routines save all registers except for AX and the flags. Other registers are modified on return only if they are returning a value to the caller. The exact register usage can be seen in the prolog 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>D11</td>
</tr>
<tr>
<td>4-7</td>
<td>1</td>
<td>Single Step</td>
<td>D11</td>
</tr>
<tr>
<td>8-B</td>
<td>2</td>
<td>Nonmaskable</td>
<td>NMI_INT</td>
</tr>
<tr>
<td>C-F</td>
<td>3</td>
<td>Breakpoint</td>
<td>D11</td>
</tr>
<tr>
<td>10-13</td>
<td>4</td>
<td>Overflow</td>
<td>D11</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>D11</td>
</tr>
<tr>
<td>1D-1F</td>
<td>7</td>
<td>Reserved</td>
<td>D11</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>D11</td>
</tr>
<tr>
<td>2C-2F</td>
<td>B</td>
<td>Communications</td>
<td>D11</td>
</tr>
<tr>
<td>30-33</td>
<td>C</td>
<td>Communications</td>
<td>D11</td>
</tr>
<tr>
<td>34-37</td>
<td>D</td>
<td>Disk</td>
<td>D11</td>
</tr>
<tr>
<td>38-3B</td>
<td>E</td>
<td>Diskette</td>
<td>DISK_INT</td>
</tr>
<tr>
<td>3C-3F</td>
<td>F</td>
<td>Printer</td>
<td>D11</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_DETERMINE</td>
</tr>
<tr>
<td>4C-4F</td>
<td>13</td>
<td>Diskette/Disk</td>
<td>DISKETEJO</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_STRAP</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 Initialization</td>
<td>VIDEO_PARMS</td>
</tr>
<tr>
<td>78-7B</td>
<td>1E</td>
<td>Diskette Parameters</td>
<td>DISK_BASE</td>
</tr>
<tr>
<td>7C-7F</td>
<td>1F</td>
<td>Video Graphics Chars</td>
<td>0</td>
</tr>
</tbody>
</table>

**8088 Software Interrupt Listing**
Vectors with Special Meanings

Interrupt Hex 1B – Keyboard Break Address

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

Control may be retained by this routine, with the following problems. The Break may have occurred during interrupt processing, so that one or more End of Interrupt commands must be sent to the 8259 controller. Also, all I/O devices should be reset in case an operation was underway at that time.

Interrupt Hex 1C – Timer Tick

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

Interrupt Hex 1D – Video Parameters

This vector points to a data region containing the parameters required for the initialization of the 6845 on the video card. Note that there are four separate tables, and all four must be reproduced if all modes of operation are to be supported. The power-on routines initialize this vector to point to the parameters contained in the ROM video routines.
Interrupt Hex 1E – Diskette Parameters

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

Interrupt Hex 1F – Graphics Character Extensions

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

Interrupt Hex 40 – Reserved

When an IBM Fixed Disk Drive Adapter is installed, the BIOS routines use interrupt hex 40 to revector the diskette pointer.

Interrupt Hex 41 – Fixed Disk Parameters

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

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

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

<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>100-17F</td>
<td>40-5F</td>
<td>Reserved</td>
</tr>
<tr>
<td>180-19F</td>
<td>60-67</td>
<td>Reserved for User Software Interrupts</td>
</tr>
<tr>
<td>1A0-1FF</td>
<td>68-7F</td>
<td>Not Used</td>
</tr>
<tr>
<td>200-217</td>
<td>80-85</td>
<td>Reserved by 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>Not Used</td>
</tr>
</tbody>
</table>

BASIC and DOS Reserved Interrupts
<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>Reserved</td>
<td>Reserved as Intra-Application Communication Area for any application</td>
</tr>
<tr>
<td>4F0-4FF</td>
<td>Reserved</td>
<td>DOS</td>
</tr>
<tr>
<td>500-5FF</td>
<td>Print Screen Status Flag Store</td>
<td></td>
</tr>
<tr>
<td>500</td>
<td>DOS</td>
<td>Print Screen Status Flag Store</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

If you do DEF SEG (Default workspace segment):

<table>
<thead>
<tr>
<th>Description</th>
<th>Offset (Hex Value)</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 (end of program text 1-1)</td>
<td>358</td>
<td>2</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 1 to 3. Do not set to 0. (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

2-8 ROM BIOS
Starting Address in Hex

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000</td>
<td>BIOS Interrupt Vectors</td>
</tr>
<tr>
<td>00080</td>
<td>Available 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>C8000</td>
<td>Disk Adapter</td>
</tr>
<tr>
<td>F0000</td>
<td>Read Only Memory</td>
</tr>
<tr>
<td>FE000</td>
<td>BIOS Program Area</td>
</tr>
</tbody>
</table>

**BIOS Memory Map**

**BIOS Programming Hints**

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 disk or diskette code, you should reset the drive adapter and retry the operation. A specified number of retries should be required on diskette reads to ensure the problem is not due to motor start-up.

When altering I/O port bit values, the programmer should change only those bits 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 applications.
Adapter Cards with System-Accessible ROM Modules

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

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

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

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

2-10   ROM BIOS
Keyboard Encoding and Usage

Encoding

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

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

Character Codes

The following character codes are passed through the BIOS keyboard routine to the system or application program. A "—1" means the combination is suppressed in the keyboard routine. The codes are returned in AL. See Appendix C for the exact codes. Also, see "Keyboard Scan Code Diagram" in Section 1.

<table>
<thead>
<tr>
<th>Key Number</th>
<th>Base Case</th>
<th>Upper Case</th>
<th>Ctrl</th>
<th>Alt</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Esc</td>
<td>Esc</td>
<td>Esc</td>
<td>-1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>!</td>
<td></td>
<td>Note 1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>@</td>
<td></td>
<td>Note 1</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>#</td>
<td></td>
<td>Note 1</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>$</td>
<td></td>
<td>Note 1</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>%</td>
<td></td>
<td>Note 1</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>^</td>
<td></td>
<td>RS(030)</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>&amp;</td>
<td></td>
<td>Note 1</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>*</td>
<td></td>
<td>Note 1</td>
</tr>
<tr>
<td>10</td>
<td>9</td>
<td>(</td>
<td></td>
<td>Note 1</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>)</td>
<td></td>
<td>Note 1</td>
</tr>
<tr>
<td>12</td>
<td>-</td>
<td>—</td>
<td></td>
<td>US(031)</td>
</tr>
<tr>
<td>13</td>
<td>=</td>
<td>+</td>
<td></td>
<td>Note 1</td>
</tr>
<tr>
<td>14</td>
<td>Backspace (008)</td>
<td>Backspace (008)</td>
<td>Del (127)</td>
<td>-1</td>
</tr>
<tr>
<td>15</td>
<td>(Note 1)</td>
<td>(Note 1)</td>
<td></td>
<td>-1</td>
</tr>
<tr>
<td>16</td>
<td>q</td>
<td>Q</td>
<td>DC1 (017)</td>
<td>Note 1</td>
</tr>
<tr>
<td>17</td>
<td>w</td>
<td>W</td>
<td>ETB (023)</td>
<td>Note 1</td>
</tr>
</tbody>
</table>

Character Codes (Part 1 of 3)
<table>
<thead>
<tr>
<th>Key Number</th>
<th>Base Case</th>
<th>Upper Case</th>
<th>Ctrl</th>
<th>Alt</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>e</td>
<td>E</td>
<td>ENQ (005)</td>
<td>Note 1</td>
</tr>
<tr>
<td>19</td>
<td>r</td>
<td>R</td>
<td>DC2 (018)</td>
<td>Note 1</td>
</tr>
<tr>
<td>20</td>
<td>t</td>
<td>T</td>
<td>DC4 (020)</td>
<td>Note 1</td>
</tr>
<tr>
<td>21</td>
<td>y</td>
<td>Y</td>
<td>EM (025)</td>
<td>Note 1</td>
</tr>
<tr>
<td>22</td>
<td>u</td>
<td>U</td>
<td>NAK (021)</td>
<td>Note 1</td>
</tr>
<tr>
<td>23</td>
<td>i</td>
<td>I</td>
<td>HT (009)</td>
<td>Note 1</td>
</tr>
<tr>
<td>24</td>
<td>o</td>
<td>O</td>
<td>SI (015)</td>
<td>Note 1</td>
</tr>
<tr>
<td>25</td>
<td>p</td>
<td>P</td>
<td>DLE (016)</td>
<td>Note 1</td>
</tr>
<tr>
<td>26</td>
<td>[</td>
<td>\</td>
<td>Esc (027)</td>
<td>-1</td>
</tr>
<tr>
<td>27</td>
<td>]</td>
<td>]</td>
<td>GS (029)</td>
<td>-1</td>
</tr>
<tr>
<td>28</td>
<td>CR</td>
<td>CR</td>
<td>LF (010)</td>
<td>-1</td>
</tr>
<tr>
<td>29 Ctrl</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>30</td>
<td>a</td>
<td>A</td>
<td>SOH (001)</td>
<td>Note 1</td>
</tr>
<tr>
<td>31</td>
<td>s</td>
<td>S</td>
<td>DC3 (019)</td>
<td>Note 1</td>
</tr>
<tr>
<td>32</td>
<td>d</td>
<td>D</td>
<td>EOT (004)</td>
<td>Note 1</td>
</tr>
<tr>
<td>33</td>
<td>f</td>
<td>F</td>
<td>ACK (006)</td>
<td>Note 1</td>
</tr>
<tr>
<td>34</td>
<td>g</td>
<td>G</td>
<td>BEL (007)</td>
<td>Note 1</td>
</tr>
<tr>
<td>35</td>
<td>h</td>
<td>H</td>
<td>BS (008)</td>
<td>Note 1</td>
</tr>
<tr>
<td>36</td>
<td>j</td>
<td>J</td>
<td>LF (010)</td>
<td>Note 1</td>
</tr>
<tr>
<td>37</td>
<td>k</td>
<td>K</td>
<td>VT (011)</td>
<td>Note 1</td>
</tr>
<tr>
<td>38</td>
<td>l</td>
<td>L</td>
<td>FF (012)</td>
<td>Note 1</td>
</tr>
<tr>
<td>39</td>
<td>;</td>
<td>;</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>40</td>
<td>'</td>
<td>'</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>41</td>
<td>~</td>
<td>~</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>42 Shift</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>43</td>
<td>\</td>
<td>\</td>
<td>FS (028)</td>
<td>-1</td>
</tr>
<tr>
<td>44</td>
<td>z</td>
<td>Z</td>
<td>SUB (026)</td>
<td>Note 1</td>
</tr>
<tr>
<td>45</td>
<td>x</td>
<td>X</td>
<td>CAN (024)</td>
<td>Note 1</td>
</tr>
<tr>
<td>46</td>
<td>c</td>
<td>C</td>
<td>ETX (003)</td>
<td>Note 1</td>
</tr>
<tr>
<td>47</td>
<td>v</td>
<td>V</td>
<td>SYN (022)</td>
<td>Note 1</td>
</tr>
<tr>
<td>48</td>
<td>b</td>
<td>B</td>
<td>STX (002)</td>
<td>Note 1</td>
</tr>
<tr>
<td>49</td>
<td>n</td>
<td>N</td>
<td>SO (014)</td>
<td>Note 1</td>
</tr>
<tr>
<td>50</td>
<td>m</td>
<td>M</td>
<td>CR (013)</td>
<td>Note 1</td>
</tr>
<tr>
<td>51</td>
<td>.</td>
<td>&lt;</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>52</td>
<td>,</td>
<td>&gt;</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>53</td>
<td>/</td>
<td>?</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>54 Shift</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>55</td>
<td>*</td>
<td>(Note 2)</td>
<td>(Note 1)</td>
<td>-1</td>
</tr>
<tr>
<td>56 Alt</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>57</td>
<td>SP</td>
<td>SP</td>
<td>SP</td>
<td>SP</td>
</tr>
<tr>
<td>58</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>Caps Lock</td>
<td>Nul (Note 1)</td>
<td>Nul (Note 1)</td>
<td>Nul (Note 1)</td>
<td>Nul (Note 1)</td>
</tr>
<tr>
<td>Nul (Note 1)</td>
<td>Nul (Note 1)</td>
<td>Nul (Note 1)</td>
<td>Nul (Note 1)</td>
<td>Nul (Note 1)</td>
</tr>
<tr>
<td>Nul (Note 1)</td>
<td>Nul (Note 1)</td>
<td>Nul (Note 1)</td>
<td>Nul (Note 1)</td>
<td>Nul (Note 1)</td>
</tr>
<tr>
<td>Nul (Note 1)</td>
<td>Nul (Note 1)</td>
<td>Nul (Note 1)</td>
<td>Nul (Note 1)</td>
<td>Nul (Note 1)</td>
</tr>
<tr>
<td>Nul (Note 1)</td>
<td>Nul (Note 1)</td>
<td>Nul (Note 1)</td>
<td>Nul (Note 1)</td>
<td>Nul (Note 1)</td>
</tr>
</tbody>
</table>

Character Codes (Part 2 of 3)
### Character Codes (Part 3 of 3)

Keys 71 to 83 have meaning only in base case, in Num Lock (or shifted) states, or in Ctrl state. It should be noted that the shift key temporarily reverses the current Num Lock state.

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

**Notes:**
1. Refer to “Extended Codes” in this section.
2. Refer to “Special Handling” in this section.
Extended Codes

Extended Functions

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

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

Keyboard Extended Functions

2-14 Keyboard Encoding
Shift States

Most shift states are handled within the keyboard routine, transparent to the system or application program. In any case, the current set of active shift states are 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 2-13, 15-27, 30-41, 43-53, 55, and 59-68 to upper case (base case if in Caps Lock state). Also, the Shift key temporarily reverses the Num Lock or non-Num-Lock state of keys 71-73, 75, 77, and 79-83.

Ctrl

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

Alt

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

The Alt key has another use. This key allows the user to enter any character code from 0 to 255 into the system from the keyboard. The user holds down the Alt key and types the decimal value of the characters desired using the numeric keypad (keys 71-73, 75-77, and 79-82). The Alt key is then released. If more than three digits are typed, a modulo-256 result is created. These three digits are interpreted as a character code and are transmitted through the keyboard routine to the system or application program. Alt is handled internal to the keyboard routine.
Caps Lock

This key shifts keys 16-25, 30-38, and 44-50 to upper case. A second depression of the Caps Lock key reverses the action. Caps Lock is handled internal to the keyboard routine.

Scroll Lock

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

Shift Key Priorities and Combinations

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

Special Handling

System Reset

The combination of the Alt, Ctrl, and Del keys will result in the keyboard routine initiating the equivalent of a “system reset” or “reboot.” System reset is handled internal to the keyboard.

Break

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

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

Print Screen

The combination of the Shift and PrtSc (key 55) keys will result in an interrupt invoking the print screen routine. This routine works in the alphanumeric or graphics mode, with unrecognizable characters printing as blanks.

Other Characteristics

The keyboard routine does its own buffering. The keyboard buffer is large enough to support a fast typist. However, if a key is entered when the buffer is full, the key will be ignored and the “bell” will be sounded.

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

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Key(s)</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Home Cursor</td>
<td>Home</td>
<td>Editors; word processors</td>
</tr>
<tr>
<td>Return to outermost menu</td>
<td>Home</td>
<td>Menu driven applications</td>
</tr>
<tr>
<td>Move cursor up</td>
<td>↑</td>
<td>Full screen editor, word processor</td>
</tr>
<tr>
<td>Page up, scroll backwards 25 lines and home</td>
<td>PgUp</td>
<td>Editors; word processors</td>
</tr>
<tr>
<td>Move cursor left</td>
<td>← Key 75</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>End</td>
<td>Editors; word processors</td>
</tr>
<tr>
<td>Move cursor down</td>
<td>↓</td>
<td>Full screen editor, word processor</td>
</tr>
<tr>
<td>Page down, scroll forward 25 lines and home</td>
<td>Pg Dn</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>
<tr>
<td>Destructive character at cursor</td>
<td>Del</td>
<td>Text, command entry</td>
</tr>
<tr>
<td>Tab forward</td>
<td>← Tab</td>
<td>Text entry</td>
</tr>
<tr>
<td>Tab reverse</td>
<td>→ Tab</td>
<td>Text entry</td>
</tr>
<tr>
<td>Clear screen and home</td>
<td>Ctrl Home</td>
<td>Command entry</td>
</tr>
<tr>
<td>Scroll up</td>
<td>↑</td>
<td>In scroll lock mode</td>
</tr>
<tr>
<td>Scroll down</td>
<td>↓</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</td>
<td>Ctrl 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>Ctrl PrtSc (Key 55)</td>
<td>Any time</td>
</tr>
<tr>
<td>Delete from cursor to EOS</td>
<td>Ctrl 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 screen</td>
</tr>
<tr>
<td>Window Left</td>
<td>Ctrl</td>
<td>When text is too wide to fit screen</td>
</tr>
<tr>
<td>Enter insert mode</td>
<td>Ins</td>
<td>Line editor</td>
</tr>
</tbody>
</table>

*Keyboard - Commonly Used Functions (Part 1 of 2)*

2-18  Keyboard Encoding
<table>
<thead>
<tr>
<th>Function</th>
<th>Key(s)</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<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</td>
<td>Stop list, stop program, and so on</td>
</tr>
<tr>
<td></td>
<td>Num Lock</td>
<td>Resumes on any key</td>
</tr>
<tr>
<td>Break interrupt</td>
<td>Ctrl 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 PgUp</td>
<td>Editors, word processors</td>
</tr>
<tr>
<td>Standard function keys</td>
<td>F1-F10</td>
<td>Primary function keys</td>
</tr>
<tr>
<td>Secondary function keys</td>
<td>Shift F1-F10</td>
<td>Extra function keys if 10 are not sufficient</td>
</tr>
<tr>
<td></td>
<td>Ctrl F1-F10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Alt F1-F10</td>
<td></td>
</tr>
<tr>
<td>Extra function keys</td>
<td>Alt Keys 2-13</td>
<td>Used when templates are put along top of keyboard</td>
</tr>
<tr>
<td></td>
<td>(1-9,0,.,:=)</td>
<td></td>
</tr>
<tr>
<td>Extra function keys</td>
<td>Alt A-Z</td>
<td>Used when function starts with same letter as one of the alpha keys</td>
</tr>
</tbody>
</table>

*Keyboard - Commonly Used Functions (Part 2 of 2)*
<table>
<thead>
<tr>
<th>Function</th>
<th>Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Carriage return</td>
<td>Ctrl</td>
</tr>
<tr>
<td>Line feed</td>
<td>Ctrl G</td>
</tr>
<tr>
<td>Bell</td>
<td>Home</td>
</tr>
<tr>
<td>Home</td>
<td></td>
</tr>
<tr>
<td>Cursor up</td>
<td></td>
</tr>
<tr>
<td>Cursor down</td>
<td></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 Home</td>
</tr>
<tr>
<td>Freeze output</td>
<td>Ctrl Num Lock</td>
</tr>
<tr>
<td>Tab advance</td>
<td></td>
</tr>
<tr>
<td>Stop execution (break)</td>
<td>Ctrl Break</td>
</tr>
<tr>
<td>Delete current line</td>
<td>Esc</td>
</tr>
<tr>
<td>Delete to end of line</td>
<td>Ctrl End</td>
</tr>
<tr>
<td>Position cursor to end of line</td>
<td>End</td>
</tr>
</tbody>
</table>

**DOS Special Functions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Suspend</td>
<td>Ctrl Num Lock</td>
</tr>
<tr>
<td>Echo to printer</td>
<td>Ctrl PrtSc</td>
</tr>
<tr>
<td>Stop echo to printer</td>
<td>Ctrl PrtSc</td>
</tr>
<tr>
<td>Exit current function (break)</td>
<td>Ctrl</td>
</tr>
<tr>
<td>Backspace</td>
<td>Break</td>
</tr>
<tr>
<td>Line feed</td>
<td>Ctrl</td>
</tr>
<tr>
<td>Cancel line</td>
<td>Esc</td>
</tr>
<tr>
<td>Copy character</td>
<td>F1 or</td>
</tr>
<tr>
<td>Copy until match</td>
<td>F2</td>
</tr>
<tr>
<td>Copy remaining</td>
<td>F3</td>
</tr>
<tr>
<td>Skip character</td>
<td>Del</td>
</tr>
<tr>
<td>Skip until match</td>
<td>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>F5</td>
</tr>
<tr>
<td>String separator in REPLACE</td>
<td>F6</td>
</tr>
<tr>
<td>End of file in keyboard input</td>
<td>F6</td>
</tr>
</tbody>
</table>

**BASIC Screen Editor Special Functions**

2-20  Keyboard Encoding
## APPENDIX A: ROM BIOS LISTINGS

<table>
<thead>
<tr>
<th>Line</th>
<th>Page</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>System ROM BIOS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Equates</td>
<td>A-2</td>
<td>12</td>
</tr>
<tr>
<td>8088 Interrupt Locations</td>
<td>A-2</td>
<td>35</td>
</tr>
<tr>
<td>Stack</td>
<td>A-2</td>
<td>67</td>
</tr>
<tr>
<td>Data Areas</td>
<td>A-2</td>
<td>76</td>
</tr>
<tr>
<td>Power-On Self-Test</td>
<td>A-5</td>
<td>239</td>
</tr>
<tr>
<td>Boot Strap Loader</td>
<td>A-20</td>
<td>1408</td>
</tr>
<tr>
<td>I/O Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Asynchronous Communications (RS-232C)</td>
<td>A-21</td>
<td>1461</td>
</tr>
<tr>
<td>Keyboard</td>
<td>A-24</td>
<td>1706</td>
</tr>
<tr>
<td>Diskette</td>
<td>A-34</td>
<td>2303</td>
</tr>
<tr>
<td>Printer</td>
<td>A-44</td>
<td>3078</td>
</tr>
<tr>
<td>Display</td>
<td>A-46</td>
<td>3203</td>
</tr>
<tr>
<td>System Configuration Analysis</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Memory Size Determination</td>
<td>A-71</td>
<td>5052</td>
</tr>
<tr>
<td>Equipment Determination</td>
<td>A-71</td>
<td>5083</td>
</tr>
<tr>
<td>Graphics Character Generator</td>
<td>A-77</td>
<td>5496</td>
</tr>
<tr>
<td>Time of Day</td>
<td>A-79</td>
<td>5630</td>
</tr>
<tr>
<td>Print Screen</td>
<td>A-81</td>
<td>5821</td>
</tr>
</tbody>
</table>

| Fixed Disk ROM BIOS | | |
| Fixed Disk I/O Interface | A-84 | 1 |
| Boot Strap Loader | A-89 | 399 |
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 THE CODE SEGMENT VIOLATE THE STRUCTURE AND DESIGN OF BIOS.

---

ABS0 SEGMENT AT 0

0000  40  STG_LOC0  LABEL BYTE
0008  40  ORG 24H
0014  40  M12_PTR  LABEL WORD
0014  40  ORG 54H
0020  46  IN0_PTR  LABEL WORD
0020  40  ORG 04H
0020  46  INT_ADDR  LABEL WORD
0020  47  INT_PTR  LABEL DWORD
0040  40  ORG 10H4
0040  49  VIDEO_INT  LABEL WORD
0074  50  ORG 10H4
0074  51  PARM_PTR  LABEL DWORD  ; POINTER TO VIDEO PARAMS
0060  52  ORG 10H4
0060  53  BASIC_PTR  LABEL WORD  ; ENTRY POINT FOR CASSETTE BASIC
0070  54  ORG 010H4  ; INTERRUPT 1EH
0070  55  DISK_POINTER  LABEL DWORD
0076  56  ORG 01FH4  ; LOCATION OF POINTER
0076  57  EXT_PTR  LABEL DWORD  ; POINTER TO EXTENSION
0040  58  ORG 60H
0040  59  DATA_AREA  LABEL BYTE  ; ABSOLUTE LOCATION OF DATA SEGMENT
0040  60  DATA_WORD  LABEL WORD
0050  61  ORG 050H
0050  62  MFG_TEST_PTR  LABEL FAR
0070  63  ORG 70H
0070  64  ROOT_LOCH  LABEL FAR

---

STACK SEGMENT AT 30H

0000 (128)

???

0100  TOS  LABEL WORD

---

STACK ENDS

66

; STACK -- USED DURING INITIALIZATION ONLY

70

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

TEXT SEGMENT

TITLE "BIOS FOR THE IBM PERSONAL COMPUTER XT"

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

0060  15  PORT_A  EQU 60H  ; 0255 PORT A ADDR
0061  16  PORT_B  EQU 61H  ; 0255 PORT B ADDR
0062  17  PORT_C  EQU 62H  ; 0255 PORT C ADDR
0063  18  CMD_PORT  EQU 63H
0064  20  INTA0  EQU 20H  ; 0259 PORT
0065  21  INTA01  EQU 21H  ; 0259 PORT
0066  22  EOI  EQU 20H
0067  22  TIMER  EQU 40H
0068  23  TIM_CTL  EQU 41H  ; 0253 TIMER CONTROL PORT ADDR
0069  24  TIMER  EQU 40H  ; 0253 TIMER/COUNTER 0 PORT ADDR
0070  25  THINT  EQU 01  ; TIMER 0 INTR RECVD MASK
0070  26  DMA00  EQU 00  ; DMA STATUS REG PORT ADDR
0070  27  DMA  EQU 00  ; DMA CH 0 ADDR, REG PORT ADDR
0076  28  MAX_PERIOD  EQU 50H
0076  29  MIN_PERIOD  EQU 41H
0076  30  KB_DIR  EQU 60H  ; KEYBOARD DATA IN PORT ADDR
0076  31  KB_DIRN  EQU 02  ; KEYBOARD INTR MASK
0076  32  KB_DATA  EQU 60H  ; KEYBOARD Scan Code PORT
0076  33  KB_CTL  EQU 61H  ; CONTROL BITS FOR KEYBOARD SENSE DATA

35

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

; ROM BIOS DATA AREAS

A-2 System BIOS
LOC OBJ  LINE  SOURCE

78  0000 14  RS232_BASE DW 4 DUP(?) ; ADDRESSES OF RS232 ADAPTERS

79  0000 14  PRINTER_BASE DW 4 DUP(?) ; ADDRESSES OF PRINTERS

----
0010 08  EQUIP_FLAG DW ? ; INSTALLED HARDWARE
0012 06  MFG_TST DB ? ; INITIALIZATION FLAG
0013 08  MEMORY_SIZE DW ? ; MEMORY SIZE IN K BYTES
0015 06  MFG_ERR_FLAG DB ? ; SCRATCHPAD FOR MANUFACTURING
0016 07  DB ? ; ERROR CODES

0017 93  KB_FLAG DB ?

0018 95  ;----- SHIFT FLAG EQUATES WITHIN KB_FLAG

0080 97  INS_STATE EQU 80H ; INSERT STATE IS ACTIVE
0080 96  CAPS_STATE EQU 40H ; CAPS LOCK STATE HAS BEEN TOGGLED
0080 99  NUM_STATE EQU 20H ; NUM LOCK STATE HAS BEEN TOGGLED
0080 100  SCROLL_STATE DB 10H ; SCROLL LOCK STATE HAS BEEN TOGGLED
0080 101  ALT_SHIFT EQU 00H ; ALTERNATE SHIFT KEY DEPRESSED
0080 102  CTL_SHIFT EQU 04H ; CONTROL SHIFT KEY DEPRESSED
0080 103  LEFT_SHIFT EQU 02H ; LEFT SHIFT KEY DEPRESSED
0080 104  RIGHT_SHIFT EQU 01H ; RIGHT SHIFT KEY DEPRESSED

0080 105  KB_FLAG_1 DB ? ; SECOND BYTE OF KEYBOARD STATUS

0080 106  INS_SHIFT EQU 80H ; INSERT KEY IS DEPRESSED
0080 109  CAPS_SHIFT EQU 40H ; CAPS LOCK KEY IS DEPRESSED
0080 110  NUM_SHIFT EQU 20H ; NUM LOCK KEY IS DEPRESSED
0080 111  SCROLL_SHIFT EQU 10H ; SCROLL LOCK KEY IS DEPRESSED
0080 112  HOLD_STATE EQU 00H ; SUSPEND KEY HAS BEEN TOGGLED

0080 113  ALT_INPUT DB ? ; STORAGE FOR ALTERNATE KEYPAD ENTRY
0080 115  BUFFER_HEAD DW ? ; POINTER TO HEAD OF KEYBOARD BUFFER
0080 116  BUFFER_TAIL DW ? ; POINTER TO TAIL OF KEYBOARD BUFFER
0080 117  KB_BUFFER DW 16 DUP(?) ; ROOM FOR IS ENTRIES

009E 118  KB_BUFFER_END LABEL WORD

0119 119  ;----- HEAD = TAIL INDICATES THAT THE BUFFER IS EMPTY

0045 121  HM_KEY EQU 6FH ; SCAN CODE FOR NUMBER LOCK
0046 123  SCROLL_KEY EQU 70H ; SCAN CODE FOR NUMBER LOCK
0048 121  ALRT_KEY EQU 56H ; ALTERNATE SHIFT KEY SCAN CODE
0010 125  CTL_KEY EQU 29H ; SCAN CODE FOR CONTROL KEY
003A 126  CAPS_KEY EQU 5BH ; SCAN CODE FOR SHIFT LOCK
002A 123  LEFT_KEY EQU 42H ; SCAN CODE FOR LEFT SHIFT
0026 126  RIGHT_KEY EQU 54H ; SCAN CODE FOR RIGHT SHIFT
0052 129  INS_KEY EQU 02H ; SCAN CODE FOR INSERT KEY
0053 130  DEL_KEY EQU 03H ; SCAN CODE FOR DELETE KEY

0132 131  ; DISKETTE DATA AREAS :
0133 132  ;--------------------------------------

003E 135  SEEK_STATUS DB ? ; DRIVE RECALIBRATION STATUS
003F 138  MOTOR_STATUS DB ? ; MOTOR STATUS

0080 139  INT_FLAG EQU 080H ; INTERRUPT OCCURRENCE FLAG
009F 140  MOTOR_STATUS DB ? ; MOTOR STATUS

0080 141  BIT 3-0 = DRIVE 3-6 NEEDS RECAL
0080 142  BIT 7 = CURRENT OPERATION IS A WRITE,
0080 143  BEFORE NEXT SEEK IF BIT IS = 0

009F 144  BIT 3-0 = DRIVE 3-6 IS CURRENTLY
009F 145  READING
009F 146  BIT 7 = CURRENT OPERATION IS A WRITE,
009F 147  REQUIRES DELAY

00A0 148  MOTOR_COUNT DB ? ; TIME OUT COUNTER FOR DRIVE TURN OFF
00A0 149  MOTOR_COUNT DB ? ; TIME OUT COUNTER FOR DRIVE TURN OFF

Appendix A

System BIOS  A-3
XXDATA SEGMENT AT SOH
  STATUS_BYTE
  DB 221
XXDATA ENDS

VIDEO DISPLAY BUFFER

ROM RESIDENT CODE

CODE SEGMENT AT 0F000H
  DB 57344 DUP(?)   ; FILL LOWEST 56K

COPYRIGHT NOTICE

INITIAL RELIABILITY TESTS -- PHASE 1

DATA DEFINITIONS

LOAD A BLOCK OF TEST CODE THROUGH THE KEYBOARD PORT

FIRST, GET THE COUNT

System BIOS  A-5
asm

E045 6AC7 267 MOV AL,0M
E047 EE 268 OUT DX,AL
E044 4A 269 DEC DX ; POINT DX AT ADDR. 60 (KB DATA)
E049 TSTI: 290
E049 E420 291 IN AL,INTA00 ; GET ERR REG
E048 22C6 292 AND AL,AH ; KB REQUEST PENDING?
E040 74F4 293 JZ TST1 ; LOOP TILL DATA PRESENT
E04F 74F6 294 IN AL,DX ; GET DATA
E050 AA 295 STOSB ; STORE IT
E051 4E 296 INC DX ; POINT DX BACK AT PORT B (61)
E052 E2EE 297 LOOP TST ; LOOP TILL ALL BYTES READ
E054 EA00355000 299 JMP NFG_TEST_RTN ; FAR JUMP TO CODE THAT WAS JUST LOADED

System BIOS

0088 B0FFFF 341 MOV AX,0FFFFH ; SETUP ONE'S PATTERN IN AX
0088 B0FF F 342 STC
0088 BE0C 343 CB: MOV DS,AX ; WRITE PATTERN TO ALL REGS
0088 BE0E 344 MOV BX,DS
0088 BE0F 345 MOV ES,BX
0088 BE10 346 MOV CX,ES
0088 BE11 347 MOV SS,CX
0088 BE12 348 MOV DX,SS
0088 BE13 349 MOV SP,DX
0088 BE14 34A MOV BP,SP
0088 BE15 34B MOV SI,BP
0088 BE16 34C MOV DI,SI
0088 BE17 353 JNC C9 ; TSTIA
0088 BE18 354 XOR AX,AX ; PATTERN MAKE IT THRU ALL REGS
0088 BE19 355 JNZ ERR01 ; NO - GO TO ERR ROUTINE
0088 BE1A 356 CLC
0088 BE1B 357 JMP C0 ; TSTIA
0088 BE1C 358 C9: ; TSTIA
0088 BE1D 359 OR AX,AX ; ZERO PATTERN MAKE IT THRU?
0088 BE1E 74F0 360 JE C10 ; YES - GO TO NEXT TEST
0088 BE1F 74F1 361 ERR01: HLT ; HALT SYSTEM

errno: 362 ---------

363 ; ROS CHECKSUM TEST I ;
; DESCRIPTION
; A CHECKSUM IS DONE FOR THE 8K
; A ROM MODULE CONTAINING PCD AND
; BIOS.

;--------------------------------------------------------
EOAE 369 C10:
370 i ZERO IN AL ALREADY
EOAE E6A0 371 OUT OAH,AL  ; DISABLE MMU INTERRUPTS
EOE0 E683 372 OUT 08H,AL  ; INITIALIZE DMA PAGE REG.
EOE2 BAD003 373 MOV DX,300H
EOE5 EE 374 OUT DX,AL  ; DISABLE COLOR VIDEO
EOE6 FECC 375 IN AL
EOE8 E208 376 MOV DL,080H
EOEA EE 377 OUT DX,AL  ; DISABLE B/W VIDEO,EN HIGH RES
EOEB B089 378 MOV AL,09H  ; SET 6255 FOR B,A=OUT, C=IN
EOEC E663 379 OUT CMODE_PORT,A
EOEF BD05 380 MOV AL,16100101B
381 i ENABLE PARITY CHECKERS AND
382 i KEYBOARD INPUTS,ENABLE HIGH
383 i BANK OF SWITCHES->PORT C(0-3)
EOC3 B001 385 MOV AL,01H
EOC5 E640 386 OUT PORT_A,AL  ; <><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><<
LOC OBJ LINE SOURCE

E10B E1F2 441 LOOP C14 ; TIMER_LOOP
E10D F4 442 HLT ; HALT SYSTEM
443
444 ;------- INITIALIZE TIMER 1 TO REFRESH MEMORY
445
E10E 6003 446 C15: MOV AL,03H ; <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>
E110 E660 447 OUT PORT_A,AL ; <<<<<<<<<<CHECKPOINT 3>>>>>
448 ; WRAP_DMA_REG
E112 E600 449 OUT DMA+0DH,AL ; SEND MASTER CLEAR TO DMA
450
451 ;------- WRAP DNA CHANNELS ADDRESS AND COUNT Registers
452
E114 B0F0 453 MOV AL,OFFH ; WRITE PATTERN FF TO ALL REGS
E116 8A08 454 C16: MOV BL,AL ; SAVE PATTERN FOR COMPARE
E118 8A86 455 MOV BH,AL ;
E11A B90080 456 MOV CH,AL ; SETUP LOOP CNT
E11D BA0000 457 MOV DX,DATA ; SETUP I/O PORT ADDR OF REG
E120 EE 458 C17: OUT DX,AL ; WRITE PATTERN TO REG, LSB
E121 50 459 PUSH AX ; SATISFY 8237 I/O TIMINGS
E122 EE 460 OUT DX,AL ; MSB OF 16 BIT REG
E123 B001 461 MOV AL,01H ; AL TO ANOTHER PAT BEFORE RD
E128 EC 462 IN AL,DX ; READ 16-BIT DMA CH REG, LSB
E126 8AEO 463 MOV AH,AL ; SAVE LSB OF 16-BIT REG
E128 EC 464 IN AL,DX ; READ MSB OF DMA CH REG
E129 3B08 465 CMP BX,AX ; PATTERN READ AS WRITTEN?
E12B 7401 466 JE C1B ; YES - CHECK NEXT REG
E12D F4 467 HLT ; NO - HALT THE SYSTEM
E12E 468 C18: ; NXT_DMA_CH
E132 42 469 INC DX ; SET I/O PORT TO NEXT CH REG
E132 E2EF 470 LOOP C17 ; WRITE PATTERN TO NEXT REG
E131 FEC0 471 INC AL ; SET PATTERN TO 0
E133 74C1 472 JZ C16 ; WRITE TO CHANNEL REGS
473
474 ;------- INITIALIZE AND START DMA FOR MEMORY REFRESH.
475
E135 8E0B 476 MOV DS,BX ; SET UP ABSO INTO DS AND ES
E137 8EC3 477 MOV ES,BX ;
E139 B0F0 478 ASSUME DS:ABS0,ES:ABS0
E13B 6E01 479 MOV AL,OFFH ; SET CNT OF 64K FOR REFRESH
E13D 50 480 OUT DMA+1,AL ;
E13E E601 481 PUSH AX ;
E140 B050 482 OUT DMA+1,AL ;
E142 B050 483 MOV AL,05H ; SET DMA MODE, CH 0, RO, ADJOINT
E142 E60B 484 OUT DMA+0BH,AL ; WRITE DMA MODE REG
E144 B000 485 MOV AL,0 ; ENABLE DMA CONTROLLER
E146 8A08 486 MOV CH,AL ; SET COUNT HIGH=00
E148 E608 487 OUT DMA+0AL,AL ; SETUP DMA COMMAND REG
E14A 50 488 PUSH AX ;
E14B E60A 489 OUT DMA+10,AL ; ENABLE DMA CH 0
E14B B012 490 MOV AL,10 ; START TIMER 1
E14F E641 491 OUT TIMER+1,AL ;
E151 B041 492 MOV AL,61H ; SET MODE FOR CHANNEL 1
E153 E60B 493 OUT DMA+0BH,AL ;
E155 50 494 PUSH AX ;
E156 E60B 495 MOV AL,08H ; GET DMA STATUS
E158 2410 496 AND AL,00000000B ; IS TIMER REQUEST THERE?
E15A 7401 497 JZ C18C ; (IT SHOULDN'T BE)
E15C F4 498 HLT ; HALT SYS.(NOT TIMER 1 OUTPUT)
E15D B042 499 MOV AL,42H ; SET MODE FOR CHANNEL 2
E15F E60B 500 OUT DMA+0DH,AL ;
E161 B043 501 MOV AL,03H ; SET MODE FOR CHANNEL 3
E163 E60B 502 OUT DMA+0EH,AL ;

480 ;-------- BASE 16K READ/WRITE STORAGE TEST
481 ; DESCRIPTION
482 ; WRITE/READ/VERIFY DATA PATTERNS
483 ; AA,55,FF,01, AND 00 TO 1ST 32K OF
484 ; STORAGE, VERIFY STORAGE ADDRESSABILITY.
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512

E165 BA1302 513 MOV DX,0013H ; ENABLE I/O EXPANSION BOX
E168 B001 514 MOV AL,09H ;
E16A EE 515 OUT DX,AL ;
516
517 MOV BX,DATA_WORDOFFSET RESET_FLAG ; SAVE 'RESET_FLAG' IN BX
LOC OBJ | LINE | SOURCE
--- | --- | ---
E16F B90000H | 520 | MDV CX,2000H  SET FOR 16K WORDS
E172 81FB3412 | 519 | CMP BX,1234H  WARM START?
E176 7416 | 520 | JE CLR_STG
E17B BC108E | 521 | MDV SP,OFFSET C2
E17B 19F104 | 522 | JMP STGTS,CHT
E18E 7412 | 523 | C26: JE HOW_BIG  STORAGE OK, DETERMINE SIZE
E190 8A00H | 524 | MDV BL,AL  SAVE FAILING BIT PATTERN
E192 B004H | 525 | MDV AL,00H  <<<<CHECKPOINT 4<<<<
E194 E600 | 526 | OUT PORT_A,AL  BASE RAM FAILURE - HANG
E196 2C97 | 527 | SUB CX,CX  BASE RAM FAILURE - HANG
E180 E2FE | 528 | C24B: LOOP C24B  FLIPPING BETWEEN D4 AND
E19A 6008 | 529 | XCHG BL,AL  FAILING BIT PATTERN
E18C EBF6 | 530 | JMP C24A
E18E 2B00H | 532 | SUB AX,AX  MAKE AX=0000
E190 F3 | 533 | REP STOSW  STORE 8K WORDS OF 0000
E191 AB | 534 | HOW_BIG:
E192 091E7204 | 535 | MDV DATA WORD_OFFSET_RESET_FLAG1,BX  RESTORE RESET FLAG
E196 B0004H | 536 | MDV DX,0400H  SET POINTER TO JUST>16KB
E19F 80000 | 537 | MDV BX,16  BASIC COUNT OF 16K
E19C 180 | 538 | FILL_LOOP:
E19C 800C | 539 | MDV ES,DX  SET SEG, REG.
E19E 2BFF | 540 | SUB DI,DI
E1AB B055A8 | 541 | MDV AX,0A55H  TEST PATTERN
E1AA 808CH | 542 | MDV CX,CX  SAVE PATTERN
E1A5 266005 | 543 | MDV ES:DI,AX  SEND PATTERN TO MEM.
E1AB 8000 | 544 | MDV AL,0FH  PUT SOMETHING IN AL
E1AA 266005 | 545 | MDV AX:ES:DI  GET PATTERN
E1AD 35C1 | 546 | XOR AX,CX  COMPARE PATTERN
E1AF 7511 | 547 | JNZ HIGH_BIG_END  GO END IF NO COMPARE
E1B1 800208 | 548 | MDV CX,2000H  SET COUNT FOR 8K WORDS
E1B4 F3 | 549 | REP STOSW  FILL 8K WORDS
E1B5 AB | 54A | HOW_BIG:
E1B6 01C0044 | 550 | ADD AX,400H  POINT TO NEXT 16KB BLOCK
E1BA 83C10 | 551 | ADD BX,16  BUMP COUNT BY 16KB
E1BD 08EA0 | 552 | CMP DH,0A0H  TOP OF RAM AREA YET? (A0000)
E1C0 70A0 | 553 | JNZ FILL_LOOP
E1C2 504 | 554 | HOW_BIG_END:
E1C2 091E1304 | 555 | MDV DATA WORD_OFFSET_MEMORY_SIZE1,BX  SAVE MEMORY SIZE
E1C4 5B4 | 557 | SETUP STACK SEG AND SP
E1C6 085000 | 559 | MDV AX,STACK  GET STACK VALUE
E1CB 0C0001 | 560 | MDV SS,AX  SET THE STACK UP
E1CB 8C0001 | 561 | MDV SP,OFFSET TOS  STACK IS READY TO GO
E1CE 5D3 | 562 | SETUP STACK SEG AND SP
E1CE 5D4 | 563 | INITIALIZE THE 0259 INTERRUPT CONTROLLER CHEM :  BASE_RAM
E1CE 5D5 | 564 | INITIALIZE THE 0259 INTERRUPT CONTROLLER CHEM :
E1CE 5D6 | 565 | CSE: MDV AL,13H  ICIM - EDGE, SHEL, ICIM
E1ED E620 | 566 | OUT INTA00,AL  ICIM
E1ED E620 | 567 | MDV AL,B  SETUP ICIM - INT TYPE 8 (6-F)
E1ED E621 | 568 | OUT INTA01,AL
E1ED B009 | 569 | MDV AL,Y  SETUP ICIM - BUFFER,0006 MODE
E1ED E621 | 570 | OUT INTA01,AL
E1DA 80FF | 571 | MDV AL,OFFH  MASK ALL INTS. OFF
E1DC E621 | 572 | OUT INTA01,AL  (VIDEO ROUTINE ENABLES INTS.)
E1E1 574 | 573 | SETUP THE INTERRUPT VECTORS TO TEMP INTERRUPT
E1E1 575 | 576 | PUSH DS
E1EF B90000H | 577 | MDV CX,32  FILL ALL 32 INTERRUPTS
E1E2 2BFF | 578 | SUB DI,DI  FIRST INTERRUPT LOCATION
E1E4 06C7 | 579 | MDV ES,DI  SET ES=0000 ALSO
E1E6 8A3FF | 580 | DS: MDV AX,OFFSET D11  MOVE ADDR OF INTR PROC TO TBL
E1EE AB | 581 | STOSW
E1EE 06CB | 582 | MDV AX,CX  GET ADDR OF INTR SEG
E1EE AB | 583 | STOSW
E1ED 02F7 | 584 | LOOP DS  LVECTBO
E1ED 02F8 | 585 | Setup ISG INTERRUPT VECTORS
E1E7 58F | 586 | INITIALIZE THE BIOS SUBROUTINE CALL INTERRUPT VECTORS
E1EF BF4000 | 587 | MDV DI,OFFSET_VIDEO_INT  SETUP ADDR TO INTR AREA
E1F2 0E | 588 | PUSH CS
E1F3 1F | 589 | POP DS  SETUP ADDR OF VECTOR TABLE
E1F4 60CB | 590 | MDV AX,DS  SET AX=SEGMENT
E1F6 BD3FF90 | 592 | MDV SI,OFFSET VECTOR_TABLE+16  START WITH VIDEO ENTRY
LOC OBJ

EIFA B91000

EIFD A5

EIFE 47

EIFF 47

E00 E2FB

E02 IF

E03 IE

E04 E462

E06 260F

E08 E4ED

E0A 80AD

E0C E661

E0E 90

E0F E462

E11 E661

E13 D2C0

E15 24F0

E17 0AC4

E19 A4E4

E1B A31004

E1E B099

E20 E663

E22 E0518

E26 80F6A

E28 7418

E2A 0F656

E2D 7503

E2F 8EFD

E32 E036

E34 E661

E36 90

E37 90

E38 E460

E3A E4F

E3C 7504

E3E FE061004

E3F

E41

E42 A11004

E45 50

E46 B030

E48 A31004

E4B 2A4E

E4D 0109

E4F B020

E51 A31004

E54 2A4E

E56 C109

E58 58

E59 A31004

E5C D430

E5E 750A

E60 BF4000

E63 C705000

E67 E94000

E6A 4E

E6C 3C30

E6E 7508

E70 3C20

E72 7502

E74 4E03

593 MOV CX,16

594 DIA: MOV SW

595 INC DI

596 INC DI

597 LOOP DIA

598 ; DETERMINE CONFIGURATION AND MFG. MODE :

599 ;------------------

600 ;

601 POP DS

602 PUSH DS

603 IN AL,PORT_C

604 AND AL,0000111B

605 MOV AH,AL

606 MOV AL,10101010B

607 OUT PORT_B,AL

608 POP PORT_A

609 NOP

610 IN AL,PORT_C

611 MOV CL,4

612 ROL AL,CL ; ROTATE TO HIGH NIBBLE

613 AND AL,11100000B

614 OR AL,AH

615 SUB AH,AH

616 MOV DATA_WORD[OFFSET EQUIP_FLAG],AX ; SAVE SWITCH INFO

617 MOV AL,99H

618 OUT CMD_PORT,AL

619 CALL KBD_RESET ; SEE IF MFG. JUMPER IN

620 CMP BL,0AH ; KEYBOARD PRESENT?

621 JE 66

622 CMP BL,06AH ; LOAD MFG. TEST REQUEST?

623 JNE D3B

624 JMP MFG_BOOT ; GO TO BOOTSTRAP IF SO

625 MOV AL,3EH

626 OUT PORT_B,AL

627 NOP

628 MOV AL,Port_A

629 AND AL,0F8H

630 MOV JNZ 66

631 MOV 632 DATA_AREA[OFFSET MFG_TEST] ; SET MANUFACTURING TEST FLAG

633 ;-------------------

634 ; INITIALIZE AND START CRT CONTROLLER 16451

635 ; TEST VIDEO READ/WRITE STORAGE.

636 ; DESCRIPTION

637 ; RESET THE VIDEO ENABLE SIGNAL.

638 ; SELECT ALPHANUMERIC MODE... 40 * 25, B & W.

639 ; READ/WRITE DATA PATTERNS TO STD. CHECK STG

640 ; ADDRESSABILITY.

641 ; ERROR = 1 LONG AND 2 SHORT BEEPS

642 ;------------------

643 ; MOV AX,DATA_WORD[OFFSET EQUIP_FLAG] ; GET SENSE SWITCH INFO

644 ; MOV AX,DATA_WORD[OFFSET EQUIP_FLAG] ; SAVE IT

645 MOV AX;DATA_WORD[OFFSET EQUIP_FLAG] ; GET SENSE SWITCH INFO

646 PUSH AX

647 MOV AL,30H

648 MOV DATA_WORD[OFFSET EQUIP_FLAG],AX

649 MOV AL,0AH

650 MOV AH,0AH

651 MOV AL,10H

652 MOV AL,20H

653 MOV DATA_WORD[OFFSET EQUIP_FLAG],AX

654 MOV AH,0AH

655 MOV AH,0AH

656 MOV DATA_WORD[OFFSET EQUIP_FLAG],AX ; RESTORE IT

657 ; AND CONTINUE

658 AND AL,30H

659 MOV E7 ; VIDE0 SW SET TO 0?

660 MOV DI,OFFSET VIDEO_INT

661 MOV [DI,OFFSET DUMMY_RETURN] ; RETURN IF NO VIDEO CARD

662 JMP E10.1 ; BYPASS VIDEO TEST

663 E7: TEST VIDEO;

664 CMP AL,30H ; B/W CARD ATTACHED?

665 JE E0 ; YES - SET MFG. FOR B/W CARD

666 INC AH ; SET COLOR MODE FOR COLOR C/D

667 CMP AL,20H

670 80X25 MODE SELECTED?

672 JNE 66 ; NO - SET MFG. FOR 40X25

674 MOV AH,3

675 MOV AH,3 ; SET MFG. FOR 8X8

A-10 System BIOS
E274 0400 670 68: XCHG AH,AL ; SET_MODE:
E274 0400 671 PUSH AX ; SAVE VIDEO MODE ON STACK
E274 0400 672 SUB AH,AH ; INITIALIZE TO ALPHANUMERIC MD
E276 CD10 673 INT 10H ; CALL VIDEO_ID
E276 50 674 POP AX ; RESTORE VIDEO SENSE SMS IN AH
E276 50 675 PUSH AX ; RESAVE VALUE
E276 50 676 MOV BX,0BB000H ; BEG VIDEO RAM ADDR B/W CD
E276 50 677 MOV DX,3B08H ; MODE REG FOR B/W
E276 50 678 MOV CX,2048 ; RAM WORD CNT FOR B/W CD
E276 B001 678 MOV AL,1 ; SET MODE FOR BW CARD
E276 80FC30 680 MOV AH,50H ; B/W VIDEO CARD ATTACHED?
E276 7409 681 JE E9 ; YES - GO TEST VIDEO STG
E276 B788 682 MOV BH,0B8H ; BEG VIDEO RAM ADDR COLOR CD
E276 B003 683 MOV DX,5D08H ; MODE REG FOR COLOR CD
E276 0220 684 MOV CH,28H ; RAM WORD CNT FOR COLOR CD
E276 FEO 685 MOV AL,0 ; SET MODE TO 0 FOR COLOR CD
E276 EE 686 E9: TEST_VIDEO_STG:
E276 EE 687 OUT DX,AL ; DISABLE VIDEO FOR COLOR CD
E279 B1372043412 688 CMP DATA_MORDIOFFSET_RESET_FLAG1,1234H ; POD INIT BY KBD RESET?
E279 8EC3 689 MOV ES,BX ; POINT ES TO VIDEO RAM STG
E27A 7407 690 JE E10 ; YES - SKIP VIDEO RAM TEST
E27A 8E0 691 MOV DS,BX ; POINT DS TO VIDEO RAM STG
E27A E0C703 692 ASSUME DS:NOTHING,ES:NOTHING
E27A 7546 693 CALL SIGTEST.CHNT ; GO TEST VIDEO B/W STG
E28A 5AAA 695 |---------------------------------------------|
E28A 50 702 E10: SETUP_VIDEO_DATA_ON_SCREEN_FOR_VIDEO:
E28A 50 703 POP AX ; LINE TEST.
E28A 50 704 PUSH AX ; DESCRIPTION:
E28A 50 705 MOV AH,0 ; ENABLE VIDEO SIGNAL AND SET MODE.
E28A 50 706 INT 10H ; DISPLAY A HORIZONTAL BAR ON SCREEN.
E28B 002070 707 MOV AX,7020H ; MB BLANKS IN REVERSE VIDEO
E28B 002070 708
E28B 002070 709
E28B 002070 710
E28B 002070 711 JMP SHORT E10A
E28C 5ECC 712 ORG 0ECC3H
E28C 5E9915 713 JMP MS_INT
E28C 714
E28C 2BFF 715 E10A:
E28C 716 SUB DI,DI ; SETUP STARTING LOC
E28C B92860 717 MOV CX,60 ; NO. OF BLANKS TO DISPLAY
E28C 9F 718 REP STOSW ; WRITE VIDEO STORAGE
E28C AD 719 |---------------------------------------------|
E28C 50 720 CRT_INTERFACE_LINES_TEST:
E28C 50 721 | DESCRIPTION:
E28C 50 722 | SENS0 ON/OFF TRANSITION OF THE:
E28C 50 723 | VIDEO ENABLE AND HORIZONTAL:
E28C 50 724 | SYNC LINES:
E28C 50 725 |------------------------------|
E28C 56 726 POP AX ; GET VIDEO SENSE SMS INFO
E28C 56 727 PUSH AX ; SAVE IT
E28C 56 728 MOV AH,50H ; B/W CARD ATTACHED?
E28C B003 729 MOV DX,03B0AH ; SETUP ADDR OF BW STATUS PORT
E28C 7403 730 JE E11 ; YES - GO TEST LINES
E28C B003 731 MOV DX,03D0AH ; COLOR CARD IS ATTACHED
E28C B003 732 E11: VIDEO_RAM_ADDR:
E28C B003 733 MOV AH,0
E28C B003 734 MOV CX,0
E28C B003 735 SUB CX,CX
E28C B003 736 E13:
E28C B003 737 IN AL,BX ; READ CRT STATUS PORT
E28C B003 738 AND AL,AL ; CHECK VIDEO/HORIZ LINE
E28C 1504 739 JNZ E14 ; ITS OK - CHECK IF IT GOES OFF
E28C 2E9 740 LOOP E13 ; LOOP TILL ON OR TIMEOUT
E28C 772 741 JMP SHORT E17 ; GO DRAW ERROR MSG
E28C B2C9 742 E14: SUB CX,CX
E28C B2C9 743 E15: IN AL,BX ; READ CRT STATUS PORT
E28C B2C9 744 E15: IN AL,BX ; READ CRT STATUS PORT
E28C B2C9 745
A-12 System BIOS
```
E358 E04E16 083 CALL E_MSG
E35B FA 084 CLI
E35C F4 085 HLT

E35F 086 ; HALT THE SYSTEM

E360 C066150402 087 ; 8253 TIMER CHECKOUT
E365 088 ; DESCRIPTION
E369 089 ; VERIFY THAT THE SYSTEM TIMER (0) DOESN'T COUNT
E36E 090 ; TOO FAST OR TOO SLOW.

E370 091 ;-------------------------------------------

E362 B0F 092 MOV DATA_AREA[OFFSET MFR_ERR_FLAG1],82H
E365 093 ; <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>
E368 094 ; <<<<<<<<<<<<TIMERP CHECKPOINT (2)>>>

E36C B160 095 MOV AL,0FEH ; MASK ALL INTRS EXCEPT LVL 0
E36F E08 096 OUT INTO;AL ; WRITE THE 8259 IMR
E372 E01 097 MOV AL,00010000B ; SEL TIM 0, LSB, MODE 0, BINARY
E375 E43 098 OUT TIP_CNTL;AL ; WRITE TIMER CONTROL MODE REG
E378 B160 099 MOV CX,16H ; SET PGM LOOP CNT
E37C 0AC1 100 MOV AL,CL ; SET TIMER 0 CNT REG
E380 E40 101 OUT TIPR0;AL ; WRITE TIMER 0 CNT REG

E37D 043 D8: 102 TEST DATA_AREA[OFFSET INTR_FLAG1].DH
E37F F066680401 103 ; DID TIMER 0 INTERRUPT OCCUR?
E37E 045 ;-------------------------------------------

E37F 7504 104 JNZ D9 ; YES - CHECK TIMER OP FOR SLOW TIME
E37E 047 LOOP D8 ; WAIT FOR INTR FOR SPECIFIED TIME
E377 E008 105 JMP D6 ; TIMER 0 INTR DIDN'T OCCUR - ERR
E379 0E9 106 DB: ;---------------------------------------
E37C B10C 107 MOV CL,12 ; SET PGM LOOP CNT
E37E 0FF 108 MOV AL,OFFH ; WRITE TIMER 0 CNT REG
E380 E60 109 OUT TIPR0;AL ; WRITE TIMER 0 CNT REG
E383 E08 110 MOV AL,0FFH ; REENABLE TIMER 0 INTERRUPTS
E386 E43 111 OUT INTO;AL
E389 E40 112 OUT TIPR0;AL
E38C F066680401 113 TEST DATA_AREA[OFFSET INTR_FLAG1].DH ; DID TIMER 0 INTERRUPT OCCUR?
E38E 050 JNZ D6 ; YES - TIMER COUNTING TOO FAST, ERR

E390 E07 114 ;-------------------------------------------
E392 E0F 115 MOV AL,OFFH ; DISABLE ALL DEVICE INTERRUPTS
E395 E61 116 OUT INTO;AL
E398 B036 117 MOV AL,36H ; SEL TIM 0, LSB,MSB,MODE 3
E39B E43 118 OUT TIPR0;AL ; WRITE TIMER MODE REG
E39E B00 119 MOV AL,0 ;
E3A0 E60 120 OUT TIPR0;AL ; WRITE LSB TO TIMER 0 REG
E3A3 E64 121 OUT TIPR0;AL ; WRITE MSB TO TIMER 0 REG

E3A6 08F 122 ;-------------------------------------------

E3A7 750C 123 JNZ D6 ; YES - TIMER COUNTING TOO FAST, ERR

E3A9 E07 124 MOV AL,OFFH ; DISABLE ALL DEVICE INTERRUPTS
E3AB E61 125 OUT INTO;AL
E3AE B00 126 MOV AL,0 ;
E3B2 E60 127 OUT TIPR0;AL ; WRITE LSB TO TIMER 0 REG
E3B5 E64 128 OUT TIPR0;AL ; WRITE MSB TO TIMER 0 REG

E3B8 7515 129 JNE F6 ; NO - DISPLAY ERROR MSG

E3C1 089 ;-------------------------------------------

E3C3 E0F 130 MOV AL,3CH ; CLR KBD, SET CLK LINE HIGH
E3C6 E61 131 MOV AL,0AH ; ENABLE KBD,CLK IN NEXT BYTE
E3C8 E61 132 OUT PORT_B;AL
E3CA 089 ;-------------------------------------------

E3CC 089 FS: ; KBD_WAIT!
```
CONTINUE TESTING

IF

CHECK ADDRESS BUS

E307 BE4EC90

MOV SI,OFFSET F1

GET MSG ADDR

E30B E0CB15

CALL E_MSG

PRINT MSG ON SCREEN

; SETUP HARDWARE INT. VECTOR TABLE

E3D 911

F7:

PUSH DS

SETUP_INT_TABLE:

E3DF 28C0

SUB AX,AX

E3E1 03C0

MOV ES,AX

E3E3 096000

MOV CX,0B

E3E6 0E

PUSH CS

SETUP DS SEG REG

E3E7 1F

POP DS

E3E8 BF3FE90

MOV SI,OFFSET VECTOR_TABLE

E3E9 BF2000

MOV DI,OFFSET INT_PTR

E3E1 F7A:

MOV AX,0A5

E3E2 47

INC DI

SKIP OVER SEGMENT

E3E3 47

INC DI

E3E4 E7F8

LOOP F7A

E3E5 4F

POP DS

; ----- SET UP OTHER INTERRUPTS AS NECESSARY

E3E6 C7060005F8

MOV NMI_PTR,OFFSET NMI_INT ; NMI INTERRUPT

E3E7 C7061005FF

MOV INTS_PTR,OFFSET PRINT_SCREEN ; PRINT SCREEN

E401 C70620000F6

MOV BASIC_PTR+2,0F600H ; SEGMENT FOR CASSETTE BASIC

E42 3---- SETUP TIMER 0 TO BLINK LED IF MANUFACTURING TEST MODE

E44 403I124901

CMP DATA_AREA[OFFSET HFG_TST1,01H] ; HFG. TEST MODE?

E44C 750A

JNZ EXP_IO

E44E C7067003CFF

MOV WORD PTR(BCH,OFFSET BLINK_INT),SETUP TIMER INT TO BLINK LED

E454 B0FE

MOV AL,OFFH ; ENABLE TIMER INTERRUPTION

E456 E621

OUT INTAH

; ----- EXPANSION I/O BOX TEST

E46 3---- CHECK TO SEE IF EXPANSION BOX PRESENT - IF INSTALLED

E46 4 EXP:IO:

(MODE WAS ENABLED EARLIER)

E46A DA1002

MOV DX,0210H ; CONTROL PORT ADDRESS

E46B D05555

MOV AX,5555H ; SET DATA PATTERN

E46E EE

OUT DX,AL

E46F 001

MOV AL,01H ; MAKE AL DIFFERENT

E471 EC

IN AL,DX ; RECOVER DATA

E472 3AC9

CMP AL,AL ; REPLY?

E474 7544

JNE E19 ; NO RESPONSE, GO TO NEXT TEST

E476 F7D0

NOT AX ; MAKE DATA=AAAA

E478 EE

OUT DX,AL

E479 001

MOV AL,01H

E47B EC

IN AL,DX ; RECOVER DATA

E47C 3ACA

CMP AL,AL

E47E 753A

JNE E19

; ----- CHECK ADDRESS BUS

E484 EXP2:

MOV BX,0001H ; LOAD HI ADDR. REG ADDRESS

E48A E51000

MOV DX,0015H ; LOAD HI ADDR. REG ADDRESS

E48E 690000

MOV CX,0016 ; GO AROUND 16 BITS

E492 E6E007

MOV CS:[BX],AL ; WRITE ADDRESS F0000+BX

E493 90

NOP

E495 EC

IN AL,DX ; READ ADDR. HIGH

E496 3AC7

CMP AL,0H

E499 7521

JNE EXP.ERR ; GO ERROR IF MISCMPARE

E502 42

INC DX

DX=016H (ADDR. LOW REG)
LOC OBJ | LINE | SOURCE
--- | --- | ---
E463 EC | 977 | IN AL, DX
E464 3AC3 | 978 | CMP AL, DL
E466 751B | 979 | JNE EXP_ERR
E468 4A | 980 | DEC DX
E469 DIE3 | 981 | SHL BX, 1
E46B E2EC | 982 | LOOP EXP3
E472 9000 | 983 | ---- CHECK DATA BUS
E474 8001 | 987 | MOV AL, 01
E472 4A | 989 | EXP4:
E473 7506 | 992 | MOV AL, 01H
E475 EE | 993 | IN AL, DX
E478 EC | 995 | JNE SHORT EXP_ERR
E45O 5E0E | 996 | SHL AL, 1
E45F EF22 | 997 | LOOP EXP4
E461 EB07 | 998 | JMP SHORT E19
E463 E0FF90 | 1000 | MOV SI, OFFSET F3C
E467 E3F15 | 1001 | CALL E_MSG
E46A | 1010 | E19:
E46A E0C15 | 1011 | CALL DDS
E460 1E | 1012 | PUSH DS
E46E 3E0C | 1013 | E20:
E474 7503 | 1014 | CMP RESET_FLAG, 0;23H
E476 E99F | 1015 | JNE E20A
E476 5E80 | 1016 | JMP ROM_SCAN
E479 5E80 | 1017 | E20A:
E47B 0100 | 1018 | MOV AX, 16
E47C EB2B | 1019 | JMP SHORT PRTE_SIZ
E47E | 1020 | E20B:
E47E 081E1300 | 1021 | MOV BX, MEMORY_SIZE
E480 038B10 | 1022 | SUB BX, 16
E485 B104 | 1023 | MOV CL, 04H
E487 D3B6 | 1024 | SHR BX, CL
E489 8CB8 | 1025 | MOV CX, BX
E48B 8000 | 1026 | MOV BX, 0400H
E48D 0277 | 1027 | E21:
E48E 08E6 | 1028 | MOV DS, BX
E490 8EC3 | 1029 | MOV ES, BX
E492 81C30004 | 1030 | ADD BX, 0400H
E496 52 | 1031 | PUSH DX
E497 51 | 1032 | PUSH CX
E498 53 | 1033 | PUSH BX
E499 50 | 1034 | PUSH AX
E49A 090020 | 1035 | MOV CX, 2000H
E49B 060F | 1036 | CALL STG1ST_CNT
E4A0 754C | 1037 | JNE E21A
E4A2 50 | 1038 | POP AX
E4A3 051000 | 1039 | ADD AX, 16
E4A6 | 1040 | PRT_SZI:
E4A6 50 | 1041 | PUSH AX
E4A7 080A00 | 1042 | MOV BX, 10
E4A8 093000 | 1043 | MOV CX, 3
E4AD | 1044 | DECIMAL_LOOP:
E4AD 302 | 1045 | XOR DX, DX
E4AF FFF3 | 1046 | DIV BX
E4B1 00C3A0 | 1047 | OR DX, 01H
E4B4 52 | 1048 | PUSH DX
E4B5 E2F6 | 1049 | LOOP DECIMAL_LOOP
E4B7 093000 | 1050 | MOV CX, 3
E4BA | 1051 | PRT_DEC_LOOP:
E4B8 50 | 1052 | POP AX
E4BB E0E14 | 1053 | CALL PRT_HEX
E4BC | 1054 | ~--- CHECK DATA BUS
E4BD | 1055 | DD & TIMES
E4BE | 1056 | MAKE DX=214H (DATA BUS REG)
E4BF | 1057 | SAVE DATA BUS VALUE
E4C0 | 1058 | SEND VALUE TO REG
E4C1 | 1059 | RETRIEVE VALUE FROM REG
E4C2 | 1060 | = TO SAVED VALUE
E4C3 | 1061 | FORM NEW DATA PATTERN
E4C4 | 1062 | LOOP TILL BIT WALKS ACROSS AL
E4C5 | 1063 | GO ON TO NEXT TEST
E4C6 | 1064 | ASSUME DS: DATA
E4C7 | 1065 | ADDITIONAL READ/ WRITE STORAGE TEST
E4C8 | 1066 | DESCRIPTION
E4C9 | 1067 | WRITE/READ DATA PATTERNS TO ANY READ/ WRITE
E4CA | 1068 | STORAGE AFTER THE FIRST 32K. STORAGE
E4CB | 1069 | ADDRESSABILITY IS CHECKED.
E4CD | 1070 | ASSUME DS: DATA
E4CE | 1071 | ADDITIONAL READ/WRITE STORAGE TEST
E4CF | 1072 | DESCRIPTION
E4D0 | 1073 | WRITE/READ DATA PATTERNS TO ANY READ/WRITE
E4D1 | 1074 | STORAGE AFTER THE FIRST 32K. STORAGE
E4D2 | 1075 | ADDRESSABILITY IS CHECKED.
E4D3 | 1076 | ADDITIONAL READ/WRITE STORAGE TEST
E4D4 | 1077 | DESCRIPTION
E4D5 | 1078 | WRITE/READ DATA PATTERNS TO ANY READ/WRITE
E4D6 | 1079 | STORAGE AFTER THE FIRST 32K. STORAGE
E4D7 | 1080 | ADDRESSABILITY IS CHECKED.
E4D8 | 1081 | ADDITIONAL READ/WRITE STORAGE TEST
E4D9 | 1082 | DESCRIPTION
E4DA | 1083 | WRITE/READ DATA PATTERNS TO ANY READ/WRITE
E4DB | 1084 | STORAGE AFTER THE FIRST 32K. STORAGE
E4DC | 1085 | ADDRESSABILITY IS CHECKED.
E4DD | 1086 | ADDITIONAL READ/WRITE STORAGE TEST
E4DE | 1087 | DESCRIPTION
E4DF | 1088 | WRITE/READ DATA PATTERNS TO ANY READ/WRITE
E4E0 | 1089 | STORAGE AFTER THE FIRST 32K. STORAGE
E4E1 | 1090 | ADDRESSABILITY IS CHECKED.
E4E2 | 1091 | ADDITIONAL READ/WRITE STORAGE TEST
E4E3 | 1092 | DESCRIPTION
E4E4 | 1093 | WRITE/READ DATA PATTERNS TO ANY READ/WRITE
E4E5 | 1094 | STORAGE AFTER THE FIRST 32K. STORAGE
E4E6 | 1095 | ADDRESSABILITY IS CHECKED.
E4E7 | 1096 | ADDITIONAL READ/WRITE STORAGE TEST
E4E8 | 1097 | DESCRIPTION
E4E9 | 1098 | WRITE/READ DATA PATTERNS TO ANY READ/WRITE
E4EA | 1099 | STORAGE AFTER THE FIRST 32K. STORAGE
E4EB | 1100 | ADDRESSABILITY IS CHECKED.
A-16  System BIOS
<table>
<thead>
<tr>
<th>LOC OBJ</th>
<th>LINE</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ES3D</td>
<td>1132</td>
<td>MOV DS,DX ; SETUP STARTING ROS ADDR</td>
</tr>
<tr>
<td>ES3D</td>
<td>1134</td>
<td>CALL ROS_CHECKSUM ; CHECK ROS</td>
</tr>
<tr>
<td>ES44</td>
<td>1136</td>
<td>JE ES ; CONTINUE IF OK</td>
</tr>
<tr>
<td>ES44</td>
<td>1137</td>
<td>CALL ROM_ERR ; POST ERROR</td>
</tr>
<tr>
<td>ES49</td>
<td>1138</td>
<td>ES ; POINT TO NEXT OK MODULE</td>
</tr>
<tr>
<td>ES49</td>
<td>1139</td>
<td>ADD DX,0200H ; ANY MORE TO DO?</td>
</tr>
<tr>
<td>ES4F</td>
<td>1140</td>
<td>DEC AH ; YES - CONTINUE</td>
</tr>
<tr>
<td>ES11</td>
<td>1141</td>
<td>JNZ E4</td>
</tr>
<tr>
<td>ES51</td>
<td>1151</td>
<td>F9:</td>
</tr>
<tr>
<td>ES51</td>
<td>1152</td>
<td>POP DS</td>
</tr>
<tr>
<td>ES52</td>
<td>1153</td>
<td>MOV AL,BYTE PTR EQUIP_FLAG ; DISKETTE PRESENT?</td>
</tr>
<tr>
<td>ES55</td>
<td>1154</td>
<td>AND AL,01H ; NO - BYPASS DISKETTE TEST</td>
</tr>
<tr>
<td>ES57</td>
<td>1155</td>
<td>JZ F15</td>
</tr>
<tr>
<td>ES59</td>
<td>1156</td>
<td>F10: ; DISK_TEST:</td>
</tr>
<tr>
<td>ES59</td>
<td>1157</td>
<td>IN AL,INTAB1</td>
</tr>
<tr>
<td>ES60</td>
<td>1158</td>
<td>AND AL,08FH ; ENABLE DISKETTE INTERRUPTS</td>
</tr>
<tr>
<td>ES60</td>
<td>1159</td>
<td>OUT INTAB1,AL</td>
</tr>
<tr>
<td>ES5F</td>
<td>1160</td>
<td>MOV AN,0 ; RESET NEC FDC</td>
</tr>
<tr>
<td>ES61</td>
<td>1161</td>
<td>MOV DL,AN ; SET FOR DRIVE 0</td>
</tr>
<tr>
<td>ES63</td>
<td>1162</td>
<td>INT 13H</td>
</tr>
<tr>
<td>ES65</td>
<td>1163</td>
<td>TEST AN,OFFH ; STATUS OK?</td>
</tr>
<tr>
<td>ES66</td>
<td>1164</td>
<td>JNZ F13 ; NO - FDC FAILED</td>
</tr>
<tr>
<td>ES67</td>
<td>1165</td>
<td>I----- TURN DRIVE 0 MOTOR ON</td>
</tr>
<tr>
<td>ES68</td>
<td>1166</td>
<td>MOV DX,03FH ; GET ADDR OF FDC CARD</td>
</tr>
<tr>
<td>ES69</td>
<td>1167</td>
<td>MOV AL,11CH ; TURN MOTOR ON, EN DMA/INT</td>
</tr>
<tr>
<td>ES6F</td>
<td>1168</td>
<td>OUT DX,AL</td>
</tr>
<tr>
<td>ES70</td>
<td>1169</td>
<td>SUB CX,CX</td>
</tr>
<tr>
<td>ES72</td>
<td>1170</td>
<td>LOOP F11 ; MOTOR_WAIT:</td>
</tr>
<tr>
<td>ES72</td>
<td>1171</td>
<td>F11:</td>
</tr>
<tr>
<td>ES74</td>
<td>1172</td>
<td>LOOP F12 ; MOTOR_WAIT:</td>
</tr>
<tr>
<td>ES74</td>
<td>1173</td>
<td>F12:</td>
</tr>
<tr>
<td>ES74</td>
<td>1174</td>
<td>MOV DX,DX</td>
</tr>
<tr>
<td>ES78</td>
<td>1175</td>
<td>MOV CH,1 ; SELECT DRIVE 0</td>
</tr>
<tr>
<td>ES79</td>
<td>1176</td>
<td>MOV B51,0 ; SELECT TRACK 1</td>
</tr>
<tr>
<td>ES79</td>
<td>1177</td>
<td>MOV SEEK_STATUS,DL</td>
</tr>
<tr>
<td>ES7E</td>
<td>1178</td>
<td>CALL SEEK ; RECALIBRATE DISKETTE</td>
</tr>
<tr>
<td>ES81</td>
<td>1179</td>
<td>MOV CX,F13 ; GO TO ERR SUBROUTINE IF ERR</td>
</tr>
<tr>
<td>ES83</td>
<td>1180</td>
<td>MOV CH,34</td>
</tr>
<tr>
<td>ES85</td>
<td>1181</td>
<td>CALL SEEK ; SEEK TO TRACK 34</td>
</tr>
<tr>
<td>ES86</td>
<td>1182</td>
<td>JNC F14 ; OK, TURN MOTOR OFF</td>
</tr>
<tr>
<td>ES50</td>
<td>1183</td>
<td>F13:</td>
</tr>
<tr>
<td>ES50</td>
<td>1184</td>
<td>MOV SF,OFFSET F3 ; GET ADDR OF MSG</td>
</tr>
<tr>
<td>ES5E</td>
<td>1185</td>
<td>CALL E_MSG ; GO PRINT ERROR MSG</td>
</tr>
<tr>
<td>ES5E</td>
<td>1186</td>
<td>I----- TURN DRIVE 0 MOTOR OFF</td>
</tr>
<tr>
<td>ES5F</td>
<td>1187</td>
<td>MOV AL,0CH ; DRP_OFF:</td>
</tr>
<tr>
<td>ES59</td>
<td>1188</td>
<td>MOV AL,0CH ; TURN DRIVE 0 MOTOR OFF</td>
</tr>
<tr>
<td>ES6E</td>
<td>1189</td>
<td>OUT DX,AL ; FDC CTL ADDRESS</td>
</tr>
<tr>
<td>ES6F</td>
<td>1190</td>
<td>I----- SETUP PRINTER AND RS232 BASE ADDRESSES IF DEVICE ATTACHED</td>
</tr>
<tr>
<td>ES69</td>
<td>1191</td>
<td>MOV DF,OFFSET KB_BUFFER ; SETUP KEYBOARD PARAMETERS</td>
</tr>
<tr>
<td>ES69</td>
<td>1192</td>
<td>MOV BUFFER_HEAD,SI</td>
</tr>
<tr>
<td>ES43</td>
<td>1193</td>
<td>MOV BUFFER_TAIL,SI</td>
</tr>
<tr>
<td>ES57</td>
<td>1194</td>
<td>MOV BUFFER_START,SI</td>
</tr>
<tr>
<td>ES58</td>
<td>1195</td>
<td>MOV ST,32 ; DEFAULT BUFFER OF 32 BYTES</td>
</tr>
<tr>
<td>ES4F</td>
<td>1196</td>
<td>MOV BUFFER_END,SI</td>
</tr>
<tr>
<td>ES52</td>
<td>1197</td>
<td>MOV DF,OFFSET PRINT_TIMER_OUT ; DEFAULT PRINTER TIMEOUT</td>
</tr>
<tr>
<td>ES56</td>
<td>1198</td>
<td>MOV POP DS</td>
</tr>
<tr>
<td>ES06</td>
<td>1199</td>
<td>POP ES</td>
</tr>
</tbody>
</table>

**System BIOS**
A-17
LOC OBJ | LINE | SOURCE
---|---|---
E5B7 D014:14 | 1200 | MOV AX,1414H; DEFAULT=20
E5BA AB | 1209 | STOSW
E5BB AB | 1210 | STOSW
E5BC B010:10 | 1211 | MOV AX,010H; RS232 DEFAULT=01
E5DF AB | 1212 | STOSW
ESC6 AB | 1213 | STOSW
ESC1 8E43 | 1214 | IN AL,INTA01
ESC3 74FC | 1215 | AND AL,0FCH; ENABLE TIMER AND KB INTS
ESC5 E621 | 1216 | OUT INTA01,AL
ESC7 E8F000 | 1217 | CMP BP,0000H; CHECK FOR BP= NON-ZERO
E5CA 7419 | 1219 | JE F I5A_0; CONTINUE IF NO ERROR
E5CC E80200 | 1220 | MOV DX,2; 2 SHORT BEEPS (ERROR)
ESC F80614 | 1221 | CALL ERR_BEEP
ESC0 BE09E000 | 1222 | MOV SI,OFFSET F3D; LOAD ERROR MSG
ESC6 E0F113 | 1223 | CALL _P_MSG
ESC9 74D9 | 1224 | ERR_WAIT;
E509 B400 | 1225 | MOV AH,00
E50B CD16 | 1226 | INT 16H; WAIT FOR 'FF' KEY
E50D 8EFCB0 | 1227 | CMP AH,50H
E510 75F7 | 1228 | JMP ERR_WAIT;
ESC8 E80E90 | 1229 | JMP F I5A;
ESC9 75E5 | 1230 | F I5A_0;
ESC5 03E120001 | 1231 | CMP MFG_TST,1; MFG MODE
ESC6 75E6 | 1232 | JE F I5A; BYPASS BEEP
ESC5 E80A00 | 1233 | MOV DX,1; 1 SHORT BEEP (NO ERRORS)
ESC F80E15 | 1234 | CALL ERR_BEEP
ESC2 A01000 | 1235 | F I5A: MOV AL,BYTE EQU_P_FLAG; GET SWITCHES
ESC5 2401 | 1236 | AND AL,00000001B; 'LOOP POST' SWITCH ON
ESC7 7503 | 1237 | JNZ F I5B; CONTINUE WITH BRING-UP
ESC9 E9F5FA | 1238 | JMP START;
ESC5 2AE4 | 1239 | F I5B: SUB AH,AH
ESC F E04900 | 1240 | MOV AL,CRT_MODE
E601 CD10 | 1241 | INT 10H; CLEAR SCREEN
E603 8E4990 | 1242 | F I5C: MOV BP,OFFSET F4; PUT_SRC_TBL
E606 B0E000 | 1243 | MOV SI,0;
ESC A46 | 1245 | F I6: MOV AX,C5: [BP]; GET Printer Base ADDR
E640 E8B5600 | 1246 | MOV AL,AAH; WRITE DATA TO PORT A
E64E 8E4A | 1247 | MOV AX,OFFSET F4 & PTR_BASE;
E64E 8E1E | 1248 | OUT DX,AL
E64E 8E1F | 1249 | PUSH DS; BUS SETTLING
E64E 8E20 | 1250 | IN AL,DX; READ PORT A
E64E 8E21 | 1251 | POP DS
E64E 8E22 | 1252 | CMP AL,AAH; DATA PATTERN SAME
E64E 8E23 | 1253 | JNE F I7; NO - CHECK NEXT PRT CO
E64E 8E24 | 1254 | MOV Printer_BASE[SI],DX; YES - STORE PRT BASE ADDR
E64E 8E25 | 1255 | INC SI; INCREMENT TO NEXT WORD
E64E 8E26 | 1256 | INC SI
E64E 8E27 | 1257 | F I7:
E64E 8E28 | 1258 | INC BP; POINT TO NEXT BASE ADDR
E64E 8E29 | 1259 | INC BP
E64E 8E2A | 1260 | CMP BP,OFFSET F4E; ALL POSSIBLE ADDRS CHECKED?
E652 8E2E | 1261 | JNE F I6; PTR_BASE;
E658 E575 | 1262 | MOV BX,0; POINTER TO RS232 TABLE
E65B BAF0A0 | 1263 | MOV DX,3F8H; CHECK IF RS232 CO 1 ATTACH?
E65C 8E2B | 1264 | IN AL,DX; READ INTR BD REG
E65C 8E2C | 1265 | TEST AL,0F8H
E65D 8E2D | 1266 | JNZ F I8
E65E 8E2E | 1267 | MOV RS232_BASE[BX],3F8H; SETUP RS232 CO #1 ADDR
E65E 8E2F | 1268 | INC BX
E65E 8E30 | 1269 | INC BX
E65E 8E31 | 1270 | F I6:
E65E 8E32 | 1271 | MOV DX,3F8H; CHECK IF RS232 CO 2 ATTACH
E65F 8E33 | 1272 | IN AL,DX; READ INTERRUPT BD REG
E65F 8E34 | 1273 | TEST AL,0F8H
E660 8E35 | 1274 | JNZ F I9; BASE_END
E661 8E36 | 1275 | MOV RS232_BASE[BX],2F8H; SETUP RS232 CO #2
E662 8E37 | 1276 | INC BX
E662 8E38 | 1277 | INC BX
E662 8E39 | 1278 | INC BX
E662 8E3A | 1279 | ; --- SET UP EQUIP FLAG TO INDICATE NUMBER OF PRINTERS AND RS232 CARDS
E663 8E3B | 1280 |
E664 8E3C | 1281 | ; BASE_END;
E665 8E3D | 1282 | MOV AX,SI; SI HAS #x NUMBER OF RS232
E666 8E3E | 1283 | MOV CL,3; SHIFT COUNT
E667 8E3F | 1284 | ROR AH,CL; ROTATE RIGHT 3 POSITIONS

A-18 System BIOS
System BIOS  A-19

LOC OBJ LINE SOURCE

E64A 0AC3 1085 OR AL, BL OR IN THE PRINTER COUNT
E64C A21100 1086 MOV BYTE PTR EQUIP_FLAG+1, AL STORE AS SECOND BYTE
E64F B0102 1087 MOV DX, 20H
E652 EC 1088 IN AL, DX
E653 90 1089 NOP
E654 90 1090 NOP
E655 90 1091 NOP
E656 A50F 1092 TEST AL, 0FH
E658 7505 1093 JNZ F20 NO_GAME_CARD
E659 0000110010 1094 OR BYTE PTR EQUIP_FLAG+1, AL
E65F 1095 F20 NO_GAME_CARD
E65F E661 1096 :------ ENABLE NMI INTERRUPTE
E660 CD30 1097 IN AL, PORT_B
E661 E661 1098 OR AL, 0FH
E663 E661 1099 OUT PORT_B, AL
E665 24CF 1100 AND AL, 0FH
E666 E661 1101 OUT PORT_B, AL
E668 B050 1102 MOV AL, 0FH
E669 E640 1103 OUT GAH, AL
E66D CD19 1104 INT 19H
E66E CD19 1105 : GO TO THE BOOT LOADER

APPENDIX A

E66F STGSTENT PROC NEAR
E66F FC 1325 CLD
E670 2BFF 1326 SUB DI, DI SET DI OFFSET 0 REL TO ES REG
E672 2BCO 1327 SUB AX, AX SETUP FOR 0-FF PATTERN TEST
E674 CD_1 1328 MOV [DI], AL
E676 BA05 1329 MOV [DI], AL ON FIRST BYTE
E676 BA05 1330 MOV AL, [DI]
E676 32C4 1331 XOR AL, AL 0.K.?
E67A 7940 1332 JNZ C7 GO ERROR IF NOT
E67C FC24 1333 INC AH
E67E BA24 1334 MOV AL, AH
E680 75F2 1335 JNZ C2_1 LOOP TILL WRAP THROUGH FF
E682 8099 1336 MOV BX, CX SAVE WORD COUNT OF BLOCK TO TEST
E684 D1E3 1337 SHL BX, 1 CONVERT TO A BYTE COUNT
E686 B0A000 1338 MOV AX, B0AAA0 INIT INITIAL DATA PATTERN TO WRITE
E689 BA55FF 1339 MOV DX, OFFFFS SETUP OTHER DATA PATTERNS TO USE
E68C FF3 1340 REP STOSW FILL STORAGE LOCATIONS IN BLOCK
E690 8B 1341 MOV AL, PORT_B
E690 0030 1342 OR AL, 0030000D TJoGGLE PARITY CHECK LATCHES
E692 E661 1343 OUT PORT_B, AL
E694 90 1344 NOP
E695 24CF 1345 AND AL, 0000111B
E697 E661 1346 OUT PORT_B, AL
E699 C3: 1347 C3:
E699 4F 1348 DEC DI POINT TO LAST BYTE JUST WRITTEN
E69A FD 1349 STD SET DIS FLAG TO GO BACKWARDS
E69B 1350 C4:
E69B 08F7 1351 MOV SI, DI INITIALIZE DESTINATION POINTER
E69D 08CB 1352 MOV CX, BX SETUP BYTE COUNT FOR LOOP
E69F C5: 1353 C5:
E69F AC 1354 LOOP 00B READ OLD TEST BYTE FROM STORAGE [SI] E6A0 32
E6A0 32C4 1355 XOR AL, AH DATA READ AS EXPECTED?
E6A2 7525 1356 JNE C7 NO - GD TO ERROR ROUTINE
E6A3 B0C2 1357 MOV AL, DL GET NEXT DATA PATTERN TO WRITE
E6A4 AA 1358 STOSB WRITE INTO LOC JUST READ [DI]+1
E6A7 EEFF 1359 LOOP CS DECENT BYTE COUNT AND LOOP CX
E6A9 2244 1360 AND AH, AH ENDING ZERO PATTERN WRITTEN TO STG ?
E6AB 7416 1362 JZ C6X YES - RETURN TO CALLER WITH AL=0
LOC OBJ  
LINE  
SOURCE

E6AD 8AE0  1363  MOV  AH,AL  ; SETUP NEW VALUE FOR COMPARE
E6AF 86F2  1364  XCHG  DH,DL  ; MOVE NEXT DATA PATTERN TO DL
E6B1 72E4  1365  AND  AH,AH  ; READING ZERO PATTERN THIS PASS?
E6B3 75D4  1366  JNZ  C6  ; CONTINUE TEST SEQUENCE TILL ZERO DATA
E6B5 8AD4  1367  MOV  DL,AL  ; ELSE SET ZERO FOR END READ PATTERN
E6B7 8E00  1368  JMP  C3  ; AND MAKE FINAL BACKWARDS PASS
E6B9  1369  C6:  
E6BA FC  1370  CLD  ; SET DIR FLAG TO GO FORWARD
E6B8 74DE  1371  INC  DI  ; SET POINTER TO END LOCATION
E6BC 4F  1372  JZ  C4  ; READ/WRITE FORWARD IN STG
E6BD BA00  1373  DEC  DI  ; ADJUST POINTER
E6BE 0400  1374  MOV  DX,00001H  ; SETUP 01 FOR PARITY BIT AND 00 FOR END
E6C1 8B06  1375  JMP  C5  ; READ/WRITE BACKWARD IN STG
E6C3  1376  C6X:  
E6C5 ED62  1377  IN  AL,PORT_C  ; DID A PARITY ERROR OCCUR?
E6C5 24C0  1378  AND  AL,0C0H  ; ZERO FLAG WILL BE OFF PARITY ERROR
E6C7 B000  1379  MOV  AL,000H  ; AL=0 DATA COMPARE OK
E6C9  1380  C7:  
E6CA FC  1381  CLD  ; SET DIRECTION FLAG TO INC
E6CA C3  1382  RET  
E6CC 1383  STGTEST_CNT  ENDP
E6CD  1384  
E6CE 52  1385  ; PRINT ADDRESS AND ERROR MESSAGE FOR ROM CHECKSUM ERRORS:
E6EC 50  1386  ;  
E6ED 8C0D  1387  ROM_ERR PROC NEAR
E6EF 8A86 1500  1388  MOV  DX,DS:0  ; GET ADDRESS POINT
E6F1 E0A5F8  1389  MOV  ES:MEG_ERR_FLAG.DH  ; <<<<<<<<CHECKPOINTS CO-F4<<<<<>
E6F2 81A0CB  1390  CMP  DX,OC800H  ; CRT CARD IN ERROR?
E6F4 7C0D  1391  JL  ROM_ERR_BEEP  ; GIVE CRT CARD FAIL BEEP
E6F5 EBFD18  1392  CALL  PTR_SEG  ; PRINT SEGMENT IN ERROR
E6F6 B0A0F00  1393  MOV  SI,OFFSET F3A  ; DISPLAY ERROR MSG
E6F7 E8C512  1394  CALL  E_MSG  
E6F8  1395  ERR_BEEP:  
E6F9 8A20D1  1396  MOV  DX,0102H  ; BEEP 1 LONG, 2 SHORT
E6FA EE812  1397  CALL  ERR_BEEP
E6FB EB5  1398  JMP  SHORT  ROM_ERR_END
E6FC  1399  ROM_ERR_END  
E6FD 5A  1400  POP  AX  
E6FE 5A  1401  POP  DX  
E6FF C3  1402  RET  
E6F7  1403  ERR_BEEP:  
E6F8 8A20D1  1404  MOV  DX,0102H  ; BEEP 1 LONG, 2 SHORT
E6F9 E0E12  1405  CALL  ERR_BEEP
E6FA EB5  1406  JMP  SHORT  ROM_ERR_END
E6FB  1407  ROM_ERR_END
E6FC  1408  ; --- INT 19  ---------------------------------------------
E6FD  1409  ; BOOT STRAP LOADER  
E6FE  1410  ; TRACK 0, SECTOR 1 IS READ INTO THE  
E6FF  1411  ; BOOT LOCATION (SEGMENT 0, OFFSET 7C0)  
E6F0  1412  ; AND CONTROL IS TRANSFERRED THERE.  
E6F1  1413  ;  
E6F2  1414  ; IF THERE IS A HARDWARE ERROR CONTROL IS  
E6F3  1415  ; TRANSFERRED TO THE ROM SEC ENTRY POINT.  
E6F4  1416  ;  
E6F5  ASSUME CS:CODE,DS:AD50  1417  ; DREG 06F2H
E6F6  1418  ;  
E6F7  1419  ; BOOTSTRAP  PROC  NEAR
E6F8  1420  ; ENABLE INTERRUPTS  
E6F9  1421  STI  
E6FA 28C0  1422  SUB  AX,AX  ; ESTABLISH ADDRESSING
E6FB 8ED5  1423  MOV  DS,AX  
E6FC  1424  bootstrap_vector  
E6FD 7067800C7EFO  1425  --- REINIT THE DISK PARAMETER TABLE VECTOR
E6FE 8C0F7A00  1426  MOV  WORD PTR DISK_POINTER, OFFSET DISK_BASE
E6FF 8C0F7A00  1427  MOV  WORD PTR DISK_POINTER+2,CS
E6F0  1428  ;  
E6F1  1429  ; --- LOAD SYSTEM FROM DISKETTE -- CK HAS RETRY COUNT
E6F2  1430  ;  
E6F3 890400  1431  MOV  CX,4  ; SET RETRY COUNT
E6F4  1432  ;  
E6F5  1433  HI:  ; IPL_SYSTEM
E6F6  1434  PUSH  CX  ; SAVE RETRY COUNT
E6F7 8A400  1435  MOV  AH,0  ; REINIT THE DISKETTE SYSTEM
E6F8 CD13  1436  INT  13H  ; DISKETTE_ID
E6F9 720F  1437  JC  HZ  ; IF ERROR, TRY AGAIN
E6FA B00102  1438  MOV  AX,201H  ; READ IN THE SINGLE SECTOR
E6FB 2002  1439  SUB  DX,DX  ; TO THE BOOT LOCATION

A-20 System BIOS
LINE                SOURCE
E710 86C2      1440 MOV ES,DX
E712 B007C      1441 MOV BX,OFFSET BOOT_LOCN
E715 B500D      1442 ; DRIVE 0, HEAD 0
E716 C013      1443 MOV CX,1
E717 59        1444 INT 13H
E71A 82E5      1445 ; DISKETTE IO
E71A 59        1446 POP CX
E71B 7504      1447 JNC H4
E71D E2E5      1448 LOOP H1
E759          1449 ; DO IT FOR RETRY TIMES
E75A          1450 ;---- UNABLE TO IPL FROM THE DISKETTE
E75B          1451
E75C          1452 H3:
E75D          1453 INT 18H
E75E          1454 ; GO TO RESIDENT BASIC
E75F          1455 ;---- IPL WAS SUCCESSFUL
E761          1456
E762          1457 H4:
E763          1458 JMP BOOT_ADDR
E769          1459 BOOT_ADDR ENDP
E76A          1460 ;--------------------------------------------------------------------------
E76B          1461 ; RS232 TO
E76C          1462 ; THIS ROUTINE PROVIDES BYTE STREAM I/O TO THE COMMUNICATIONS PORT
E76D          1463 ; PORT ACCORDING TO THE PARAMETERS:
E76E          1464 ; (AH)=0 INITIALIZE THE COMMUNICATIONS PORT
E76F          1465 ; (AL) HAS PARAMETERS FOR INITIALIZATION
E770          1466 ;--------------------------------------------------------------------------
E771          1467 ; 7 6 5 4 3 2 1 0
E772          1468 ; BLT RATE -- >ParITY<-- STOPBIT --WORD LENGTH--
E773          1469 ; 000 - 110 X0 - NONE 0 - 1 10 - 7 BITS
E774          1470 ; 001 - 150 01 - ODD 1 - 2 11 - 8 BITS
E775          1471 ; 010 - 300 11 - EVEN
E776          1472 ; 011 - 60005
E777          1473 ; 100 - 1200
E778          1474 ; 101 - 2400
E779          1475 ; 110 - 4800
E77A          1476 ; 111 - 9600
E77B          1477 ;--------------------------------------------------------------------------
E77C          1478 ; ON RETURN, CONDITIONS SET AS IN CALL TO COMMP STATUS (AH=3)
E77D          1479 ; (AH)=3 SEND THE CHARACTER IN (AL) OVER THE COMM LINE
E77E          1480 ; (AL) REGISTER IS PRESERVED
E77F          1481 ; ON EXIT, BIT 7 OF AH IS SET IF THE ROUTINE WAS UNABLE
E780          1482 ; TO TRANSLATE THE BYTE OF DATA OVER THE LINE.
E781          1483 ; IF BIT 7 OF AH IS NOT SET, THE REMAINDER OF AH
E782          1484 ; IS SET AS IN A STATUS REQUEST, REFLECTING THE
E783          1485 ; CURRENT STATUS OF THE LINE.
E784          1486 ;--------------------------------------------------------------------------
E785          1487 ; (AH)=2 RECEIVE A CHARACTER FROM COMM LINE BEFORE
E786          1488 ; RETURNING TO CALLER
E789          1489 ; ON EXIT, AH HAS THE CURRENT LINE STATUS AS SET BY THE
E78A          1490 ; THE STATUS ROUTINE, EXCEPT THAT THE ONLY BITS
E78B          1491 ; LEFT ON ARE THE ERROR BITS (7,4,3,2,1)
E78C          1492 ; IF AH HAS BIT 7 ON (TIME OUT) THE REMAINING
E78D          1493 ; BITS ARE NOT PREDICTABLE.
E78E          1494 ; THUS, AH IS NON ZERO ONLY WHEN AN ERROR
E78F          1495 ; OCCURRED.
E78G          1496 ;--------------------------------------------------------------------------
E78H          1497 ; (AH)=3 RETURN THE COMM PORT STATUS IN (AX)
E78I          1498 ; AH CONTAINS THE LINE STATUS
E78J          1499 ; BIT 7 = TIME OUT
E78K          1500 ; BIT 6 = TRANS SHIFT REGISTER EMPTY
E78L          1501 ; BIT 5 = TRN HOLDING REGISTER EMPTY
E78M          1502 ; BIT 4 = BREAK DETECT
E78N          1503 ; BIT 3 = FRAMING ERROR
E78O          1504 ; BIT 2 = PARITY ERROR
E78P          1505 ; BIT 1 = OVERRUN ERROR
E78Q          1506 ; BIT 0 = DATA READY
E78R          1507 ; AL CONTAINS THE MODI STATUS
E78S          1508 ; BIT 7 = RECEIVED LINE SIGNAL DETECT
E78T          1509 ; BIT 6 = RING INDICATOR
E78U          1510 ; BIT 5 = DATA SET READY
E78V          1511 ; BIT 4 = CLEAR TO SEND
E78W          1512 ; BIT 3 = DELTA RECEIVE LINE SIGNAL DETECT
E78X          1513 ; BIT 2 = TRAILING EDGE RING DETECTOR
E78Y          1514 ; BIT 1 = DELTA DATA SET READY
E78Z          1515 ; BIT 0 = DELTA CLEAR TO SEND
E78A          1516 ; (DX) = PARAMETER INDICATING WHICH RS232 CARD (0,1 ALLOWED)

System BIOS  A-21
LaC OBJ E729 E729 E729 1704 Ens 0003 E720 8001 E7: F CODa E731 6000 E733 3000 E735 1800 E739 DeaD E739 E73B S2 E73C 56 E73D 57 E73F 51 aBF2 aBFA E742 D156 EBlon SB14 0602 7413 OAE4 7445 74t> 7446,1, SB14 98300 f762 f762 S9 f763 Sf f764 SF f765 Sf f766 SA IF f768 CF f769 8614 f76a 8614 42 2E6A4501 A-22 System BIOS
COT1 PORT STATUS ROUTINE

E797 EE 1594 OUT DX,AL  ; SET MS OF DIV TO 0
E798 4A 1595 DEC DX
E799 2E8A05 1596 MOV AL,SC1011  ; GET LOW ORDER OF DIVISOR
E79C EE 1597 OUT DX,AL  ; SET LOW OF DIVISOR
E79D 03C203 1598 ADD DX,3
E79F 8AC4 1599 MOV AL,AH  ; GET PARMS BACK
E792 2A1F 1600 AND AL,0FH  ; STRIP OFF THE BAUD BITS
E794 EE 1601 OUT DX,AL  ; MODEM CONTROL TO OUT BITS
E796 4A 1602 DEC DX
E796 4A 1603 DEC DX
E797 B000 1604 MOV AL,0
E799 EE 1605 OUT DX,AL  ; INTERRUPT ENABLES ALL OFF
E79A EB49 1606 JMP SHORT A18  ; COM_STATUS

1607 i----- SEND CHARACTER IN (AL) OVER COMM LINE

E79C 1610 A5: PUSH AX  ; SAVE CHAR TO SEND
E79C 50 1611 MOV AX,0  ; SAVE CHAR TO SEND
E7AC 1620 A7: POP CX  ; RESTORE DATA BIITES
E7AA 1621 MOV AL,CX  ; RESTORE DATA BIITES
E7FA 1622 MOV AL,CL  ; RESTORE DATA BIITES
E7FB 1623 AB: OR AH,0BH  ; INDICATE TIME OUT
E7BE 1624 MOV BX,0  ; RETURN
E7B4 1625 JMP A3  ; RETURN
E7B4 1626 A9: INC BX  ; CLEAR TO SEND
E7B4 4A 1627 DEC BX  ; MODEM STATUS REGISTER
E7B5 1628 A10: MOV BH,20H  ; IS TRANSMITTER READY
E7B6 1629 MOV BH,0AH  ; DATA SET READY & CLEAR TO SEND
E7AA 1630 CALL WAIT_FOR_STATUS  ; ARE BOTH TRUE
E7AF 7400 1631 JE A9  ; YES, READY TO TRANSMIT CHAR
E7AC 1632 A11: MOV AH,0  ; MODEM CONTROL REGISTER
E7AC 1633 SUB DX,5  ; MODEM CONTROL REGISTER
E7AC 1634 POP CX  ; MODEM CONTROL REGISTER
E7AC 1635 MOV AL,CX  ; MODEM CONTROL REGISTER
E7AC 1636 MOV AL,CL  ; MODEM CONTROL REGISTER
E7AC 1637 JMP A3  ; MODEM CONTROL REGISTER

1638 i----- RECEIVE CHARACTER FROM COMM LINE

E7B6 1640 E800  ; MODEM CONTROL REGISTER
E7B6 B720 1641 MOV BX,0  ; MODEM CONTROL REGISTER
E7B7 1642 MOV AH,0  ; MODEM CONTROL REGISTER
E7B7 E8000 1643 MOV AH,0  ; MODEM CONTROL REGISTER
E7BB 1644 MOV BX,0  ; MODEM CONTROL REGISTER
E7BB 1645 MOV AH,0  ; MODEM CONTROL REGISTER
E7BC 1646 MOV BX,0  ; MODEM CONTROL REGISTER
E7BC 1647 A13: MOV BH,20H  ; DATA SET READY & CLEAR TO SEND
E7BC 1648 MOV BH,0AH  ; MODEM CONTROL REGISTER
E7BC 1649 CALL WAIT_FOR_STATUS  ; MODEM CONTROL REGISTER
E7BD 750B 1650 JNZ A8  ; MODEM CONTROL REGISTER
E7BD 1651 A15: MOV AH,0  ; MODEM CONTROL REGISTER
E7BD 1652 DEC DX  ; MODEM CONTROL REGISTER
E7BD 1653 JMP A16:  ; MODEM CONTROL REGISTER
E7B5 1654 MOV BH,1  ; RECEIVE BUFFER FULL
E7BA E8000 1655 MOV AH,0  ; RECEIVE BUFFER FULL
E7BA 750B 1656 JNZ A8  ; MODEM CONTROL REGISTER
E7BA 1657 A17: MOV AH,0  ; MODEM CONTROL REGISTER
E7B7 80E41E 1658 AND AL,00011110B  ; TEST FOR ERR CONDITIONS ON RECV CHAR
E7BD 0F14 1659 MOV DX,RS1232_BASE+1I  ; DATA PORT
E7A1 EC 1660 IN AL,DX  ; GET CHARACTER FROM LINE
E7A1 E903 1661 JMP A3  ; MODEM STATUS REGISTER

1662 i----- COMM PORT STATUS ROUTINE

E7A6 1664 E800  ; CONTROL PORT
E7A6 0814 1665 MOV DX,RS1232_BASE+1I  ; CONTROL PORT
E7A6 4A05 1666 ADD DX,5  ; CONTROL PORT
E7AE EC 1667 IN AL,DX  ; GET LINE CONTROL STATUS
E7AE 8A00 1668 MOV AH,AL  ; PUT IN AH FOR RETURN
E7AF 42 1669 MOV AH,AL  ; PUT IN AH FOR RETURN
E7B0 42 1670 INC DX  ; POINT TO MODEM STATUS REGISTER

Appendix A

System BIOS  A-23
LOC OBJ    LINE    SOURCE

E7EC EC    1671    IN AL,DX ; GET MODERN CONTROL STATUS
E7EF E970FF    1672    JMP AX ; RETURN
E7F    1673    ┌─────────────────────────────────────────────┐
E7F4    1674    │ WAIT FOR STATUS ROUTINE │
E7F5    1675    │                        │
E7F6    1676    │ ENTRY:                │
E7F7    1677    │ BH=STATUS BIT(S) TO LOOK FOR: │
E7F8    1678    │ DX=ADDR. OF STATUS REG:      │
E7F9    1679    │ EXIT:                   │
E7FA    1680    │ ZERO FLAG ON = STATUS FOUND:│
E7FB    1681    │ ZERO FLAG OFF = TIMEOUT:    │
E7FC    1682    │ AH=LAST STATUS READ:       │
E7FD    1683    │#----------------------------------------------------------------------------------#
E7FE    1684    │ E7F2    1685    │# WAIT_FOR_STATUS_PROC NEAR #
E7FF    1686    │ E7F5    1687    │ E7F6    1688    │ E7F7    1689    │ E7F8    1690    │ E7F9    1691    │ E7FA    1692    │ E7FB    1693    │ E7FC    1694    │ E7FD    1695    │ E7FE    1696    │ E7FF    1697    │ E7F0    1698    │ E7F1    1699    │ E7F2    1700    │ E7F3    1701    │ E7F4    1702    │ E7F5    1703    │ E7F6    1704    │ E7F7    1705    │ E7F8    1706    │ E7F9    1707    │ E7FA    1708    │ E7FB    1709    │ E7FC    1710    │ E7FD    1711    │ E7FE    1712    │ E7FF    1713    │ E7F0    1714    │ E7F1    1715    │ E7F2    1716    │ E7F3    1717    │ E7F4    1718    │ E7F5    1719    │ E7F6    1720    │ E7F7    1721    │ E7F8    1722    │ E7F9    1723    │ E7FA    1724    │ E7FB    1725    │ E7FC    1726    │ E7FD    1727    │ E7FE    1728    │ E7FF    1729    │ E7F0    1730    │ E7F1    1731    │ E7F2    1732    │ E7F3    1733    │ E7F4    1734    │ E7F5    1735    │ E7F6    1736    │ E7F7    1737    │ E7F8    1738    │ E7F9    1739    │ E7FA    1740    │ E7FB    1741    │ E7FC    1742    │ E7FD    1743    │ E7FE    1744    │ E7FF    1745    │ E7F0    1746    │ E7F1    1747    │ E7F2    1748    │
E7F7 EC    1677    │                   │
E7F8 E970FF    1678    │ SUB CX,CX        │
E7F9    1679    │ MOV AH,AL        │
E7FA    1680    │ CMP AL,BH        │
E7FB    1681    │ JO WFS_END       │
E7FC    1682    │ LOOP WFS1        │
E7FD    1683    │ DEC BL           │
E7FE    1684    │ JNZ WFS0         │
E7FF    1685    │ OR BH,BH         │
E7F0    1686    │ MOV BL,RSZ32Milliseconds Out       │
E7F1    1687    │ PUSH OS          │
E7F2    1688    │ CALL DDS         │
E7F3    1689    │ JZ ASCII_READ    │
E7F4    1690    │ DEC AH           │
E7F5    1691    │ JZ ASCII_STATUS  │
E7F6    1692    │ JZ SHAFT_STATUS  │
E7F7    1693    │ JMP SHORT INT10_END ; EXIT  │
E7F8    1694    │ IN AL,DX         │
E7F9    1695    │ MOV AX,44552524FS22E20 2852455354045 20302022463122 2040455929 802E 00 0A ; ASCII READ
E7FA    1696    │                │
E7FB    1697    │                │
E7FC    1698    │                │
E7FD    1699    │                │
E7FE    1700    │                │
E7FF    1701    │                │
E7F0    1702    │                │
E7F1    1703    │                │
E7F2    1704    │                │
E7F3    1705    │                │
E7F4    1706    │                │
E7F5    1707    │                │
E7F6    1708    │                │
E7F7    1709    │                │
E7F8    1710    │                │
E7F9    1711    │                │
E7FA    1712    │                │
E7FB    1713    │                │
E7FC    1714    │                │
E7FD    1715    │                │
E7FE    1716    │                │
E7FF    1717    │                │
E7F0    1718    │                │
E7F1    1719    │                │
E7F2    1720    │                │
E7F3    1721    │                │
E7F4    1722    │                │
E7F5    1723    │                │
E7F6    1724    │                │
E7F7    1725    │                │
E7F8    1726    │                │
E7F9    1727    │                │
E7FA    1728    │                │
E7FB    1729    │                │
E7FC    1730    │                │
E7FD    1731    │                │
E7FE    1732    │                │
E7FF    1733    │                │
E7F0    1734    │                │
E7F1    1735    │                │
E7F2    1736    │                │
E7F3    1737    │                │
E7F4    1738    │                │
E7F5    1739    │                │
E7F6    1740    │                │
E7F7    1741    │                │
E7F8    1742    │                │
E7F9    1743    │                │
E7FA    1744    │                │
E7FB    1745    │                │
E7FC    1746    │                │
E7FD    1747    │                │
E7FE    1748    │                │
E7FF    1749    │                │
E7F0    1750    │                │
E7F1    1751    │                │

A-24 System BIOS
In this section, the Assembly code for the System BIOS is presented. The code is written in assembly language and includes various instructions for handling inputs, such as scan codes and shift keys, and managing data buffers. The code snippet demonstrates the use of various system BIOS functions and instructions to process input data.

The code snippet includes:
- Instructions for handling scan codes and shift keys.
- Management of data buffers for input data.
- Use of system BIOS functions for input processing.

The code is structured to handle various input scenarios, such as scanning for keys, managing buffer states, and handling shift keys.

For a detailed understanding, the user should refer to the System BIOS documentation and assembly language tutorials.
LOC OBJ | LINE | SOURCE
----------|------|-------------------
E893 FF  |      |                   
E894 1E  |      |                   
E895 FF  |      |                   
E896 FF  | 1806 | DB -1,-1,-1,31,-1,127,-1,17 
E897 FF  |      |                   
E898 FF  |      |                   
E899 1F  |      |                   
E89A FF  |      |                   
E89B FF  |      |                   
E89C FF  |      |                   
E89D 11  |      |                   
E89E 17  | 1807 | DB 23,5,18,20,25,21,9,15 
E89F 05  |      |                   
E8A0 12  |      |                   
E8A1 14  |      |                   
E8A2 19  |      |                   
E8A3 15  |      |                   
E8A4 09  |      |                   
E8A5 0F  |      |                   
E8A6 10  | 1808 | DB 16,27,29,10,-1,1,19 
E8A7 1B  |      |                   
E8A8 1D  |      |                   
E8A9 0A  |      |                   
E8AA FF  |      |                   
E8AB 01  |      |                   
E8AC 13  |      |                   
E8AD 04  | 1809 | DB 4,6,7,0,10,11,12,-1,-1 
E8AE 06  |      |                   
E8AF 07  |      |                   
E8B0 05  |      |                   
E8B1 0A  |      |                   
E8B2 0B  |      |                   
E8B3 0C  |      |                   
E8B4 FF  |      |                   
E8B5 FF  |      |                   
E8B6 FF  | 1810 | DB -1,-1,20,26,24,3,22,2 
E8B7 FF  |      |                   
E8B8 1C  |      |                   
E8B9 1A  |      |                   
E8BA 10  |      |                   
E8BB 03  |      |                   
E8BC 16  |      |                   
E8BD 02  |      |                   
E8BE 0E  | 1811 | DB 14,13,-1,-1,-1,-1,-1,-1 
E8BF 0D  |      |                   
E8C0 FF  |      |                   
E8C1 FF  |      |                   
E8C2 FF  |      |                   
E8C3 FF  |      |                   
E8C4 FF  |      |                   
E8C5 FF  |      |                   
E8C6 2D  | 1812 | DB ' ',' ','-' 
E8C7 FF  |      |                   
E8C8     | 1813 |                   
E8C9 5E  | 1814 | K9 LABEL BYTE 
E8CB     | 1815 | DB 94,95,96,97,98,99,100,101,101 
E8CC 6D  |      |                   
E8CD 63  |      |                   
E8CE 64  |      |                   
E8CF 65  |      |                   
E8D0 66  | 1816 | DB 102,103,-1,-1,119,-1,132,-1 
E8D1 67  |      |                   
E8D2 FF  |      |                   
E8D3 FF  |      |                   
E8D4 77  |      |                   
E8D5 FF  |      |                   
E8D6 04  |      |                   
E8D7 FF  |      |                   
E8DB 72  | 1817 | DB 115,-1,116,-1,117,-1,118,-1 
E8DD FF  |      |                   
E8DE 74  |      |                   
E8DF FF  |      |                   
E8EC 75  |      |                   
E8ED FF  |      |                   

A-26 System BIOS
<table>
<thead>
<tr>
<th>ADDR</th>
<th>DB/WORD</th>
<th>EFFECT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0076</td>
<td>FF</td>
<td>1810 DB -1</td>
</tr>
<tr>
<td>0077</td>
<td>FF</td>
<td>1819 LC TABLE</td>
</tr>
<tr>
<td>0078</td>
<td>K10</td>
<td>1820 LABEL BYTE</td>
</tr>
<tr>
<td>0079</td>
<td></td>
<td>1821 DB 01BH,'1234567890-=',0AH,0H</td>
</tr>
<tr>
<td>007A</td>
<td>3323333353637</td>
<td>38930203D</td>
</tr>
<tr>
<td>0080</td>
<td></td>
<td>00</td>
</tr>
<tr>
<td>0081</td>
<td></td>
<td>08</td>
</tr>
<tr>
<td>0082</td>
<td></td>
<td>1822 DB 'qwertyuiop{',0DH=-1,'asdfghjkl:',027H</td>
</tr>
<tr>
<td>0083</td>
<td></td>
<td>08</td>
</tr>
<tr>
<td>0084</td>
<td>61736666676866A</td>
<td>68C3B</td>
</tr>
<tr>
<td>0085</td>
<td></td>
<td>1823 DB 0DH=-1,0CH,'zxcvbnma',-1,'*',-1, ' '</td>
</tr>
<tr>
<td>0086</td>
<td></td>
<td>08</td>
</tr>
<tr>
<td>0087</td>
<td>7A78637662666D</td>
<td>0C22</td>
</tr>
<tr>
<td>0088</td>
<td></td>
<td>08</td>
</tr>
<tr>
<td>0089</td>
<td>60</td>
<td>1824 DB 1288',37,05EH,'&amp;(',00H,0</td>
</tr>
<tr>
<td>008A</td>
<td></td>
<td>08</td>
</tr>
<tr>
<td>008B</td>
<td>262A28246F2B</td>
<td>08</td>
</tr>
<tr>
<td>008C</td>
<td></td>
<td>27</td>
</tr>
<tr>
<td>008D</td>
<td>5C</td>
<td>1825 DB 'QWERTYUIOP{}'</td>
</tr>
<tr>
<td>008E</td>
<td></td>
<td>00</td>
</tr>
<tr>
<td>008F</td>
<td></td>
<td>FF</td>
</tr>
<tr>
<td>0090</td>
<td></td>
<td>00</td>
</tr>
<tr>
<td>0091</td>
<td></td>
<td>FF</td>
</tr>
<tr>
<td>0092</td>
<td></td>
<td>21402324</td>
</tr>
<tr>
<td>0093</td>
<td>25</td>
<td>1826 DB 84,85,86,87,88,89,90</td>
</tr>
<tr>
<td>0094</td>
<td></td>
<td>55</td>
</tr>
<tr>
<td>0095</td>
<td></td>
<td>56</td>
</tr>
<tr>
<td>0096</td>
<td></td>
<td>57</td>
</tr>
<tr>
<td>0097</td>
<td></td>
<td>58</td>
</tr>
<tr>
<td>0098</td>
<td></td>
<td>59</td>
</tr>
<tr>
<td>0099</td>
<td></td>
<td>5A</td>
</tr>
<tr>
<td>009A</td>
<td></td>
<td>5B</td>
</tr>
<tr>
<td>009B</td>
<td></td>
<td>5C</td>
</tr>
<tr>
<td>009C</td>
<td></td>
<td>5D</td>
</tr>
<tr>
<td>009D</td>
<td></td>
<td>5E</td>
</tr>
<tr>
<td>009E</td>
<td></td>
<td>5F</td>
</tr>
<tr>
<td>009F</td>
<td></td>
<td>60</td>
</tr>
<tr>
<td>00A0</td>
<td>5C</td>
<td>1827 DB 27,'1288',37,05EH,'&amp;(',00H,0</td>
</tr>
<tr>
<td>00A1</td>
<td></td>
<td>08</td>
</tr>
<tr>
<td>00A2</td>
<td></td>
<td>55</td>
</tr>
<tr>
<td>00A3</td>
<td></td>
<td>56</td>
</tr>
<tr>
<td>00A4</td>
<td></td>
<td>57</td>
</tr>
<tr>
<td>00A5</td>
<td></td>
<td>58</td>
</tr>
<tr>
<td>00A6</td>
<td></td>
<td>59</td>
</tr>
<tr>
<td>00A7</td>
<td></td>
<td>5A</td>
</tr>
<tr>
<td>00A8</td>
<td>5B</td>
<td>1828 DB 07EH=-1,'zxcvbnma',-1,0,0, ' '</td>
</tr>
<tr>
<td>00A9</td>
<td></td>
<td>08</td>
</tr>
<tr>
<td>00AA</td>
<td>6F</td>
<td>1829 DB 109,110,111,112,113</td>
</tr>
<tr>
<td>00AB</td>
<td></td>
<td>08</td>
</tr>
<tr>
<td>00AC</td>
<td></td>
<td>6F</td>
</tr>
<tr>
<td>00AD</td>
<td></td>
<td>70</td>
</tr>
<tr>
<td>00AE</td>
<td></td>
<td>71</td>
</tr>
<tr>
<td>00AF</td>
<td></td>
<td>08</td>
</tr>
<tr>
<td>00B0</td>
<td></td>
<td>08</td>
</tr>
<tr>
<td>00B1</td>
<td>60</td>
<td>1830 DB 04,05,06,07,08,09,90</td>
</tr>
<tr>
<td>00B2</td>
<td></td>
<td>09</td>
</tr>
<tr>
<td>00B3</td>
<td>54</td>
<td>1831 DB 91,92,93</td>
</tr>
<tr>
<td>00B4</td>
<td></td>
<td>08</td>
</tr>
<tr>
<td>00B5</td>
<td>64</td>
<td>1832 DB 104,105,106,107,108</td>
</tr>
<tr>
<td>00B6</td>
<td>65</td>
<td>1833 DB 109,110,111,112,113</td>
</tr>
<tr>
<td>00B7</td>
<td>66</td>
<td>1834 DB 104,105,106,107,108</td>
</tr>
<tr>
<td>00B8</td>
<td>67</td>
<td>1835 DB 109,110,111,112,113</td>
</tr>
<tr>
<td>00B9</td>
<td>68</td>
<td>1836 DB 104,105,106,107,108</td>
</tr>
<tr>
<td>00BA</td>
<td>69</td>
<td>1837 DB 109,110,111,112,113</td>
</tr>
<tr>
<td>00BB</td>
<td>6A</td>
<td>1838 DB 104,105,106,107,108</td>
</tr>
<tr>
<td>00BC</td>
<td>6B</td>
<td>1839 DB 109,110,111,112,113</td>
</tr>
<tr>
<td>00BD</td>
<td>6C</td>
<td>1840 DB 104,105,106,107,108</td>
</tr>
<tr>
<td>00BE</td>
<td>6D</td>
<td>1841 DB 109,110,111,112,113</td>
</tr>
<tr>
<td>00BF</td>
<td>6E</td>
<td>1842 DB 104,105,106,107,108</td>
</tr>
<tr>
<td>00C0</td>
<td>6F</td>
<td>1843 DB 109,110,111,112,113</td>
</tr>
<tr>
<td>00C1</td>
<td>70</td>
<td>1844 DB 104,105,106,107,108</td>
</tr>
<tr>
<td>00C2</td>
<td>71</td>
<td>1845 DB 109,110,111,112,113</td>
</tr>
<tr>
<td>00C3</td>
<td>72</td>
<td>1846 DB 104,105,106,107,108</td>
</tr>
<tr>
<td>00C4</td>
<td>73</td>
<td>1847 DB 109,110,111,112,113</td>
</tr>
<tr>
<td>00C5</td>
<td>74</td>
<td>1848 DB 104,105,106,107,108</td>
</tr>
<tr>
<td>00C6</td>
<td>75</td>
<td>1849 DB 109,110,111,112,113</td>
</tr>
<tr>
<td>00C7</td>
<td>76</td>
<td>1850 DB 104,105,106,107,108</td>
</tr>
<tr>
<td>00C8</td>
<td>77</td>
<td>1851 DB 109,110,111,112,113</td>
</tr>
<tr>
<td>00C9</td>
<td>78</td>
<td>1852 DB 104,105,106,107,108</td>
</tr>
<tr>
<td>00CA</td>
<td>79</td>
<td>1853 DB 109,110,111,112,113</td>
</tr>
<tr>
<td>00CB</td>
<td>7A</td>
<td>1854 DB 104,105,106,107,108</td>
</tr>
<tr>
<td>00CC</td>
<td>7B</td>
<td>1855 DB 109,110,111,112,113</td>
</tr>
<tr>
<td>00CD</td>
<td>7C</td>
<td>1856 DB 104,105,106,107,108</td>
</tr>
<tr>
<td>00CE</td>
<td>7D</td>
<td>1857 DB 109,110,111,112,113</td>
</tr>
<tr>
<td>00CF</td>
<td>7E</td>
<td>1858 DB 104,105,106,107,108</td>
</tr>
<tr>
<td>00D0</td>
<td>7F</td>
<td>1859 DB 109,110,111,112,113</td>
</tr>
<tr>
<td>00D1</td>
<td>80</td>
<td>1860 DB 104,105,106,107,108</td>
</tr>
<tr>
<td>00D2</td>
<td>81</td>
<td>1861 DB 109,110,111,112,113</td>
</tr>
<tr>
<td>00D3</td>
<td>82</td>
<td>1862 DB 104,105,106,107,108</td>
</tr>
<tr>
<td>00D4</td>
<td>83</td>
<td>1863 DB 109,110,111,112,113</td>
</tr>
<tr>
<td>00D5</td>
<td>84</td>
<td>1864 DB 104,105,106,107,108</td>
</tr>
<tr>
<td>00D6</td>
<td>85</td>
<td>1865 DB 109,110,111,112,113</td>
</tr>
<tr>
<td>00D7</td>
<td>86</td>
<td>1866 DB 104,105,106,107,108</td>
</tr>
<tr>
<td>00D8</td>
<td>87</td>
<td>1867 DB 109,110,111,112,113</td>
</tr>
<tr>
<td>00D9</td>
<td>88</td>
<td>1868 DB 104,105,106,107,108</td>
</tr>
<tr>
<td>00DA</td>
<td>89</td>
<td>1869 DB 109,110,111,112,113</td>
</tr>
</tbody>
</table>

System BIOS A-27
; KEYBOARD INTERRUPT TEST

; Key FOUND.

; ORG 0E967H

; KB_INT PROC FAR

; STI ; ALLOW FURTHER INTERRUPTS

; PUSH AX

; IN AL, KB_DATA ; READ THE CHARACTER

; PUSH AX ; SAVE IT

; IN AL, KB_CTL ; GET THE CONTROL PORT

; MOV AH, AL ; SAVE VALUE

; OR AL, 080H ; RESET BIT FOR KEYBOARD

; OUT KB_CTL, AL

; XCHG AH, AL ; GET BACK ORIGINAL CONTROL

; POP AX ; RECOVER SCAN CODE

; MOV AH, AL ; SAVE SCAN CODE IN AN ALSO

; CALL DDS ; FORWARD DIRECTION

; CMP AL, 0FFH ; IS THIS AN OVERRUN CHAR

; JNZ K16 ; NO, TEST FOR SHIFT KEY

; JMP K62 ; BUFFER_FULL_BEEP

; TEST FOR SHIFTS

; K16: ; TEST_SHIFT

; AND AL, 07FH ; TURN OFF THE BREAK BIT

; PUSH CS

; MOV DI, OFFSET K6 ; ESTABLISH ADDRESS OF SHIFT TABLE

; MOV CX, 06L ; LENGTH

; REPNE SCASB ; LOOK THROUGH THE TABLE FOR A MATCH

; MOV AL, AH ; RECOVER SCAN CODE

; JE K17 ; JUMP IF MATCH FOUND

; JMP K25 ; IF NO MATCH, THEN SHIFT NOT FOUND

; SHIFT KEY FOUND

; SUB DI, OFFSET K6+1 ; ADJUST PTR TO SCAN CODE MATCH

; MOV AH, CS:K7DI11 ; GET MASK INTO AH

; TEST AL, 06H ; TEST FOR BREAK KEY

; JE K23 ; BREAK_SHIFT_FOUND

; JMP K50 ; IF NO MATCH, THEN SHIFT NOT FOUND

; SHIFT MAKE FOUND, DETERMINE SET OR TOGGLE

; CMP AH, SCROLL_SHIFT ; IF SCROLL SHIFT OR ABOVE, TOGGLE KEY

; JAE K18 ; SET SHIFT ON

; A-28 System BIOS
LOC OBJ   LINE SOURCE

1904
E902 00651700 1905 OR KB_FLAG, AH ; TURN ON SHIFT BIT
E906 0E9000 1906 JMP K26 ; INTERRUPT_RETURN
1907
E909 1908 ; ----- TOGGLE SHIFT KEY, TEST FOR 1ST MAKE OR NOT
1909
E909 F6061700004 1911 TEST KB_FLAG, CTL_SHIFT ; CHECK CTL SHIFT STATE
E90E 7516 1912 JNZ K25 ; JUMP IF KEY NOT DEPRESSED
E90E 3C52 1913 CMP AL, INS_KEY ; CHECK FOR INSERT KEY
E912 7522 1914 JNZ K22 ; JUMP IF NOT INSERT KEY
E914 F6061700006 1915 TEST KB_FLAG, ALT_SHIFT ; CHECK FOR ALTERNATE SHIFT
E915 755A 1916 JNZ K25 ; JUMP IF ALTERNATE SHIFT
E916 F6061700020 1917 K19: TEST KB_FLAG, H&H_STATE ; CHECK FOR BASE STATE
E918 7500 1918 JNZ K22 ; JUMP IF NUM LOCK IS ON
E91E F6061700003 1919 TEST KB_FLAG, LEFT_SHIFT+ RIGHT_SHIFT
E92F 74D0 1920 JZ K22 ; JUMP IF NUMERIC, NOT INSERT
1921
E999 1922 K20: ; NUMERIC ZERO, NOT INSERT KEY
E999 830052 1923 MOV AX, 030H ; PUT OUT AN ASCII ZERO
E99C E90601 1924 JMP K57 ; BUFFER_FILL
E99F 1925 K21: ; RIGHT NUMERIC
E9FF F6061700003 1926 TEST KB_FLAG, LEFT_SHIFT+ RIGHT_SHIFT
EA04 74F3 1927 JZ K90 ; JUMP NUMERIC, NOT INSERT
1928
EA06 1929 K22: ; SHIFT TOGGLE KEY HTS: PROCESS IT
EA06 84261800 1930 TEST AH,KB_FLAG_1 ; IS KEY ALREADY DEPRESSED
EA0A 7540 1931 JNZ K26 ; JUMP IF KEY ALREADY DEPRESSED
EA0C 00261800 1932 OR KB_FLAG_1, AH ; INDICATE THAT THE KEY IS DEPRESSED
EA10 30261700 1933 XOR KB_FLAG,AH ; TOGGLE THE SHIFT STATE
EA14 3C52 1934 CMP AL,INS_KEY ; TEST FOR 1ST MAKE OF INSERT KEY
EA16 7541 1935 JNE K26 ; JUMP IF NOT INSERT KEY
EA18 0B0052 1936 MOV AX,INS_KEY+80H ; SET SCAN CODE INTO AH, 0 INTO AL
EA1B 880701 1937 JMP K57 ; PUT INTO OUTPUT BUFFER
1938
EA1E 1939 K23: ; BREAK-SHIFT-FOUND
EA1E 80F010 1940 CMP AH,SCROLL_SHIFT ; IS THIS A TOGGLE KEY
EA21 731A 1941 JAE K24 ; YES, HANDLE BREAK TOGGLE
EA23 F604 1942 NOT AH ; INVERT MASK
EA25 20261700 1943 AND KB_FLAG,AH ; TURN OFF SHIFT BIT
EA29 3C00 1944 CMP AL,ALT_KEY+60H ; IS THIS ALTERNATE SHIFT RELEASE
EA2B 752C 1945 JNE K26 ; INTERRUPT_RETURN
1946
EA2D 1949 ; ----- ALTERNATE SHIFT KEY RELEASED, GET THE VALUE INTO BUFFER
EA2D A01900 1950 MOV AL,ALT_INPUT
EA30 8400 1951 MOV AH,0 ; SCAN CODE OF 0
EA32 80261900 1952 MOV AL,ALT_INPUT,AH, AH ; ZERO OUT THE FIELD
EA36 3C00 1953 CMP AL,0 ; WAS THE INPUT 0?
EA38 741F 1954 JE K26 ; INTERRUPT_RETURN
EA3A E9101 1955 JMP K58 ; IT WASN'T, SO PUT IN BUFFER
EA3D 1957 K24: ; BREAK-TOGGLE
EA3D F604 1958 NOT AH ; INVERT MASK
EA3F 20261800 1959 AND KB_FLAG_1, AH ; INDICATE NO LONGER DEPRESSED
EA43 8814 1960 JMP SHORT K26 ; INTERRUPT_RETURN
1961
EA45 1962 ; ----- TEST FOR HOLD STATE
EA45 1963 K25: ; NO-SHIFT-FOUND
EA45 3C00 1964 CMP AL,60H ; TEST FOR BREAK KEY
EA47 731D 1965 JAE K26 ; NOTHING FOR BREAK CHAR FROM HERE ON
EA49 F606180008 1966 TEST KB_FLAG_1,HOLD_STATE ; ARE WE IN HOLD STATE
EA4E 7417 1967 JZ K26 ; BRANCH AROUND TEST IF NOT
EA50 3C45 1968 JMP AL,MEM_KEY
EA52 7405 1969 JE K26 ; CAN'T END HOLD ON H&M_LOCK
EA5A 80261800F 1971 AND KB_FLAG_1,HOLD_STATE ; TURN OFF THE HOLD STATE BIT
EA5B 1972 K26: ; INTERRUPT_RETURN
EA59 FA 1973 CLI ; TURN OFF INTERRUPTS
EA5A 8020 1974 MOV AL,EO ; END OF INTERRUPT COMMAND
EA5C E620 1975 OUT 020H,AL ; SEND COMMAND TO INT CONTROL PORT
EA5E 1976 K27: ; INTERRUPT_RETURN-NO-EOI
EASE 07 1977 POP ES
EASF IF 1978 POP DS
EA60 SF 1979 POP DI
EA61 SE 1980 POP SI

System BIOS  A-29
LOC OBJ  LINE  SOURCE

EA62 5A  1961  POP DX
EA63 59  1962  POP CX
EA64 5B  1963  POP BX
EA65 5B  1994  POP AX  ; RESTORE STATE
EA66 CF  1985  IRET  ; RETURN, INTERRUPTS BACK ON
1986  ; WITH FLAG CHANGE
1987
1988  i---- NOT IN HOLD STATE, TEST FOR SPECIAL CHAR
1989
EA67  1990  K2B:  ; NO-HOLD-STATE
EA67 F6061700D  1991  TEST KB_FLAG,ALT_SHIFT  ; ARE WE IN ALTERNATE SHIFT
EA6C 7565  1992  JNZ K29  ; JUMP IF ALTERNATE SHIFT
EA6E E99100  1993  JMP K36  ; JUMP IF NOT ALTERNATE
1994
1995 i---- TEST FOR RESET KEY SEQUENCE (CTL ALT DEL)
1996
EA71  1997  K29:  ; TEST-RESET
EA71 F60617004  1998  TEST KB_FLAG,CTL_SHIFT  ; ARE WE IN CONTROL SHIFT ALSO
EA76 7433  1999  JZ K31  ; NO_RESET
EA78 3C53  2000  CMP AL,DEL_KEY  ; SHIFT STATE IS THERE, TEST KEY
EA7A 752F  2001  JNE K31  ; NO_RESET
2002
2003 i---- CTL-ALT-DEL HAS BEEN FOUND, DO I/O CLEANUP
2004
EA7C C706720003412  2005  MOV RESET_FLAG,123WH  ; SET FLAG FOR RESET FUNCTION
EA82 E98E000F0  2006  JMP RESET  ; JUMP TO POWER ON DIAGNOSTICS
2007
2008 i---- ALT-INPUT-TABLE
EA87  2009  K30 LABEL BYTE
EA87 S2  2010  DB 82,79,80,81,75,76,77
EA88 4F  2011  DB 71,72,73  ; 10 NUMBERS ON KEYPAD
EA8F 4B
EA90 49
2012 i---- SUPER-SHIFT-TABLE
EA91 10  2013  DB 16,17,18,19,20,21,22,23  ; A-Z TYPEWRITER CHARs
EA92 11
EA93 12
EA94 13
EA95 14
EA96 15
EA97 16
EA98 17
EA99 18
EA9A 19
EA9B 1E
EA9C 1F
EA9D 20
EA9E 21
EA9F 22
EA97 23
EA98 24  2015  DB 36,37,38,44,45,46,47,48
EA92 25
EA93 26
EA94 2C
EA95 2D
EA96 2E
EA97 2F
EA98 30
EA99 31  2016  DB 49,50
EA9A 32
2017
2018 i---- IN ALTERNATE SHIFT, RESET NOT FOUND
EA9B  2019  K31:  ; NO-RESET
EA9B 3C39  2020  CMP AL,57  ; TEST FOR SPACE KEY
EA9D 7505  2021  JNC K32  ; NOT THERE
EA9F B020  2022  MOV AL, ' '  ; SET SPACE CHAR
EAB1 E92101  2023  JMP K57  ; BUFFER_FILL
2024
2025
2026 i---- LOOK FOR KEY PAD ENTRY
2027

A-30  System BIOS
EA49 2028 K32: ; ALT-KEY-PAD
EA4B BF87EA 2029 MOVC DI,OFFSET K30 ; ALT-INPUT-TABLE
EA87 B9A000 2030 MOVC CX,10 ; LOOK FOR ENTRY USING KEYPAD
EAAB F2 2031 REPNE SCA5B ; LOOK FOR MATCH
EA8D AE 2032 JNE K33 ; NO-ALT-KEYPAD
EAC2 B187E000 2033 SUB DI,OFFSET K30+1 ; DI NOW HAS ENTRY VALUE
EAC5 B1400A 2034 MOVC AL,ALT_INPUT ; GET THE CURRENT BYTE
EAC7 840A 2035 MOVC AH,10 ; MULTIPLY BY 10
EACF 840C 2036 ADD AH ; ADD IN THE LATEST ENTRY
EACB A21900 2037 MOVC AL,ALT_INPUT,AL ; STORE IT AWAY
EAEC ED09 2038 JMP K26 ; THROW AWAY THAT KEYSROKE

EA00 2040 -------- LOOK FOR SUPERSHIFT ENTRY
EA02 C606190000 2041 MOVC AL,ALT_INPUT,0 ; ZERO ANY PREVIOUS ENTRY INTO INPUT
EA05 B91A00 2044 MOVCL CX,26 ; DI,ES ALREADY POINTING
EA0B F2 2046 REPNE SCA5B ; LOOK FOR MATCH IN ALPHABET
EA0D AE 2047 JNE K34 ; NOT FOUND, FUNCTION KEY OR OTHER
EA0F B000 2048 MOVC AL,0 ; ASCII CODE OF ZERO
EA1E E9F400 2049 JMP K57 ; PUT IT IN THE BUFFER
EA1F 2050
EA20 2051 -------- LOOK FOR TOP ROW OF ALTERNATE SHIFT
EA21 2052 K34: ; ALT-TOP-ROM
EA23 3C02 2054 CMP AL,E ; KEY WITH '1' ON IT
EA25 720C 2055 JB K35 ; NOT ONE OF INTERESTING KEYS
EA2E 3C0E 2056 CMP AL,14 ; IS IT IN THE REGION
EA2F 730F 2057 JAE K35 ; ALT-FUNCTION
EA49 00C476 2058 ADD AH,118 ; CONVERT PSEUDO SCAN CODE TO RANGE
EA5C B000 2059 MOVC AL,0 ; INDICATE AS SUCH
EA6E 2060 JMP K57 ; BUFFER_FILL
EA75 2061
EA7F 2062 -------- TRANSLATE ALTERNATE SHIFT PSEUDO SCAN CODES
EA81 2063 K35: ; ALT-FUNCTION
EA83 3C3B 2064 CMP AL,59 ; TEST FOR IN TABLE
EA83 730D 2065 JAE K37 ; ALT-CONTINUE
EA84 2066
EA86 730F 2067 K36: ; CLOSED-RETURN
EA87 E961FF 2068 JMP K26 ; IGNORE THE KEY
EA88 2069 K37: ; ALT-CONTINUE
EA8A 3C47 2070 CMP AL,71 ; IN KEYPAD REGION
EA8B 73F9 2071 JAE K36 ; IF SO, IGNORE
EA8C 00F4E9 2072 MOV BL,OFFSET K13 ; ALT SHIFT PSEUDO SCAN TABLE
EA8F E91001 2073 JMP K63 ; TRANSLATE THAT
EA94 2074
EA95 2075 -------- NOT IN ALTERNATE SHIFT
EA96 2076
EB02 2077 K38: ; NOT-ALT-PAD
EB02 F406170004 2078 TEST KB_FLAG,CTL_SHIFT ; ARE WE IN Control SHIFT
EB07 7458 2079 JZ K44 ; NOT-CTL-SHIFT
EB08 2080
EB0D 2081 -------- Control SHIFT, TEST SPECIAL CHARACTERS
EB10 2082 K39: ; NO-BREAK
EB13 3C46 2084 CMP AL,SCROLL_KEY ; TEST FOR BREAK
EB15 7510 2085 JNE K39 ; NO-BREAK
EB1B 01E80000 2086 MOVC BX,BUFFER_START ; RESET BUFFER TO EMPTY
EB1B 01E1400 2087 MOVC BX,BUFFER HEAD,BX
EB1B 01E1C00 2088 MOVC BX,BUFFER TAIL,BX
EB1D C606710000 2089 MOVC BIOS_BREAK,60H ; TURN ON BIOS_BREAK BIT
EB1E C01B 2090 INT IBH ; BREAK INTERRUPT VECTOR
EB20 28C0 2091 SUB AL,AX ; PUT OUT DUMMY CHARACTER
EB22 E00000 2092 JMP K57 ; BUFFER_FILL
EB25 2093 K39: ; NO-BREAK
EB25 3C46 2094 CMP AL,NUM_KEY ; LOOK FOR PAUSE KEY
EB27 7511 2095 JNE K41 ; NO-PAUSE
EB29 00E100000 2096 OR KB_FLAG,1,HOLD STATE ; TURN ON THE HOLD FLAG
EB2E B020 2097 MOVC AL,EOI ; END OF INTERRUPT TO CONTROL PORT
EB30 6520 2098 OUT 020H,AL ; ALLOW FURTHER KEYSTROKE INTS
EB35 2099
EB3A 209A -------- DURING PAUSE INTERVAL, TURN CRT BACK ON
EB3F 20A5
EB3F 20A6
EB3F 20A7
EB3F 20A8
EB3F 20A9
EB3F 20AA
EB3F 20AB
EB3F 20AC
EB3F 20AD
EB3F 20AE
EB3F 20AF
EB3F 20B0
EB3F 20B1
EB3F 20B2
EB3F 20B3
EB3F 20B4
EB3F 20B5
EB3F 20B6
EB3F 20B7
EB3F 20B8
EB3F 20B9
EB3F 20BA
EB3F 20BB
EB3F 20BC
EB3F 20BD
EB3F 20BE
EB3F 20BF
EB3F 20C0
EB3F 20C1
EB3F 20C2
EB3F 20C3
EB3F 20C4
EB3F 20C5
EB3F 20C6
EB3F 20C7
EB3F 20C8
EB3F 20C9
EB3F 20CA
EB3F 20CB
EB3F 20CC
EB3F 20CD
EB3F 20CE
EB3F 20CF
EB3F 20D0
EB3F 20D1
EB3F 20D2
EB3F 20D3
EB3F 20D4
EB3F 20D5
EB3F 20D6
EB3F 20D7
EB3F 20D8
EB3F 20D9
EB3F 20DA
EB3F 20DB
EB3F 20DC
EB3F 20DD
EB3F 20DE
EB3F 20DF
EB3F 20E0
EB3F 20E1
EB3F 20E2
EB3F 20E3
EB3F 20E4
EB3F 20E5
EB3F 20E6
EB3F 20E7
EB3F 20E8
EB3F 20E9
EB3F 20EA
EB3F 20EB
EB3F 20EC
EB3F 20ED
EB3F 20EE
EB3F 20EF
EB3F 20F0
EB3F 20F1
EB3F 20F2
EB3F 20F3
EB3F 20F4
EB3F 20F5
EB3F 20F6
EB3F 20F7
EB3F 20F8
EB3F 20F9
EB3F 20FA
EB3F 20FB
EB3F 20FC
EB3F 20FD
EB3F 20FE
EB3F 20FF
EB3F 2100
EB3F 2101
fB3? 7407 2103; YES, NOTHING TO DO
EB39 8AD003 2104; PORT FOR COLOR CARD
EB3A 06500 2105; GET THE VALUE OF THE CURRENT MODE
EB3F EE 2106; SET THE CRT MODE, SO THAT CRT IS ON
EB40 2107 K40: ; PAUSE-LOOP
EB40 F06100008 2108; TEST KB_FLAG_1,HOLD_STATE
EB45 75F9 2109; JNZ K40; LOOP UNTIL FLAG TURNED OFF
EB47 E914FF 2110; JMP K27; INTERRUPT_RETURN_NO_EOI
EB4A 2111 K41: ; NO-PAUSE
EB4B 2112
EB4C 2113 i----- TEST SPECIAL CASE KEY 55
EB4C 3C37 2114; CMP AL,55; NOT-KEY-55
EB4C 7506 2115; JNE K42; START-STOP PRINTING SWITCH
EB51 E90100 2116; JMP K57; BUFFER_FILL
EB52 2117
EB53 2118 i----- SET UP TO TRANSLATE CONTROL SHIFT
EB54 K42: ; NOT-KEY-55
EB54 B00E6 2119; MOV BX,OFFSET K8; SET UP TO TRANSLATE CTRL
EB57 3C3B 2120; CMP AL,59; IS IT IN TABLE
EB59 7276 2121; JB K56; CTRL-TABLE-TRANSLATE
EB5B K43: ; CTRL-TABLE-TRANSLATE
EB5B BBC66B 2122; MOV BX,OFFSET K9; CTRL TABLE SCAN
EB5E E9BC00 2123; JMP K63; TRANSLATE_SCAN
EB5F 2124
EB60 2125 i----- NOT IN CONTROL SHIFT
EB61 K44: ; NOT-CTL-SHIFT
EB61 3C47 2126; CMP AL,71; TEST FOR KEYPAD REGION
EB63 732C 2127; JAE 046; HANDLE KEYPAD REGION
EB65 F06170003 2128; TEST KB_FLAG_LEFT_SHIFT+RIGHT_SHIFT
EB66 745A 2129; CMP AL,59; IS FUNCTION KEY
EB67 740B 2130; JE K50; BACK TAB KEY
EB68 3C3F 2131; CMP AL,15; BACK TAB KEY
EB6E 7505 2132; JNE K46; NOT-BACK-TAB
EB70 B0000F 2133; MOV AX,15*256; SET PSEUDO SCAN CODE
EB73 E960 2134; JMP SHORT K57; BUFFER_FILL
EB75 2135
EB76 2136 ; NOT-BACK-TAB
EB77 3C37 2137; CMP AL,55; PRINT SCREEN KEY
EB77 7509 2138; JNE K46; NOT-PRINT-SCREEN
EB7A 2139
EB7B 2140 i----- ISSUE INTERRUPT TO INDICATE PRINT SCREEN FUNCTION
EB79 B020 2141; MOV AL,EDI; END OF CURRENT INTERRUPT
EB7B E620 2142; OUT O2H,AL; SO FURTHER THINGS CAN HAPPEN
EB7D C005 2143; INT 5H; ISSUE PRINT SCREEN INTERRUPT
EB7F E90C8E 2144; JMP K27; GO BACK WITHOUT EOI OCCURRING
EB82 2145
EB82 3C3B 2146; CMP AL,59; FUNCTION KEYS
EB84 7206 2147; JB K47; NOT-UPPER-FUNCTION
EB86 803589 2148; MOV BX,OFFSET K12; UPPER CASE PSEUDO SCAN CODES
EB89 E99100 2149; JMP K63; TRANSLATE_SCAN
EB8B 2150
EB8C 2151 ; NOT-UPPER-FUNCTION
EB8C BB10E9 2152; MOV BX,OFFSET K11; POINT TO UPPER CASE TABLE
EB8F E940 2153; JMP SHORT K56; OK, TRANSLATE THE CHAR
EB94 2154
EB95 2155 ; KEYPAD KEYS, MUST TEST NUM LOCK FOR DETERMINATION
EB99 B010 2156; MOV AL,EDI; END OF CURRENT INTERRUPT
EB9B E620 2157; OUT O2H,AL; SO FURTHER THINGS CAN HAPPEN
EB9D C005 2158; INT 5H; ISSUE PRINT SCREEN INTERRUPT
EB9F E90C8E 2159; JMP K27; GO BACK WITHOUT EOI OCCURRING
EBB2 2160
EBB2 3C3B 2161; CMP AL,59; FUNCTION KEYS
EBB4 7206 2162; JB K47; NOT-UPPER-FUNCTION
EBB6 803589 2163; MOV BX,OFFSET K12; UPPER CASE PSEUDO SCAN CODES
EBB9 E99100 2164; JMP K63; TRANSLATE_SCAN
EBC0 2165
EBC1 2166 ; NOT-UPPER-FUNCTION
EBC6 BB10E9 2167; MOV BX,OFFSET K11; POINT TO UPPER CASE TABLE
EBE8 E940 2168; JMP SHORT K56; OK, TRANSLATE THE CHAR
EBE9 2169
EBE9 2170 ; KEYPAD KEYS, MUST TEST NUM LOCK FOR DETERMINATION
EBF1 2171
EBF1 F06170020 2172; TEST KB_FLAG,HLM_STATE; ARE WE IN NUM LOCK
EBF6 7250 2173; JNZ K52; TEST FOR SURE
EBF9 F06170003 2174; TEST KB_FLAG_LEFT_SHIFT+RIGHT_SHIFT; ARE WE IN STATE
EBF9 7250 2175; JNZ K53; IF SHIFTED, REALLY NUM STATE
EBF9 2176
EBF9 2177 ; IF SHIFTED, REALLY NUM STATE
EBA1 750B 2178; JE K50; MINUS
EBA3 3C34 2179; CMP AL,70;
EBA5 740C 2180; JE K51;
EBA8 2181
EBA8 ECK47 2182; SUB AL,71; CONVERT ORIGIN
A-32 System BIOS
LOC OBJ   LINE SOURCE

EB48  EB76E9  2100   MOV  BX,OFFSET K15 ; BASE CASE TABLE
EBAC  EB771   2101   JMP  SHORT K64 ; CONVERT TO PSEUDO SCAN
EBAE  EB812   2102   K50:  ; ALREADY
EBAE  EB820A  2103   MOV  AX,74H; NOT-LOWER-FUNCTION
EBB1  EB822   2104   JMP  SHORT K57 ; BUFFER_FILL
EBB3  EB85F   2105   K51:  ; FINAL-RESULT
EBB3  EB85F   2106   MOV  AX,7B4H; PLUS
EBBE  EB81D   2107   JMP  SHORT K56 ; TRANSLATE_CHAR
2108
2109  1------ MIGHT BE NUM LOCK, TEST SHIFT STATUS
2110
EBB8  EB681   2111   TEST  KB_FLAG,LEFT_SHIFT=RIGHT_SHIFT
EBBD  EB750   2112   JNZ  K49 ; SHIFTED TEMP OUT OF NUM STATE
EBBF  EB794   2113   K53:  ; REALLY_NUM_STATE
EBBF  EB7C4   2114   SUB  AL,70 ; CONVERT ORIGIN
EBC1  EB869E  2115   MOV  BX,OFFSET K14 ; NUM STATE TABLE
EBC4  EB880   2116   JMP  SHORT K56 ; TRANSLATE_CHAR
2117
2118  1------ PLAIN OLD LOWER CASE
2119
EBC6  EB201   2119   K54:  ; NOT-SHIFT
EBC6  EB3C8   2120   CMP  AL,59 ; TEST FOR FUNCTION KEYS
EBC8  EB704   2121   JB   K55 ; NOT-LOWER-FUNCTION
EBCA  EB000   2122   MOV  AL,0 ; SCAN CODE IN AN ALREADY
EBCC  EB07    2123   JMP  SHORT K57 ; BUFFER_FILL
EBCE  EB206   2124   K55:  ; NOT-LOWER-FUNCTION
EBCE  EB206   2125   MOV  BX,OFFSET K10 ; LC TABLE
2126
2127  1------ TRANSLATE THE CHARACTER
2128
EBD1  EB211   2127   DEC  AL ; CONVERT ORIGIN
EBD3  EB213   2128   XLAT  CS:K11 ; CONVERT THE SCAN CODE TO ASCII
2129
2130  1------ PUT CHARACTER INTO BUFFER
2131
EBD5  EB215   2131   K57:  ; BUFFER-FILL
EBD5  EB216   2132   CMP  AL,-1 ; IS THIS AN IGNORE CHAR
EBD7  EB219   2133   JE   K59 ; YES, DO NOTHING WITH IT
EBD9  EB220   2134   CMP  AL,-1 ; LOOK FOR -1 PSEUDO SCAN
EBDC  EB221   2135   JE   K59 ; NEAR_INTERRUPT_RETURN
2136
2137  1------ HANDLE THE CAPS LOCK PROBLEM
2138
EBDE  EB225   2137   TEST  KB_FLAG,CAPS_STATE ; ARE WE IN CAPS LOCK STATE
EBE3  EB227   2138   JZ   K61 ; SKIP IF NOT
2139
2140  1------ IN CAPS LOCK STATE
2141
EBE5  EB229   2142   TEST  KB_FLAG,LEFT_SHIFT=RIGHT_SHIFT ; TEST FOR SHIFT STATE
EBEA  EB232   2143   JZ   K60 ; IF NOT SHIFT, CONVERT LOWER TO UPPER
2144
2145  1------ CONVERT ANY UPPER CASE TO LOWER CASE
2146
EBEC  EB233   2146   CMP  AL,'A' ; FIND OUT IF ALPHABETIC
EBEE  EB236   2147   JB   K61 ; NOT_CAPS_STATE
EBEF  EB237   2148   CMP  AL,'Z' ; NOT-CAPS_STATE
EBF2  EB238   2149   JA   K61 ; NOT-CAPS_STATE
EBF4  EB239   2150   ADD  AL,AL-1 ; CONVERT TO LOWER CASE
EBF6  EB240   2151   JMP  SHORT K61 ; NOT-CAPS_STATE
EBF8  EB241   2152   JMP  SHORT K61 ; NOT-CAPS_STATE
EBF9  EB242   2153   K59:  ; NEAR_INTERRUPT_RETURN
2154
2155  1------ INTERRUPT_RETURN
2156
EBF0  EB244   2156   JMP  K26 ; INTERRUPT_RETURN
2157
2158  1------ CONVERT ANY LOWER CASE TO UPPER CASE
2159
EBF0  EB245   2159   ADD  AL,AL-1 ; FIND OUT IF ALPHABETIC
EBF0  EB246   2160   JB   K61 ; NOT_CAPS_STATE
EBF0  EB247   2161   CMP  AL,'Z' ; NOT-CAPS_STATE
EC01  EB250   2162   JA   K61 ; NOT-CAPS_STATE
EC03  EB252   2163   SUB  AL,AL-1 ; CONVERT TO UPPER CASE
ECOS  EB255   2164   CMP  AL,AL-1 ; CONVERT TO UPPER CASE
EC05  EB255   2165   ADD  AL,AL-1 ; GET THE END POINTER TO THE BUFFER
EC09  EB256   2166   MOV  BX,BUFFER_TAIL ; SAVE THE VALUE
EC0B  EB256   2167   CALL  K4 ; ADVANCE THE TAIL
EC0 3B1E1A00 2257  
EC1F 7413 2258  
EC16 8904 2259  
EC1A 93C6E 2260  
EC1D 2261  
EC1E 2262  
EC1F 2263  
EC1G 2264  
EC1H 2265  
EC1I 2266  
EC1J 2267  
EC1K 2268  
EC1L 2269  
EC1M 2270  
EC1N 2271  
EC1O 2272  
EC1P 2273  
EC1Q 2274  
EC1R 2275  
EC1S 2276  
EC1T 2277  
EC1U 2278  
EC1V 2279  
EC1W 2280  
EC1X 2281  
EC1Y 2282  
EC1Z 2283  
EC20 2284  
EC21 2285  
EC22 2286  
EC23 2287  
EC24 2288  
EC25 2289  
EC26 2290  
EC27 2291  
EC28 2292  
EC29 2293  
EC2A 2294  
EC2B 2295  
EC2C 2296  
EC2D 2297  
EC2E 2298  
EC2F 2299  
EC30 2300  
EC31 2301  
EC32 2302  
EC33 2303  
EC34 2304  
EC35 2305  
EC36 2306  
EC37 2307  
EC38 2308  
EC39 2309  
EC3A 2310  
EC3B 2311  
EC3C 2312  
EC3D 2313  
EC3E 2314  
EC3F 2315  
EC40 2316  
EC41 2317  
EC42 2318  
EC43 2319  
EC44 2320  
EC45 2321  
EC46 2322  
EC47 2323  
EC48 2324  
EC49 2325  
EC4A 2326  
EC4B 2327  
EC4C 2328  
EC4D 2329  
LOC OBJ  
LINE  
SOURCE  

A-34 System BIOS
DISK_POINTER

EC94 C606410000 2394 I10V DISKETTE_STATUS,0;
RESET

EC7C 50 2378 POP BP

ECAF

EC9E

EC81 5B

EC7A

EC66 E81COO 2370 CALL

EC5A

EC59

EC59

EC59

EC59

EC59

EC59

EC5C

EC58

EC57

EC5F 55

EC5E

EC60 52

EC61 88EC

EC63 E8F50D

EC66 EB1C00

EC69 E0C400

EC6C E8F001

EC6F 80266000

EC73 8A641000

EC77 80FC01

EC7A F5

EC7C 5D

EC7D 5F

EC7E 5E

EC7F 5F

EC80 59

EC81 5D

EC82 CA0200

EC85

2330  ; (C,H,R,N), WHERE C = TRACK NUMBER, H=HEAD NUMBER,
2331  ; R = SECTOR NUMBER, N= NUMBER OF BYTES PER SECTOR
2332  ; (00=128, 01=256, 02=512, 03=1024). THERE MUST BE ONE
2333  ; ENTRY FOR EVERY SECTOR ON THE TRACK. THIS INFORMATION
2334  ; IS USED TO FIND THE REQUESTED SECTOR DURING READ/WRITE
2335  ; ACCESS.
2336  ;
2337  ; DATA VARIABLE -- DISK_POINTER
2338  ; DOUBLE WORD POINTER TO THE CURRENT SET OF DISKETTE PARAMETERS
2339  ; OUTPUT
2340  ; AH = STATUS OF OPERATION
2341  ; STATUS BITS ARE DEFINED IN THE EQUATES FOR
2342  ; DISKETTE_STATUS VARIABLE IN THE DATA SEGMENT OF THIS
2343  ; MODULE.
2344  ; CY = 0 SUCCESSFUL OPERATION (AH=0 ON RETURN)
2345  ; CY = 1 FAILED OPERATION (AH HAS ERROR REASON)
2346  ; FOR READ/WRITE/VERIFY
2347  ; DS,BX,DX,CH.CL PRESERVED
2348  ; AL = NUMBER OF SECTORS ACTUALLY READ
2349  ; ***** AL MAY NOT BE CORRECT IF TIME OUT ERROR OCCURS
2350  ; NOTE: IF AN ERROR IS REPORTED BY THE DISKETTE CODE, THE
2351  ; APPROPRIATE ACTION IS TO RESET THE DISKETTE, THEN RETRY
2352  ; THE OPERATION. ON READ ACCESS, NO MOTOR START DELAY
2353  ; IS TAKEN, SO THAT THREE RETRIES ARE REQUIRED ON READS
2354  ; TO ENSURE THAT THE PROBLEM IS NOT DUE TO MOTOR
2355  ; START-UP.
2356
2357  ;------------------------------------------------------------
2358  ; ASSUME CS:CODE,DS:DATA,ES:DATA
2359  ; ORG DISCWIN
2360  ; DISKETTE_ID
2361  ; PROC FAR
2362  ; CALL J1; CALL THE REST TO ENSURE DS RESTORED
2363  ; MOV BX,4; GET THE MOTOR WAIT PARAMETER
2364  ; CALL GET_PARM
2365  ; MOV MOTOR_COUNT,AH; SET THE TIMER COUNT FOR THE MOTOR
2366  ; MOV AH,DISKETTE_STATUS; GET STATUS OF OPERATION
2367  ; CMP AH,1; SET THE CARRY FLAG TO INDICATE
2368  ; SUCCESS OR FAILURE
2369  ; POP DX; RESTORE ALL REGISTERS
2370  ; POP BX; RECOVER ADDRESS
2371  ; POP DX; THRO AWAY SAVED FLAGS
2372  ; RET 2; THRO AWAY SAVED FLAGS
2373  ;------------------------------------------------------------
2374  ; J1 PROC NEAR
2375  ; MOV DH,AL; SAVE # SECTORS IN DH
2376  ; MOV DWORD PTR ES:[DISK_STATUS+AH],0; INDICATE A READ OPERATION
2377  ; OR AH,AX; AH=0
2378  ; JNZ DISK_RESET; IF ERROR IF ABOVE
2379  ; MOV AH,1; AH=1
2380  ; REC AH; AH=1
2381  ; MOV AH,2; TEST FOR DRIVE IN 0-3 RANGE
2382  ; CMP DL,4; GET THE MOTOR WAIT PARAMETER
2383  ; DEC DX; RESTORE ALL REGISTERS
2384  ; MOV AH,3; AH=3
2385  ; MOV AH,4; AH=4
2386  ; MOV AH,5; AH=5
; BADCOMMAND
; ERROR CODE, NO SECTORS TRANSFERRED
; UNDEFINED OPERATION

;----- RESET THE DISKETTE SYSTEM

;----- DISKETTE STATUS ROUTINE

;----- DISKETTE READ
LOC OBJ

LINE SOURCE

LD8 1------ DISKETTE FORMAT
2464 2465
ED18 2466 DISK_FORMAT PROC NEAR
ED18 2467 ED28 OR MOTOR_STATUS,0FH ; INDICATE WRITE OPERATION
ED19 2468 MOV AL,04AH ; WILL WRITE TO THE DISKETTE
ED1A 2469 CALL DMA_SETUP ; SET UP THE DMA
ED1B 2470 MOV AH,0DH ; ESTABLISH THE FORMAT COMMAND
ED1C 2471 JMP SHORT RH_OPN ; DO THE OPERATION
ED1D 2472 J10: ; CONTINUATION OF RH_OPN FOR HMT
ED1E 2473 MOV BX,7 ; GET THE
ED1F 2474 CALL GET_PARM ; BYTES/SECTOR VALUE TO NEC
ED20 2475 MOV BX,P ; GET THE
ED21 2476 CALL GET_PARM ; SECTORS/TRACK VALUE TO NEC
ED22 2477 MOV BX,15 ; GET THE
ED23 2478 CALL GET_PARM ; GAP LENGTH VALUE TO NEC
ED24 2479 MOV BX,17 ; GET THE FILLER BYTE
ED25 2480 JMP J16 ; TO THE CONTROLLER
ED26 2481 DISK_FORMAT ENDP
ED27 2482
ED28 2483 )------ DISKETTE WRITE ROUTINE
ED29 2484
ED30 2485 DISK_WRITE PROC NEAR
ED31 2486 ED28 OR MOTOR_STATUS,0FH ; INDICATE WRITE OPERATION
ED32 2487 MOV AL,04AH ; DMA WRITE COMMAND
ED33 2488 CALL DMA_SETUP ; CALL DMA_SETUP
ED34 2489 MOV AH,0CH ; NEC COMMAND TO WRITE TO DISKETTE
ED35 2490 DISK_WRITE ENDP
ED36 2491
ED37 2492 )------ ALLOW WRITE ROUTINE TO FALL INTO RH_OPN
ED38 2493
ED39 2494 ; THIS ROUTINE PERFORMS THE READ/WRITE/VERIFY OPERATION
ED40 2495 )-----------------------------------------------
ED41 2496 RH_OPN PROC NEAR
ED42 2497 ED28 MOV DISKETTE_STATUS,DMA_BOUNDARY ; SET ERROR
ED43 2498 MOV AH,0 ; NO SECTORS TRANSFERRED
ED44 2499 RET ; RETURN TO MAIN ROUTINE
ED45 2500
ED46 2501 J11: ; DO RH_OPN
ED47 2502 PUSH AX ; SAVE THE COMMAND
ED48 2503 CALL
ED49 2504
ED50 2505 )------ TURN ON THE MOTOR AND SELECT THE DRIVE
ED51 2506
ED52 2507 ED28 PUSH CX ; SAVE THE T/S PARMS
ED53 2508 ED28 MOV CL,DL ; GET DRIVE NUMBER AS SHIFT COUNT
ED54 2509 MOV AL,1 ; MASK FOR DETERMINING MOTOR BIT
ED55 2510 SAL AL,CL ; SHIFT THE MASK BIT
ED56 2511 CLI ; NO INTERRUPTS WHILE DETERMINING
ED57 2512 MOV CSM,00001H ; MOTOR STATUS
ED58 2513
ED59 2514 MOV MOTOR_COUNT,OFFH ; SET LARGE COUNT DURING OPERATION
ED60 2515 ED28 TEST AL,MOTOR_STATUS ; TEST THAT MOTOR FOR OPERATING
ED61 2516 MOV AH,0 ; IF RUNNING, SKIP THE WAIT
ED62 2517 AND MOTOR_STATUS,OFFH ; TURN OFF ALL MOTOR BITS
ED63 2518 OR MOTOR_STATUS,AL ; TURN ON THE CURRENT MOTOR
ED64 2519 STI ; INTERRUPTS BACK ON
ED65 2520 MOV AL,01H ; MASK BIT
ED66 2521 SAL AL,CL ; DEVELOP BIT MASK FOR MOTOR ENABLE
ED67 2522 OR AL,DL ; GET DRIVE SELECT BITS IN
ED68 2523 OR AL,0CH ; NO RESET, ENABLE DMA/INT
ED69 2524 PUSH DX ; SAVE REG
ED6A 2525 MOV DX,03F2H ; CONTROL PORT ADDRESS
ED6B 2526 OUT DX,AL
ED6C 2527 POP DX ; RECOVER REGISTERS
ED6D 2528
ED6E 2529 )------ WAIT FOR MOTOR IF WRITE OPERATION
ED6F 2530
ED70 2531 ED28 TEST MOTOR_STATUS,00H ; IS THIS A WRITE
ED71 2532 JZ J14 ; NO, CONTINUE WITHOUT WAIT
ED72 2533 MOV BA,LO ; GET THE MOTOR WAIT
ED73 2534 ED28 CALL GET_PARM ; PARAMETER
ED74 2535 OR AH,0AH ; TEST FOR NO WAIT
ED75 2536 J12: ; TEST_WAIT_TIME
ED76 2537 JE J14 ; EXIT WITH TIME EXPIRED
ED77 2538 JZ J13 ; EXIT WITH TIME EXPIRED
ED78 2539 J3: ; SET UP 1/8 SECOND LOOP TIME
ED79 2540 LOOP J13 ; WAIT FOR THE REQUIRED TIME
ED7A 2541
ED7B 2542
ED7C 2543
ED7D 2544
ED7E 2545
ED7F 2546

Appendix A

System BIOS A-37
A-38 System BIOS
LOC OBJ  LINE  SOURCE

EE0F 0410  2630  MOV  AH, BAD_CRC
EE11 721C  2639  JC  J19  ; RH_FAIL
EE13 00E0  2640  SAL  AL, 1  ; TEST FOR DNA OVERRUN
EE15 0440  2641  MOV  AH, BAD_DMA
EE17 7216  2642  JC  J19  ; RH_FAIL
EE19 00E0  2643  SAL  AL, 1
EE1B 00E0  2644  SAL  AL, 1  ; TEST FOR RECORD NOT FOUND
EE1D 0440  2645  MOV  AH, RECORD_NOT_FOUND
EE1F 720E  2646  JC  J19  ; RH_FAIL
EE21 00E0  2647  SAL  AL, 1
EE23 0440  2648  MOV  AH, WRITE_PROTECT
EE25 7208  2649  JC  J19  ; RH_FAIL
EE27 00E0  2650  SAL  AL, 1  ; TEST MISSING ADDRESS MARK
EE29 B404  2651  MOV  AH, BAD_ADDR_MARK
EE2B 7202  2652  JC  J19  ; RH_FAIL
EE2D 5266  2654  ;----- NEC MUST HAVE FAILED
EE2D B420  2655  MOV  AH, BAD_NEC
EE2F 5266  2657  MOV  AH, BAD_NEC
EE2F 00264100  2658  J19:  ; RH_FAIL
EE33 E87001  2659  CALL  HM_TRANS  ; HOW MANY WERE REALLY TRANSFERRED
EE36 5266  2660  J20:  ; RH_ERR
EE36 C3  2661  RET  ; RETURN TO CALLER
EE37 5266  2662  J21:  ; RH_ERR_RES
EE37 E8F0F1  2663  CALL  RESULTS  ; FLUSH THE RESULTS BUFFER
EE3A C3  2664  RET
EE3E 5266  2665  ;----- OPERATION WAS SUCCESSFUL
EE3B 5266  2666  J22:  ; OPN_OK
EE3B E87001  2667  CALL  HM_TRANS  ; HOW MANY GOT MOVED
EE3E 32E4  2668  XOR  AH, AH  ; NO ERRORS
EE40 C3  2669  RET
EE40 5267  2670  ;------------------------------------------------------------------------
EE41 5267  2671  NEC_OUTPUT  PROC  NEAR
EE41 5267  2672  NEC_OUTPUT  PROC
EE41 S2  2673  PUSH  DX  ; SAVE REGISTERS
EE42 S1  2674  PUSH  CX
EE43 BAF403  2675  MOV  DX, 03F4H  ; STATUS PORT
EE44 33C9  2676  XOR  CX, CX  ; COUNT FOR TIME OUT
EE48 5267  2677  J23:  ; PHONE_HOME
EE4B EC  2678  IN  AL, DX  ; GET STATUS
EE49 A040  2679  TEST  AL, 040H  ; TEST DIRECTION BIT
EE49 7C26  2680  J26  ; DIRECTION OK
EE4A E2F9  2681  LOOP  J23
EE4E 5267  2682  J24:  ; TIME_ERROR
EE4F 000E410000  2683  OR  DISKETTE_STATUS, TIME_OUT
EE54 59  2684  PNP  CX
EE55 5A  2685  PNP  DX  ; SET ERROR CODE AND RESTORE REGS
EE56 50  2686  PNP  AX  ; DISCARD THE RETURN ADDRESS
EE57 F9  2687  STC  ; INDICATE ERROR TO CALLER
EE58 C3  2688  RET
EE59 5267  2689  J25:  ; DISCARD
EE59 33C9  2690  XOR  CX, CX  ; RESET THE COUNT
EE5B 5267  2691  J26:  ; COUNT DOWN AND TRY AGAIN
EE5B EC  2692  IN  AL, DX  ; GET THE STATUS
EE5C A060  2693  TEST  AL, 060H  ; IS IT READY
EE5E 7504  2694  JNC  J27  ; YES, GO OUTPUT
EE60 E2F9  2695  LOOP  J24

Appendix A

System BIOS  A-39
PROGRAM

; INPUT

; This routine fetches the indexed pointer from the disk base

; Block pointed at by the data variable disk pointer, a byte from

; That table is then moved into ah, the index of that byte being

; The parm in bx

; Entry

; Bx = index of byte to be fetched * 2

; If the low bit of bx is on, the byte is immediately output

; To the nec controller

; Exit --

; ah, byte from block

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

; GET_PARM PROC NEAR

; source

ASSUME DS:DATA

GET_PARM:

; seek

; this routine will move the head on the named drive to the

; named track. if the drive has not been accessed since the

; drive reset command was issued, the drive will be recalibrated.

; input

; (dl) = drive to seek on

; (ch) = track to seek to

; output

; cy = 0 success

; cy = 1 failure -- diskette status set accordingly

; (ax) destroyed

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

; seek

; move al,1

; establish mask for recal test

; push cx

; save input values

; mov cl,dl

; get drive value into cl

; rol al,cl

; shift it by the drive value

; pop cx

; recover track value

; mov ah,0si+bx1

; get the word

; pop ds

; restore segment

; assume ds:data

; jc nec_output

; if flag set, output to controller

; ret

; return to caller

ENDP

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

; get_parm

; getparm

; source

GET_PARM:

; seek

; mov al,1

; establish mask for recal test

; push cx

; save input values

; mov cl,dl

; get drive value into cl

; rol al,cl

; shift it by the drive value

; pop cx

; recover track value

; mov ah,0si+bx1

; get the word

; pop ds

; restore segment

; assume ds:data

; jc nec_output

; if flag set, output to controller

; ret

; return to caller

ENDP
System BIOS  A-41

LOC OBJ  LINE  SOURCE

2792 ;----- WAIT FOR HEAD SETTLE
2793
2794 EEB0 9C
2795 2796 EEB1 BD200
2797 CALL GET_PARM
2798 EEB7 51
2799 PUSH CX
2800 SAVE STATUS FLAGS
2801 GET HEAD SETTLE PARAMETER
2802 EE8B 129
2803 JZ J29:
2804 RETURN TO CALLER
2805 EE8B 1270
2806 OR AH, AH
2807 TEST FOR TIME EXPIRED
2808 EEB8 7406
2809 JZ J31
2810 EE8B 3EYE
2811 LOOP J30:
2812 DELAY FOR 1 MS
2813 EE8C EFC
2814 DEC AX
2815 DECREMENT THE COUNT
2816 EE8C EBF3
2817 JMP J29:
2818 DO IT SOME MORE
2819 EE8E S9
2820 POP CX
2821 RECOVER STATE
2822 EE8E 9D
2823 POPF
2824 SEEK ERROR
2825 EE8E C7
2826 POP AX
2827 RETURN FROM ADDITION
2828 EE8E C3
2829 RET
2829
2830 SEEK ENDP
2831
2832 -------------------------------
2833 | DMA_SETUP |
2834 -------------------------------
2835
2836 EE8C 51
2837 CALL DHA_SETUP
2838 ; INITIALIZE THE INTERRUPTS
2839
2840 EE8C 5B
2841 CLI
2842 ; INTERRUPTS OFF
2843
2844 EE8C E66C
2845 OUT DMA+12, AL
2846 ; GET THE FIRST/LAST I/F
2847
2848 EE8C 50
2849 PUSH AX
2850 ; SAVE THE REGISTER
2851
2852 EE8C 56
2853 MOV AX, ES
2854 ; NO MORE INTERRUPTS
2855
2856 EE8C E66B
2857 MOV AX, DS
2858 ;(high) DATA ADDRESS
2859
2860 EE8C E6C0
2861 MOV AX, DS
2862 ;(low) DATA ADDRESS
2863
2864 EE8C B104
2865 MOV CL, 4
2866 ; SHIFT COUNT
2867
2868 EE8C D3C0
2869 ROL AX, CL
2870 ; ROTATE LEFT
2871
2872 EE8C 84E8
2873 MOV CH, CL
2874 ; GET HIGHEST NYBLE OF ES TO CH
2875
2876 EE8C 24F0
2877 AND AL, 0FH
2878 ; ZERO THE LOW NYBLE FROM SEGMENT
2879
2880 EE8E 3C3
2881 ADD AX, 03H
2882 ; TEST FOR CARRY FROM ADDITION
2883
2884 EE8E 7302
2885 JNC J33:
2886 ; CARRY MEANS HIGH 4 BITS MUST BE INC
2887
2888 EE8E 59
2889 PUSH AX
2890 ; SAVE START ADDRESS
2891
2892 EE8E E604
2893 OUT DMA+4, AL
2894 ; OUTPUT LOW ADDRESS
2895
2896 EE8E 84C4
2897 MOV AX, AL
2898 ; DATA ADDRESS
2899
2899 EE8E E604
2900 OUT DMA+4, AL
2901 ; OUTPUT HIGH ADDRESS
2902
2903 EE8E 84C5
2904 MOV AX, AL
2905 ; DATA ADDRESS
2906
2907 EE8E 24F0
2908 AND AL, 0FH
2909 ; OUTPUT THE HIGH 4 BITS TO
2910
2911 EE8E 6601
2912 OUT 0B3H, AL
2913 ; THE PAGE REGISTER
2914
2915 EE8E 6645
2916 MOV AH, DI
2917 ; NUMBER OF SECTORS
2918
2919 EE8E 24C0
2920 SUB AL, AL
2921 ; TIMES 256 INTO AX
2922
2923 EE8E D16B
2924 SHR AX, 1
2925 ; SECTORS = 128 INTO AX
2926
2927 EE8E 550
2928 PUSH AX
2929 ; DATA ADDRESS
2930
2931 EE8E BD00
2932 MOV BX, 0
2933 ; GET THE BYTES/SECTOR PARM
2934
2935 EE8E E872FF
2936 CALL GET_PARM
2937 ; USE AS SHIFT COUNT (0-128, 1-256 ETC)
2938
2939 EE8E 8ACC
2940 MOV CL, AH
2941 ; SHIFT COUNT VALUE
2942
2943 EE8E 55C5
2944 PUSH AX
2945 ; DATA ADDRESS
2946
2947 EE8E D3E0
2948 SHR AX, CL
2949 ; MUL OFF BY CORRECT AMOUNT
2950
2951 EE8E 460
2952 DEC AX
2953 ; -1 FOR DMA VALUE
2954
2955 EE8E 0050
2956 PUSH AX
2957 ; SAVE COUNT VALUE
2958
2959 EE8E 6E65
2960 OUT DMA+5, AL
2961 ; LOW BYTE OF COUNT
2962
2963 EE8E 04C4
2964 MOV AX, AL
2965 ; DATA ADDRESS
2966
2967 EE8E 6E65
2968 OUT DMA+5, AL
2969 ; HIGH BYTE OF COUNT
2970
2971 EE8E 6837
2972 STI
2973 ; INTERRUPTS BACK ON
2974
2975 EE8E 059
2976 POP CX
2977 ; RECOVER COUNT VALUE
2978
2979 EE8E 05F0
2980 POP AX
2981 ; RECOVER ADDRESS VALUE
2982
2983 EE8E 03C1
2984 ADD AX, CX
2985 ; ADD, TEST FOR 64K OVERFLOW
2986
2987 EE8E 59
2988 POP AX
2989 ; RECOVER REGISTER
2990
2991 EE8E 0002
2992 MOV AX, AL
2993 ; MODE FOR 6237
2994
2995 EE8E 660A
2996 OUT DMA+10, AL
2997 ; INITIALIZE THE DISKETTE CHANNEL

Appendix A
EF11 C3
2569 RET ; RETURN TO CALLER.
2570 I CFL SET BY ABOVE IF ERROR
2571 DMA_SETUP ENDP
2572 i------------------------------------------ i
2573 | CHK_STAT_2 | i
2574 | THIS ROUTINE HANDLES THE INTERRUPT RECEIVED AFTER A | i
2575 | RECALIBRATION, SEEK, OR RESET TO THE ADAPTER. | i
2576 | THE INTERRUPT IS WAITED FOR, THE INTERRUPT STATUS SENSED, | i
2577 | AND THE RESULT RETURNED TO THE CALLER. | i
2578 | INPUT | i
2579 | NONE | i
2580 | OUTPUT | i
2581 | CY = 0 SUCCESS | i
2582 | CY = 1 FAILURE -- ERROR IS IN DISKETTE_STATUS | i
2583 | (AX) DESTROYED | i
2584 i------------------------------------------ i

EF12
2005 CHK_STAT_2 PROC NEAR ; WAIT FOR THE INTERRUPT
2006 CALL WAIT_INT
2007 JC JS4 ; IF ERROR, RETURN IT
2008 E8 06H ; SENSE INTERRUPT STATUS COMMAND
2009 E8B D0H ; CALL HECT_OUTPUT
2010 CALL RESULTS ; READ IN THE RESULTS
2011 E8 720A ; JC JS4 ; CHK2_RETURN
2012 E8 04200 ; MOV AL,NEC_STATUS ; GET THE FIRST STATUS BYTE
2013 E8 2460 ; AND AL,060H ; ISOLATE THE BITS
2014 E8 7402 ; CMP AL,060H ; TEST FOR CORRECT VALUE
2015 E8 7402 ; JZ JS5 ; IF ERROR, GO MARK IT
2016 E8 0F6 ; CLC ; GOOD RETURN
2017 E8B 9F ; RET ; RETURN TO CALLER
2018 E8B 9F ; CHK2_ERROR
2019 E8C 000410040 ; OR DISKETTE_STATUS,REQ_SEEK
2020 E8F 9F ; STC ; ERROR RETURN CODE
2021 E8F 9F ; RET ; Disconnect
2022 E8F 9F ; CHK_STAT_2 ENDP
2023 i------------------------------------------ i
2024 | WAIT_INT | i
2025 | THIS ROUTINE WAITS FOR AN INTERRUPT TO OCCUR. A TIME OUT | i
2026 | ROUTINE TAKES PLACE DURING THE WAIT, SO THAT AN ERROR MAY BE | i
2027 | RETURNED IF THE DRIVE IS NOT READY. | i
2028 | INPUT | i
2029 | NONE | i
2030 | OUTPUT | i
2031 | CY = 0 SUCCESS | i
2032 | CY = 1 FAILURE -- DISKETTE_STATUS IS SET ACCORDINGLY | i
2033 | (AX) DESTROYED | i
2034 i------------------------------------------ i

EF35
2056 WAIT_INT PROC NEAR ; TURN OFF INTERRUPTS, JUST IN CASE
2057 STI
2058 PUSH BX
2059 PUSH CX
2060 MOV BL,0
2061 XOR CX,CX
2062 FOR 2 SECOND WAIT
2063 E8A JS6
2064 E8F 6064E0080 ; TEST SEEK_STATUS,INT_FLAG ; TEST FOR INTERRUPT OCCURRING
2065 E8F 750C ; JNZ JS7 ; NO INTERRUPT OCCURRED
2066 E8F 742F ; LOOP JS6 ; COUNT DOWN WHILE WAITING
2067 E8F 754F ; DEC BL ; SECOND LEVEL COUNTER
2068 E8F 75FF ; JNZ JS6 ; INTERRUPT OCCURRED
2069 E8F 8004100000 ; OR DISKETTE_STATUS,TIME_OUT ; NOTHING HAPPENED
2070 E8F 9C ; STC ; ERROR RETURN
2071 E8F 9C ; JS7:
2072 E8F 9C ; PUSHF ; SAVE CURRENT CARRY
2073 E8F 8026E007F ; AND SEEK_STATUS,NOT_INT_FLAG ; TURN OFF INTERRUPT FLAG
2074 E8F 9C ; POPF ; RECOVER CARRY
2075 E8F 8046 ; POP CX
2076 E8F 5B ; POP BX ; RECOVER REGISTERS
2077 E8F 5B ; RET ; GOOD RETURN CODE COMES
2078 E8F 9C ; FROM TEST INST
2079 E8F 9C ; WAIT_INT ENDP
2080 i------------------------------------------ i

A-42 System BIOS
EF57 2947 ORG $EF57H
EF57 2948 DISK_INT PROC FAR
EF57 FB 2949 STI ; RE ENABLE INTERRUPTS
EF56 1E 2950 PUSH DS
EF59 50 2951 PUSH AX
EF5A 0EF0CA 2952 CALL DOS
EF5D 0003E30000 2953 OR $SEEK_STATUS.INT_FLAG
EF62 0000 2954 MOV AL,0AH ; END OF INTERRUPT MARKER
EF64 E620 2955 OUT 20H,AL ; INTERRUPT CONTROL PORT
EF66 55 2956 POP AX
EF67 1F 2957 POP DS ; RECOVERY SYSTEM
EF68 0F 2958 IRET ; RETURN FROM INTERRUPT
EF69 2959 ;----------------------------------------------------
EF69 2960 DISK_INT ENDP
EF69 2961 ;RESULTS ;-----------------------------------------
EF69 2962 THIS ROUTINE WILL READ ANYTHING THAT THE NEC CONTROLLER HAS ;
EF69 2963 TO SAY FOLLOWING AN INTERRUPT. ;
EF69 2964 ;INPUT ;
EF69 2965 ;OUTPUT ;
EF69 2966 ; CY = 0 SUCCESSFUL TRANSFER ;
EF69 2967 ; CY = 1 FAILURE -- TIME OUT IN WAITING FOR STATUS ;
EF69 2968 ; NEC_STATUS AREA HAS STATUS BYTE LOADED INTO IT ;
EF69 2969 ; (AH) DESTROYED ;----------------------------------
EF6A 2970 RESULTS PROC NEAR
EF6A FC 2971 CLD
EF6B 0F6200 2972 MOV DI,OFFSET NEC_STATUS ; POINTER TO DATA AREA
EF6D 51 2973 PUSH CX ; SAVE COUNTER
EF6F 53 2974 PUSH BX
EF70 8307 2975 MOV BL,7 ; MAX STATUS BYTES
EF72 2976 ;----- WAIT FOR REQUEST FOR MASTER
EF72 2977 J3A: ; INPUT_LOOP
EF72 33C9 2978 XOR CX,CX ; COUNTER
EF74 BAF403 2979 MOV DX,03F4H ; ST BUS PORT
EF77 2980 J39: ; WAIT FOR MASTER
EF78 A500 2981 TEST AL,06H ; MASTER READY
EF7A 750C 2982 JNZ J40A ; TEST_DIR
EF7C E0F9 2983 LOOP J39 ; WAIT_MASTER
EF7E 00E410000 2984 OR DISKETTE_STATUS,TIME_OUT
EF7F 2985 J40: ; RESULTS_ERROR
EF80 3F 2986 STC ; SET ERROR RETURN
EF84 51 2987 POP DX
EF85 5A 2988 POP CX
EF86 59 2989 POP BX
EF87 C3 2990 RET
EF94 2991 ;----- TEST THE DIRECTION BIT
EF94 2992 J42: ; INPUT_STAT
EF94 33C9 2993 XOR CX,CX ; COUNTER
EF96 BAF403 2994 MOV DX,03F4H ; STATUS PORT
EF97 2995 J39: ; WAIT FOR MASTER
EF98 EC 2996 IN AL,DX ; GET STATUS
EF9A A500 2997 TEST AL,06H ; MASTER READY
EF9C 7507 2998 JNZ J42 ; OK TO READ STATUS
EF9D 3004 J41: ; NEC_FAIL
EF9F 00E410020 2999 OR DISKETTE_STATUS.BAD_NEC
EFA2 E8EF 3000 JMP J40 ; RESULTS_ERROR
EFB7 2999 ;----- READ IN THE STATUS
EFB7 3000 J40A: ; INPUT_STAT
EFB8 EC 3001 IN AL,DX ; GET STATUS REG AGAIN
EFB9 A500 3002 TEST AL,040H ; TEST DIRECTION BIT
EFBA 7507 3003 JNZ J42 ; OK TO READ STATUS
EFBB 3004 J41: ; NEC_FAIL
EFBC 00E410020 3005 OR DISKETTE_STATUS.BAD_NEC
EFD0 29A4 ; RESULTS_ERROR
EFD2 E8EF 3006 JMP J40 ; RESULTS_ERROR
EFD7 3007 ;----- READ IN THE STATUS
EFD7 3008 J42: ; INPUT_STAT
EFDC 42 3011 INC DX ; POINT AT DATA PORT
EFDF EC 3012 IN AL,DX ; GET THE DATA
EFEE 0005 3013 MOV [DI],AL ; STORE THE BYTE
EFF0 47 3014 INC DI ; INCREMENT THE POINTER
EF99 B90A00 3015 MOV CX,10 ; LOOP TO KILL TIME FOR NEC
EF9C E2FE 3016 LOOP J43
EF9C 4A 3017 DEC DX ; POINT AT STATUS PORT
EF9F EC 3018 IN AL,DX ; GET STATUS
EFA0 A810 3019 TEST AL,010H ; TEST FOR NEC STILL BUSY
EFA2 7406 3020 JZ J44 ; RESULTS DONE
EFA4 FECB 3021 DEC BL ; DECREMENT THE STATUS COUNTER
EFA6 75CA 3022 JNZ J58 ; GO BACK FOR MORE

System BIOS A-43
A-44 System BIOS
LaC
EFD6 51
EFD4
HDS
EFDZ
EfF7 SE
EFFE
EFF9
EFFS
EFFS
EFF3 7428 3132
EFEF 743F
EFE9
EFE5
EFE2
EFED
EFDD
EFE7
FOl7 42
EH8 50
FOIS
F018
F015 3160
EFFC
FOO
f007 E2F7
FOl5 BODO
F022 EC
F021 42
FOIE 885408
FOIE 3172
FOlD 50
FOlD 3170
F009
FOD8
F007
F006 59
F005 5A
F004 SE
F003 5B
F002 1F
F001 CF
F000 40
F00D
F00C
F00B
F00A
F009
F008
F007 75F1
F006 60CC01
F005 5D
F004 4D
F003 4C
F002 4B
F001 D3
F000 DC

3100 ; DATA AREA  PRINTER_BASE CONTAINS THE BASE ADDRESS OF THE PRINTER :
3101 ; CARD(S) AVAILABLE (LOCATED AT BEGINNING OF DATA SEGMENT, :
3102 ; 400H ABSOLUTE, 3 WORDS)
3103 ;
3104 ; DATA AREA PRINT_TIM_OUT (BYTE) MAY BE CHANGED TO CAUSE DIFFERENT :
3105 ; TIME-OUT WAITS. DEFAULT=20
3106 ;
3107 ; REGISTERS AH IS MODIFIED
3108 ; ALL OTHERS UNCHANGED
3109 ;---------------------------------------------------------------------.---
3110 ; ASSUME CS:CODE,DS:DATA
3111 ; ORG 6F70H
3112 ;-------------------
3113 ; PRINTER_ID PROC FAR
3114 ; interrupts back on
3115 ;-------------------
3116 ;-------------------
3117 ;-------------------
3118 ;-------------------
3119 ;-------------------
3120 ;-------------------
3121 ;-------------------
3122 ;-------------------
3123 ;-------------------
3124 ;-------------------
3125 ;-------------------
3126 ;-------------------
3127 ;-------------------
3128 ;-------------------
3129 ;-------------------
3130 ;-------------------
3131 ;-------------------
3132 ;-------------------
3133 ;-------------------
3134 ;-------------------
3135 ;-------------------
3136 ;-------------------
3137 ;-------------------
3138 ;-------------------
3139 ;-------------------
3140 ;-------------------
3141 ;-------------------
3142 ;-------------------
3143 ;-------------------
3144 ;-------------------
3145 ;-------------------
3146 ;-------------------
3147 ;-------------------
3148 ;-------------------
3149 ;-------------------
3150 ;-------------------
3151 ;-------------------
3152 ;-------------------
3153 ;-------------------
3154 ;-------------------
3155 ;-------------------
3156 ;-------------------
3157 ;-------------------
3158 ;-------------------
3159 ;-------------------
3160 ;-------------------
3161 ;-------------------
3162 ;-------------------
3163 ;-------------------
3164 ;-------------------
3165 ;-------------------
3166 ;-------------------
3167 ;-------------------
3168 ;-------------------
3169 ;-------------------
3170 ;-------------------
3171 ;-------------------
3172 ;-------------------
3173 ;-------------------
3174 ;-------------------
3175 ;-------------------
3176 ;-------------------

Appendix A
System BIOS A-45
F025 B004F8 3177 AND AL,OF8H ; TURN OFF UNUSED BITS
F026 B004F9 3179 POP DX ; STATUS_SET
F027 B004FA 317F MOV AL,DL ; RECOVER AL REG
F028 B004FB 3180 MOV AL,DL ; GET CHARACTER INTO AL
F029 B004FC 3181 XOR AH,48H ; FLIP A COUPLE OF BITS
F02A B004FD 3182 JMP B1 ; RETURN FROM ROUTINE

F030 B004FE 31A3 00H ; ------ INITIALIZE THE PRINTER PORT
F031 B004FF 31A0 PUSH AX ; SAVE AL
F032 B00500 31A8 INC DX ; POINT TO OUTPUT PORT
F033 B00501 31A9 INC DX
F034 B00502 3190 MOV AX,AL ; SET INIT LINE LOW
F035 B00503 3191 OUT DX,AL
F036 B00504 3192 MOV AX,1000
F037 B00505 3193 B9H ; INIT_LOOP
F038 B00506 3194 DEC AX ; LOOP FOR RESET TO TAKE
F039 B00507 3195 JNZ B9H ; INIT_LOOP
F03A B00508 3196 MOV AL,0CH ; NO INTERRUPTS, NON AUTO LF,
F03B B00509 3197 ; INIT HIGH
F03C B0050A 3198 OUT DX,AL
F03D B0050B 3199 JMP B6 ; PRT_STATUS_1

3200 B0050C 3204 ; VIDEO_ID ;
3201 ;
3202 ;
3203 ;
3204 ;
3205 ; THESE ROUTINES PROVIDE THE CRT INTERFACE ;
3206 ; THE FOLLOWING FUNCTIONS ARE PROVIDED: ;
3207 ; (AH)=0 SET MODE (AL) CONTAINS MODE VALUE ;
3208 ; (AL)=0 40X25 BW (POWER ON DEFAULT) ;
3209 ; (AL)=2 80X25 BW ;
3210 ; (AL)=3 80X25 COLOR ;
3211 ; (AL)=4 GRAPHICS MODES ;
3212 ; (AL)=4 320X200 COLOR ;
3213 ; (AL)=5 320X200 BW ;
3214 ; (AL)=6 640X200 BW ;
3215 ; CRT MODE=7 800X25 BW CARD (USED INTERNAL TO VIDEO ONLY) ;
3216 ; *** NOTE BW MODES OPERATE SAME AS COLOR MODES, BUT ;
3217 ; COLOR BURST IS NOT ENABLED ;
3218 ;
3219 ; (AH)=1 SET CURSOR TYPE ;
3220 ; (CH) = BITS 4-0 > START LINE FOR CURSOR ;
3221 ; ** HARDWARE WILL ALWAYS CAUSE BLIN ;
3222 ; ** SETTING BIT 5 OR 6 WILL CAUSE ERRATIC ;
3223 ; BLINKING OR NO CURSOR AT ALL ;
3224 ; (CL) = BITS 4-0 > END LINE FOR CURSOR ;
3225 ; (AH)=2 SET CURSOR POSITION ;
3226 ; (DH,DL) = ROW,COLUMN (0,0) IS UPPER LEFT ;
3227 ; (BH) = PAGE NUMBER (MUST BE 0 FOR GRAPHICS MODES) ;
3228 ; (AH)=3 READ CURSOR POSITION ;
3229 ; (BH) = PAGE NUMBER (MUST BE 0 FOR GRAPHICS MODES) ;
3230 ; ON EXIT (DH,DL) = ROW,COLUMN OF CURRENT CURSOR ;
3231 ; (CH,CL) = CURSOR MODE CURRENTLY SET ;
3232 ; (AH)=4 READ LIGHT PEN POSITION ;
3233 ; ON EXIT: ;
3234 ; (AH) = 0 -- LIGHT PEN SWITCH NOT DOWN/NOT TRIGGERED ;
3235 ; (AH) = 1 -- VALID LIGHT PEN VALUE IN REGISTERS ;
3236 ; (DH,DL) = ROW,COLUMN OF CHARACTER LP POSN ;
3237 ; (CH) = MASTER LINE (0-199) ;
3238 ; (DX) = PIXEL COLUMN (0-319,639) ;
3239 ; (AH)=5 SELECT ACTIVE DISPLAY PAGE (VALID ONLY FOR ALPHA MODES) ;
3240 ; (AL) = NEW PAGE VAL (0-7 FOR MODES 081, 0-3 FOR MODES 231) ;
3241 ; (AH)=6 SCROLL ACTIVE PAGE UP ;
3242 ; (AL) = NUMBER OF LINES, INPUT LINES BLANKED AT BOTTOM ;
3243 ; OF WINDOW ;
3244 ; AL = 0 MEANS BLANK ENTIRE WINDOW ;
3245 ; (CH,CL) = ROW,COLUMN OF UPPER LEFT CORNER OF SCROLL ;
3246 ; (DH,DL) = ROW,COLUMN OF LOWER RIGHT CORNER OF SCROLL ;
3247 ; (BH) = ATTRIBUTE TO BE USED ON BLANK LINE ;
3248 ; (AH)=7 SCROLL ACTIVE PAGE DOWN ;
3249 ; (AL) = NUMBER OF LINES, INPUT LINES BLANKED AT TOP ;
3250 ; OF WINDOW ;
3251 ; AL = 0 MEANS BLANK ENTIRE WINDOW ;
3252 ; (CH,CL) = ROW,COLUMN OF UPPER LEFT CORNER OF SCROLL ;
3253 ; (DH,DL) = ROW,COLUMN OF LOWER RIGHT CORNER OF SCROLL ;

3254 ;

A-46 System BIOS
Appendix A

System BIOS  A-47

LOC OBJ  LINE  SOURCE

3254  ;  (BH) = Attribute to be used on blank line
3255  ;
3256  ;  Character Handling Routines
3257  ;
3258  ;  (AH) = 8 READ ATTRIBUTE/CHARACTER AT CURRENT CURSOR POSITION
3259  ;  (BH) = DISPLAY PAGE (VALID FOR ALPHA Modes only)
3260  ;  ON EXIT:
3261  ;  (AL) = CHAR READ
3262  ;  (AH) = Attribute of Character READ (Alpha Modes only)
3263  ;  (AH) = 9 WRITE ATTRIBUTE/CHARACTER AT CURRENT CURSOR POSITION
3264  ;  (BH) = DISPLAY PAGE (VALID FOR ALPHA Modes only)
3265  ;  (CX) = COUNT OF CHARACTERS TO WRITE
3266  ;  (AL) = CHAR TO WRITE
3267  ;  (BL) = Attribute of Character (Alpha)/Color of Char (Graphics)
3268  ;  See note on write dot for bit 7 of BL = 1.
3269  ;  (AH) = 10 WRITE CHARACTER ONLY AT CURRENT CURSOR POSITION
3270  ;  (BH) = DISPLAY PAGE (VALID FOR ALPHA Modes only)
3271  ;  (CX) = COUNT OF CHARACTERS TO WRITE
3272  ;  (AL) = CHAR TO WRITE
3273  ;  For READ/WRITE Character Interface While in Graphics Mode, the
3274  ;  characters are formed from a Character Generator Image
3275  ;  maintained in the System ROM. Only the first 128 chars
3276  ;  are contained there. To READ/WRITE the second 128
3277  ;  chars, the user must initialize the pointer at
3278  ;  interrupt 1FH location 0000CH to point to the 1K Byte
3279  ;  table containing the code points for the second
3280  ;  128 chars (128-255).
3281  ;
3282  ;  For Write Character Interface in Graphics Mode, the replication:
3283  ;  Factor contained in (CX) on entry will produce valid
3284  ;  results only for Characters contained on the same row.
3285  ;  Continuation to succeeding lines will NOT produce
3286  ;  correctly.
3287  ;
3288  ;  Graphics Interface
3289  ;  (AH) = 11 SET COLOR PALETTE
3290  ;  (BH) = Palette color ID being set (0-127)
3291  ;  (BL) = Color value to be used with that color ID
3292  ;  Note: For the current color card, this entry point
3293  ;  has meaning only for 320x200 Graphics.
3294  ;  Color ID = 0 selects the background color (0-15);
3295  ;  Color ID = 1 selects the palette to be used:
3296  ;  0 = Green (1)/Red (2)/Yellow (3)
3297  ;  1 = Cyan (4)/Magenta (5)/White (6)
3298  ;  In 40x25 or 80x25 Alpha modes, the value set
3299  ;  for Palette Color 0 indicates the
3300  ;  Border color to be used (Values 0-31, 1
3301  ;  Where 16-31 select the high intensity
3302  ;  Background set.
3303  ;  (AH) = 12 WRITE DOT
3304  ;  (DX) = Row number
3305  ;  (CX) = Column number
3306  ;  (AL) = Color value
3307  ;  If bit 7 of AL = 1, then the color value is
3308  ;  Exclusive or'd with the current contents of
3309  ;  THE DOT
3310  ;  (AH) = 13 READ DOT
3311  ;  (DX) = Row number
3312  ;  (CX) = Column number
3313  ;  (AL) returns the dot read
3314  ;
3315  ;  ASCII Teletype Routine for Output
3316  ;
3317  ;  (AH) = 14 WRITE Teletype to Active Page
3318  ;  (AL) = Char to Write
3319  ;  (BL) = Foreground Color in Graphics Mode
3320  ;  Note -- Screen width is controlled by previous mode set
3321  ;
3322  ;  (AH) = 15 Current Video State
3323  ;  Returns the current video state
3324  ;  (AL) = Mode currently set (See AH=0 for explanation)
3325  ;  (AH) = Number of Character Columns on Screen
3326  ;  (BH) = Current Active Display Page
3327  ;
3328  ;  CS,55,55,55,55,55,55,0X,0X,0X,0X PRESERVED DURING CALL
3329  ;
3330  ;  All Others Destroyed
LOCATION OBJECT SOURCE

3331 ASSUME CS: CODE, DS: DATA, ES: VIDEO_RAM
3332 ORG OFFEH
3333 M1 LABEL WORD ; TABLE OF ROUTINES WITHIN VIDEO I/O
3334 DW OFFSET SET_MODE
3335 DW OFFSET SET_TYPE
3336 DW OFFSET SET_CPOS
3337 DW OFFSET READ_CURSOR
3338 DW OFFSET READ_UPLM
3339 DW OFFSET ACT_DISP_PAGE
3340 DW OFFSET SCROLL_UP
3341 DW OFFSET SCROLL_DOWN
3342 DW OFFSET READ_AC_CURRENT
3343 DW OFFSET WRITE_AC_CURRENT
3344 DW OFFSET SET_COLOR
3345 DW OFFSET WRITE_DOT
3346 DW OFFSET WRITE_DODT
3347 DW OFFSET WRITE_COLOR
3348 DW OFFSET VIDEO_STATE

HIL $EH
ORG OF06SH
VIDEO_IO PROC NEAR

3354 STI ; INTERRUPTS BACK ON
3355 CLO ; SET DIRECTION FORWARD
3356 PUSH ES ; SAVE SEGMENT REGISTERS
3357 PUSH DS
3358 PUSH DX
3359 PUSH CX
3360 PUSH BX
3361 PUSH SI
3362 PUSH DI
3363 PUSH AX ; SAVE AX VALUE
3364 MOV AL,AH ; GET INTO LOW BYTE
3365 XOR AH, AH ; ZERO TO HIGH BYTE
3366 SHL AX,1 ; LEFT FOR TABLE LOOKUP
3367 MOV SI,AX ; PUT INTO SI FOR BRANCH
3368 CMP AX,00H ; TEST FOR WITHIN RANGE
3369 JB M1 ; BRANCH AROUND BRANCH
3370 POP AX ; THROW AWAY THE PARAMETER
3371 JMP VIDEO_RETURN ; DO NOTHING IF NOT IN RANGE

HZ: CALL ODS
3372 E004H
3373 E005H
3374 MOV AX,0800H ; SEGMENT FOR COLOR CARD
3375 MOV DI,EQUIP_FLAG ; GET EQUIPMENT SETTING
3376 MOV AX,01H ; SEED FOR BW CARD?
3377 MOV DI,01H ; IS SETTING FOR BW CARD?
3378 MOV AH,080H ; SEGMENT FOR BM CARD
3379 MOV ES,AX ; SET UP TO POINT AT VIDEO_RAM AREAS
3380 POP AX ; RECOVER VALUE
3381 MOV AH,CRT_MODE ; GET CURRENT MODE INTO AH
3382 JMP WORD PTR CS:[SHOFFSET M1]

3383 MOV AX,ES:AX
3384 MOV AX,ES:AX
3385 MOV AX,ES:AX
3386 MOV AX,ES:AX
3387 MOV AX,ES:AX
3388 MOV AX,ES:AX
3389 MOV AX,ES:AX
3390 MOV AX,ES:AX
3391 MOV AX,ES:AX
3392 MOV AX,ES:AX
3393 MOV AX,ES:AX
3394 MOV AX,ES:AX
3395 MOV AX,ES:AX
3396 MOV AX,ES:AX
3397 MOV AX,ES:AX
3398 MOV AX,ES:AX
3399 MOV AX,ES:AX
3400 MOV AX,ES:AX
3401 MOV AX,ES:AX

SOURCE
ASSIGN CS: CODE, DS: DATA, ES: VIDEO_RAM
MI LABEL
LABEL WORD
TABLE OF ROUTINES WITHIN VIDEO I/O
OW OFFSET SET_HOOE
OW OFFSET SET_CTYPE
OW OFFSET SET_CPOS
OW OFFSET READ_CURSOR
OW OFFSET READ_UPLM
OW OFFSET ACT_DISP_PAGE
OW OFFSET SCROLL_UP
OW OFFSET SCROLL_DOWN
OW OFFSET READ_AC_CURRENT
OW OFFSET WRITE_AC_CURRENT
OW OFFSET SET_COLOR
OW OFFSET WRITE_DOT
OW OFFSET WRITE_DODT
OW OFFSET WRITE_COLOR
OW OFFSET VIDEO_STATE

VIDEO_IO PROC NEAR

st : CALL DDS
3375 MOV DI,EQUIP_FLAG
3376 MOV AX, 01H
3377 MOV DI, 01H
3378 MOV AH, 080H
3379 MOV AH, CRT_MODE
3380 JMP WORD PTR CS:[SHOFFSET M1]

3381 MOV AX, ES:AX
3382 MOV AX, ES:AX
3383 MOV AX, ES:AX
3384 MOV AX, ES:AX
3385 MOV AX, ES:AX
3386 MOV AX, ES:AX
3387 MOV AX, ES:AX
3388 MOV AX, ES:AX
3389 MOV AX, ES:AX
3390 MOV AX, ES:AX
3391 MOV AX, ES:AX
3392 MOV AX, ES:AX
3393 MOV AX, ES:AX
3394 MOV AX, ES:AX
3395 MOV AX, ES:AX
3396 MOV AX, ES:AX
3397 MOV AX, ES:AX
3398 MOV AX, ES:AX
3399 MOV AX, ES:AX
3400 MOV AX, ES:AX
3401 MOV AX, ES:AX

A-48 System BIOS
Table of Regen Lengths

<table>
<thead>
<tr>
<th>LOC OBJ</th>
<th>LINE</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>F0A0 1C</td>
<td>3402</td>
<td>DB 1CH,2,7,6,7</td>
</tr>
<tr>
<td>F0A0 02</td>
<td>F0A0 07</td>
<td>F0A0 06</td>
</tr>
<tr>
<td>F0B1 00</td>
<td>F0B2 00</td>
<td>F0B5 00</td>
</tr>
<tr>
<td>D010</td>
<td>3404</td>
<td>M4 EQU $-VIDEO_PARMS</td>
</tr>
<tr>
<td>F0B4 71</td>
<td>3405</td>
<td>DB 71H,50H,5AH,0AH,1FH,6,19H</td>
</tr>
<tr>
<td>F0B5 50</td>
<td>F0B6 5A</td>
<td>F0B7 5A</td>
</tr>
<tr>
<td>F0B9 06</td>
<td>F0BA 19</td>
<td></td>
</tr>
<tr>
<td>F0B0 1C</td>
<td>F0B0 02</td>
<td>F0B0 07</td>
</tr>
<tr>
<td>F0B0 09</td>
<td>F0B0 10</td>
<td>F0B0 11</td>
</tr>
<tr>
<td>F0B0 13</td>
<td>F0B0 14</td>
<td>F0B0 15</td>
</tr>
<tr>
<td>F0B0 17</td>
<td>F0B0 18</td>
<td>F0B0 19</td>
</tr>
<tr>
<td>F0B0 21</td>
<td>F0B0 22</td>
<td>F0B0 23</td>
</tr>
<tr>
<td>F0B0 25</td>
<td>F0B0 26</td>
<td>F0B0 27</td>
</tr>
<tr>
<td>F0B0 29</td>
<td>F0B0 2A</td>
<td>F0B0 2B</td>
</tr>
<tr>
<td>F0B0 2D</td>
<td>F0B0 2E</td>
<td>F0B0 2F</td>
</tr>
<tr>
<td>F0B0 31</td>
<td>F0B0 32</td>
<td>F0B0 33</td>
</tr>
<tr>
<td>F0B0 35</td>
<td>F0B0 36</td>
<td>F0B0 37</td>
</tr>
<tr>
<td>F0B0 39</td>
<td>F0B0 3A</td>
<td>F0B0 3B</td>
</tr>
<tr>
<td>F0B0 40</td>
<td>F0B0 41</td>
<td>F0B0 42</td>
</tr>
<tr>
<td>F0B0 44</td>
<td>F0B0 45</td>
<td>F0B0 46</td>
</tr>
<tr>
<td>F0B0 48</td>
<td>F0B0 49</td>
<td>F0B0 4A</td>
</tr>
<tr>
<td>F0B0 4C</td>
<td>F0B0 4D</td>
<td>F0B0 4E</td>
</tr>
<tr>
<td>F0B0 50</td>
<td>F0B0 51</td>
<td>F0B0 52</td>
</tr>
<tr>
<td>F0B0 54</td>
<td>F0B0 55</td>
<td>F0B0 56</td>
</tr>
<tr>
<td>F0B0 58</td>
<td>F0B0 59</td>
<td>F0B0 5A</td>
</tr>
<tr>
<td>F0B0 5C</td>
<td>F0B0 5D</td>
<td>F0B0 5E</td>
</tr>
<tr>
<td>F0B0 60</td>
<td>F0B0 61</td>
<td>F0B0 62</td>
</tr>
<tr>
<td>F0B0 64</td>
<td>F0B0 65</td>
<td>F0B0 66</td>
</tr>
<tr>
<td>F0B0 68</td>
<td>F0B0 69</td>
<td>F0B0 6A</td>
</tr>
</tbody>
</table>
F02 50
F03 50

;----- C_REG_TAB
F04 3420
F04 3421 M7 LABEL BYTE ; TABLE OF MODE SETS
F04 3422 DB 2CH,2E8H,2DH,2E6H,2AH,2EH,27H,26H
F04 3423 FOFA IE
F05 2B
F06 ID
F07 2A
F08 EA
F09 IE
F0A 1E
F0B 29

F0C 3432 SET_MODE PROC NEAR
F0C 3433 MOV DI,0304H ; ADDRESS OF COLOR CARD
F0C 3434 F0FF 8500
F0C 3435 MOV BL,0 ; MODE SET FOR COLOR CARD
F0C 3436 F101 03FF30
F0C 3437 CMP DI,30H ; IS BM CARD INSTALLED
F0C 3438 F106 7506
F0C 3439 JNE M8 ; OK WITH COLOR
F0C 343A F100 8204
F0C 343B MOV AL,7 ; INDICATE BM CARD MODE
F0C 343C F102 B204
F0C 343D MOV DL,0D4H ; ADDRESS OF BM CARD (384)
F0C 343E F10A FEC3
F0C 343F INC BL ; MODE SET FOR BM CARD
F0C 3440 F10C 5441
F0C 3441 M8:
F0C 3442 MOV AH,AL ; SAVE MODE IN AH
F0C 3443 F10E 24900
F0C 3444 MOV CR1_MODE,AL ; SAVE IN GLOBAL VARIABLE
F0C 3445 F115 8405300
F0C 3446 MOV ADDR_40053D,DX ; SAVE ADDRESS OF BASE
F0C 3447 F116 5466
F0C 3448 PUSH DS ; SAVE POINTER TO DATA SEGMENT
F0C 3449 F117 52
F0C 344A PUSH AX ; SAVE MODE
F0C 344B F118 83C204
F0C 344C ADD DX,4 ; POINT TO CONTROL REGISTER
F0C 344D F11B BAC3
F0C 344E MOV AL,AL ; GET MODE SET FOR CARD
F0C 344F F11D EE
F0C 3450 OUT DX,AL ; RESET VIDEO
F0C 3451 F11E 5A
F0C 3452 POP DX ; BACK TO BASE REGISTER
F0C 3453 F11F 28C0
F0C 3454 SUB AX,AX ; SET UP FOR ABS SEGMENT
F0C 3455 F121 EDD5
F0C 3456 MOV DS,AX ; ESTABLISH VECTOR TABLE ADDRESSING
F0C 3457 F123 5456
F0C 3458 ASSUME DS:ABS50
F0C 3459 F125 C51E7400
F0C 345A LDS BX,PARM_PTR ; GET POINTER TO VIDEO PARMs
F0C 345B F127 5B
F0C 345C POP AX ; RECOVER PARMs
F0C 345D F128 849000
F0C 345E MOV CX,MA ; LENGTH OF EACH ROW OF TABLE
F0C 345F F129 840C92
F0C 3460 CMP AH,2 ; DETERMINE WHICH ONE TO USE
F0C 3461 F12E 7210
F0C 3462 JC M9 ; MODE IS 0 OR 1
F0C 3463 F130 03D9
F0C 3464 ADD BX,CX ; MOVE TO NEXT ROW OF INIT TABLE
F0C 3465 F132 03FC04
F0C 3466 CMP AH,4 ; MODE IS 2 OR 3
F0C 3467 F135 7209
F0C 3468 JC M9 ; MODE IS 2 OR 3
F0C 3469 F137 83D9
F0C 346A ADD BX,CX ; MOVE TO GRAPHICS ROW OF INIT_TABLE
F0C 346B F139 03FC07
F0C 346C CMP AH,7 ; MODE IS 4,5, OR 6
F0C 346D F13C 7202
F0C 346E JC M9 ; MODE IS 4,5, OR 6
F0C 346F F13E 03D9
F0C 3470 ADD BX,CX ; MOVE TO BM CARD ROW OF INIT_TABLE
F0C 3471
F0C 3472 M9: OUT_INIT
F0C 3473 F140 50
F0C 3474 PUSH AX ; SAVE MODE IN AH
F0C 3475 F141 32E4
F0C 3476 XOR AH,AX ; AH WILL SERVE AS REGISTER
F0C 3477 F143 7575
F0C 3478 INC BX ; NUMBER DURING LOOP
F0C 3479
F0C 347A M10: INIT LOOP
F0C 347B F143 BAC4
F0C 347C MOV AL,AH ; GET 6451 Register Number
F0C 347D F145 EE
F0C 347E OUT DX,AL
F0C 347F F146 42
F0C 3480 INC DX ; POINT TO DATA PORT
F0C 3481 F147 F6C4
F0C 3482 INC AH ; NEXT REGISTER VALUE
F0C 3483 F148 4007
F0C 3484 MOV AL,(BX) ; GET TABLE VALUE
F0C 3485 F14A EE
F0C 3486 OUT DX,AL ; OUT TO CHIP
F0C 3487 F14C 43
F0C 3488 INC BX ; NEXT IN TABLE
F0C 3489 F14D 4A
F0C 348A DEC DX ; BACK TO POINTER REGISTER
F0C 348B F14E 2FF3
F0C 348C LOOP M10 ; DO THE WHOLE TABLE
F0C 348D F150 50
F0C 348E POP AX ; GET MODE BACK
F0C 348F F151 1F
F0C 3490 POP DS ; RECOVER SEGMENT VALUE
F0C 3491 F152 35FF
F0C 3492 ASSUME DS:DATA
F0C 3493 F153 3494
F0C 3494 XOR DI,DI ; SET UP POINTER FOR REGEN

A-50 System BIOS
AND SET UP OVERSCAN REGISTER

FL54 93E4600 3496 MOV CRT_START.OI ; START ADDRESS SAVED IN GLOBAL
FL55 936620000 3497 MOV ACTIVE_PAGE.O ; SET PAGE VALUE
FL5B 900020 3498 MOV CX,8192 ; NUMBER OF WORDS IN COLOR CARD
FL60 89F04 3499 CMP AH,4 ; TEST FOR GRAPHICS
FL63 70B0 3500 JC M12 ; NO GRAPHICS_INIT
FL65 89F07 3501 CMP AH,7 ; TEST FOR BM CARD
FL68 7044 3502 JE M11 ; BM_CARD_INIT
FL6A 33C0 3503 XOR AX,AX ; FILL FOR GRAPHICS MODE
FL6C B050 3504 JMP SHORT M13 ; CLEAR BUFFER
FL6E 3505 M12: ; BM_CARD_INIT
FL6E 8506 MOV CX,08H ; BUFFER SIZE ON BM CARD
FL70 3507 M12: ; NO_GRAPHICS_INIT
FL70 B2007 3508 MOV AX, 'F' + '2' + '5'6 ; FILL CHAR FOR ALPHA
FL75 3509 M13: ; CLEAR BUFFER
FL75 F3 3510 REP STOSW ; FILL THE REGEN BUFFER WITH BLANKS
FL74 AB 3511

3512 ;------ ENABLE VIDEO AND CORRECT PORT SETTING
3513
3514 FL75 C706600000706 3514 MOV CURSOR_MODE.607H ; SET CURRENT CURSOR MODE
3515 FL76 6000 3515 MOV AL,CRT_MODE ; GET THE MODE
3516 FL77 32E4 3516 XOR AH,AH ; INTO AX REGISTER
3517 FL80 88F0 3517 MOV SI,AX ; TABLE POINTER, INDEXED BY MODE
3518 FL82 80166300 3518 MOV DX,ADDR_6045 ; PREPARE TO OUTPUT TO
3519 FL84 3519 ; VIDEO ENABLE PORT
3520 FL86 89084CO0 3520 ADD DX,4
3521 FL89 8E4A04F0 3521 MOV AL,C.S:[SI]+OFFSET M7
3522 FL8E EE 3522 OUT DX,AL ; SET VIDEO ENABLE PORT
3523 FL90 A62500 3523 MOV CRT_MODE_SET_AL ; SAVE THAT VALUE
3524 3524
3525 FL92 8E4A4ECF0 3525 ;------ DETERMINE NUMBER OF COLUMNS, BOTH FOR ENTIRE DISPLAY
3526 FL97 32E6 3526 ; AND THE NUMBER TO BE USED FOR TTY INTERFACE
3527 FL99 A5A000 3527
3528 FL9C B160000 3528 MOV AL,C.S:[SI]+OFFSET M6
3529 FL9E 32E6 3529 XOR AH,AH
3530 FL9F A5A000 3530 MOV CRT_COLS.AX ; NUMBER OF COLUMNS IN THIS SCREEN
3531 3531
3532 FL9C B160000 3532 ;------ SET CURSOR POSITIONS
3533 FL9D 32E4 3533 AND SI,0EH ; WORD OFFSET INTO CLEAR LENGTH TABLE
3534 FLAD 8E4DCE4F0 3534 MOV CX,C.S:[SI]+OFFSET M5 ; LENGTH TO CLEAR
3535 FLA5 89E6C00 3535 MOV DX,ADDR_6045 ; SAVE LENGTH OF CRT -- NOT USED FOR BM
3536 FLA9 89O0800 3536 MOV CX,8 ; CLEAR ALL CURSOR POSITIONS
3537 FLAC EF5000 3537 MOV DI,OFFSET_CURSOR_POSN
3538 FLAF 1E 3538 PUSH DS ; ESTABLISH SEGMENT
3539 F180 07 3539 POP ES ; ADDRESSING
3540 F1B1 33C0 3540 XOR AX,AX
3541 F183 F3 3541 REP STOSW ; FILL WITH ZEROS
3542 F184 AB 3542

3543 3543
3544 FLB5 42 3544 ;----- SET UP OVERSCAN REGISTER
3545 3545
3546 FLB6 03D0 3546 INC DX ; SET OVERSCAN PORT TO A DEFAULT
3547 FLB6 03D0 3547 MOV AL,38H ; VALUE OF 38H FOR ALL MODES
3548 3548
3549 FLBD 803F900006 3549 CMP CRT_MODE.6 ; SEE IF THE MODE IS 640X200 BM
3550 FLBD 7502 3550 JNZ M14 ; IF IT ISN'T 640X200, THEN GOTO REGULAR
3551 FLBF 803F 3551 MOV AL,3FH ; IF IT IS 640X200, THEN PUT IN 3FH
3552 F1C1 3552 M14:
3553 F1C1 IE 3553 OUT DX,AL ; OUTPUT THE CORRECT VALUE TO 39 PORT
3554 F1CC A96600 3554 MOV CRT_PALETTE.AL ; SAVE THE VALUE FOR FUTURE USE
3555 3555
3556 FLCS 3556 ;------ NORMAL RETURN FROM ALL VIDEO RETURNS
3557 FLCS 3557
3558 FLCS 5F 3558 VIDEO_RETURN:
3559 FLCS 5E 3559 POP DI
3560 FLCS 5E 3560 POP SI
3561 FLCS 5B 3561 POP BX
3562 F1CB 3562 M15: ; VIDEO_RETURN_C
3563 FLCS 59 3563 POP CX
3564 FLCS 5A 3564 POP DX
3565 FLCA 1F 3565 POP DS
3566 FLCC 07 3566 POP ES ; RECOVER SEGMENTS
3567 F1CC CF 3567 INT 21H ; ALL DONE
3568 3568 SET_MODE ENDP
3569 3569 ;---- SET_TYPE
3570 3570

Appendix A

System BIOS A-51
This route sets the cursor value.

Input.

If (CX) has cursor value CH-start line, CL-stop line.

Output.

None.

Routine outputs the CX register to the 6845 register.

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.

Routine sets the current cursor.

Establish loop count.

Word offset.

Use index register.

Save pointer.

CMP ACTIVE_PAGE, BH.

JNZ M17.

MOV AX, DX.

Get row/column to AX.

Call M18.

Cursor Set.

JMP VIDEO_RETURN.

Set cursor position, AX has row/column for cursor.

Call position.

Determine location in regen buffer.

MOV CX, AX.

Add CX, CERT_START.

Divide by 2 for char only count.

MOV AH, 14.

Call M16.

Output the value to the 6845.

RET.

M16 END.

Routine sets the active display page, allowing the.

Full use of the RAM set aside for the video attachment.

Input.

AL has the new active display page.

Output.

The 6845 is reset to display that page.

Routine sets the active page value.

ACT_DISP_PAGE PROC NEAR

ACT_DISP_PAGE PROC NEAR

System BIOS
THIS ROUTINE READS COLOR, AND IF BH=1, THE PALETTE SELECTION IS MADE
BASED ON THE LOW BIT OF BL.

IF BH=0, THE BACKGROUND COLOR VALUE IS SET
FROM THE LOW BITS OF BL (0-31).

IF BH=1, THE PALETTE SELECTION IS MADE

BASED ON THE LOW BIT OF BL:

THE COLOR SELECTION IS UPDATED

THE COLOR SELECTION IS UPDATED

HANDLE COLOR 0 BY SETTING THE BACKGROUND COLOR

AND AL.EDM

TURN OFF LOW 5 BITS OF CURRENT

AL.EDF

TURN OFF HIGH 3 BITS OF INPUT VALUE

AL.EL

PUT VALUE INTO REGISTER

AL.EL

OUTPUT THE PALETTE

AL.ED

OUTPUT COLOR SELECTION TO 3D9 PORT

AL.ED

SAVE THE COLOR VALUE

JMP VIDEO_RETURN
F26A 3725 M29:
F26A 240F 3726 AND AL,0DFH ; TURN OFF PALETTE SELECT BIT
F26C DDEB 3727 SHR BL,1 ; TEST THE LOW ORDER BIT OF BL
F26E 73F3 3728 JNC M19 ; ALREADY DONE
F270 0C20 3729 OR AL,20H ; TURN ON PALETTE SELECT BIT
F272 EBEF 3730 JMP M19 ; GO DO IT
F271 SET_COLOR ENDP
F272 3731
F273 3732 :------------------------------------------
F274 3733 ; VIDEO STATE :
F275 3734 ; RETURNS THE CURRENT VIDEO STATE IN AX :
F276 3735 ; AH = NUMBER OF COLUMNS ON THE SCREEN :
F277 3736 ; AL = CURRENT VIDEO MODE :
F278 3737 ; BH = CURRENT ACTIVE PAGE :
F279 3738 ;------------------------------------------
F27A 3739 VIDEO_STATE PROC NEAR
F27B 3740 MOV AH,BYTE PTR CRT_COLS ; GET NUMBER OF COLUMNS
F27D 3741 MOV AL,CRT_MODE ; CURRENT MODE
F27F 3742 MOV BH,ACTIVE_PAGE ; GET CURRENT ACTIVE PAGE
F281 3743 POP DI ; RECOVER REGISTERS
F283 3744 POP SI
F285 3745 POP CX ; DISCARD SAVED BX
F287 E943FF 3746 JMP M15 ; RETURN TO CALLER
F288 3747 VIDEO_STATE ENDP
F289 3748
F28A 3749 ; POSITION :
F28B 3750 ; THIS SERVICE ROUTINE CALCULATES THE REGEN :
F28D 3751 ; BUFFER ADDRESS OF A CHARACTER IN THE ALPHA MODE :
F28F 3752 ; INPUT :
F291 3753 ; AX = ROW, COLUMN POSITION :
F293 3754 ; OUTPUT :
F295 3755 ; AX = OFFSET OF CHAR POSITION IN REGEN BUFFER
F297 3756 ;------------------------------------------
F298 3757 POSITION PROC NEAR
F299 3758 PUSH BX ; SAVE REGISTER
F29B 3759 MOV BX,AX
F29D 3760 MOV AL,AL ; ROWS TO AL
F29F F2644A00 3761 MUL BYTE PTR CRT_COLS ; DETERMINE BYTES TO ROW
F2A1 3762 XOR BH,BH
F2A3 3763 ADD AX,BX ; ADD IN COLUMN VALUE
F2A5 3764 SAL AX,1 ; * 2 FOR ATTRIBUTE BYTES
F2A7 3765 POP BX
F2A9 3766 BET
F2AA 3767 POSITION ENDP
F2AB 3768
F2AC 3769 ; SCROLL UP :
F2AD 3770 ; THIS ROUTINE MOVES A BLOCK OF CHARACTERS UP :
F2AE 3771 ; ON THE SCREEN :
F2AF 3772 ; INPUT :
F2B1 3773 ; (AH) = CURRENT CRT MODE :
F2B3 3774 ; (AL) = NUMBER OF ROWS TO SCROLL :
F2B5 3775 ; (CX) = ROW/COLUMN OF UPPER LEFT CORNER :
F2B7 3776 ; (DX) = ROW/COLUMN OF LOWER RIGHT CORNER :
F2B9 3777 ; (BH) = ATTRIBUTE TO BE USED ON BLANKED LINE :
F2BB 3778 ; (DS) = DATA SEGMENT :
F2BD 3779 ; (ES) = REGEN BUFFER SEGMENT :
F2BF 3780 ; OUTPUT :
F2C1 3781 ; NONE -- THE REGEN BUFFER IS MODIFIED
F2C3 3782 ;------------------------------------------
F2C4 3783 ASSUME CS:CODE,DS:DATA,ES:DATA
F2C6 3784 SCROLL_UP PROC NEAR
F2C8 3785 MOV BL,AL ; SAVE LINE COUNT IN BL
F2CA 3786 CMP AH,4 ; TEST FOR GRAPHICS MODE
F2CB 7200 3787 JC N1 ; HANDLE SEPARATELY
F2CD 3788 CMP AH,7 ; TEST FOR BM CARD
F2CF 7403 3789 JE N1
F2D1 E9F001 3790 JMP GRAPHICS_UP
F2D3 3791 N1:
F2D5 3792 PUSH BX ; SAVE FILL ATTRIBUTE IN BH
F2D7 3793 MOV AX,CX ; UPPER LEFT POSITION
F2D9 E83700 3794 CALL SCROLL_POSITION ; DO SETUP FOR SCROLL
F2DA 7431 3795 JZ N7 ; BLANK_FIELD
F2DB 03F0 3796 ADD SI,AX ; FROM ADDRESS
F2DF B3E06 3797 MOV AH,0H ; # ROWS IN BLOCK
F2E1 2A33 3798 SUB AH,BL ; # ROWS TO BE MOVED
F2E3 3799 N2: ; ROW_LOOP
F2E5 E87200 3800 CALL N10 ; MOVE ONE ROW
F2E7 03F5 3801 ADD SI,SP

A-54 System BIOS
LOC OBJ
LINE
SOURCE
F2BA 03FD
3002
ADD DI,DP
; POINT TO NEXT LINE IN BLOCK
F2BA FECC
3003
DEC AH
; COUNT OF LINES TO MOVE
F2B0 75F5
3004
JNZ N2
; ROM_LOOP
F2BE
3005
N3:
; CLEAR_ENTRY
F2BE SA
3006
POP AX
; RECOVER ATTRIBUTE IN AH
F2BF 80D0
3007
MOV AL,;'14
; FILL WITH BLANKS
F2CI
3008
N4:
; CLEAR_LOOP
F2CI E60000
3009
CALL HII
; CLEAR THE ROW
F2C4 03FD
3010
ADD DI,DP
; POINT TO NEXT LINE
F2CA FEDB
3011
DEC DL
; COUNTER OF LINES TO SCROLL
F2CA 75F7
3012
JNZ N4
; CLEAR_LOOP
F2CA
3013
N5:
; SCROLL_END
F2CA E80C07
3014
CALL DDS
F2CD 00349000001
3015
CMP CRT_MODE,1
; IS THIS THE BLACK AND WHITE CARD
F2DE 7947
3016
JE N6
; IF SO, SKIP THE MODE RESET
F2DF 405000
3017
MOV AL,CRT_MODE_SET
; GET THE VALUE OF THE MODE SET
F2D7 BAO003
3018
MOV DX,030DH
; ALWAYS SET COLOR CARD PORT
F2DA EE
3019
OUT DX,AL
F2DB
3020
N6:
; VIDEO_RET_HERE
F2DB E9F7FE
3021
JMP VIDEO_RETURN
F2DE
3022
N7:
; BLANK_FIELD
F2DE 6A0E
3023
MOV BL,DL
; GET ROW COUNT
F2E0 EB0C
3024
JMP N3
; GO CLEAR THAT AREA
F2E5 SCROLL_UP
3025
ENDP
3026
; ----- HANDLE COMMON SCROLL SET UP HERE
F2E2
3029
SCROLL_POSITION PROC NEAR
F2E2 00349000002
3030
CMP CRT_MODE,2
; TEST FOR SPECIAL CASE HERE
F2E7 7110
3031
JB N9
; HAVE TO HANDLE 80X2S SEPARATELY
F2E9 00349000003
3032
CMP CRT_MODE,3
F2EE 7711
3033
JA N9
3034
F2E5 3035
; ----- 80X2S COLOR CARD SCROLL
3036
F2F0 52
3037
PUSH DX
F2F1 BADA03
3038
MOV DX,30AH
; GUARANTEED TO BE COLOR CARD HERE
F2F4 50
3039
PUSH AX
F2F5
3040
N8:
; WAIT_DISP_ENABLE
F2F5 EC
3041
IN AL,DX
; GET PORT
F2F6 A808
3042
TEST AL,8
; WAIT FOR VERTICAL RETRACE
F2F8 74FF
3043
JZ N8
; WAIT_DISP_ENABLE
F2FA 0D25
3044
MOV AL,ESH
F2FC B0DB
3045
MOV DL,00DH
; DX=3DB
F2FE EE
3046
OUT DX,AL
; TURN OFF VIDEO
F2FF 5A
3047
POP AX
; DURING VERTICAL RETRACE
F300 5A
3048
POP DP
F301
3049
N9:
; CALL POSITION
F301 E801FF
3050
CALL POSITION
; CONVERT TO REGEN POINTER
F304 03064E00
3051
ADD AX,CRT_START
; OFFSET OF ACTIVE PAGE
F306 08F0
3052
MOV SI,AX
; TO ADDRESS FOR SCROLL
F308 08F0
3053
MOV SI,AX
; FROM ADDRESS FOR SCROLL
F30C 20D1
3054
SUB DX,CX
; DX = # RNS, WOLS IN BLOCK
F30E F6C6
3055
INC DI
F310 F6C2
3056
INC DL
; INCREMENT FOR # ORIGIN
F312 3EDD
3057
XOR CH,CH
; SET HIGH BYTE OF COUNT TO ZERO
F314 0B2E4A00
3058
MOV BP,CRT_COLS
; GET NUMBER OF COLUMNS IN DISPLAY
F316 03ED
3059
ADD BP,BP
; TIMES 2 FOR ATTRIBUTE BYTE
F31A 8C03
3060
MOV AL,BL
; GET LINE COUNT
F31C FD664A00
3061
MUL BYTE PTR CRT_COLS
; DETERMINE OFFSET FROM ADDRESS
F320 03C0
3062
ADD AX,AX
; #2 FOR ATTRIBUTE BYTE
F322 06
3063
PUSH ES
; ESTABLISH ADDRESSING TO REGEN BUFFER
F323 1F
3064
POP DS
; FOR BOTH POINTERS
F324 00F000
3065
CMP BL,0
; 0 SCROLL MEANS BLANK FIELD
F327 C3
3066
RET
; RETURN WITH FLAGS SET
3067
SCROLL_POSITION ENDP
3068
; ----- MOVE ROW
3069
F328
3070
F328 NA
3071
PROC NEAR
; GET # OF COLS TO MOVE
F328 BACA
3072
MOV CL,DL
F32A S6
3073
PUSH SI
F32B 57
3074
PUSH DI
; SAVE START ADDRESS
F32C F3
3075
REP MOVSW
; MOVE THAT LINE ON SCREEN
F32D A5
3076
F32E SF
3077
PUSH SI
; RECOVER ADDRESSES
System BIOS A-55
LOC OBJ  LINE  SOURCE

F330 C3  3878  RET
3879  N10  ENDP
3880
3881  |----- CLEAR_ROW
3882
F331  3883  N11  PROC NEAR
F331  3884  6ACA  MOV  CL,DL  ; GET 8 COLUMNS TO CLEAR
F333  3885  5SF  PUSH  DI
F334  3886  F3  REP  STOSW  ; STORE THE FILL CHARACTER
F335  3887  A0
F336  3888  SF  POP  DI
F337  3889  C3
F338  3890  N11  ENDP
3891  |-----------------------------
3892  | SCROLL_DOWN
3893  | ; THIS ROUTINE MOVES THE CHARACTERS WITHIN A
3894  | DEFINED BLOCK DOWN ON THE SCREEN, FILLING THE
3895  | TOP LINES WITH A DEFINED CHARACTER
3896  | ; INPUT
3897  | ; (AL) = CURRENT CRT MODE
3898  | ; (AH) = NUMBER OF LINES TO SCROLL
3899  | ; (CX) = UPPER LEFT CORNER OF REGION
3890  | ; (DX) = LOWER RIGHT CORNER OF REGION
3891  | ; (BH) = FILL CHARACTER
3892  | ; (DS) = DATA SEGMENT
3893  | ; (ES) = REGEN SEGMENT
3894  | ; OUTPUT
3895
F338  3896  3897  SCROLL_DOWN  PROC NEAR
F339  3898  FD
3899  3890  6ADD  MOV  BL,AL  ; LINE COUNT TO BL
F33B  3891  0FC04  CMP  AH,4  ; TEST FOR GRAPHICS
F33E  3892  72BA  JC  N12
F340  3893  80FC07  CMP  AH,7  ; TEST FOR BM CARD
F343  3894  7403  JZ  N12
F345  3895  EP4601  JE  N12
F348  3896  N12:
F349  3897  53  PUSH  BX  ; CONTINUE_DOWN
F34C  3898  8BC2  MOV  AX,DX  ; LOWER RIGHT CORNER
F34F  3899  894FF  CALL  SCROLL_POSITION  ; GET REGEN LOCATION
F34E  3900  7420  JZ  N16
F350  3901  2DF0  SUB  SI,AX  ; SI IS FROM ADDRESS
F353  3902  6AF6  MOV  AH,DX  ; GET TOTAL # ROWS
F355  3903  2AE3  SUB  AH,BL  ; COUNT TO MOVE IN SCROLL
F356  3904
F357  3905  2AE6  N13:
F358  3906  SCROLL_DOWN
3907  STD  ; DIRECTION FOR SCROLL DOWN
3908  6ADD  MOV  BL,AL  ; LINE COUNT TO BL
3909  0FC04  CMP  AH,4  ; TEST FOR GRAPHICS
3910  72BA  JC  N12
3911  80FC07  CMP  AH,7  ; TEST FOR BM CARD
3912  7403  JZ  N12
3913  EP4601  JE  N12
F348  3914  N12:
F349  3915  53  PUSH  BX  ; CONTINUE_DOWN
F34C  3916  8BC2  MOV  AX,DX  ; LOWER RIGHT CORNER
F34F  3917  894FF  CALL  SCROLL_POSITION  ; GET REGEN LOCATION
F34E  3918  7420  JZ  N16
F350  3919  2DF0  SUB  SI,AX  ; SI IS FROM ADDRESS
F353  3920  6AF6  MOV  AH,DX  ; GET TOTAL # ROWS
F355  3921  2AE3  SUB  AH,BL  ; COUNT TO MOVE IN SCROLL
F356  3922
F357  3923  2AE6  N13:
F358  3924  SCROLL_DOWN
3925  STD  ; DIRECTION FOR SCROLL DOWN
3926  6ADD  MOV  BL,AL  ; LINE COUNT TO BL
3927  0FC04  CMP  AH,4  ; TEST FOR GRAPHICS
3928  72BA  JC  N12
3929  80FC07  CMP  AH,7  ; TEST FOR BM CARD
3930  7403  JZ  N12
3931  EP4601  JE  N12
F348  3932  N12:
F349  3933  53  PUSH  BX  ; CONTINUE_DOWN
F34C  3934  8BC2  MOV  AX,DX  ; LOWER RIGHT CORNER
F34F  3935  894FF  CALL  SCROLL_POSITION  ; GET REGEN LOCATION
F34E  3936  7420  JZ  N16
F350  3937  2DF0  SUB  SI,AX  ; SI IS FROM ADDRESS
F353  3938  6AF6  MOV  AH,DX  ; GET TOTAL # ROWS
F355  3939  2AE3  SUB  AH,BL  ; COUNT TO MOVE IN SCROLL
F356  3940  SCROLL_DOWN
3941  ENDP
3942
F330  3943  FD
3944  3897  SCROLL_DOWN  PROC NEAR
3945  STD  ; DIRECTION FOR SCROLL DOWN
3946  6ADD  MOV  BL,AL  ; LINE COUNT TO BL
3947  0FC04  CMP  AH,4  ; TEST FOR GRAPHICS
3948  72BA  JC  N12
3949  80FC07  CMP  AH,7  ; TEST FOR BM CARD
3950  7403  JZ  N12
3951  EP4601  JE  N12
F348  3952  N12:
F349  3953  53  PUSH  BX  ; CONTINUE_DOWN
F34C  3954  8BC2  MOV  AX,DX  ; LOWER RIGHT CORNER
F34F  3955  894FF  CALL  SCROLL_POSITION  ; GET REGEN LOCATION
F34E  3956  7420  JZ  N16
F350  3957  2DF0  SUB  SI,AX  ; SI IS FROM ADDRESS
F353  3958  6AF6  MOV  AH,DX  ; GET TOTAL # ROWS
F355  3959  2AE3  SUB  AH,BL  ; COUNT TO MOVE IN SCROLL
F356  3960  SCROLL_DOWN
3961  ENDP
3962
F330  3963  FD
3964  3897  SCROLL_DOWN  PROC NEAR
3965  STD  ; DIRECTION FOR SCROLL DOWN
3966  6ADD  MOV  BL,AL  ; LINE COUNT TO BL
3967  0FC04  CMP  AH,4  ; TEST FOR GRAPHICS
3968  72BA  JC  N12
3969  80FC07  CMP  AH,7  ; TEST FOR BM CARD
3970  7403  JZ  N12
3971  EP4601  JE  N12
F348  3972  N12:
F349  3973  53  PUSH  BX  ; CONTINUE_DOWN
F34C  3974  8BC2  MOV  AX,DX  ; LOWER RIGHT CORNER
F34F  3975  894FF  CALL  SCROLL_POSITION  ; GET REGEN LOCATION
F34E  3976  7420  JZ  N16
F350  3977  2DF0  SUB  SI,AX  ; SI IS FROM ADDRESS
F353  3978  6AF6  MOV  AH,DX  ; GET TOTAL # ROWS
F355  3979  2AE3  SUB  AH,BL  ; COUNT TO MOVE IN SCROLL
F356  3980  SCROLL_DOWN
3981  ENDP
3982
F330  3983  FD
3984  3897  SCROLL_DOWN  PROC NEAR
3985  STD  ; DIRECTION FOR SCROLL DOWN
3986  6ADD  MOV  BL,AL  ; LINE COUNT TO BL
3987  0FC04  CMP  AH,4  ; TEST FOR GRAPHICS
3988  72BA  JC  N12
3989  80FC07  CMP  AH,7  ; TEST FOR BM CARD
3990  7403  JZ  N12
3991  EP4601  JE  N12
F348  3992  N12:
F349  3993  53  PUSH  BX  ; CONTINUE_DOWN
F34C  3994  8BC2  MOV  AX,DX  ; LOWER RIGHT CORNER
F34F  3995  894FF  CALL  SCROLL_POSITION  ; GET REGEN LOCATION
F34E  3996  7420  JZ  N16
F350  3997  2DF0  SUB  SI,AX  ; SI IS FROM ADDRESS
F353  3998  6AF6  MOV  AH,DX  ; GET TOTAL # ROWS
F355  3999  2AE3  SUB  AH,BL  ; COUNT TO MOVE IN SCROLL
F356  3999  SCROLL_DOWN
3999  ENDP
F374 3955 ASSUME CS:CODE, DS:DATA, ES:DATA

F374 00FC04 3957 CMP AH, 4  ; IS THIS GRAPHICS?
F377 7268 3958 JC P1  ; IS THIS BM CARD?
F379 80FC07 3959 CMP AH, 7  ; IS THIS BM CARD?
F37C 7403 3960 JE P1  ; IS THIS BM CARD?
F37E 69A002 3961 JMP GRAPHICS_READ
F381 081000 3962 P1: CALL FIND_POSITION
F384 08F3 3963 MOV SI, BX  ; ESTABLISH ADDRESSING IN SI
F386 0B163300 3964  ;----- WAIT FOR HORIZONTAL RETRACE
F388 03C206 3965 MOV DX, ADDR_6445  ; GET BASE ADDRESS
F38D 06 3966 ADD DX, 6  ; POINT AT STATUS PORT
F390 1F 3967 PUSH ES  ; GET SEGMENT FOR QUICK ACCESS
F393 E94802 3968 JMP GRAPHICS_READ
F396 0E0000 3969 PI: , READ_AT_CONTINUE
F399 EC 3970 CALL FIND_POSITION
F3A2 8BFB 3971 MOV DI, BX  ; ESTABLISH ADDRESSING IN DI REG
F3AB C3 3972 POP DI  ; GET STATUS
F3B1 031E400 3973 ADD BX, CSET_LEN  ; LENGTH OF BUFFER
F3BE 030E400 3974 ADD BX, CR len  ; COUNT OF CHARACTERS TO WRITE
F3C1 E800 3975 CALL FIND_POSITION  ; DETERMINE LOCATION IN REGEN
F3C4 030E 3976 ADD BX, AX  ; ADD TO START OF REGEN
F3C7 C3 3977 RET

Appendix A

System BIOS A-57
LOC OBJ  LINE  SOURCE

F301  4031  P7:  ; WRITE_LOOP

F302  4032

F303  4033  ;----- WAIT FOR HORIZONTAL RETRACE

F304  4034

F305  88166300  4035  MOV AX,ADDR_6845  ; GET BASE ADDRESS

F306  83C206  4036  ADD DX,6  ; POINT AT STATUS PORT

F307  0B  4037  P8:  

F308  4038  IN AL,DX  ; GET STATUS

F309  4039  TEST AL,1  ; IS IT LOW

F310  75FB  4040  JNZ P8  ; WAIT UNTIL IT IS

F311  FA  4041  CLI  ; NO MORE INTERRUPTS

F312  4042  P9:  

F313  4043  IN AL,DX  ; GET STATUS

F314  4044  TEST AL,1  ; IS IT HIGH

F315  74FB  4045  JZ P9  ; WAIT UNTIL IT IS

F316  60C3  4046  MOV AX,BX  ; RECOVER THE CHAR/ATTR

F317  AB  4047  STOSB  ; PUT THE CHAR/ATTR

F318  FB  4048  STI  ; INTERRUPTS BACK ON

F319  E2EB  4049  LOOP P7  ; AS MANY TIMES AS REQUESTED

F320  EVD9FD  4050  JMP VIDEO_RETURN

F321  4051  WRITE_C_CURRENT ENDP

F322  4052  ;-------------------------------

F323  4053  ; WRITE_C_CURRENT PROC NEAR

F324  4054  F3EC  4055  CMP AH,4  ; IS THIS GRAPHICS

F325  80FC04  4056  JC P10  

F326  720B  4057  JE P10  

F327  F3F1  4058  CMP AH,7  ; IS THIS BW CARD

F328  80FC07  4059  JNC P10  

F329  7051  4060  JZ P10  

F330  F3F4  4061  JE P10  

F331  7651  4062  CMP AX,0000  ; IS CHAR/ATTR

F332  F3F6  4063  JNC P10  

F333  07F01  4064  JMP GRAPHICS_WRITE

F334  4065

F335  F3F9  4066  ;-------------------------------

F336  4067

F337  4068  F3EC  4069  CMP AH,6  ; IS THIS GRAPHICS

F338  80FC04  4070  JC P11  

F339  720B  4071  JE P11  

F340  F3F1  4072  CMP AH,7  ; IS THIS BW CARD

F341  80FC07  4073  JNC P11  

F342  7051  4074  JZ P11  

F343  F3F4  4075  JE P11  

F344  7651  4076  CMP AX,0000  ; IS CHAR/ATTR

F345  F3F6  4077  JNC P11  

F346  07F01  4078  JMP GRAPHICS_WRITE

F347  4079

F348  F3F9  4080  P10:  

F349  50  4081  ; WRITE_LOOP

F350  4082

F351  F402  4083  ;----- WAIT FOR HORIZONTAL RETRACE

F352  80166300  4084  MOV DX,ADDR_6845  ; GET BASE ADDRESS

F353  83C206  4085  ADD DX,6  ; POINT AT STATUS PORT

F354  0B  4086  P12:  

F355  4087  IN AL,DX  ; GET STATUS

F356  4088  TEST AL,1  ; IS IT LOW

F357  75FB  4089  JNZ P12  ; WAIT UNTIL IT IS

F358  FA  4090  CLI  ; NO MORE INTERRUPTS

F359  4091  P13:  

F360  4092  IN AL,DX  ; GET STATUS

F361  4093  TEST AL,1  ; IS IT HIGH

F362  74FB  4094  JZ P13  ; WAIT UNTIL IT IS

F363  74C3  4095  MOV AL,0BL  ; RECOVER CHAR

F364  4096  STOSB  ; RECOVER THE CHAR/ATTR

F365  4097  STI  ; INTERRUPTS BACK ON

F366  47  4098  INC DI  ; BUMP POINTER PAST ATTRIBUTE

F367  E2EB  4099  LOOP P11  ; AS MANY TIMES AS REQUESTED

F368  409A  409A  JMP VIDEO_RETURN

F369  409B  ;-------------------------------

F370  409C

F371  409D  ; READ DOT -- WRITE DOT

F372  409E  ; THESE ROUTINES WILL WRITE A DOT, OR READ THE DOT AT

F373  409F  ; THE INDICATED LOCATION

F374  40A0  ; ENTRY --

F375  40A1  ; DX = ROM (0-199) (THE ACTUAL VALUE DEPENDS ON THE MODE)
4108  CX = COLUMN (0-639) (THE VALUES ARE NOT RANGE CHECKED)

4109  AL = DOT VALUE TO WRITE (1, 2 OR 4 BITS DEPENDING ON MODE)

4110  REQ'D FOR DOT VALUE ONLY, RIGHT JUSTIFIED

4111  BIT 7 OF AL=1 INDICATES XOR THE VALUE INTO THE LOCATION

4112  DS = DATA SEGMENT

4113  ES = SEGMENT

4114  

4115  EXIT

4116  AL = DOT VALUE READ, RIGHT JUSTIFIED, READ ONLY

4117  

---

4118  ASSUME CS:CODE, DS:DATA, ES:DATA

4119  READ_DOT  PROC NEAR

4120  CALL R3          ; DETERMINE BYTE POSITION OF DOT

4121  MOV AL:[SI]      ; GET THE BYTE

4122  AND AL,AH        ; MASK OFF THE OTHER BITS IN THE BYTE

4123  SHL AL,CL       ; LEFT JUSTIFY THE VALUE

4124  MOV CL,DL        ; GET NUMBER OF BITS IN RESULT

4125  ROL AL,CL       ; RIGHT JUSTIFY THE RESULT

4126  JMP VIDEO_RETURN ; RETURN FROM VIDEO IO

---

4127  READ_DOT  ENDP

---

4128  

Appendix A

System BIOS  A-59
LOC OBJ

4185 | I-----------
4186 | SET UP THE REGISTERS ACCORDING TO THE MODE
4187 | CH = MASK FOR LOW OF COLUMN ADDRESS (7/3 FOR HIGH/RED) :
4188 | CL = # OF ADDRESS BITS IN COLUMN VALUE (3/2 FOR H/M) :
4189 | BL = MASK TO SELECT BITS FROM POINTED BYTE (80H/CON FOR H/M) :
4190 | BH = NUMBER OF VALID BITS IN POINTED BYTE (.1/2 FOR H/M) :
4191 |
4192 |
4193 | MOV BX,2BH
4194 | MOV CX,302H ; SET PARMS FOR MED RES
4195 | CMP CRT_MODE,6
4196 | JC RS ; HANDLE IF MED AREA
4197 | MOV BX,1BH ; USE
4198 | MOV CX,703H ; SET PARMS FOR HIGH RES
4199 |
4200 | I----- DETERMINE BIT OFFSET IN BYTE FROM COLUMN MASK
4201 |
4202 | R5:
4203 | AND CH,DL ; ADDRESS OF PEL WITHIN BYTE TO CH
4204 |
4205 | I----- DETERMINE BYTE OFFSET FOR THIS LOCATION IN COLUMN
4206 |
4207 | SHR DX,CL ; SHIFT BY CORRECT AMOUNT
4208 | ADD SI,DX ; INCREMENT THE POINTER
4209 | MOV DH,BH ; GET THE # OF BITS IN RESULT TO DH
4210 |
4211 | I----- MULTIPLY BH (VALID BITS IN BYTE) BY CH (BIT OFFSET)
4212 |
4213 | SUB CL,CL ; ZERO INTO STORAGE LOCATION
4214 | R6:
4215 | MOV AL,1 ; LEFT JUSTIFY THE VALUE
4216 | IN AL (FOR WRITE)
4217 | ADD CL,CH ; ADD IN THE BIT OFFSET VALUE
4218 | DEC BH ; LOOP CONTROL
4219 | JNZ R6 ; ON EXIT, CL HAS SHIFT COUNT
4220 | ; TO RESTORE BITS
4221 | MOV AH,DL ; GET MASK TO AN
4222 | SHR AH,CL ; MOVE THE MASK TO CORRECT LOCATION
4223 | POP BX ; RECOVER REG
4224 | RET ; RETURN WITH EVERYTHING SET UP
4225 | R3:
4226 | ; SCROLL UP
4227 | ; THIS ROUTINE SCROLLS UP THE INFORMATION ON THE CRT
4228 | ; ENTRY
4229 | ; CH,CL = UPPER LEFT CORNER OF REGION TO SCROLL
4230 | ; DH,DL = LOWER RIGHT CORNER OF REGION TO SCROLL
4231 | ; BOTH OF THE ABOVE ARE IN CHARACTER POSITIONS
4232 | ; BH = FIL L VALUE FOR BLANKED LINES
4233 | ; AL = # LINES TO SCROLL (ALO MEANS BLANK THE ENTIRE
4234 | ; FIELD)
4235 | ; DS = DATA SEGMENT
4236 | ; ES = REG SEGMENT
4237 | ; EXIT
4238 | ; NOTHING, THE SCREEN IS SCROLLED
4239 |
4240 | ;-----------------------
4241 | GRAPHICS_UP PROC NEAR
4242 | MOV BL,AL ; SAVE LINE COUNT IN BL
4243 | MOV AX,CX ; GET UPPER LEFT POSITION INTO AX REG
4244 |
4245 | I----- USE CHARACTER SUBROUTINE FOR POSITIONING
4246 | I----- ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE
4247 |
4248 | CALL GRAPHIC_PDN
4249 | MOV DI,AX ; SAVE RESULT AS DESTINATION ADDRESS
4250 |
4251 | I----- DETERMINE SIZE OF WINDOW
4252 |
4253 | SUB DX,CX ; ADJUST VALUES
4254 | ADD DX,10H
4255 | SAL DH,1 ; MULTIPLE # ROWS BY 4
4256 | SAL DH,1 ; SIZE & VERT DOTS/CHAR
4257 | SAL DH,1 ; AND EVEN/ODD ROWS
4258 |
4259 | I----- DETERMINE CRT MODE
4260 |
4261 | CMP CRT_MODE,6 ; TEST FOR MEDIUM RES

A-60 System BIOS
<table>
<thead>
<tr>
<th>LOC OBJ</th>
<th>LINE</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>FA0D 7304</td>
<td>4262</td>
<td>JNC R7 ; FIND_SOURCE</td>
</tr>
<tr>
<td>FA0D 4264</td>
<td>----- MEDIUM RES UP</td>
<td></td>
</tr>
<tr>
<td>FA0D D0E2</td>
<td>4266</td>
<td>SAL DL,1 ; # COLUMNS = 2, SINCE 2 BYTES/CHAR</td>
</tr>
<tr>
<td>FA0D 4267</td>
<td>SAL DI,1 ; OFFSET # SINCE 2 BYTES/CHAR</td>
<td></td>
</tr>
<tr>
<td>FA0D 4269</td>
<td>----- DETERMINE THE SOURCE ADDRESS IN THE BUFFER</td>
<td></td>
</tr>
<tr>
<td>FA0D 4270</td>
<td>----- LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD FIELDS</td>
<td></td>
</tr>
<tr>
<td>FA0D 4275</td>
<td>FA0D 4266</td>
<td>PUSH ES ; GET SECTIONS BOTH POINTING TO REG</td>
</tr>
<tr>
<td>FA0D 4277</td>
<td>FA0D 4267</td>
<td>POP DS</td>
</tr>
<tr>
<td>FA0D 4278</td>
<td>FA0D 4268</td>
<td>MOV DL,0 ; ZERO TO HIGH OF CNT REG</td>
</tr>
<tr>
<td>FA0D 427E</td>
<td>FA0D 4269</td>
<td>MOV DI,0 ; MULTIPLE NUMBER OF LINES BY 4</td>
</tr>
<tr>
<td>FA0D 427F</td>
<td>FA0D 426A</td>
<td>MOV DI,0 ; IF ZERO, THEN BLANK ENTIRE FIELD</td>
</tr>
<tr>
<td>FA0D 427A</td>
<td>FA0D 426B</td>
<td>MOV DL,0 ; GET NUMBER OF LINES IN AL</td>
</tr>
<tr>
<td>FA0D 427B</td>
<td>FA0D 426C</td>
<td>MOV AH,0 ; ; 80 BYTES/ROW</td>
</tr>
<tr>
<td>FA0D 427C</td>
<td>FA0D 426D</td>
<td>MOV AL,0 ; DETERMINE OFFSET TO SOURCE</td>
</tr>
<tr>
<td>FA0D 427D</td>
<td>FA0D 426E</td>
<td>MOV DI,0 ; SET UP SOURCE</td>
</tr>
<tr>
<td>FA0D 427E</td>
<td>FA0D 426F</td>
<td>ADD SI,AX ; ADD IN OFFSET TO IT</td>
</tr>
<tr>
<td>FA0D 427F</td>
<td>FA0D 4270</td>
<td>MOV AH,0 ; NUMBER OF ROWS IN FIELD</td>
</tr>
<tr>
<td>FA0D 4280</td>
<td>FA0D 4271</td>
<td>MOV AH,0 ; DETERMINE NUMBER TO MOVE</td>
</tr>
<tr>
<td>FA0D 4281</td>
<td>FA0D 4272</td>
<td>SUB AH,BL ; NUMBER OF ROWS TO MOVE</td>
</tr>
<tr>
<td>FA0D 4282</td>
<td>FA0D 4273</td>
<td>MOV AH,0 ; CONTINUE TILL ALL MOVED</td>
</tr>
<tr>
<td>FA0D 4283</td>
<td>FA0D 4274</td>
<td>MOV AL,0 ; CLEAR_ENTRY</td>
</tr>
<tr>
<td>FA0D 4284</td>
<td>FA0D 4275</td>
<td>MOV AL,0 ; ATTRIBUTE TO FILL WITH</td>
</tr>
<tr>
<td>FA0D 4285</td>
<td>FA0D 4276</td>
<td>MOV AL,0 ; CLEAR_THAT_ROW</td>
</tr>
<tr>
<td>FA0D 4286</td>
<td>FA0D 4277</td>
<td>MOV DI,0 ; POINT TO NEXT LINE</td>
</tr>
<tr>
<td>FA0D 4287</td>
<td>FA0D 4278</td>
<td>MOV DI,0 ; NUMBER OF LINES TO FILL</td>
</tr>
<tr>
<td>FA0D 4288</td>
<td>FA0D 4279</td>
<td>MOV BL,0 ; CLEAR_LOOP</td>
</tr>
<tr>
<td>FA0D 4289</td>
<td>FA0D 427A</td>
<td>JMP VIDEO_RETURN ; EVERYTHING DONE</td>
</tr>
<tr>
<td>FA0D 428A</td>
<td>FA0D 427B</td>
<td>MOV BL,0 ; BLANK_FIELD</td>
</tr>
<tr>
<td>FA0D 428B</td>
<td>FA0D 427C</td>
<td>MOV BL,0 ; SET_BLANK_COUNT</td>
</tr>
<tr>
<td>FA0D 428C</td>
<td>FA0D 427D</td>
<td>MOV BL,0 ; EVERYTHING IN_FIELD</td>
</tr>
<tr>
<td>FA0D 428D</td>
<td>FA0D 427E</td>
<td>JMP R9 ; CLEAR_FIELD</td>
</tr>
</tbody>
</table>

Appendix A

System BIOS A-61
LOC OBJ     LINE     SOURCE

F4FA 61C20101 4359     ADD DX,101H     ; ADJUST VALUES
F4FE 006      4340     SAL DX,1     ; MULTIPLE # ROWS BY 4
F500 006      4341     SAL DX,1     ; SINCE A VERT DOTS/CHAR
F4E4          4342     SAL DX,1     ; AND EVEN/ODD ROWS
F4E4          4343     ----- DETERMINE CRT MODE
F502 003E490006 4344     CMP CRT_MODE,6     ; TEST FOR MEDIUM RES
F507 7305      4345     JNC R12     ; FIND_SOURCE_DOWN
F4E4          4346     ---- MEDIUM RES DOWN
F509 002      4347     SAL DL,1     ; # COLUMNS * 2, SINCE
F50B 01E7      4348     SAL DL,1     ; 2 BITES/CHAR (OFFSET OK)
F50D 47        4349     INC DI     ; OFFSET *2 SINCE 2 BYTES/CHAR
F4E4          4350     ---- DETERMINE THE SOURCE ADDRESS IN THE BUFFER
F50E          4351     R12:     ; FIND_SOURCE_DOWN
F50E 06      4352     PUSH ES     ; BOTH SEGMENTS TO REGEN
F50F LF      4353     POP DS      ; ZERO TO HIGH OF COUNT REG
F510 2AE      4354     SUB CH,CH     ; POINT TO LAST ROW OF PIXELS
F512 0C1F0000 4355     ADD DI,240     ; MULTIPLY NUMBER OF LINES BY 4
F516 0DE5     4356     SAL BL,1     ;
F518 0DE6     4357     SAL BL,1     ;
F51A 7242E     4358     JZ R16     ; IF ZERO, THEN BLANK ENTIRE FIELD
F51C 8AC3     4359     MOV AH,80     ; GET NUMBER OF LINES IN AL
F51E 0650     4360     MOV AH,80     ; 80 BYTES/ROM
F520 0EA4     4361     MULT AH     ; DETERMINE OFFSET TO SOURCE
F522 06FF     4362     MOV SI,DI     ; SET UP SOURCE
F524 28F0     4363     SUB SI,AX     ; SUBTRACT THE OFFSET
F526 0A96     4364     MOV AH,4DH     ; NUMBER OF ROWS IN FIELD
F528 2AE3     4365     MOV AH,4DH     ; DETERMINE NUMBER TO MOVE
F4E4          4366     ---- LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD FIELDS
F52A          4367     R13:     ; ROW_LOOP_DOWN
F52A E82100    4368     CALL R17     ; MOVE ONE ROW
F52D 01E5020   4369     SUB SI,200H+80     ; MOVE TO NEXT ROW
F531 81E5F020  4370     SUB SI,200H+80     ;
F535 FECC      4371     DEC AH     ; NUMBER OF ROWS TO MOVE
F537 75F1      4372     JNZ R13     ; CONTINUE TILL ALL MOVED
F4E4          4373     ---- FILL IN THE VACATED LINE(S)
F539          4374     R16:     ; CLEAR_ENTRY_DOWN
F539 0AC7     4375     MOV AL,OH     ; ATTRIBUTE TO FILL WITH
F53B          4376     CALL R1C     ; CLEAR_LOOP_DOWN
F53B 0E0000    4377     CALL R1C     ; CLEAR A ROM
F53E 01E5020   4378     SUB DI,200H+80     ; POINT TO NEXT LINE
F542 FECD     4379     DEC BL     ; NUMBER OF LINES TO FILL
F544 75F5      4380     JNZ R15     ; CLEAR_LOOP_DOWN
F546 FC       4381     CLD     ; RESET THE DIRECTION FLAG
F547 E97D0C    4382     JMP VIDEO_RETURN     ; EVERYTHING DONE
F54A          4383     ----- BLANK_FIELD_DOWN
F54A 0ADD     4384     MOV BL,OH     ; SET BLANK COUNT TO EVERYTHING
F54B          4385     JMP R14     ; IN FIELD
F54C EEBB     4386     JMP R14     ; CLEAR THE FIELD
F4E4          4387     ---- ROUTINE TO MOVE ONE ROW OF INFORMATION
F54E          4400     R17:     ; PROCNear
F54E 00A       4401     MOV CL,DL     ; NUMBER OF BYTES IN THE ROW
F550 56       4402     PUSH SI     ;
F551 57       4403     PUSH DI     ; SAVE POINTERS
F552 F3       4404     REP MOVSB     ; MOVE THE EVEN FIELD
F553 44       4405     REP MOVSB     ;
F554 5F       4406     POP DI     ;
F555 5E       4407     POP SI     ;
F556 61C0000   4408     ADD SI,200H     ; POINT TO THE ODD FIELD
F55A 61C7000   4409     ADD DI,200H     ;
F55E 56       4410     PUSH SI     ;
F55F 57       4411     PUSH DI     ; SAVE THE POINTERS
F560 0ACA     4412     MOV CL,DL     ; COUNT BACK
F562 F3       4413     REP MOVSB     ; MOVE THE ODD FIELD

A-62 System BIOS
Appendix A

LOC OBJ  LINE  SOURCE

A4  4415  POP   DI
A5  4416  POP   SI 
          ; POINTERS BACK
C3  4417  RET   ; RETURN TO CALLER

F567   4418  R17  ENDP

F567 BACA  4419  ;----- CLEAR A SINGLE ROW

F569 S7  4420  MOV  CL,DL  ; NUMBER OF BYTES IN FIELD
F56A F3  4421  REP   STOSB  ; STORE THE NEW VALUE
F56B AA  4422  POP   DI   ; POINT BACK
F56D 81C70020  4423  ADD  DI,000H  ; POINT TO ODD FIELD
F570 S7  4424  PUSH  DI
F572 BACA  4425  MOV  CL,DL
F573 F3  4426  REP   STOSB  ; FILL THE ODD FIELD
F575 AA  4427  POP   DI
F576 F5  4428  POP   DI
F577 C3  4429  RET   ; RETURN TO CALLER

4430  R10  ENDP

GRAPHICS_WRITE PROC NEAR

4433  ;----- DETERMINE REGION TO GET CODE POINTS FROM

4434  ;----- IMAGE IS IN SECOND HALF. IN USER RAM

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

GRAPHICS_WRITE PROC NEAR

4435  ; GRAPHICS_WRITE

4436  ; THIS ROUTINE WRITES THE ASCII CHARACTER TO THE

4437  ; CURRENT POSITION ON THE SCREEN.

4438  ; ENTRY

4439  ; AL = CHARACTER TO WRITE

4440  ; BL = COLOR ATTRIBUTE TO BE USED FOR FOREGROUND COLOR

4441  ; IF BIT 7 IS SET, THE CHAR IS XOR'D INTO THE REGEN

4442  ; BUFFER (0 IS USED FOR THE BACKGROUND COLOR)

4443  ; CX = NUMBER OF CHARS TO WRITE

4444  ; DS = DATA SEGMENT

4445  ; ES = REGEN SEGMENT

4446  ; EXIT

4447  ; NOTHING IS RETURNED

4448  ;

4449  ; GRAPHICS_READ

4450  ; THIS ROUTINE READS THE ASCII CHARACTER AT THE CURRENT

4451  ; CURSOR POSITION ON THE SCREEN BY MATCHING THE DOTS ON

4452  ; THE SCREEN TO THE CHARACTER GENERATION CODE POINTS

4453  ; ENTRY

4454  ; NONE (0 IS ASSUMED AS THE BACKGROUND COLOR)

4455  ; EXIT

4456  ; AL = CHARACTER READ AT THAT POSITION (0 RETURNED IF

4457  ; NONE FOUND)

4458  ;

4459  ; FOR BOTH ROUTINES, THE IMAGES USED TO FORM CHARS ARE

4460  ; CONTAINED IN ROM FOR THE 1ST 128 CHAR. TO ACCESS CHARS

4461  ; IN THE SECOND HALF, THE USER MUST INITIALIZE THE VECTOR AT

4462  ; INTERRUPT 1FH (LOCATION 0607CH) TO POINT TO THE USER

4463  ; SCRAMbled TABLE OF GRAPHIC IMAGES (6X8 BOXES).

4464  ; FAILURE TO DO SO WILL CAUSE INSTRANGE RESULTS

4465  ;

4466  ;

4467  ;

4468  ;

4469  ;

4470  ;

4471  ;

4472  ;

4473  ;

4474  ;

4475  ;

4476  ;

4477  ;

4478  ;

4479  ;

4480  ;

4481  ;

4482  ;

4483  ;

4484  ;

4485  ;

4486  ;

4487  ;

4488  ;

System BIOS  A-63
System BIOS

A-64

LOC OBJ | LINE | SOURCE

F50B 4449 | 51: | EXTEND_CHAR
F50B 2C00 | 4491 | SUB AL,60H | ZERO ORIGIN FOR SECOND HALF
F50B 1E | 4492 | PUSH DS | SAVE DATA POINTER
F50E 20F6 | 4493 | SUB SI,SI | ESTABLISH VECTOR ADDRESSING
F50E 8D0E | 4494 | MOV DS,SI | GET THE OFFSET OF THE TABLE
F50C 5367C00 | 4495 | ASSUME DS:ABSO | SEGMENT OF THE TABLE
F50C 6CDA | 4496 | MOV DX,DS | GET THE SEGMENT OF THE TABLE
F500 1F | 4497 | POP DS | RECOVER DATA SEGMENT
F508 52 | 4498 | PUSH DX | SAVE TABLE SEGMENT ON STACK
F501 4501 |
F509 4502 | 52: | DETERMINE_MODE
F50A 4503 | DETERMINE GRAPHICS MODE IN OPERATION
F50A 4504 |
F50A 4505 | SAL AX,1 | VALUE BY 8
F50A 4506 | SAL AX,1 | VALUE BY 8
F50A 4507 |
F50A 4508 | ADD SI,AX | ST HAS OFFSET OF DESIRED CODES
F50A 4509 | CMP CR_M,0,6 | TABLE
F50A 4510 | POP DS | RECOVER TABLE POINTER SEGMENT
F50A 4511 | JC 57 | TEST FOR MEDIUM RESOLUTION MODE
F50A 4512 |
F50A 4513 | HIGH RESOLUTION MODE |
F50A 4514 |
F50A 4515 | 53: | HIGH CHAR
F50A 4516 | PUSH DI | SAVE REGEN POINTER
F50A 4517 | PUSH SI | SAVE CODE POINTER
F50A 4518 | MOV DH,4 | NUMBER OF TIMES THROUGH LOOP
F50A 4519 | 54: |
F50A 4520 | LODSB | GET BYTE FROM CODE POINTS
F50A 4521 | TEST BL,0OH | SHOULD WE USE THE FUNCTION
F50A 4522 | JNZ 56 | TO PUT CHAR IN
F50A 4523 | STOSB | STORE IN REGEN BUFFER
F50A 4524 | LODSB |
F50A 4525 | 55: |
F50A 4526 | MOV ES:(DI+200H-1),AL | STORE IN SECOND HALF
F50A 4527 | ADD DI,79 | MOVE TO NEXT ROW IN REGEN
F50A 4528 | DEC DH | DONE WITH LOOP
F50A 4529 |
F50A 4530 | POP SI |
F50A 4531 | POP DI | RECOVER REGEN POINTER
F50A 4532 | INC DI | POINT TO NEXT CHAR POSITION
F50A 4533 | LOOP S3 | MORE CHARs TO WRITE
F50A 4534 | JMP VIDEO_RETURN |
F50A 4535 |
F50A 4536 | XOR AL,ES:(DI) | EXCLUSIVE OR WITH CURRENT
F50A 4537 | STOSB | STORE THE CODE POINT
F50A 4538 | LODSB | AGAIN FOR ODD FIELD
F50A 4539 | XOR ES:(DI+200H-1) |
F50A 4540 | JMP SS | BACK TO MAINSTREAM
F50A 4541 |
F50A 4542 | MEDIUM RESOLUTION WRITE |
F50A 4543 |
F50A 4544 | 57: | MED_BUS_WRITE
F50A 4545 | MOV DL,BL | SAVE HIGH COLOR BIT
F50A 4546 | SAL DI,1 | OFFSET+2 SINCE 2 BYTES/CHAR
F50A 4547 | CALL S19 | EXPAND BL TO FULL WORD OF COLOR
F50A 4548 | 58: | MED_CHAR
F50A 4549 | PUSH DI | SAVE REGEN POINTER
F50A 4550 | PUSH SI | SAVE THE CODE POINTER
F50A 4551 | MOV DH,4 | NUMBER OF LOOPS
F50A 4552 | 59: |
F50A 4553 | LODSB |
F50A 4554 | CALL 521 | DOUBLE UP ALL THE BITS
F50A 4555 | AND AL,8X | CONVERT THEM TO FOREGROUND
F50A 4556 | XOR (0 BACK) |
F50A 4557 | TEST DL,0OH | IS THIS XOR FUNCTION
F50A 4558 | JZ 510 | NO, STORE IT IN AS IT IS
F50A 4559 | XOR AL,ES:(DI) | DO FUNCTION WITH HALF
F50A 4560 | XOR AL,ES:(DI+1) | AND WITH OTHER HALF
F50A 4561 | 510: |
F50A 4562 | MOV ES:(DI+1),AL | STORE FIRST BYTE
F50A 4563 | MOV ES:(DI+1),AL | STORE SECOND BYTE
F50A 4564 | LODB | GET CODE POINT
F50F 4565 | CALL S21
GRAPHICS_READ PROC NEAR

GRAPHICS_WRITE

;----- MEOIUH RESOLUTION READ

;----- GET VALUES FROM REGEN BUFFER AND CONVERT TO CODE POINT

;----- MEDIUM RESOLUTION READ

;----- SAVE AREA HAS CHARACTER IN IT, MATCH IT

;----- FIND_CHAR

;----- ESTABLISH ADDRESSING

;----- CODE POINTS IN CS

;----- OF SAVE AREA

;----- CURRENT CODE POINT BEING MATCHED

F5FE 23C3 4966 AND AL,AX ; CONVERT TO COLOR
F600 F4C200 4967 TEST DL,00H ; AGAIN, IS THIS XOR FUNCTION
F603 70A4 4968 JZ SI1 ; NO, JUST STORE THE VALUES
F605 262A50020 4969 XOR AH,ES:[DI+200H] ; FUNCTION WITH FIRST HALF
F60A 262B50100 4970 XOR AL,ES:[DI+201H] ; AND WITH SECOND HALF
F60F 4971 SI1:
F60F 260A50020 4972 MOV ES:[DI+200H],AL ; STORE IN SECOND PORTION OF BUFFER
F614 262B50100 4973 MOV ES:[DI+201H]+1,AL
F619 03C750 4974 ADD DI,00H ; POINT TO NEXT LOCATION
F61C FECE 4975 DEC BH ; KEEP GOING
F61E 75C1 4976 JNZ SI9
F620 5E 4977 POP SI ; RECOVER CODE POINTER
F621 5F 4978 POP DI ; RECOVER REGEN POINTER
F622 47 4979 INC DI ; POINT TO NEXT CHAR POSITION
F623 47 4980 INC DI
F624 E2B7 4981 LOOP 5B ; MORE TO WRITE
F626 E99CFB 4982 JMP VIDEO_RETURN
F553 GRAPHICS_WRITE ENDP
F505 : GRAPHICS_READ:
F629 E0D060 4983 CALL SI2
F62C 08F0 4984 MOV SI,AX ; SAVE IN SI
F62E 03EC08 4985 SUB SP,8 ; ALLOCATE SPACE TO SAVE THE
F591 2BEC 4986 MOV BP,SI ; READ CODE POINT
F593 : SI TO SAVE AREA
F595 5534 ; ----- DETERMINE GRAPHICS MODES
F533 003E40006 4996 CMP CR_MODE,6
F53B 06 4997 PUSH ES
F539 IF 4998 POP DS ; POINT TO REGEN SEGMENT
F53A 721A 4999 JC SI3 ; MEDIUM RESOLUTION
F599 4600
F501 : ----- HIGH RESOLUTION READ
F502 4603 : ----- GET VALUES FROM REGEN BUFFER AND CONVERT TO CODE POINT
F504 4606 MOV DH,4 ; NUMBER OF PASS
F50C 4608 MOV SI,[SI] ; GET FIRST BYTE
F50E 006060 4609 MOV [BP],AL ; SAVE IN STORAGE AREA
F514 45 460A INC BP ; NEXT LOCATION
F516 086000 4610 MOV AL,[SI+200H] ; GET LOWER REGION BYTE
F518 086000 4611 MOV [BP],AL ; ADJUST AND STORE
F51A 45 4612 INC BP
F51C 83C550 4613 ADD SI,60 ; POINTER INTO REGEN
F51F 4EFE 4614 DEC DH ; LOOP CONTROL
F561 75EB 4615 JNZ SI2 ; DO IT SOME MORE
F563 EB1790 4616 JMP SI5 ; GO MATCH THE SAVED CODE POINTS
F567 4618 ; ----- MEDIUM RESOLUTION READ
F56F 4620 SI3: ; MEM_RES_READ
F566 116 4621 SAL SI,1 ; OFFSET*2 SINCE 2 BYTES/CHAR
F568 B604 4622 MOV DH,4 ; NUMBER OF PASS
F56A 4623 SI4:
F56A EB0800 4624 CALL SI3 ; GET PAIR BYTES FROM REGEN
F56C 4625 INTO SINGLE SAVE
F56D 01C6020 4626 ADD SI,200H ; GO TO LOWER REGION
F56E EB00100 4627 CALL SI3 ; GET THIS PAIR INTO SAVE
F574 81EB001F 4628 SUB SI,200H-80 ; ADJUST POINTER BACK INTO UPPER
F576 FECE 4629 DEC DH
F56A 75EE 4630 JNZ SI4 ; KEEP GOING UNTIL ALL 8 DONE
F56C 4632 ; ----- SAVE AREA HAS CHARACTER IN IT, MATCH IT
F56C 4634 SI5: ; FIND_CHAR
F56C BFAEEA90 4635 MOV DI,OFFSET CRT_CHAR_GEN ; ESTABLISH ADDRESSING
F570 0E 4636 PUSH CS
F571 07 4637 POP ES ; CODE POINTS IN CS
F572 83ED06 4638 SUB BP,8 ; ADJUST POINTER TO BEGINNING
F574 08F5 4639 MOV SI,8P
F577 FC 4640 CLD ; ENSURE DIRECTION
F578 B000 4642 MOV AL,0 ; CURRENT CODE POINT BEING MATCHED
LOC OBJ  LINE  SOURCE
F47A  4643  S16:   
F47A 16  4644  PUSH SS ; ESTABLISH ADDRESSING TO STACK
F47B  4645  POP DS ; FOR THE STRING COMPARE
F47C BA8000  4646  MOV DX,128 ; NUMBER TO TEST AGAINST
F47F  4647  S17:  
F47F  4648  PUSH SI ; SAVE SAVE AREA POINTER
F480  4649  PUSH DI ; SAVE CODE POINTER
F481 B90000  4650  MOV CX,8 ; NUMBER OF BYTES TO MATCH
F484 F  4651  REP REPE CMPSB ; COMPARE THE 8 BITE
F485  4652  POP DI ; RECOVER THE POINTERS
F486  4653  POP SI ;
F488  4654  JZ S16 ; IF ZERO FLAG SET, THEN MATCH OCCURRED
F48A FEC0  4655  INC AL ; NO MATCH, MOVE ON TO NEXT
F48C 83C70B  4656  ADD DX,8 ; NEXT CODE POINT
F48F  4657  DEC DX ; LOOP CONTROL
F490 75ED  4658  JNZ S17 ; DO ALL OF THEM
F469  4659  
F460  4660  i----- CHAR NOT MATCHED, MIGHT BE IN USER SUPPLIED SECOND HALF
F461  4661  
F462  4662  CMP AL,0 ; AL <> 0 IF ONLY 1ST HALF SCANNED
F463  4663  JE S16 ; IF = 0, THEN ALL HAS BEEN SCANNED
F464  4664  SUB AX,AX
F465  4665  MOV DS,AX ; ESTABLISH ADDRESSING TO VECTOR
F466  4666  ASSUME DS:ASDS
F468  4667  LES DI,EXT_PTR ; GET POINTER
F469  4668  MOV AX,ES
F46A  4669  OR AX,DI ; IF ALL 0, THEN DOESN'T EXIST
F46C 7404  466A  JZ S16 ; NO SENSE LOOKING
F46E  466B  MOV AL,128 ; ORIGIN FOR SECOND HALF
F470  466C  EBD2  466D  JMP S16 ; GO BACK AND TRY FOR IT
F473  466E  ASSUME DS:DATA
F46F  466F  
F470  4670  i----- CHARACTER IS FOUND ( AL=0 IF NOT FOUND )
F471  4671  S10:
F472  4672  
F476  4676  
F47A  4677  S10:  
F47A 03C40B  4678  ADD SP,B ; READJUST THE STACK, THROW AWAY SAVE
F47A E917FB  4679  JMP VIDEO_RETURN ; ALL DONE
F47B  4680  GRAPHICS_READ ENDP
F480  4681  i--------------------------------------------------------------------------------------------------
F484  4691  S19  PROC NEAR
F486  4692  AND BL,3 ; ISOLATE THE COLOR BITS
F488  4693  MOV AL,AL ; COPY TO AL
F489  4694  PUSH CX ; SAVE REGISTER
F48A B90300  4695  MOV CX,3 ; NUMBER OF TIMES TO DO THIS
F48B  4696  S20:  
F48C  4697  SAL AL,1
F48D  4698  SAL AL,1 ; LEFT SHIFT BY 2
F48E  4699  OR BL,AL ; ANOTHER COLOR VERSION INTO BL
F490  4700  LOOP S20 ; ALL OF BL
F491  4701  MOV BX,BL ; FILL UPPER PORTION
F493  4702  POP CX ; REGISTER BACK
F494  4703  RET ; ALL DONE
F495  4704  S19 ENDP
F495  4705  i--------------------------------------------------------------------------------------------------
F496  4706  EXPAND_BYTE  
F498  4707  THIS ROUTINE TAKES THE BYTE IN AL AND DOUBLES
F499  4708  ALL OF THE BITS, TURNING THE 8 BITS INTO
F49A  4709  16 BITS. THE RESULT IS LEFT IN AX
F49B  4710  i--------------------------------------------------------------------------------------------------
F49C  4711  PROC NEAR
F49E  4712  PUSH DX ; SAVE REGISTERS
F49F  4713  PUSH CX
F4A0  4714  PUSH BX
F4A2  4715  SUB DX,DX ; RESULT REGISTER
F4A3 90100  4716  MOV CX,1 ; MASK REGISTER
F4A4  4717  S22:  
F4A4  4718  MOV BX,AX ; BASE INTO TEMP

A-66  System BIOS
LOC OBJ | LINE | SOURCE
F6CB 2309 | 4719 | AND BX,CX USE MASK TO EXTRACT A BIT
F6CF 0803 | 4720 | OR DX,BX PUT INTO RESULT REGISTER
F6D1 010 | 4721 | SHL AX,1 SHIFT BASE AND MASK BY 1
F6D3 010 | 4722 | SHL CX,1 BASE TO TEMP
F6D5 080B | 4723 | MOV BX,AX EXTRACT THE SAME BIT
F6D7 2309 | 4724 | AND BX,CX PUT INTO RESULT
F6D9 0803 | 4725 | OR DX,BX SHIFT ONLY MASK NOW,
F6DA 010 | 4726 | SHL CX,1 MOVING TO NEXT BASE
F6DC 73C | 4727 | JNC SI2 USE MASK BIT COMING OUT TO TERMINATE
F6DF 080C | 4728 | MOV AX,DX RESULT TO FARM REGISTER
F6E1 50 | 4729 | POP BX RECOVER REGISTERS
F6E2 59 | 4730 | POP CX
F6E3 5A | 4731 | POP DX
F6E4 C3 | 4732 | RET ALL DONE
F6E5 | 4733 | S21 ENDP
F6E6 B4 | 4734 | S23 PROC NEAR
F6E7 8A4401 | 4735 | MOV AX,[SI] GET FIRST BYTE
F6EA 090CD0 | 4736 | MOV AX,[SI]+1 GET SECOND BYTE
F6ED 6200 | 4737 | MOV DL,0 RESULT REGISTER
F6E8 524 | 4738 | S24:
F6E9 05C1 | 4739 | TEST AX,CX IS THIS SECTION BACKGROUND?
F6F1 F8 | 4740 | CLC CLEAR CARRY IN HOPES THAT IT IS
F6F2 7401 | 4741 | JZ S25 IF IT IS BACKGROUND
F6F4 F9 | 4742 | STC WASN'T, SO SET CARRY
F6F5 0002 | 4743 | S25: RCL DL,1 MOVE THAT BIT INTO THE RESULT
F6F7 010 | 4744 | SHR CX,1 MOVE THE MASK TO THE RIGHT BY 2 BITS
F6F9 010 | 4745 | JNC S24 DO IT AGAIN IF NOT
F6FD 055000 | 4746 | MOV [BP],DL STORE RESULT IN SAVE AREA
F700 45 | 4747 | JNC BP ADJUST POINTER
F701 C3 | 4748 | RET ALL DONE
F702 | 4749 | S23 ENDP
F702 A10000 | 4750 | MOV AX,CURSOR_POSH GET CURRENT CURSOR
F703 | 4751 | GRAPH_POSH LABEL NEAR
F705 53 | 4752 | PUSH BX SAVE REGISTER
F706 0808 | 4753 | MOV BX,AX SAVE A COPY OF CURRENT CURSOR
F708 D4C | 4754 | MOV AL,AX GET R9S TO AL
F70A F624400 | 4755 | MUL BYTE PTR CRT_COLS MULTIPLY BY BYTES/COLUMN
F70E 010 | 4756 | SHL AX,1 MULTIPLY * 4 SINCE 4 R9S/BYTE
F710 010 | 4757 | SHL AX,1
F712 2AFF | 4758 | SUB BH,BH ISOLATE COLUMN VALUE
F714 03C3 | 4759 | ADD AX,BX DETERMINE OFFSET
F716 5B | 4760 | POP BX RECOVER POINTER
F717 C3 | 4761 | RET ALL DONE
F718 | 4762 | S26 ENDP
LOC OBJ

---

F71A  WRITE_TTY PROC NEAR
F71A 50  PUSH AX ; SAVE REGISTERS
F71A 50  PUSH AX ; SAVE CHAR TO WRITE
F71A B403  MOV AH,3
F71A 8A366200  MOV BH,ACTIVE_PAGE ; GET THE CURRENT ACTIVE PAGE
F71A CD10  INT 10H ; READ THE CURRENT CURSOR POSITION
F72A 58  POP AX ; RECOVER CHAR
F72A 26  I----- DX NOW HAS THE CURRENT CURSOR POSITION
F72A C008  CMP AL,0 ; IS IT A BACKSPACE
F72A 7542  JE UB ; BACK_SPACE
F72A C000  CMP AL,0DH ; IS IT CARRIAGE RETURN
F72A 7957  JE UF ; CAR_RET
F72A C0A0  CMP AL,0AH ; IS IT A LINE FEED
F72A 7547  JE U10 ; LINE_FEED
F72A C007  CMP Al,07H ; IS IT A BELL
F73A 745A  JE UI1 ; BELL
F72A 3C08  I----- WRITE THE CHAR TO THE SCREEN
F73B B40A  MOV AH,10 ; WRITE CHAR ONLY
F73B 90100  MOV CX,1 ; ONLY ONE CHAR
F73B CD10  INT 10H ; WRITE THE CHAR
F73B 3C09  I----- POSITION THE CURSOR FOR NEXT CHAR
F73A FEC2  INC DL
F73C 3A144A00  CMP DL,BYTE PTR CRT_COLS ; TEST FOR COLUMN OVERFLOW
F74D 7535  JNZ U7 ; SET_CURSOR
F74D B200  MOV DL,0 ; COLUMN FOR CURSOR
F74D 00E10  CMP DH,24
F74D 752A  JNZ U6 ; SET_CURSOR_INC
F74D 5C  I----- SCROLL REQUIRED
F74D 49  MOV AH,2
F74D 602  MOV U1;
F74D CD10  INT 10H ; SET THE CURSOR
F74D 59  I----- DETERMINE VALUE TO FILL WITH DURING SCROLL
F74D 0D900  MOV AL,CRT_MODE ; GET THE CURRENT MODE
F75A 3C04  CMP AL,4
F75A 7206  JC U2 ; READ_CURSOR
F75A C007  CMP AL,7
F75A 6700  MOV BH,0 ; FILL WITH BACKGROUND
F75A 7506  JNE U3 ; SCROLL_UP
F75A 6408  MOV AH,0 ; READ_CURSOR
F75C CD10  INT 10H ; STORE CUR/ATTR AT CURRENT CURSOR
F75C 6AFC  MOV BH,AH
F760 72U3 ; SCROLL_UP

A-68  System BIOS
LOC  OBJ  LINE  SOURCE

F760  B80106  4873  MOV  AX,60H  // SCROLL ONE LINE
F763  B8C9  4874  SUB  CX,CX  // UPPER LEFT CORNER
F765  B818  4875  MOV  DL,24  // LOWER RIGHT ROW
F767  B9144A00  4876  MOV  DL,BYTE PTR CRT_COLS  // LOWER RIGHT COLUMN
F768  FECA  4877  DEC  DL
F760  4878  U4:  // VIDEO-CALL-RETURN
F760  CD10  4879  INT  10H  // SCROLL UP THE SCREEN
F76F  4880  US:  // TTY-RETURN
F76F  8B  4881  POP  AX  // RESTORE THE CHARACTER
F770  EE92FA  4882  JMP  VIDEO_RETURN  // RETURN TO CALLER
F773  4883  U6:  // SET-CURSOR-INC
F773  FEC6  4884  INC  DH  // NEXT ROW
F775  4885  U7:  // SET-CURSOR
F775  B902  4886  MOV  AH,2  // ESTABLISH THE NEW CURSOR
F777  E8F4  4887  JMP  U4  // BACK SPACE FOUND
F490  4888
F779  4890  U6:  // ALREADY AT END OF LINE
F779  80FA00  4892  CMP  DL,0
F77C  76F7  4893  JE  U7  // SET-CURSOR
F77E  FECA  4894  DEC  DL  // NO -- JUST MOVE IT BACK
F780  EBF3  4895  JMP  U7  // SET-CURSOR
F780  4896  U4:  // CARRIAGE RETURN FOUND
F490  4898
F782  B200  4899  MOV  DL,0  // MOVE TO FIRST COLUMN
F784  EBEF  489A  JMP  U7  // SET-CURSOR
F784  489B  U7:  // LINE FEED FOUND
F490  489D
F786  489E  U10:  // BOTTOM OF SCREEN
F786  80FE1A  489F  CMP  DH,24
F789  75E8  4907  JNE  U6  // YES, SCROLL THE SCREEN
F78B  E88C  4908  JMP  U1  // NO, JUST SET THE CURSOR
F490  4909
F78D  4910  U11:  // BELL FOUND
F491  4912
F78D  B302  4913  MOV  BL,2  // SET UP COUNT FOR BEEP
F78F  E07602  4914  CALL  BEEP  // SOUND THE POP BELL
F792  E80B  4915  JMP  US  // TTY_RETURN
F491  4916  WRITE_TTY  ENDP
F491  4917  // LIGHT PEN
F491  4918  // THIS ROUTINE TESTS THE LIGHT PEN SWITCH AND THE LIGHT
F491  4919  // PEN TRIGGER. IF BOTH ARE SET, THE LOCATION OF THE LIGHT
F491  4920  // PEN IS DETERMINED. OTHERWISE, A RETURN WITH NO
F491  4921  // INFORMATION IS MADE.
F491  4922  // ON EXIT
F491  4923  // (AH) = 0 IF NO LIGHT PEN INFORMATION IS AVAILABLE
F491  4924  // (AH) = 1 IF LIGHT PEN IS AVAILABLE
F491  4925  // BX,CX,DX ARE DESTROYED
F491  4926  // (AH) = 1 IF LIGHT PEN INFORMATION IS AVAILABLE
F491  4927  // IDH,DL = ROW,COLUMN OF CURRENT LIGHT PEN
F491  4928  // POSITION
F491  4929  // (CH) = RASTER POSITION
F491  4930  // (BC) = BEST GUESS AT PIXEL HORIZONTAL POSITION
F491  4931  // ASSUME CS:CODE,DS:DATA
F493  4932  // SUBTRACT_TABLE
F494  4933  // DECREMENT INDEX OF TABLE
F494  4934  DB  3,3,5,5,3,3,4
F774  4935  V1  // LABEL BYTE
F774  4936  03
F774  4937  03
F775  4938  03
F775  4939  03
F775  493A  03
F779  493B  04
F78C  493C
F794  493D  READ_LPN  PROC
F794  493E  PROC
F794  493F
F794  4940  MOV  AH,0  // SET NO LIGHT PEN RETURN CODE
F794  4941  MOV  DX,ADDR_6845  // GET BASE ADDRESS OF 6845
F794  4942  ADD  DX,6  // POINT TO STATUS REGISTER

System BIOS  A-69
FOAS EC  4943    IN AL,DX  ; SET STATUS REGISTER
FOAB A04  4944    TEST AL, A  ; TEST LIGHT PEN SWITCH
FOAB 75E  4945    JNZ V6  ; NOT SET, RETURN
4946
4947   ;---- TEST FOR LIGHT PEN TRIGGER
4948
F7AA A02  4949    TEST AL, 2  ; TEST LIGHT PEN TRIGGER
F7AC 7503  4950    JNZ V7A  ; RETURN WITHOUT RESETTING TRIGGER
F7AE E90100  4951    JMP V7
4952
4953   ;---- TRIGGER HAS BEEN SET, READ THE VALUE IN
4954
F7B1  4955    V7A:
4956    MOV AH,16  ; LIGHT PEN REGISTERS ON 6045
4957
4958   ;---- INPUT REGS POINTED TO BY AH, AND CONVERT TO ROW COLUMN IN DX
4959
F7B3 8B166300  4960    MOV DX,ADDR_6045  ; ADDRESS REGISTER FOR 6045
4961    MOV AL,AH  ; REGISTER TO READ
4962
4963    MOV DX,AL  ; SET IT UP
4964    MOV DX,AL  ; DATA REGISTER
4965    MOV CH,AL  ; GET THE VALUE
4966    MOV CH,AL  ; SAVE IN CX
4967
4968    MOV AX,AH  ; SECOND DATA REGISTER
4969    MOV AL,AH  ; POINT TO DATA REGISTER
4970    MOV AL,DX  ; GET SECOND DATA VALUE
4971
4972    MOV AH, CH  ; AX HAS INPUT VALUE
4973
4974   ;---- AX HAS THE VALUE READ IN FROM THE 6045
4975
F7C8 8A1E4900  4976    MOV BL,CRT_MODE
4977    MOV BH,BH  ; MODE VALUE TO BX
4978
4979    MOV CL,CS:Vl[BX])  ; DETERMINE AMOUNT TO SUBTRACT
4980    MOV AX,BX  ; TAKE IT AWAY
4981
4982    MOV BX,BX  ; DETERMINE BX
4983    MOV AL,AH  ; DETERMINE AX
4984    MOV AH,1  ; AX HAS VALUE
4985
4986   ;---- DETERMINE MODE OF OPERATION
4987
F7E1  4988    V2:
4989    MOV CL,3  ; DETERMINE_MODE
4990
4991    CMP CRT_MODE,4  ; DETERMINE IF GRAPHICS OR ALPHA
4992    JB V4  ; ALPHA_PEN
4993
4994    CMP CRT_MODE,7  ; DETERMINE GRAPHICS_MODE
4995
4996
4997    MOV DL,40  ; DIVISOR FOR GRAPHICS
4998
4999    DIV DL  ; DETERMINE ROW(AX) AND COLUMN(AX)
5000
5001   ;---- DETERMINE GRAPHIC ROW POSITION
5002
F7F5 8A0E  5003    MOV CH,AL  ; SAVE ROW VALUE IN CH
5004
5005    MOV CH,CM  ; #2 FOR EVENWOOD FIELD
5006
5007    ADD CH,CM  ; COLUMN VALUE TO BX
5008    MOV BH,BH  ; MULTIPLY BY 8 FOR MEDIUM RES
5009    MOV CL,4  ; SHIFT VALUE FOR HIGH RES
5010    MOV AX,AL  ; COLUMN VALUE TIMES 2 FOR HIGH RES
5011
5012    SHL BX,CL  ; MULTIPLY #16 FOR HIGH RES
5013
5014   ;---- DETERMINE ALPHA CHAR POSITION
5015
F7F0 8A04  5016    MOV DL,AL  ; COLUMN VALUE FOR RETURN
5017
5018    MOV DH,AL  ; ROW VALUE
5019
5020    SHR DH,1  ; DIVIDE BY 4
5021
5022    SHR DH,1  ; FOR VALUE IN 0-24 RANGE

A-70  System BIOS
LOC OBJ  |  LINE | SOURCE
---|---|---
F812 EB12 | 5920 | JMP SHORT V5 | LIGHT_PEN_RETURN_SET
F821 | 5922 | ------ ALPHA MODE ON LIGHT PEN |
F814 | 5924 | V6: | ALPHA_PEN |
F814 F6364A00 | 5925 | DIV BYTE PTR CRT_COLS | DETERMINE ROW,COLUMN VALUE |
F818 8AF0 | 5926 | MOV DH,AL | ROWS TO DH |
F81A BAD4 | 5927 | MOV DL,AL | COLS TO DL |
F81C D2E0 | 5928 | SAL AL,CL | MULTIPLY ROWS = 0 |
F81E 6A26 | 5929 | MOV CH,AL | GET RASTER VALUE TO RETURN REG |
F820 BADC | 5930 | MOV BL,AL | COLUMN VALUE |
F822 3FF | 5931 | XOR BH,DL | TO BX |
F824 D3E3 | 5932 | SAL BX,CL |
F826 | 5933 | V5: | LIGHT_PEN_RETURN_SET |
F826 B401 | 5934 | MOV AH,1 | INDICATE EVERYTHING SET |
F828 | 5935 | V6: | LIGHT_PEN_RETURN |
F82A 82E | 5936 | PUSH DX | SAVE RETURN VALUE (IN CASE) |
F82C 08166300 | 5937 | MOV DX,ADDR_6045 | GET BASE ADDRESS |
F82E 65C07 | 5938 | ADD DX,7 | POINT TO RESET PARAM |
F830 EE | 5939 | OUT DX,AL | ADDRESS, NOT DATA, IS IMPORTANT |
F831 5A | 5940 | POP DX | RECOVER VALUE |
F832 | 5941 | V7: | RETURN_NO_RESET |
F832 5F | 5942 | POP DI |
F833 SE | 5943 | POP SI |
F834 IF | 5944 | POP DS | DISCARD SAVED BX,CX,DX |
F835 IF | 5945 | POP DS |
F836 IF | 5946 | POP DS |
F837 IF | 5947 | POP DS |
F838 07 | 5948 | POP ES |
F839 CF | 5949 | IRET |
F840 | 5950 | READ_OPEN | ENDP |
F841 | 5951 | | |
F841 | 5952 | --- INT 12 ---- ------------ | |
F841 | 5953 | MEMORY_SIZE_DET | |
F841 | 5954 | : THIS ROUTINE DETERMINES THE AMOUNT OF MEMORY IN THE SYSTEM | |
F841 | 5955 | : AS REPRESENTED BY THE SWITCHES ON THE PLANNER. NOTE THAT THE | |
F841 | 5956 | : SYSTEM MAY NOT BE ABLE TO USE I/O MEMORY UNLESS THERE IS A FULL | |
F841 | 5957 | : COMPLEMENT OF 64K BYTES ON THE PLANNER. | |
F841 | 5958 | : INPUT | |
F841 | 5959 | : NO REGISTERS | |
F841 | 5960 | : THE MEMORY_SIZE VARIABLE IS SET DURING POWER ON DIAGNOSTICS | |
F841 | 5961 | : ACCORDING TO THE FOLLOWING HARDWARE ASSUMPTIONS: | |
F841 | 5962 | : PORT 60 BITS 3,2 = 00 - 16K BASE RAM | |
F841 | 5963 | : 01 - 32K BASE RAM | |
F841 | 5964 | : 02 - 48K BASE RAM | |
F841 | 5965 | : 11 - 64K BASE RAM | |
F841 | 5966 | : PORT 62 BITS 3-0 INDICATE AMOUNT OF I/O RAM IN 32K INCREMENTS | |
F841 | 5967 | : E.G., 0000 = NO RAM IN I/O CHANNEL | |
F841 | 5968 | : 0010 - 64K RAM IN I/O CHANNEL, ETC. | |
F841 | 5969 | : OUTPUT | |
F841 | 5970 | : \( |AX| = \) NUMBER OF CONTIGUOUS 1K BLOCKS OF MEMORY | |
F841 | 5971 | : | |
F841 | 5972 | \( |AX| = \) NUMBER OF CONTIGUOUS 1K BLOCKS OF MEMORY | |
F841 | 5973 | \( |AX| = \) NUMBER OF CONTIGUOUS 1K BLOCKS OF MEMORY | |
F841 | 5974 | MEMORY_SIZE_DET PROC FAR | |
F841 | 5975 | | |
F841 FB | 5976 | STI | INTERRUPTS BACK ON |
F842 IE | 5977 | PUSH DS | SAVE SEGMENT |
F843 E01302 | 5978 | CALL DDS | |
F845 A11500 | 5979 | MOV AX,MEMORY_SIZE | GET VALUE |
F849 IF | 5980 | POP DS | RECOVER SEGMENT |
F84A CF | 5981 | IRET | RETURN TO CALLER |
F84A | 5982 | MEMORY_SIZE_DET ENDP | |
F84B | 5983 | | |
F84B | 5984 | --- INT 11 ---- ------------ | |
F84B | 5985 | : EQUIPMENT DETECTION | |
F84B | 5986 | : THIS ROUTINE ATTEMPTS TO DETERMINE WHAT OPTIONAL | |
F84B | 5987 | : DEVICES ARE ATTACHED TO THE SYSTEM. | |
F84B | 5988 | : INPUT | |
F84B | 5989 | : NO REGISTERS | |
F84B | 5990 | : THE EQUIP_FLAG VARIABLE IS SET DURING THE POWER ON | |
F84B | 5991 | : DIAGNOSTICS USING THE FOLLOWING HARDWARE ASSUMPTIONS: | |
F84B | 5992 | : PORT 60 = LOW ORDER BYTE OF EQUIPMENT | |
F84B | 5993 | : PORT 3FA = INTERRUPT 2D REGISTER OF 6250 | |
F84B | 5994 | : PORT 378 = OUTPUT PORT OF PRINTER -- 6255 PORT THAT | |
F84B | 5995 | : CAN BE READ AS WELL AS WRITTEN | |
F84B | 5996 | : OUTPUT | |
A-72  System BIOS
<table>
<thead>
<tr>
<th>LOC OBJ</th>
<th>LINE</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFT 313031</td>
<td>5250</td>
<td>E0 DB '101',13.10 ; SYSTEM BOARD ERROR</td>
</tr>
<tr>
<td>F902 OD</td>
<td>5251</td>
<td>E1 DB '201',13.10 ; MEMORY ERROR</td>
</tr>
<tr>
<td>F903 OD</td>
<td>5252</td>
<td>F9A DB 'ROM',13.10 ; ROM CHECKSUM ERROR</td>
</tr>
<tr>
<td>F904 29323031</td>
<td>5253</td>
<td>F3C DB '1001',13.10 ; EXPANSION IO BOX ERROR</td>
</tr>
<tr>
<td>F905 OD</td>
<td>5254</td>
<td>D1 DB 'PARITY CHECK 2',13.10</td>
</tr>
<tr>
<td>F906 OD</td>
<td>5255</td>
<td>D2 DB 'PARITY CHECK 1',13.10</td>
</tr>
<tr>
<td>F907 3F3F3F3F</td>
<td>5256</td>
<td>D2A DB '?????',13.10</td>
</tr>
</tbody>
</table>

---

**A-74 System BIOS**
SUBROUTINES

F90B  XPC_BYTE  PROC NEAR
F90B  50  PUSH AX  ; SAVE FOR LOW NIBBLE DISPLAY
F90C  B104  MOV CL,4  ; SHIFT COUNT
F90E  D2E8  SHR AL,CL  ; NIBBLE SWAP
F90F  08300  CALL XLAT_PR  ; DO THE HIGH NIBBLE DISPLAY
F99  58  POP AX  ; REMOVE NIBBLE DISPLAY
F99A  240F  AND AL,0FH  ; ISOLATE TO LOW NIBBLE
F99E  50  CALL XLAT_PR  ; FALL INTO LOW NIBBLE CONVERSION
F99F  27  SHR AL,CL  ; NIBBLE SWAP
F99F  2F  CALL XPC_BYTE  ; CONVERT 00-0F TO ASCII CHARACTER
F99F  38  ADD AL,040H  ; ADD FIRST CONVERSION FACTOR
F99F  27  CALL XLAT_PR  ; ADD CONVERSION AND ADJUST LOW NIBBLE
F99F  27  CALL XLAT_PR  ; ADD CONVERSION AND ADJUST LOW NIBBLE
F99F  C7  CALL XPC_BYTE  ; CONVERT 00-0F TO ASCII CHARACTER
F99F  2F  CALL XPC_BYTE  ; CONVERT 00-0F TO ASCII CHARACTER
F99F  27  CALL XLAT_PR  ; ADD CONVERSION AND ADJUST LOW NIBBLE
F99F  C7  CALL XPC_BYTE  ; CONVERT 00-0F TO ASCII CHARACTER
F99F  2F  CALL XPC_BYTE  ; CONVERT 00-0F TO ASCII CHARACTER
F99F  38  ADD AL,040H  ; ADD FIRST CONVERSION FACTOR
F99F  27  CALL XLAT_PR  ; ADD CONVERSION AND ADJUST LOW NIBBLE

F9A3  F4  LABEL WORD  ; PRINTER SOURCE TABLE
F9A3  DC03  DW 3ECH
F9A5  7003  DW 370H
F9A7  7002  DW 270H
F9A9  F4E  LABEL WORD
F9A9  F4  LABEL WORD

F9A9  B8EE  MOV BP,SI  ; SET BP NON-ZERO TO FLAG ERR
F9AB  E01C00  CALL P_MSG  ; PRINT MESSAGE
F9AEC  15  PUSH DS
F9AF  E40700  CALL DOS
F9B2  A01000  MOV AL,BYTE PTR EQUTP_FLAG  ; LOOP/HALT ON ERROR
F9B5  2401  AND AL,01H  ; SWITCH ON?
F9B7  750F  JNZ G12  ; NO - RETURN
F9B9  FA  MOV MFG_HALT: CL1  ; YES - HALT SYSTEM
F9BA  B089  MOV AL,08H
F9BC  E663  OUT CMD_PORT,AL
F9BE  B085  MOV AL,1000001B  ; DISABLE KB
F9BF  E661  OUT PORT_B,AL
F9C2  A01500  MOV AL,MFG_ERR_FLAG  ; RECOVER ERROR INDICATOR
F9C5  E660  OUT PORT_A,AL  ; SET INTO 8255 REG
F9C7  F4  HLT  ; HALT SYS
F9CA  G12  POP DS  ; WRITE_MSG;
F9CB  C3  RET
F9C9  E_MSG  ENDP

F9CA  P_MSG  PROC NEAR
F9CA  G12:  MOV AL,CS:[SI]  ; PUT CHAR IN AL
F9CD  46  INC SI  ; POINT TO NEXT CHAR
F9CE  50  PUSH AX  ; SAVE PRINT CHAR
F9CF  EBCAFF  CALL PRT_HEX
F9D2  50  POP AX  ; RECOVER PRINT CHAR
F9D3  3COA  CMP AL,10  ; WAS IT LINE FEED?
F9D5  75F3  JNE G12A  ; NO,KEEP PRINTING STRING
F9D7  C3  RET
F9D3  P_MSG  ENDP

System BIOS  A-75
This page contains assembly code for a routine to sound a beeper. The code is designed to handle various scenarios such as sounding a short beep (short tones) or a long beep (long tones) based on input parameters. The code includes calls to other routines for functions like clearing the display, setting the keyboard, and checking the status of the system.

The routine `ERR_BEEP` is defined first, followed by `BEEP_PROC_NEAR` and several subroutines for handling different conditions. The code is structured to allow for flexibility in responding to system failures or errors, such as a short beep for a simple warning or a long beep for a more critical issue.

For instance, `F608` to `F609` involve saving system flags and disabling system interrupts, while `F610` to `F619` include delays and flag operations for timing and control purposes. These subroutines are crucial for ensuring that the system responds correctly to various error conditions, with each subroutine tailored to manage a specific aspect of the beep generation or system recovery process.

Overall, the code is designed to be modular, with each subroutine having a specific function that can be called upon to handle different error conditions, ensuring that the system remains stable and informative in the face of failures.
; INT 1A - The INTERRUPT HANDLER for the 8086

ORG OFFFFH

TIME_OF_DAY PROC FAR

STI ; Interrupts back on

CALL DOS ; Save segment

OR AH, AH ; AH = 0

JZ T2 ; READ_TIME

DEC AH ; AH = 1

JZ T3 ; SET_TIME

T1: ; INTERRUPTS BACK ON

JMP T1 ; RE_INT

T2: ; READ_TIME

JMP T1 ; SET_TIME

T3: ; INTERRUPTS BACK ON

JMP T1 ; RE_INT

CLD ; No timer interrupts while writing

MOV AX, TIMER_HIGH

JMP T1 ; RET

T4: ; No interrupts while writing

MOV AX, TIMER_LOW

JMP T1 ; RET

T5: ; Timer overflow, and reset the flag

GET OVERFLOW, AND SET THE FLAG

JMP T1 ; RET

T6: ; Timer overflow

JMP T1 ; RET

T7: ; Timer overflow

JMP T1 ; RET

T8: ; Timer overflow

JMP T1 ; RET

T9: ; Timer overflow

JMP T1 ; RET

TIME_OF_DAY ENDP

TIME_OF_DAY PROC FAR

; 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. 10.2 INTERRUPTS EVERY SECOND.

; THE INTERRUPT HANDLER MAINTAINS A COUNT OF INTERRUPTS

; SINCE POWER ON TIME, WHICH MAY BE USED TO ESTABLISH

; THE TIME OF DAY.

; THE INTERRUPT HANDLER ALSO DECREMENTS THE FLAG

; CONTROL COUNT OF THE DISKETTE, AND WHEN IT EXPIRES

; WILL TURN OFF THE DISKETTE MOTOR, AND RESET THE

; MOTOR RUNNING FLAGS.

; THE INTERRUPT HANDLER WILL ALSO INVOKE A USER ROUTINE

; THROUGH INTERRUPT ICH AT EVERY TIME TICK. THE USER

; MUST CODE A ROUTINE AND PLACE THE CORRECT ADDRESS IN

; THE VECTOR TABLE.

ORG OFFEAH

TIMER_INT PROC FAR
The image contains a document with text related to system BIOS and programming. The text appears to be a listing of assembly language instructions and comments. The natural text is not directly transcribed as it seems to be a screenshot of a document rather than a readable text. The content includes hexadecimal values and comments indicating various system BIOS functions and parameters. The document is likely a part of a larger guide or manual for system BIOS programming.
IHT 5 ---------------------------------------------------------------
- ---
- ---
- ---
- ---
------------------- ____ --_ - ____ _
- - - -- - ---

FF23 PROC NEAR
FF23 1E ASSUME DS:DATA
FF24 DD PUSH DS
FF25 50 PUSH DX : SAVE REG AX CONTENTS
FF26 E030FB CALL DBB
FF27 B0D0 MOV AL,96H : READ IH-SERVICE REG
FF28 E620 OUT INTAB0.AL ; (FIND OUT WHAT LEVEL BEING
FF29 90 HNP : SERVICED)
FF2E E420 IN AL,INTAB0 ; GET LEVEL
FF30 0AC0 MOV AH,AL ; SAVE IT
FF31 0AC4 OR AL,AL ; DO? (NO HARDWARE ISR ACTIVE)
FF32 7504 JNZ HL_INT
FF33 D0F MOV AH,OFFH
FF34 EA0A JMP SHORT SET_INTR_Flag ; SET FLAG TO FF IF NON-HW
FF3A 800 MOV HL_INT:
FF3E E421 IN AL,INTAB1 ; GET MASK VALUE
FF40 0AC4 OR AL,AL ; MASK OFF_LVL BEING SERVICED
FF42 E620 OUT INTAB0.AL
FF44 0000 JMP SHORT SET_INTR_FLAG: ; SET FLAG TO FF IF NON-HW
FF4A 80266B00 MOV INTAB0.AL,AH ; SET FLAG
FF4B 50 POP AX : RESTORE REG AX CONTENTS
FF4C 5A POP DX
FF4D 1F POP DS
FF4E 81 DUMMY_RETURN: ; NEED IRET FOR VECTOR TABLE
FF4F 80 IRET
FF50 D11 ENDP
FF51
FF5B 81 ; DUMMY RETURN FOR ADDRESS COMPATIBILITY :
FF5C 87

FF5D ORG OFF58H
FF5E IRET

FF61 ; INT S --------------------------------------------------
FF62 ; THIS LOGIC WILL BE INVOKED BY INTERRUPT 65H TO PRINT THE :
FF63 ; SCREEN. THE CURSOR POSITION AT THE TIME THIS ROUTINE IS INVOKED :
FF64 ; WILL BE SAVED AND RESTORED UPON COMPLETION. THE ROUTINE IS :
FF65 ; INTENDED TO RUN WITH INTERRUPTS ENABLED. IF A SUBSEQUENT :
FF66 ; ‘PRINT SCREEN’ KEY IS DEPRESSED DURING THE TIME THIS ROUTINE :
FF67 ; IS PRINTING IT WILL BE IGNORED. :
FF68 ; ADDRESS 50:0 CONTAINS THE STATUS OF THE PRINT SCREEN: :
FF69 ; THIS LATER FOR CURSOR LIMITS :
FF6A ; ORIGIN MARKED AS PRINT SCREEN IS IN PROGRESS :
FF6B ; 25S ERROR ENCOUNTERED DURING PRINTING :

ASSUME CS:CODE,DS:ODATA
FF6C ORG OFF58H
FF6D PRINT_SCREEN PROC FAR
FF6E STI ; MUST RUN WITH INTERRUPTS ENABLED
FF6F 50 PUSH DS ; MUST USE 50:0 FOR DATA AREA STORAGE
FF75 5A PUSH BX
FF76 58 PUSH CX
FF77 51 PUSH DX
FF78 8500 MOV AX,ODATA
FF79 D068 MOV DS,AX
FF7A 80E0 CMP STATUS_BYTE,1 ; SEE IF PRINT ALREDY IN PROGRESS
FF7B 4F JNZ STATUS_BYTE,1 ; JUMP IF PRINT ALREDY IN PROGRESS
FF7C 8600 MOV STATUS_BYTE,1 ; INDICATE PRINT NOW IN PROGRESS
FF7D 8600 MOV AX,15 ; WILL REQUEST THE CURRENT SCREEN MODE

System BIOS A-81
AT THIS POINT WE KNOW THE COLUMNS/LINE ARE IN [AX] AND THE PAGE IF APPLICABLE IN [BH]. THE STACK HAS DS,AX,BX,CX,DX PUSHED. [AX] HAS VIDEO MODE : [AH]-MODE

WILL MAKE USE OF [CX] REGISTER TO MAKE USE OF [CX] REGISTER TO

CONTROL ROW & COLUMNS

CALL CRLF ; CARRIAGE RETURN LINE FEED ROUTINE

SAVE SCREEN BOUNDS

MOV AH,3 ; WILL NOW READ THE CURSOR.

INT 10H ; AND PRESERVE THE POSITION

POP CX ; RECALL SCREEN BOUNDS

PUSH DX ; RECALL [BH]-VISUAL PAGE

XOR DX,DX ; WILL SET CURSOR POSITION TO (0,0)

THE LOOP FROM PRI10 TO THE INSTRUCTION PRIOR TO PRI20 :

IS THE LOOP TO READ EACH CURSOR POSITION FROM THE

SCREEN AND PRINT.

MAKE A BLANK

SAVE CURSOR POSITION

INDICATE PRINTER 1

INDICATE PRINT CHAR IN [AH]

PRINT THE CHARACTER

RECALL CURSOR POSITION

TEST AH, 25H ; TEST FOR PRINTER ERROR

JNZ ERR10 ; JUMP IF ERROR DETECTED

INC DL ; ADVANCE TO NEXT COLUMN

CMP CL,DL ; SEE IF AT END OF LINE

JNZ PRI15 ; JUMP IF VALID CHAR

MOV AL,1 ; MAKE A BLANK

PUSH DX ; SAVE CURSOR POSITION

XOR DX,DX ; INDICATE PRINTER 1

XOR AH, AH ; TO INDICATE PRINT CHAR IN [AH]

MOV AH, 3 ; TO INDICATE READ CHARACTER

INT 10H ; CHARACTER NOW IN [AL]

OR AL, AL ; SEE IF VALID CHAR

JNZ PRI15 ; JUMP IF VALID CHAR

MOV AL, 1 ; MAKE A BLANK

PUSH DX ; SAVE CURSOR POSITION

XOR DX, DX ; INDICATE PRINTER 1

XOR AH, AH ; TO INDICATE PRINT CHAR IN [AH]

MOV AH, 3 ; TO INDICATE READ CHARACTER

INT 10H ; CHARACTER NOW IN [AL]

POP AX ; RECALL CURSOR POSITION

POP DX ; RECALL CURSOR POSITION

JMP CRLF ; LINE FEED CARRIAGE RETURN

JNZ PRI11 ; IF NOT PROCEED

INC DL ; ADVANCE TO NEXT COLUMN

INC AH ; advance position

CMP AH, 0 ; IF LAST COLUMN

JNZ PRI10 ; IF NOT CONTINUE

POP AL ; RECALL CURSOR POSITION

MOV AH, 2 ; TO INDICATE CURSOR SET REQUEST

INT 10H ; CURSOR POSITION RESTORED

MOV STATUS_BYTE, 0 ; INDICATE FINISHED

JMP SHORT EXIT ; EXIT THE ROUTINE

JMP SHORT EXIT ; EXIT THE ROUTINE

GET CURSOR POSITION

MOV AH, 2 ; TO REQUEST CURSOR SET

INT 10H ; CURSOR POSITION RESTORED

MOV STATUS_BYTE, OFFH ; INDICATE ERROR

EXIT:

RESTORE ALL THE REGISTERS USED

IF PRINT CYCLE

INT 19H ; PRINT SCREEN ENDP

----- CARRIAGE RETURN, LINE FEED SUBROUTINE

----- CARRIAGE RETURN, LINE FEED SUBROUTINE

CRLF PROC NEAR

POP DX ; PRINTER 0

XOR DX,DX ; WILL NOW SEND INITIAL LF,CR

POP AX ; TO PRINTER

POP DS ; TO PRINTER

MOV AL, 120 ; LF

A-82 System BIOS
PRINT A SEGMENT VALUE TO LOOK LIKE A 20 BIT ADDRESS:

PRINT A '0'

PRINT A '1'

PRINT A SPACE

POWER ON RESET VECTOR:

POWER ON RESET:

JMP

DB '11/08/82'

VECTOR ENDS

END
$TITLE(FIXED DISK BIOS FOR IBM DISK CONTROLLER)$

<table>
<thead>
<tr>
<th>Line</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>INT 13 ---------------------------------------------</td>
</tr>
<tr>
<td>2</td>
<td>FIXED DISK I/O INTERFACE</td>
</tr>
<tr>
<td>3</td>
<td>THIS INTERFACE PROVIDES ACCESS TO 5 1/4&quot; FIXED DISKS</td>
</tr>
<tr>
<td>4</td>
<td>THROUGH THE IBM FIXED DISK CONTROLLER.</td>
</tr>
<tr>
<td>5</td>
<td>----------------------------------------------------------------</td>
</tr>
<tr>
<td>6</td>
<td>THE BIOS ROUTINES ARE MEANT TO BE ACCESSED THROUGH</td>
</tr>
<tr>
<td>7</td>
<td>SOFTWARE INTERRUPTS ONLY. ANY ADDRESSES PRESENT IN</td>
</tr>
<tr>
<td>8</td>
<td>THE LISTINGS ARE INCLUDED ONLY FOR COMPLETENESS.</td>
</tr>
<tr>
<td>9</td>
<td>NOT FOR REFERENCE. APPLICATIONS WHICH REFERENCE</td>
</tr>
<tr>
<td>10</td>
<td>ABSOLUTE ADDRESSES WITHIN THE CODE SEGMENT: VIOLATE</td>
</tr>
<tr>
<td>11</td>
<td>THE STRUCTURE AND DESIGN OF BIOS.</td>
</tr>
<tr>
<td>12</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>13</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>14</td>
<td>The BIOS routines are meant to be accessed through</td>
</tr>
<tr>
<td>15</td>
<td>software interrupts only. Any addresses present in</td>
</tr>
<tr>
<td>16</td>
<td>the listings are included only for completeness,</td>
</tr>
<tr>
<td>17</td>
<td>not for reference. applications which reference</td>
</tr>
<tr>
<td>18</td>
<td>absolute addresses within the code segment:</td>
</tr>
<tr>
<td>19</td>
<td>violate the structure and design of BIOS.</td>
</tr>
<tr>
<td>20</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>21</td>
<td>INPUT (AH = HEX VALUE)</td>
</tr>
<tr>
<td>22</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>23</td>
<td>(AH)=00 RESET DISK (DL=80H, 81H) / DISKETE</td>
</tr>
<tr>
<td>24</td>
<td>(AH)=01 READ THE STATUS OF THE LAST DISK OPERATION</td>
</tr>
<tr>
<td>25</td>
<td>INTO (AL) NOTE: DL &lt; 80H - DISKETE DL &gt; 80H - DISK</td>
</tr>
<tr>
<td>26</td>
<td>(AH)=02 READ THE DESIRED SECTORS INTO MEMORY</td>
</tr>
<tr>
<td>27</td>
<td>(AH)=03 WRITE THE DESIRED SECTORS FROM MEMORY</td>
</tr>
<tr>
<td>28</td>
<td>(AH)=04 VERIFY THE DESIRED SECTORS</td>
</tr>
<tr>
<td>29</td>
<td>(AH)=05 FORMAT THE DESIRED TRACK</td>
</tr>
<tr>
<td>30</td>
<td>(AH)=06 FORMAT THE DESIRED TRACK AND SET BAD SECTOR</td>
</tr>
<tr>
<td>31</td>
<td>(AH)=07 FORMAT THE DRIVE STARTING AT THE DESIRED</td>
</tr>
<tr>
<td>32</td>
<td>TRACK (AH)=08 RETURN THE CURRENT DRIVE PARAMETERS</td>
</tr>
<tr>
<td>33</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>34</td>
<td>(AH)=09 INITIALIZE DRIVE PAIR CHARACTERISTICS</td>
</tr>
<tr>
<td>35</td>
<td>INTERRUPT 41 POINTS TO DATA BLOCK</td>
</tr>
<tr>
<td>36</td>
<td>(AH)=0A READ LONG</td>
</tr>
<tr>
<td>37</td>
<td>(AH)=0B WRITE LONG</td>
</tr>
<tr>
<td>38</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>39</td>
<td>NOTE: READ AND WRITE LONG ENCOMPASS 512 + 4 BYTES</td>
</tr>
<tr>
<td>40</td>
<td>ECC</td>
</tr>
<tr>
<td>41</td>
<td>(AH)=0C SEEK</td>
</tr>
<tr>
<td>42</td>
<td>(AH)=0D ALTERNATE DISK RESET (SEE DL)</td>
</tr>
<tr>
<td>43</td>
<td>(AH)=0E READ SECTOR BUFFER</td>
</tr>
<tr>
<td>44</td>
<td>(AH)=0F WRITE SECTOR BUFFER.</td>
</tr>
<tr>
<td>45</td>
<td>(AH)=10 RECOMMENDED PRACTICE BEFORE FORMATTING</td>
</tr>
<tr>
<td>46</td>
<td>(AH)=11 DRIVE READY</td>
</tr>
<tr>
<td>47</td>
<td>(AH)=12 CONTROLLER RAM DIAGNOSTIC</td>
</tr>
<tr>
<td>48</td>
<td>(AH)=13 DRIVE DIAGNOSTIC</td>
</tr>
<tr>
<td>49</td>
<td>(AH)=14 CONTROLLER INTERNAL DIAGNOSTIC</td>
</tr>
<tr>
<td>50</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>51</td>
<td>REGISTERS USED FOR FIXED DISK OPERATIONS</td>
</tr>
<tr>
<td>52</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>53</td>
<td>(DL) - DRIVE NUMBER (00H-7FH FOR DISK, VALUE CHECKED)</td>
</tr>
<tr>
<td>54</td>
<td>(OH) - HEAD NUMBER (0-7 ALLOWED, NOT VALUE CHECKED)</td>
</tr>
<tr>
<td>55</td>
<td>(CH) - CYLINDER NUMBER (0-1023, NOT VALUE CHECKED)</td>
</tr>
<tr>
<td>56</td>
<td>(CL) - SECTOR NUMBER (1-17, NOT VALUE CHECKED)</td>
</tr>
<tr>
<td>57</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>58</td>
<td>NOTE: HIGH 2 BITS OF CYLINDER NUMBER ARE PlACED</td>
</tr>
<tr>
<td>59</td>
<td>IN THE HIGH 2 BITS OF THE CL REGISTER (10 BITS TOTAL)</td>
</tr>
<tr>
<td>60</td>
<td>(AL) - NUMBER OF SECTORS (MAXIMUM POSSIBLE RANGE 1-80H,</td>
</tr>
<tr>
<td>61</td>
<td>FOR READ/ WRITE LONG 1-7FH)</td>
</tr>
<tr>
<td>62</td>
<td>(ES:BX) - ADDRESS OF BUFFER FOR READS AND WRITES.</td>
</tr>
<tr>
<td>63</td>
<td>(ES:BX) - ADDRESS OF BUFFER FOR READS AND WRITES.</td>
</tr>
<tr>
<td>64</td>
<td>(ES:BX) - ADDRESS OF BUFFER FOR READS AND WRITES.</td>
</tr>
<tr>
<td>65</td>
<td>(ES:BX) - ADDRESS OF BUFFER FOR READS AND WRITES.</td>
</tr>
<tr>
<td>66</td>
<td>(ES:BX) - ADDRESS OF BUFFER FOR READS AND WRITES.</td>
</tr>
<tr>
<td>67</td>
<td>OUTPUT</td>
</tr>
<tr>
<td>68</td>
<td>AH = STATUS OF CURRENT OPERATION</td>
</tr>
<tr>
<td>69</td>
<td>STATUS BITS ARE DEFINED IN THE EQUATES BELOW</td>
</tr>
<tr>
<td>70</td>
<td>CY = 0 SUCCESSFUL OPERATION (AH=0 ON RETURN)</td>
</tr>
<tr>
<td>71</td>
<td>CY = 1 FAILED OPERATION (AH HAS ERROR REASON)</td>
</tr>
<tr>
<td>72</td>
<td>!NOTE: ERROR 1H INDICATES THAT THE DATA READ HAD A</td>
</tr>
</tbody>
</table>
| 73   | RECOVERABLE ERROR WHICH WAS CORRECTED BY THE ECC ALGORITHM. THE DATA IS PROBABLY GOOD. HOWEVER THE BIOS ROUTINE INDICATES AN ERROR TO ALLOW THE CONTROLLING PROGRAM A CHANCE TO DECIDE FOR ITSELF. THE ERROR MAY NOT RECUR IF THE DATA IS
REWITTEN. (AL) CONTAINS THE BURST LENGTH.

IF DRIVE PARAMETERS WERE REQUESTED,

DL = NUMBER OF CONSECUTIVE ACKNOWLEDGING DRIVES ATTACHED (0-2)

(CONTROLLER CARD ZERO TALLY ONLY)

DH = MAXIMUM USEABLE VALUE FOR HEAD NUMBER

CH = MAXIMUM USEABLE VALUE FOR CYLINDER NUMBER

CL = MAXIMUM USEABLE VALUE FOR SECTOR NUMBER

AND CYLINDER NUMBER HIGH BITS

REGISTERS WILL BE PRESERVED EXCEPT WHEN THEY ARE USED TO RETURN

INFORMATION.

NOTE: IF AN ERROR IS REPORTED BY THE DISK CODE, THE APPROPRIATE

ACTION IS TO RESET THE DISK, THEN RETRY THE OPERATION.

-----------------------------------------------------------------------------------

00FF

SENSE_FAIL EQU OFFH ; SENSE OPERATION FAILED

0000

UNDER_ERR EQU O08H ; UNDEFINED ERROR OCCURRED

0001

TIME_OUT EQU 00FH ; ATTACHMENT FAILED TO RESPOND

0002

BAD_SEEK EQU 040H ; SEEK OPERATION FAILED

0011

BAD_CNTL EQU 020H ; CONTROLLER HAS FAILED

0012

DATA_CORRECTED EQU 011H ; ECC CORRECTED DATA ERROR

0013

BAD_ECC EQU 010H ; BAD ECC ON DISK READ

0018

BAD_TRACK EQU 00EH ; BAD TRACK FLAG DETECTED

0019

DATA_BOUNDARY EQU 008H ; ATTEMPT TO DMA ACROSS 64K BOUNDARY

001A

INIT_FAIL EQU 007H ; DRIVE PARAMETER ACTIVITY FAILED

001B

BAD_RESET EQU 005H ; RESET FAILED

001C

RECORD_NOT_FOUND EQU 004H ; REQUESTED SECTOR NOT FOUND

001D

BAD_ADDR_MARK EQU 002H ; ADDRESS MARK NOT FOUND

001E

BAD_CMD EQU 013H ; BAD COMMAND PASSED TO DISK I/O

-----------------------------------------------------------------------------------

< Appendix A >

---

0034

ORG 00H*4 ; FIXED DISK INTERRUPT VECTOR

0036

HOISK_INT LABEL DWORD ; DISK INTERRUPT VECTOR

0040

ORG_VECTOR LABEL DWORD ; DISKETTE INTERRUPT VECTOR

004C

ORG 10H*4 ; DISKETTE INTERRUPT VECTOR

0056

BOOT_VEC LABEL DWORD ; DISKETTE INTERRUPT VECTOR

0060

DISKETTE_PARM LABEL DWORD ; DISKETTE INTERRUPT VECTOR

0064

DISK_LABEL LABEL DWORD ; NEW DISKETTE INTERRUPT VECTOR

0068

DISK_VECTOR LABEL DWORD ; FIXED DISK PARAMETER VECTOR

006C

HF_TBL_VEC LABEL DWORD ; FIXED DISK PARAMETER VECTOR

0070

DISKETTE_STATUS ; FIXED DISK STATUS BYTE

0074

HF_NUM ? ? ? ; COUNT OF FIXED DISK DRIVES

0078

CONTROL_BYTE DB ? ; CONTROL BYTE DRIVE OPTIONS

007C

PORT_OFFSET DB ? ; PORT OFFSET

---

0090

DATA_SEGMENTS AT 40H

0094

ORG 42H ; FIXED DISK INTERRUPT VECTOR

0098

CMBLOCK LABEL BYTE ; DISKETTE INTERRUPT VECTOR

009C

NO_ERROR DB ? DUP(); OVERLAYS DISKETTE STATUS

00A0

TIMER_LOW DW ? ; TIMER LOW WORD

00A4

RESET_FLAG DW ? ; 123H IF KEYBOARD RESET UNDERRUN

00A8

DISK_STATUS DB ? ; FIXED DISK STATUS BYTE

00AC

DB ? ; COUNT OF FIXED DISK DRIVES

00AE

CONTROL_BYTE DB ? ; CONTROL BYTE DRIVE OPTIONS

00B2

PORT_OFFSET DB ? ; PORT OFFSET

---

00C0

CODE SEGMENT

00C4

HARDWARE SPECIFIC VALUES

---

150

FIXED DISK BIOS  A-85
A-86  Fixed Disk BIOS
0027 FA 220 CLI
0028 A14C00 229 MOV AX, WORD PTR ORG_VECTOR ; GET DISKETTE VECTOR
0029 B30001 230 MOV WORD PTR DISK_VECTOR, AX ; INTO INT 40H
002A E0400 231 MOV AX, WORD PTR ORG_VECTOR+2
0031 A30001 232 MOV WORD PTR DISKVECTOR+2, AX
0032 C7064C0008 233 MOV WORD PTR ORG_VECTOR, OFFSET DISK_ID ; HDISK HANDLER
0033 C06E400 234 MOV WORD PTR ORG_VECTOR+2, CS
0034 E06007 235 MOV AX, OFFSET HD_INT ; HDISK INTERRUPT
0035 A30001 236 MOV WORD PTR HDISK_INT, AX
0036 C06E3600 237 MOV WORD PTR HDISK_INT+2, CS
0037 6066460000 1 MOV WORD PTR BOOT_VEC.OFFSET BOOT Strap ; BOOTSTRAP
0038 C06C6600 239 MOV WORD PTR BOOT_VEC+2, CS
0039 C7064D4170D3 240 MOV WORD PTR HF_TBL_VEC.OFFSET FD_TBL ; PARAMETER TBL
003A C06E0601 241 MOV WORD PTR HF_TBL_VEC+2, CS
003B 5D 242 STI

003C 243
003D B04000 245 MOV AX, DATA ; ESTABLISH SEGMENT
003E 06006D 246 MOV DS, AX
003F C606760000 247 MOV DISK_STATUS, O ; RESET THE STATUS INDICATOR
0040 C606750000 248 MOV HF_NUM, O ; ZERO COUNT OF DRIVES
0041 C606430000 249 MOV CHD_BLOCK+1, O ; DRIVE ZERO, SET VALUE IN BLOCK
0042 C606770000 250 MOV PORT_OFF, O ; ZERO CARD OFFSET
0043 251
0044 252
0045 B92500 254 MOV DX, 25H ; RETRY COUNT
0046 0079 255 L4: CALL HD_RESET_1 ; RESET CONTROLLER
0047 7C 7305 256 JNC L7 ; TRY RESET AGAIN
0048 7E E2F9 257 LOOP L6
0049 E0 89F0 258 JMP ERROR_EX
004A 259
004B 03 25A MOV CX, 25H
004C 090100 25B MOV CX, 1
004D 8A0000 25C MOV DX, 00H
004E 25D
004F B80012 262 MOV AX, 1200H ; CONTROLLER DIAGNOSTICS
0050 CC 0D13 263 INT 13H
0051 08 E730 264 JNC P7
0052 9A EF90 265 JMP ERROR_EX
0053 266
0054 0909 B00014 267 MOV AX, 1400H ; CONTROLLER DIAGNOSTICS
0055 96 CD13 268 INT 13H
0056 90 7303 269 JNC P9
0057 9A EA5000 270 JMP ERROR_EX
0058 271
0059 0D C7066C000000 272 MOV AX, TIMER_LOW_0 ; ZERO TIMER
005A A3 17200 273 MOV AX, reset_FLAG
005B A6 3D3112 274 CMP AX, 1234H ; KEYBOARD RESET
005C 49 7506 275 JNE P5
005D 00AB C7066C09A01 276 MOV AX, TIMER_LOW+100
005E 80B1 277 P6: MOV AX, TIMER_LOW+100
005F 80B1 278 IN AL, 021H ; TIMER
0060 83 24FE 279 AND AL, 0FEH ; ENABLE TIMER
0061 8065 E621 280 OUT 021H, AL ; START TIMER
0062 80B7 281 P4: CALL HD_RESET_1 ; RESET CONTROLLER
0063 80BA 7207 282 JC P10
0064 0B0C D0010 283 MOV AX, 1000H ; READY
0065 80BF CD13 284 INT 13H
0066 0C01 7300 285 JNC P2
0067 0C03 870100 286 MOV AX, TIMER_LOW
0068 0C06 3D00E1 287 CMP AX, 1466D ; 25 SECONDS
0069 0C09 72EC 288 JB P4
006A 0C0C 87590 289 JMP ERROR_EX
006B 0C0E B90100 290 MOV AX, CK_1
006C 0D1A 8A0000 291 MOV AX, TIMER_LOW
006D 292
006E B80011 293 MOV AX, 1100H ; RECALIBRATE
006F 0070 CD13 294 INT 13H
0070 0079 7267 295 JC ERROR_EX
0071 296
0072 B80009 300 MOV AX, 0900H ; SET DRIVE PARAMETERS
0073 007E CD13 301 INT 13H
0074 00E0 7260 302 JC ERROR_EX
0075 303
0076 B800C8 304 MOV AX, 0C0800H ; DMA TO BUFFER
LOC OBJ | LINE | SOURCE
-------------------|-------|---------------------
0065 REC0 | 305   | MOV ES,AX           | ; SET SEGMENT
0067 28DB | 306   | SUB BX,BX          |  
0069 B0000F | 307   | MOV AX,0F00H       | ; WRITE SECTOR BUFFER
006C C013 | 308   | INT 13H            |  
00EE 7252 | 309   | JC ERROR_EX        |  
00F0 FE067500 | 310  | INC HF_NM          | ; DRIVE ZERO RESPONDED
00F4 BAI032 | 311   | MOV DX,213H        | ; EXPANSION BOX
00F7 B000 | 312   | MOV AL,0           |  
00F9 EE | 313   | OUT DX,AL          | ; TURN BOX OFF
00FA BAI031 | 314   | MOV DX,321H        | ; TEST IF CONTROLLER
00FD EC | 315   | IN AL,DX           | ... IS IN THE SYSTEM UNIT
00FE D00F | 316   | AND AL,0FH         |  
0100 C02F | 317   | CMP AL,0FH         |  
0102 7406 | 318   | JE BOX_ON          |  
0104 67066C00A401 | 319 | MOV TIMER_LOW,420D | ; CONTROLLER IS IN SYSTEM UNIT
010A | 320   | BOX_ON:            |  
010A BAI032 | 321   | MOV DX,213H        | ; EXPANSION BOX
010F EE | 322   | MOV AL,0FFH        |  
0110 B90100 | 323  | OUT DX,AL          | ; TURN BOX ON
0113 B0100 | 324   | MOV DX,081H        |  
0116 | 325   | PS:                |  
0116 B8C0 | 326   | SUB AX,AX          | ; RESET
0118 C013 | 327   | INT 13H            |  
011A 7240 | 328   | JC POD_DONE        |  
011C B0011 | 329   | MOV AX,01100H      | ; RECAL
011F C013 | 330   | INT 13H            |  
0121 730B | 331   | JNC PS             |  
0123 A16C00 | 332  | MOV AX,TIMER_LOW   |  
0126 300B00 | 333  | CMP AX,4460        | ; 25 SECONDS
0129 72EF | 334   | JB P3              |  
012B 8A1302 | 335  | MOV DX,213H        | ; EXPANSION BOX
0130 7310 | 336   | JA POD_DONE        |  
0133 FA0100 | 337  | MOV DX,081H        |  
0136 28CO | 338   | SUB AX,AX          | ; RESET
0138 0600 | 339   | MOV DX,081H        |  
013B | 340   | PS:                |  
013E B00009 | 341   | MOV AX,0900H       | ; INITIALIZE CHARACTERISTICS
0141 C013 | 342   | INT 13H            |  
0143 7240 | 343   | JC POD_DONE        |  
0145 FE067500 | 344 | INC HF_NM          | ; TALLY ANOTHER DRIVE
0149 61FA8100 | 345 | CMP DX,(0FH + 5_MAX_FILE - 1) |  
014D 7310 | 346   | JAE POD_DONE       |  
0150 42 | 347   | INC DX             |  
0154 8D04 | 348   | JMP P3             |  
0159 | 349   |                   |  
015E | 350   |                   | ;----- POD ERROR
0162 ERROR: | 351   |                   |  
0164 20F00 | 352   | MOV BP,0FH         | ; POD ERROR FLAG
0166 28C0 | 353   | MOV AX,0           |  
0167 8BF0 | 354   | MOV SI,AX          |  
0169 B9060090 | 355 | MOV CX:F17L        | ; MESSAGE CHARACTER COUNT
016D B700 | 356   | MOV BH,0           | ; PAGE ZERO
016F | 357   |                   |  
0170 62CF4 | 358   | LOOP OUT.CH        | ; DO MORE
0179 FA | 359   |                   |  
017B | 360   |                   | ; POD_DONE:
017D FA | 361   |                   |  
017F E421 | 362   | MOV AH,140         | ; VIDEO OUT
0184 C010 | 363   | INT 10H            | ; DISPLAY CHARACTER
0188 46 | 364   | INC SI             | ; NEXT CHAR
0189 27FA | 365   | LOOP OUT.CH        | ; DO MORE
018B F9 | 366   | STC                |  
018D 351    | 367   |                   |  
0191 31373031  | 368   |                   |  
0196 DD  | 369   |                   |  
019A 0A | 370   |                   |  
019C | 371   |                   |  
019E E421 | 372   | MOV AL,021H        | ; BE SURE TIMER IS DISABLED
019F 8C81 | 373   | OR AL,01H          |  
01A1 E421 | 374   | OUT 021H,AL        |  
01A3 FB | 375   | STI                |  
01A4 6A500 | 376   | CALL DSBL          |  
01A7 CB | 377   | RET                |  
01AD | 378   |                   |  
01B0 31373031 | 379  | F17 DB '1701',00H,0AH |  
01B6 0D | 380   |                   |  
01B8 DA | 381   |                   |  
01C2 0F | 382   |                   |  
01C4 0F | 383   |                   |  
01C6 | 384   |                   |  
01C8 | 385   |                   |  
01CA 0F | 386   |                   |  
01CB 0F | 387   |                   |  
01CD | 388   |                   |  
01CE | 389   |                   |  
01CF | 390   |                   |  
01D0 0F | 391   |                   |  
01D2 0F | 392   |                   |  
01D3 | 393   |                   |  
01D4 | 394   |                   |  
01D5 | 395   |                   |  
01D6 | 396   |                   |  
01D7 | 397   |                   |  
01D8 | 398   |                   |  
01D9 | 399   |                   |  
01DA | 391   |                   |  
01DB | 392   |                   |  
01DC | 393   |                   |  
01DD | 394   |                   |  
01DE | 395   |                   |  
01DF | 396   |                   |  
01E0 | 397   |                   |  
01E1 | 398   |                   |  
01E2 | 399   |                   |  
01E3 | 400   |                   |  
01E4 | 401   |                   |  
01E5 | 402   |                   |  
01E6 | 403   |                   |  
01E7 | 404   |                   |  
01E8 | 405   |                   |  
01E9 | 406   |                   |  
01EA | 407   |                   |  
01EB | 408   |                   |  
01EC | 409   |                   |  
01ED | 410   |                   |  
01EE | 411   |                   |  
01EF | 412   |                   |  
01F0 | 413   |                   |  
01F1 | 414   |                   |  
01F2 | 415   |                   |  
01F3 | 416   |                   |  
01F4 | 417   |                   |  
01F5 | 418   |                   |  
01F6 | 419   |                   |  
01F7 | 420   |                   |  
01F8 | 421   |                   |  
01F9 | 422   |                   |  
01FA | 423   |                   |  
01FB | 424   |                   |  
01FC | 425   |                   |  
01FD | 426   |                   |  
01FE | 427   |                   |  
01FF | 428   |                   |  
0200 88 Fixed Disk BIOS

A-88 Fixed Disk BIOS
THE FIXED DISK BOOT STRAP >

A Pointer to this and Diskette Parameter Vectors

The Interrupt 19

The Diskette Table

The Last Two Bytes of the Block

If the above fails control is passed to resident basic

1---- ATTEMPT BOOTSTRAP FROM DISKETTE

IPL_SYSTEM

5 Save Retry Count

Drive Zero

Try Fixed Disk

Unable to IPL from the Diskette

IPL was successful
0100 E007C0000 457 JMP BOOT_LOCN
0100 E007C0000 458
0100 E007C0000 459 i----- ATTEMPT BOOTSTRAP FROM FIXED DISK
0100 E007C0000 460
0100 E007C0000 461 H5:
0100 E007C0000 462 SUB AX,AX ; RESET DISKETTE
0100 E007C0000 463 MOV DX,DX
0100 E007C0000 464 INT 13H ; SET RETRY COUNT
0100 E007C0000 465 MOV CX,3 ; IPL SYSTEM
0100 E007C0000 466 H6:
0100 E007C0000 467 PUSH CX ; SAVE RETRY COUNT
0100 E007C0000 468 MOV DX,0001H ; FIXED DISK ZERO
0100 E007C0000 469 MOV AX,AX ; RESET THE FIXED DISK
0100 E007C0000 470 INT 13H ; FILE ID CALL
0100 E007C0000 471 JC H7 ; IF ERROR, TRY AGAIN
0100 E007C0000 472 MOV AX,0201H ; READ IN THE SINGLE SECTOR
0100 E007C0000 473 SUB AX,AX ; RESET DISKETTE
0100 E007C0000 474 MOV ES,ES
0100 E007C0000 475 MOV BX,OFFSET BOOT_LOCN+510D
0100 E007C0000 476 MOV AX,WORD prR BOOT_LOCN+510D
0100 E007C0000 477 H7:
0100 E007C0000 478 POP AX ; RECOVER RETRY COUNT
0100 E007C0000 479 LOOP H6 ; DO IT FOR RETRY TIMES
0100 E007C0000 480 H8:
0100 E007C0000 481 MOV AX,WORD prR BOOT.LOCN+510D
0100 E007C0000 482 CMP AX,0AABH ; TEST FOR GENERIC BOOT BLOCK
0100 E007C0000 483 INT 13H ; FILE CALL
0100 E007C0000 484 JPF2
0100 E007C0000 485 DB 510H ; DRIVE NUMBER
0100 E007C0000 486 890100
0100 E007C0000 487 MOV BX,OFFSET DISKETTE_TBL
0100 E007C0000 488 MOV BX,OFFSET DISKETTE_TBL
0100 E007C0000 489 DB 11001111B ; SRT+C, HD UNLOAD=OFF - 1ST SPEC BYTE
0100 E007C0000 490 DB 2 ; HD LOAD=1, MODE=MDM - 2ND SPEC BYTE
0100 E007C0000 491 DB 25H ; WAIT AFTER GPM TIL MOTOR OFF
0100 E007C0000 492 DB 2 ; 512 BYTES PER SECTOR
0100 E007C0000 493 DB 6 ; END (LAST SECTOR ON TRACK)
0100 E007C0000 494 DB 02H ; GAP LENGTH
0100 E007C0000 495 DB 0FH ; DTL
0100 E007C0000 496 DB 650H ; GAP LENGTH FOR FORMAT
0100 E007C0000 497 DB 06H ; FILL BYTE FOR FORMAT
0100 E007C0000 498 DB 25 ; HEAD SETTLE TIME (MILLISECONDS)
0100 E007C0000 499 DB 4 ; MOTOR START TIME (1/8 SECOND)
0100 E007C0000 500 DB 07H ; SAVE OFFSET
0100 E007C0000 501 MOV AX,PORT_OFF
0100 E007C0000 502 860400
0100 E007C0000 503 MOV AX,DATA ; SAVE SEGMENT
0100 E007C0000 504 8F0D00
0100 E007C0000 505 MOV AX,DATA
0100 E007C0000 506 8F0D00
0100 E007C0000 507 MOV AX,DATA ; SAVE OFFSET
0100 E007C0000 508 8F0D00
0100 E007C0000 509 MOV AX,DATA
0100 E007C0000 510 MOV AX,DATA
0100 E007C0000 511 MOV AX,DATA
0100 E007C0000 512 MOV AX,DATA
0100 E007C0000 513 MOV AX,PORT_OFF
0100 E007C0000 514 PUSH AX ; SAVE OFFSET
0100 E007C0000 515 DB 06H ; PORT_OFFSET
0100 E007C0000 516 MOV PORT_OFFSET,OH
0100 E007C0000 517 CALL PORT_3
0100 E007C0000 518 DB 02H ; SUB AL,AL
0100 E007C0000 519 OUT DX,AL ; RESET INT/DMA MASK
0100 E007C0000 520 MOV PORT_OFFSET,4H
0100 E007C0000 521 CALL PORT_3
0100 E007C0000 522 SUB AL,AL ; SET DMA MODE TO DISABLE
0100 E007C0000 523 MOV PORT_OFFSET,4H
0100 E007C0000 524 CALL PORT_3
0100 E007C0000 525 MOV PORT_OFFSET,4H
0100 E007C0000 526 CALL PORT_3
0100 E007C0000 527 OUT DX,AL ; SET DMA MODE TO DISABLE
0100 E007C0000 528 MOV PORT_OFFSET,4H
0100 E007C0000 529 CALL PORT_3
0100 E007C0000 530 MOV PORT_OFFSET,4H
0100 E007C0000 531 CALL PORT_3
0100 E007C0000 532 MOV PORT_OFFSET,4H
0100 E007C0000 533 OUT DMA+10,AL

A-90 Fixed Disk BIOS
FA  CLI  AL:021H  ; DISABLE INTERRUPTS
0421  IN  AL:020H  ; 02H
000C  OR  AL:02H  ; 02H
0016  OUT  021H:AL  ; DISABLE INTERRUPT 5
001F  STI  ; ENABLE INTERRUPTS
08  POP  AX  ; RESTORE OFFSET
700  MOV  PORT_OFF, AH  ; RESTORE SEGMENT
1F  POP  DS  ;
C3  RET
DSBL  ENDP
---------------------------------------------------------------------

0256  DISK_ID PROC FAR
540  ASSUME DS:NOTHING, ES:NOTHING
551  CMP  DL:0DH  ; TEST FOR FIXED DISK DRIVE
552  JAE  HARD_DISK  ; YES, HANDLE HERE
553  INT  40H  ; DISKETTE HANDLER
554  RET  ;
555  HARD_DISK:
556  ASSUME OS: DATA  ;
557  CMP  AH, 80H  ; GET PARAMETERS IS A SPECIAL CASE
558  JAE  GET_PARM_H
559  JMP  GET_PARM_M  ;
560  JMP  GET_PARM_L
561  JMP  GET_PARM_1
562  JMP  GET_PARM_2
563  JMP  GET_PARM_3
564  JMP  GET_PARM_4
565  JMP  GET_PARM_5
566  JMP  GET_PARM_6
567  JMP  GET_PARM_7
568  JMP  GET_PARM_8
569  JMP  GET_PARM_9
570  PUSH  BX  ; SAVE REGISTERS DURING OPERATION
571  PUSH  CX  ;
572  PUSH  DX  ;
573  PUSH  DS  ;
574  PUSH  ES  ;
575  PUSH  SI  ;
576  PUSH  DI  ;
577  CALL  DISK_IO_CONT  ; PERFORM THE OPERATION
578
579  86A0H  CALL  DISK_IO_CONT  ; PERFORM THE OPERATION
580  50  PUSH  AX  ; BE SURE DISABLES OCCURRED
581  860FF  CALL  DSBL  ;
582  80000H  MOV  AX:DATA  ; ESTABLISH SEGMENT
583  8206  MOV  DS:AX  ;
584  8B  POP  AX  ;
585  8267400H  MOV  AH:DISK_STATUS  ; GET STATUS FROM OPERATION
586  86FC01  CMP  AH, 1  ; SET THE CARRY FLAG TO INDICATE
587  89  8F  5A  JMP  GET_PARM_L  ; SUCCESS OR FAILURE
588  8A  POP  DI  ; RESTORE REGISTERS
589  89  5F  59  POP  SI  ;
58A  67  59  POP  ES  ;
58B  5F  59  POP  DS  ;
58C  5A  59  POP  DX  ;
58D  79  5A  POP  CX  ;
58E  5B  5A  POP  BX  ;
58F  5C  RET  Z  ; THROW AWAY SAVED FLAGS
590
591  DISK_IO ENDP
592
593  "FUNCTION TRANSFER TABLE"
594  "DISK_RESET"
595  "RETURN_STATUS"
596  "DISK_READ"
597  "DISK_WRITE"
598  "DISK_VERF"
599  "FMT_TRK"
600  "BAD_COMMAND"
601  "RO_LONG"
602  "WR_LONG"
603
604  "DOS_RESET"
605  "NORMAL"
A-92  Fixed Disk BIOS
Appendix A

Fixed Disk BIOS  A-93
LOC OBJ

LINE SOURCE

A-94 Fixed Disk BIOS
## Section A: Translation Table

### Section A.1: Drive Type 00

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>03E7</td>
<td>FD_TBL:</td>
</tr>
<tr>
<td>03E8</td>
<td>3201</td>
</tr>
<tr>
<td>03E9 02</td>
<td>OD</td>
</tr>
<tr>
<td>03EA 3201</td>
<td>OD</td>
</tr>
<tr>
<td>03EC 0000</td>
<td>OD</td>
</tr>
<tr>
<td>03EE 0B</td>
<td>OD</td>
</tr>
<tr>
<td>03EF 00</td>
<td>OD</td>
</tr>
<tr>
<td>03F0 0C</td>
<td>OD</td>
</tr>
<tr>
<td>03F1 04</td>
<td>OD</td>
</tr>
<tr>
<td>03F2 2D</td>
<td>OD</td>
</tr>
<tr>
<td>03F3 00000000</td>
<td>OD</td>
</tr>
</tbody>
</table>

### Section A.2: Drive Type 01

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>03F7 7701</td>
<td>OD</td>
</tr>
<tr>
<td>03F9 0B</td>
<td>OD</td>
</tr>
<tr>
<td>03FA 7701</td>
<td>OD</td>
</tr>
<tr>
<td>03FC 0000</td>
<td>OD</td>
</tr>
<tr>
<td>03FE 0D</td>
<td>OD</td>
</tr>
<tr>
<td>03FF 05</td>
<td>OD</td>
</tr>
<tr>
<td>0400 0C</td>
<td>OD</td>
</tr>
<tr>
<td>0401 04</td>
<td>OD</td>
</tr>
<tr>
<td>0402 2D</td>
<td>OD</td>
</tr>
<tr>
<td>0403 00000000</td>
<td>OD</td>
</tr>
</tbody>
</table>

### Section A.3: Drive Type 02

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0407 3201</td>
<td>OD</td>
</tr>
<tr>
<td>0409 06</td>
<td>OD</td>
</tr>
<tr>
<td>040A 0000</td>
<td>OD</td>
</tr>
<tr>
<td>040C 0001</td>
<td>OD</td>
</tr>
<tr>
<td>040E 0B</td>
<td>OD</td>
</tr>
<tr>
<td>040F 05</td>
<td>OD</td>
</tr>
<tr>
<td>0410 0C</td>
<td>OD</td>
</tr>
<tr>
<td>0411 04</td>
<td>OD</td>
</tr>
<tr>
<td>0412 2D</td>
<td>OD</td>
</tr>
<tr>
<td>0413 00000000</td>
<td>OD</td>
</tr>
</tbody>
</table>

### Section A.4: Drive Type 03

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0417 3201</td>
<td>OD</td>
</tr>
<tr>
<td>0419 04</td>
<td>OD</td>
</tr>
</tbody>
</table>

---

**Notes:**
- The default table is vectored in for AN INTERRUPT 19H (BOOTSTRAP).
- For DRIVE 0, DRIVE 1, etc., the corresponding vector is set into INTERRUPT 41.
- The values in the table are used to translate certain commands into INT 19H.
LOC OBJ LINE SOURCE
041A 3201 919 DH 0306D
041C 0000 920 DW 0000D
041E 0B 921 DB 00H
041F 05 922 DB 05H
0420 0C 923 DB 0CH ; STANDARD
0421 B4 924 DB 08AH ; FORMAT DRIVE
0422 20 925 DB 02H ; CHECK DRIVE
0423 00000000 926 DB 0.0.0.0
0427 928 INIT_DRV PROC NEAR
0429 930 ;------- DO DRIVE ZERO
0427 C606420000 932 MOV CMD_BLOCK+0,INIT_DRV_CMD
042C C6064300000 933 MOV CMD_BLOCK+1,0
0431 E81000 934 CALL INIT_DRV_R
0434 7200 935 JC INIT_DRV_OUT
0436 937 ;------- DO DRIVE ONE
0436 C606420000C 939 MOV CMD_BLOCK+0,INIT_DRV_CMD
043B C6064302000 940 MOV CMD_BLOCK+1,00100000B
0440 000100 941 CALL INIT_DRV_R
0443 942 INIT_DRV_OUT: 942
0444 C3 943 RET
0446 944 INIT_DRV ENDP
0448 945 ;------- SEND DRIVE PARAMETERS MOST SIGNIFICANT BYTE FIRST
0450 BF01000 946 MOV DI:1
0454 947 CALL INIT_DRV_S
0456 BF0200 948 JC B3
0458 BF0300 949 MOV DI:0
045B BF0400 950 JC B3
045E BF0500 951 MOV DI:3
0462 BF0600 952 JC B3
0466 BF0700 953 MOV DI:4
046A BF0800 954 JC B3
046E BF0900 955 MOV DI:5
0472 BF0A00 956 JC B3
0476 BF0B00 957 MOV DI:6
047A BF0C00 958 JC B3
047E BF0D00 959 MOV DI:7
0482 BF0E00 960 JC B3
0486 BF0F00 961 MOV DI:8
048A BF1000 962 JC B3
048E BF1100 963 MOV DI:9
0492 BF1200 964 JC B3
0496 BF1300 965 MOV DI:10
049A BF1400 966 JC B3
049E BF1500 967 MOV DI:11
04A2 BF1600 968 JC B3
04A6 BF1700 969 MOV DI:12
04A9 BF1800 970 JC B3
04AE BF1900 971 MOV DI:13
04B2 BF1A00 972 JC B3
04B6 BF1B00 973 MOV DI:14
04BC BF1C00 974 JC B3
04C2 BF1D00 975 MOV DI:15
04C7 BF1E00 976 JC B3
04D2 BF1F00 977 MOV DI:16
04D7 BF2000 978 JC B3
04D9 BF2100 979 MOV DI:17
04DC BF2200 980 JC B3
04E2 BF2300 981 MOV DI:18
04E7 BF2400 982 JC B3
04E9 BF2500 983 MOV DI:19
04E9 BF2600 984 JC B3
04EE BF2700 985 MOV DI:20
04F0 BF2800 986 JC B3
04F4 BF2900 987 MOV DI:21
04F9 BF2A00 988 JC B3
04FA BF2B00 989 MOV DI:22
04FB BF2C00 990 JC B3
04FC BF2D00 991 MOV DI:23
04FD BF2E00 992 JC B3
04FE BF2F00 993 MOV DI:24
04FF BF3000 994 JC B3
0500 BF3100 995 MOV DI:25

A-96 Fixed Disk BIOS
Loe OBJ LINE SOURCE

049B 7215 996 JC BS
049D BF00 998 MOV DI,8 ; DRIVE STEP OPTION
04AA 263A01 999 MOV AL,ES:[BX + DI]
04A3 A760 1000 MOV CONTROL_BYTE,AL
1001
04A6 2BC9 1002 SUB CX,CX
04A8 1003 BS:
04A8 E03D02 1004 CALL PORT_1
04AB 1C 1005 IN AL,DX
04AC A02 1006 TEST AL,RI IOMUX ; STATUS INPUT MODE
04AE 7599 1007 JNZ BS
04B0 E2F6 1008 LOOP BS
04B2 1009 BS:
04B2 C604740007 1010 MOV DISK_STATUS,INIT_FAIL ; OPERATION FAILED
04B7 F9 1011 STC
04B8 C3 1012 RET
1013
04B9 1014 BS:
04B9 E08502 1015 CALL PORT_0
04BC 1C 1016 IN AL,DX
04BD 2402 1017 JB AL, Z ; MASK ERROR BIT
04BF 75F1 1018 JNZ BS
04C1 C3 1019 RET
1020 ASSUME ES:NOTHING
1021 INIT_DRV_S ENDP
1022
1023 ; ----- SEND THE BYTE OUT TO THE CONTROLLER
1024
04C2 1025 INIT_DRV_S PROC NEAR
04C2 E0C501 1026 CALL HD_WAIT_REQ
04C5 7207 1027 JC DI
04C7 E0A702 1028 CALL PORT_0
04CA 26A401 1029 MOV AL,ES:[BX + DI]
04CD EE 1030 OUT DX,AL
04CE 1031 DI:
04CE C3 1032 RET
1033 INIT_DRV_S ENDP
1034
1035 ;---------------------------------------
1036 ; READ LONG (AH = OAH) :
1037 ;---------------------------------------
1038
04C9 1039 RD_LONG PROC NEAR
04C9 E19400 1040 CALL CHK_LONG
04D2 724B 1041 JC GB
04D4 C6042000E5 1042 MOV CHD_BLOCK+RD_LONG_CMD
04D9 B047 1043 MOV AL,DMA_READ
04DB EB46 1044 JMP SHORT DMA_OPN
1045 RD_LONG ENDP
1046
1047 ;---------------------------------------
1048 ; WRITE LONG (AH = OEH) :
1049 ;---------------------------------------
1050
04D6 1051 WR_LONG PROC NEAR
04D6 E00000 1052 CALL CHK_LONG
04D9 7250 1053 JC GB
04DB C6042000E6 1054 MOV CHD_BLOCK+WR_LONG_CMD
04DE B048 1055 MOV AL,DMA_WRITE
04F0 EB5A 1056 JMP SHORT DMA_OPN
1057 WR_LONG ENDP
1058
1059 ;---------------------------------------
1060 ; SEEK (AH = OCH) :
1061 ;---------------------------------------
1062
04F3 1063 INIT_DRV_S PROC NEAR
04F3 C6042000DB 1064 MOV CHD_BLOCK,SEEK_CMD
04F7 B33D 1065 JMP SHORT DMA_OPN
1066
1067 ;---------------------------------------
1068 ;---------------------------------------
1069
Fixed Disk BIOS  A-97
<table>
<thead>
<tr>
<th>LOC OBJ</th>
<th>LINE</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1073</td>
<td>DISK_SEEK</td>
<td>ENDP</td>
</tr>
<tr>
<td>1074</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1075</td>
<td></td>
<td>-----------------------------</td>
</tr>
<tr>
<td>1076</td>
<td></td>
<td>READ SECTOR BUFFER (AH = 0EH)</td>
</tr>
<tr>
<td>1077</td>
<td></td>
<td>-----------------------------</td>
</tr>
<tr>
<td>1078</td>
<td></td>
<td></td>
</tr>
<tr>
<td>04F9</td>
<td>1079</td>
<td>RD_BUFF PROC NEAR</td>
</tr>
<tr>
<td>04F9</td>
<td>1080</td>
<td>MOV CMD_BLOCK+0,RD_BUFF_CMD</td>
</tr>
<tr>
<td>04FE</td>
<td>1081</td>
<td>MOV CMD_BLOCK+4,1 ; ONLY ONE BLOCK</td>
</tr>
<tr>
<td>0503</td>
<td>1082</td>
<td>MOV AL,DMA_READ</td>
</tr>
<tr>
<td>0505</td>
<td>1083</td>
<td>JMP SHORT DMA_OPN</td>
</tr>
<tr>
<td>1084</td>
<td>RD_BUFF ENDP</td>
<td></td>
</tr>
<tr>
<td>1085</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1086</td>
<td></td>
<td>-----------------------------</td>
</tr>
<tr>
<td>1087</td>
<td></td>
<td>WRITE SECTOR BUFFER (AH = 0EH)</td>
</tr>
<tr>
<td>1088</td>
<td></td>
<td>-----------------------------</td>
</tr>
<tr>
<td>1089</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0507</td>
<td>1090</td>
<td>WR_BUFF PROC NEAR</td>
</tr>
<tr>
<td>0507</td>
<td>1091</td>
<td>MOV CMD_BLOCK+0,WR_BUFF_CMD</td>
</tr>
<tr>
<td>050C</td>
<td>1092</td>
<td>MOV CMD_BLOCK+4,1 ; ONLY ONE BLOCK</td>
</tr>
<tr>
<td>0511</td>
<td>1093</td>
<td>MOV AL,DMA_WRITE</td>
</tr>
<tr>
<td>0513</td>
<td>1094</td>
<td>JMP SHORT DMA_OPN</td>
</tr>
<tr>
<td>1095</td>
<td>WR_BUFF ENDP</td>
<td></td>
</tr>
<tr>
<td>1096</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1097</td>
<td></td>
<td>-----------------------------</td>
</tr>
<tr>
<td>1098</td>
<td></td>
<td>TEST DISK READY (AH = 010H)</td>
</tr>
<tr>
<td>1099</td>
<td></td>
<td>-----------------------------</td>
</tr>
<tr>
<td>1100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0515</td>
<td>1101</td>
<td>TST_DRV PROC NEAR</td>
</tr>
<tr>
<td>0515</td>
<td>1102</td>
<td>MOV CMD_BLOCK+0,TST_DRV_CMD</td>
</tr>
<tr>
<td>051A</td>
<td>1103</td>
<td>JMP SHORT NDMA_OPN</td>
</tr>
<tr>
<td>1104</td>
<td>TST_DRV ENDP</td>
<td></td>
</tr>
<tr>
<td>1105</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1106</td>
<td></td>
<td>-----------------------------</td>
</tr>
<tr>
<td>1107</td>
<td></td>
<td>RECALIBRATE (AH = 011H)</td>
</tr>
<tr>
<td>1108</td>
<td></td>
<td>-----------------------------</td>
</tr>
<tr>
<td>1109</td>
<td></td>
<td></td>
</tr>
<tr>
<td>051C</td>
<td>1110</td>
<td>HDISK_RECAL PROC NEAR</td>
</tr>
<tr>
<td>051C</td>
<td>1111</td>
<td>MOV CMD_BLOCK+0,HDISK_RECAL_CMD</td>
</tr>
<tr>
<td>0521</td>
<td>1112</td>
<td>JMP SHORT NDMA_OPN</td>
</tr>
<tr>
<td>1113</td>
<td>HDISK_RECAL ENDP</td>
<td></td>
</tr>
<tr>
<td>1114</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1115</td>
<td></td>
<td>-----------------------------</td>
</tr>
<tr>
<td>1116</td>
<td></td>
<td>CONTROLLER RAM DIAGNOSTICS (AH = 012H)</td>
</tr>
<tr>
<td>1117</td>
<td></td>
<td>-----------------------------</td>
</tr>
<tr>
<td>1118</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0523</td>
<td>1119</td>
<td>RAM_DIAG PROC NEAR</td>
</tr>
<tr>
<td>0523</td>
<td>1120</td>
<td>MOV CMD_BLOCK+0,RAM_DIAG_CMD</td>
</tr>
<tr>
<td>0528</td>
<td>1121</td>
<td>JMP SHORT NDMA_OPN</td>
</tr>
<tr>
<td>1122</td>
<td>RAM_DIAG ENDP</td>
<td></td>
</tr>
<tr>
<td>1123</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1124</td>
<td></td>
<td>-----------------------------</td>
</tr>
<tr>
<td>1125</td>
<td></td>
<td>DRIVE DIAGNOSTICS (AH = 013H)</td>
</tr>
<tr>
<td>1126</td>
<td></td>
<td>-----------------------------</td>
</tr>
<tr>
<td>1127</td>
<td></td>
<td></td>
</tr>
<tr>
<td>052A</td>
<td>1128</td>
<td>CHK_DRV PROC NEAR</td>
</tr>
<tr>
<td>052A</td>
<td>1129</td>
<td>MOV CMD_BLOCK+0,CHK_DRV_CMD</td>
</tr>
<tr>
<td>052F</td>
<td>1130</td>
<td>JMP SHORT NDMA_OPN</td>
</tr>
<tr>
<td>1131</td>
<td>CHK_DRV ENDP</td>
<td></td>
</tr>
<tr>
<td>1132</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1133</td>
<td></td>
<td>-----------------------------</td>
</tr>
<tr>
<td>1134</td>
<td></td>
<td>CONTROLLER INTERNAL DIAGNOSTICS (AH = 014H)</td>
</tr>
<tr>
<td>1135</td>
<td></td>
<td>-----------------------------</td>
</tr>
<tr>
<td>1136</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0531</td>
<td>1137</td>
<td>CNTRL_DIAG PROC NEAR</td>
</tr>
<tr>
<td>0531</td>
<td>1138</td>
<td>MOV CMD_BLOCK+0,CNTRL_DIAG_CMD</td>
</tr>
<tr>
<td>1139</td>
<td>CNTRL_DIAG ENDP</td>
<td></td>
</tr>
<tr>
<td>1140</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1141</td>
<td></td>
<td>-----------------------------</td>
</tr>
<tr>
<td>1142</td>
<td></td>
<td>SUPPORT ROUTINES</td>
</tr>
<tr>
<td>1143</td>
<td></td>
<td>-----------------------------</td>
</tr>
<tr>
<td>1144</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0536</td>
<td>1145</td>
<td>NDMA_OPN:</td>
</tr>
<tr>
<td>0536</td>
<td>1146</td>
<td>MOV AL,02H</td>
</tr>
<tr>
<td>053B</td>
<td>1147</td>
<td>CALL COMMAND ; ISSUE THE COMMAND</td>
</tr>
<tr>
<td>053C</td>
<td>1148</td>
<td>JC G11</td>
</tr>
<tr>
<td>053D</td>
<td>1149</td>
<td>JMP SHORT G3</td>
</tr>
</tbody>
</table>

A-98  Fixed Disk BIOS
053F 1150 G0: MOV DISK_STATUS, DMA_BOUN DARY
0544 C3 1152 RET
0545 1153 DNA_DM:
0545 E85701 1154 CALL DMA_SETUP ; SET UP FOR DMA OPERATION
0546 72F5 1155 JC GO
054A 8003 1156 MOV AL, 03H
054C E81300 1157 CALL COMMAND ; ISSUE THE COMMAND
054F 720D 1158 JC G11
0551 B003 1159 MOV AL, 03H
0553 E60A 1160 OUT DMA+10, AL ; INITIALIZE THE DISK CHANNEL
0555 1161 G3: OUT DMA+10, AL
0555 E421 1162 IN AL, 02H
0557 240F 1163 AND AL, 00FH
0559 E621 1164 OUT 02H, AL
055B E0A01 1165 CALL WAIT_INT
055E 1166 G11: CALL ERROR_CHK
0561 C3 1168 RET
1169
1170 i------------------------------------------------------------
1171 ; COMMAND
1172 ; THIS ROUTINE OUTPUTS THE COMMAND BLOCK
1173 ; INPUT
1174 ; AL = CONTROLLER DMA/INTERRUPT REGISTER MASK
1175 ;
1176 ;--------------------------------------------------------------
1177
0562 1178 COMMAND PROC NEAR
0562 BE4200 1179 MOV $I, OFFSET CMD_BLOCK
0565 E01B02 1180 CALL PORT_2
0568 EE 1181 OUT DX, AL ; CONTROLLER SELECT PULSE
0569 E01C02 1182 CALL PORT_3
056C EE 1183 OUT DX, AL
056D 28C9 1184 SUB CX, CX ; WAIT COUNT
056F E01C02 1185 CALL PORT_1
0572 1186 WAIT_BUSY:
0572 EC 1187 IN AL, DX ; GET STATUS
0573 240F 1188 AND AL, 00FH
0575 CB0C 1189 CMP AL, R1_BUSY OR R1_BUS OR R1_REQ
0577 7409 1190 JE C1
0579 28F7 1191 LOOP WAIT_BUSY
057B C606740080 1192 MOV DISK_STATUS, TIME_OUT
0580 F9 1193 STC
0581 C3 1194 RET ; ERROR RETURN
0582 1195 C1: CLD
0583 890600 1196 MOV CX, 6 ; BYTE COUNT
0586 1197 CH3: MOV AL, 00H
0586 E0E001 1198 CALL PORT_0
0589 AC 1199 LODSB ; GET THE NEXT COMMAND BYTE
058A EE 1200 OUT DX, AL ; OUT IT GOES
058B E2F9 1202 LOOP CH3 ; DO MORE
058D E0E001 1203 CALL PORT_1 ; STATUS
0590 EC 1204 IN AL, DX
0591 A601 1205 TEST AL, R1_REQ
0593 7406 1206 JE CH7
0595 C606740020 1208 MOV DISK_STATUS, BAD_CHAN
059A F9 1209 STC
059B 1210 CH7: RET
059B C3 1211
1212 COMMAND ENDP
1213
1214 i------------------------------------------------------------
1215 ; SENSE STATUS BYTES
1216 i
1217 i BYTE 0
1218 i BIT 7 ADDRESS VALID, WHEN SET
1219 i BIT 6 SPARE, SET TO ZERO
1220 i BITS 5-4 ERROR TYPE
1221 i BITS 3-0 ERROR CODE
1222 i
1223 i BYTE 1
1224 i BITS 7-6 ZERO
1225 i BITS 5 DRIVE (0-1)
1226 i BITS 4-0 HEAD NUMBER

Fixed Disk BIOS  A-99
A-100  Fixed Disk BIOS
0002  1304 TYPE3_LEN  EQU $-TYPE3_TABLE
0017  1308 TYPE3_TABLE  LABEL BYTE
0017  1309 DB  BAD_CNTL:BAD_CNTL:BAD_ECC
0035  1307 TYPE3_LEN  EQU  $-TYPE3_TABLE
0109  1309 ------ TYPE 0 ERROR
0614  1313 TYPE_O:  
0614  1322 MOV  BX,OFFSET TYPE2_TABLE
0629  1331 MOV  CX,AX
0632  1334 JMP  UNDEF_ERR_L
0635  1337 XLAT  CS:TYPE0_TABLE  TABLE LOOKUP
063B  1340 MOV  DISK_STATUS:AL  SET ERROR CODE
0640  1343  RET
066A  1353 TYPE_2:
066A  1358 MOV  BX,OFFSET TYPE2_TABLE
066D  1359 CMP  AL,TYPE2_LEN  ; CHECK IF ERROR IS DEFINED
0670  1360 JAE  UNDEF_ERR_L
0672  1361 XLAT  CS:TYPE2_TABLE  ; TABLE LOOKUP
0673  1362 MOV  DISK_STATUS:AL  ; SET ERROR CODE
0676  1363 RET
0679  1366 ------ TYPE 3 ERROR
067C  1370 ------ TYPE 4 ERROR
068D  1375 TYPE_3:
068D  1378 MOV  BX,OFFSET TYPE3_TABLE
0697  1382 MOV  BX,OFFSET TYPE3_TABLE
0698  1387 MOV  AL,CL
0699  1388  RET
069A  1389  PROC  HD_WAIT_REQ  NEAR

0688 2BC9 1361 SUB CX,CX
0690 EC 1363 L1: IN AL,DX
0691 A601 1365 TEST AL,R1_REQ
0693 7508 1366 JNZ L2
0695 E8H0 1367 LOOP L1
0697 50640080 1368 MOV DISK_STATUS,TIME_OUT
069C F9 1369 STC
069D 50 1370 L2: POP CX
069F 50 1371 POP CX
069F 50 1372 CALL PORT_1
069F 50 1373 HD_WAIT_REQ ENDP

069F 1404 IDHA_SETUP PROC NEAR
069F 50 1405 PUSH AX
069F A04600 1406 MOV AL,CMD_BLOCK+4
069F 1407 CMP AL,81H ; BLOCK COUNT OUT OF RANGE
069F 50 1409 POP AX
069F 7202 1410 JS J1
069F FA 1411 STC
069F C3 1412 RET
069F 50 1413 PUSH AX
069F 50 1414 PUSH AX
069F 50 1415 OUT DMA+12,AL ; SET THE FIRST/LAST F/F
069F 50 1416 PUSH AX
069F 50 1417 POP AX
069F 50 1418 OUT DMA+12,AL ; OUTPUT THE MODE BYTE
069F 50 1419 MOV AX,ES ; GET THE ES VALUE
069F 8104 1420 MOV CL,4 ; SHIFT COUNT
069F 50 1421 ROL AX,CL ; ROTATE LEFT
069F 50 1422 MOV CH,AL ; GET HIGHEST NYBBLE OF ES TO CH
069F 50 1423 AND AL,0FH ; ZERO THE LOW NYBBLE FROM SEGMENT
069F 50 1424 ADD AX,0H ; TEST FOR CARRY FROM ADDITION
069F 50 1425 INC AX
069F 50 1426 INC CH ; CARRY MEANS HIGH 4 BITS MUST BE INC
069F 50 1427 JS3: PUSH AX ; SAVE START ADDRESS
069F 50 1428 POP AX
069F 50 1429 OUT DMA+12,AL ; OUTPUT LOW ADDRESS
069F 50 142A MOV AH,0 ; MULTIPLE BY 512 BYTES PER SECTOR
069F 50 142B DEC AL ; AND DECREMENT VALUE BY ONE
069F 50 142C MOV AH,AL
069F 50 142D MOV AH,OFFH
069F 50 142E MOV AX,516D ; ONE BLOCK (512) PLUS 4 BYTES ECC
069F 50 142F MOV AL,CMD_BLOCK+4 ; RECOVER BLOCK COUNT
069F 50 1430 MOV AL,AL ; MULTIPLY BY 512 BYTES PER SECTOR
069F 50 1431 MOV AL,AL ; AND DECREMENT VALUE BY ONE
069F 50 1432 MOV AH,AL
069F 50 1433 MOV AH,OFFH
069F 50 1434 MOV AX,516D
069F 50 1435 MOV AX,516D
069F 50 1436 MOV AX,516D
069F 50 1437 MOV AX,516D
069F 50 1438 MOV AX,516D
069F 50 1439 MOV AX,516D
069F 50 143A MOV AX,516D
069F 50 143B MOV AX,516D
069F 50 143C MOV AX,516D
069F 50 143D MOV AX,516D
069F 50 143E MOV AX,516D
069F 50 143F MOV AX,516D
069F 50 1440 MOV AX,516D
069F 50 1441 MOV AX,516D
069F 50 1442 MOV AX,516D
069F 50 1443 MOV AX,516D
069F 50 1444 MOV AX,516D
069F 50 1445 MOV AX,516D
069F 50 1446 MOV AX,516D
069F 50 1447 MOV AX,516D
069F 50 1448 MOV AX,516D
069F 50 1449 MOV AX,516D
069F 50 144A MOV AX,516D
069F 50 144B MOV AX,516D
069F 50 144C MOV AX,516D
069F 50 144D MOV AX,516D
069F 50 144E MOV AX,516D
069F 50 144F MOV AX,516D
069F 50 1450 MOV AX,516D
069F 50 1451 MOV AX,516D
069F 50 1452 MOV AX,516D
069F 50 1453 MOV AX,516D
069F 50 1454 MOV AX,516D
069F 50 1455 MOV AX,516D
069F 50 1456 MOV AX,516D
069F 50 1457 MOV AX,516D

A-102 Fixed Disk BIOS
06EE 0AFF 1450 SUB BH,BH
06F0 0A1E4600 1459 MOV BL,CM_BLOCK+4
06F4 5E 1460 PUSH DX
06F5 FFE3 1461 MOV BX ; BLOCK COUNT TIMES S16
06F7 5A 1462 POP DX
06F8 5B 1463 POP BX
06F9 48 1464 DEC AX ; ADJUST
06FA 1465 JZ0:
06FA 50 1467 PUSH AX ; SAVE COUNT VALUE
06FB E607 1468 OUT DMA7,AL ; LOW BYTE OF COUNT
06FD 84C4 1469 MOV AL,AH
06FF E607 1470 OUT DMA7,AL ; HIGH BYTE OF COUNT
0701 FB 1471 STI ; INTERRUPTS BACK ON
0702 59 1472 POP CX ; RECOVER COUNT VALUE
0703 58 1473 POP AX ; RECOVER ADDRESS VALUE
0704 03C1 1474 ADD AX,CX ; ADD, TEST FOR 64K OVERFLOW
0706 59 1475 POP CX ; RECOVER REGISTER
0707 C3 1476 RET ; RETURN TO CALLER, CPL SET BY ABOVE IF ERROR

1477 DMA_SETUP
1478 ENDP
1479
1480 ;--------- WAIT_INT PROC NEAR
1481
1482 STI ; TURN ON INTERRUPTS
1483 PRESERVE REGISTERS
1484 PUSH BX
1485 PUSH CX
1486 PUSH SI
1487 PUSH DS
1488 ASSUME DS:DUMMY
1489 SUB AX,AX ; NO DUMMY
1490 28C0 1493 SUB AX,AX
1491 06D0 1494 MOV DS,AX ; ESTABLISH SEGMENT
1492 1C360401 1495 LES SI,HI_TBL_VEC
1496 ASSUME DS:DATA
1497 0716 1F 1498 POP DS
1499 0717 E84400 1499 CALL PORT_1
1500
1501 ; SET TIMEOUT VALUES
1502 0718 5A254200 1501 SUB BH,BH
1503 0719 2665C809 1502 MOV BL,BYTE PTR ES:[SI][9] ; STANDARD TIME OUT
1504 071A 08FCC4 1503 MOV AH,FMTDRV_CMD
1505 071B 7506 1504 JMP SHORT H4
1506 071C 2665C80A 1505 MOV BL,BYTE PTR ES:[SI][0AH] ; FORMAT DRIVE
1507 071D 7506 1506 JMP SHORT H4
1508 071E 80FC33 1507 MOV AH,CHK_DRV_CMD
1509 071F 7504 1508 JS H4
1510 0720 2665C80B 1509 MOV BL,BYTE PTR ES:[SI][0BH] ; CHECK DRIVE
1511 0721 750D 1510 JMP SHORT H4
1512 0722 28C9 1511 MOV AX,AL ; ERROR BIT
1513 0723 1512 SUB CX,CX
1514 ;------ WAIT FOR INTERRUPT
1515 0724 1513
1516 0725 0B4400 1514 W1:
1517 0726 EC 1515 CALL PORT_1
1518 0727 0A254200 1516 IN AL,DX
1519 0728 2420 1517 AND AL,020H
1520 0729 3C20 1518 JMP SHORT H4
1521 072A 740A 1519 MOV AL,020H ; DID INTERRUPT OCCUR
1522 072B 5E24 1520 CJNE W1 ; INNER LOOP
1523 072C 5B23 1521 LOOP W1
1524 072D 75F1 1522 MOV DX,W1 ; OUTER LOOP
1525 072E 560740080 1523 MOV DISK_STATUS.TIME_OUT
1526 072F 1524 OUT DX,AL
1527 0730 0B4300 1525 W2:
1528 0731 EC 1526 CALL PORT_0
1529 0732 0A254200 1527 IN AL,DX
1530 0733 2420 1528 AND AL,020H
1531 0734 8567400 1529 CALL PORT_3 ; INTERRUPT MASK REGISTER
1532 0735 3C20 1530 MOV AL,020H ; ZERO
1533 0736 5E23 1531 LOOP W1
1534 0737 0B5B 1532 OUT DX,AL ; RESTORE REGISTERS

Appendix A

Fixed Disk BIOS A-103
LOC OBJ   LINE   SOURCE
075C 07    1535   POP  ES
075D 59    1536   POP  CX
075E 5B    1537   POP  BX
075F C3    1538   RET
0759 89    1539   WAIT_INT ENDP
               1540
0760 153A   HD_INT PROC NEAR
0761 56    153B   PUSH AX
0762 B200    153C   MOV AX, [AL+E0H]
0763 65    153D   OUT INT_CTL_PORT, AL
0764 0759    153E   MOV AL, 02H
0765 E60A    153F   OUT DMA+10H, AL
0766 E621    1540   IN AL, 021H
0767 8C20    1541   OR AL, 020H
0768 E621    1542   OUT 021H, AL
0769 56    1543   POP  AX
0770 C5    1544   MOV AX, 0
0771 89    1545   OUT AX, PORT
               1546
0771 U21I    1547   MOV AL, 0
0772 50    1548   ADD AX, OFFSET PORT
               1549
0773 7506    1550   JNZ SW2.HIGH
0774 80E4300 1551   MOV AX, AH
0775 FA26     1552   CALL PORT 2
0776 2403    1553   AND AL, 011B
0777 B104    1554   MOV CL, 4
               1555
077E UA2003   1556   CALL PORT 0
077F 50    1557   PUSH AX
0780 2A46    1558   MOV AX, [AL+AH]
0781 A07700   1559   MOV AL, PORT_OFF
               1560
0782 50    1561   ADD AX, OFFSET PORT
               1562
0783 8A26     1563   CALL PORT 1
0784 2402    1564   AND AL, 011B
0785 7516    1565   JNZ SW2.LOW
0786 BAE630B 1566   MOV AX, AH
0787 FA26     1567   CALL PORT 2
0788 2403    1568   AND AL, 011B
0789 B104    1569   MOV CL, 4
               1570
078A UA2003   1571   CALL PORT 0
078B 50    1572   PUSH AX
078C 2A46    1573   MOV AX, [AL+AH]
078D A07700   1574   MOV AL, PORT_OFF
               1575
078E 50    1576   ADD AX, OFFSET PORT
               1577
078F 8A26     1578   CALL PORT 1
0790 2402    1579   AND AL, 011B
0791 7516    1580   JNZ SW2.LOW
0792 BAE630B 1581   MOV AX, AH
0793 FA26     1582   CALL PORT 2
0794 2403    1583   AND AL, 011B
0795 B104    1584   MOV CL, 4
               1585
0796 U21I    1586   MOV AL, 0
0797 50    1587   ADD AX, OFFSET PORT
               1588
0798 8A2630B 1589   MOV AX, AH
0799 FA26     1590   CALL PORT 2
079A 2403    1591   AND AL, 011B
079B B104    1592   MOV CL, 4
               1593
079C EA2003   1594   CALL PORT 0
079D 50    1595   PUSH AX
079E 2A46    1596   MOV AX, [AL+AH]
079F A07700   1597   MOV AL, PORT_OFF
               1598
07A0 50    1599   ADD AX, OFFSET PORT
               159A
07A1 8A26     159B   CALL PORT 1
07A2 2402    159C   AND AL, 011B
07A3 B104    159D   MOV CL, 4
               159E
07A4 EA2003   159F   CALL PORT 0
07A5 50    1600   PUSH AX
07A6 2A46    1501   MOV AX, [AL+AH]
07A7 A07700   1502   MOV AL, PORT_OFF
               1503
07A8 50    1604   ADD AX, OFFSET PORT
               1505
07A9 8A26     1606   CALL PORT 1
07AA 2402    1607   AND AL, 011B
07AB B104    1608   MOV CL, 4
               1609
07AC EA2003   160A   CALL PORT 0
07AD 50    160B   PUSH AX
07AE 2A46    160C   MOV AX, [AL+AH]
07AF A07700   160D   MOV AL, PORT_OFF
               160E
07B0 50    160F   ADD AX, OFFSET PORT
               1610
07B1 8A26     1611   CALL PORT 1
07B2 2402    1612   AND AL, 011B
07B3 B104    1613   MOV CL, 4
               1614
07B4 1505   1615   ...
LOC OBJ | LINE | SOURCE
--- | --- | ---
07AC 02E0 | 1612 | SHL AL, CL
07AE 2AE4 | 1613 | SUB AH, AH
07B0 C3 | 1614 | RET
07B1 | 1615 | SWZ_OFFS_ERR:
07B1 F9 | 1616 | STC
07B2 C3 | 1617 | RET
07B3 3038F31362F30 | 1618 | SWZ_OFFS ENDP
07B3 | 1619 | RET
07B5 08/16/82 | 1620 | DB '08/16/82' RELEASE MARKER
07B8 | 1621 | END_ADDRESS
--- | 1622 | LABEL BYTE
07BD | 1623 | CODE ENDS
1624 | END

Appendix A

Fixed Disk BIOS   A-105
Notes:
### 8088 Register Model

<table>
<thead>
<tr>
<th>AX:</th>
<th>AH</th>
<th>AL</th>
<th>Accumulator</th>
<th>General Register File</th>
</tr>
</thead>
<tbody>
<tr>
<td>BX:</td>
<td>BH</td>
<td>BL</td>
<td>Base</td>
<td></td>
</tr>
<tr>
<td>CX:</td>
<td>CH</td>
<td>CL</td>
<td>Count</td>
<td></td>
</tr>
<tr>
<td>DX:</td>
<td>DH</td>
<td>DL</td>
<td>Data</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SP</td>
<td></td>
<td>Stack Pointer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BP</td>
<td></td>
<td>Base Pointer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SI</td>
<td></td>
<td>Source Index</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DI</td>
<td></td>
<td>Destination Index</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IP</td>
<td></td>
<td>Instruction Pointer</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>FLAGSH</th>
<th>FLAGSL</th>
<th>Status Flags</th>
<th>Segment Register File</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CS</td>
<td></td>
<td>Code Segment</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DS</td>
<td></td>
<td>Data Segment</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SS</td>
<td></td>
<td>Stack Segment</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ES</td>
<td></td>
<td>Extra Segment</td>
<td></td>
</tr>
</tbody>
</table>

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

```
15 7 0
  X X X X OF DF IF TF SF ZF X AF X PF X CF
```

X = Don’t Care

- **AF**: Auxiliary Carry - BCD
- **CF**: Carry Flag
- **PF**: Parity Flag
- **SF**: Sign Flag
- **ZF**: Zero Flag

### Flag Register

- **DF**: Direction Flag (Strings)
- **IF**: Interrupt Enable Flag
- **OF**: Overflow Flag (CF ⊕ SF)
- **TF**: Trap - Single Step Flag

---

**B-2 8088 Instruction Reference**
Operand Summary

"reg" field Bit Assignments:

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

Second Instruction Byte Summary

<table>
<thead>
<tr>
<th>mod</th>
<th>Displacement</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>DISP = 0*, disp-low and disp-high are absent</td>
</tr>
<tr>
<td>01</td>
<td>DISP = disp-low sign-extended to 16-bits, disp-high is absent</td>
</tr>
<tr>
<td>10</td>
<td>DISP = disp-high: disp-low</td>
</tr>
<tr>
<td>11</td>
<td>r/m is treated as a &quot;reg&quot; field</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>r/m</th>
<th>Operand Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>(BX) + (SI) + DISP</td>
</tr>
<tr>
<td>001</td>
<td>(BX) + (DI) + DISP</td>
</tr>
<tr>
<td>010</td>
<td>(BP) + (SI) + DISP</td>
</tr>
<tr>
<td>011</td>
<td>(BP) + (DI) + DISP</td>
</tr>
<tr>
<td>100</td>
<td>(SI) + DISP</td>
</tr>
<tr>
<td>101</td>
<td>(DI) + DISP</td>
</tr>
<tr>
<td>110</td>
<td>(BP) + DISP*</td>
</tr>
<tr>
<td>111</td>
<td>(BX) + DISP</td>
</tr>
</tbody>
</table>

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

Logical Memory Space

Offset Address

Selected Segment Register

Displacement

Adder

Physical Address Latch

Displacement

Word

MSB

LSB

BYTE

Extra Data Segment

Data Segment

Stack Segment

Code Segment

FFFFFH

64KB

Segment Override Prefix

0 0 1 reg 1 1 0

Use of Segment Override

<table>
<thead>
<tr>
<th>Operand Register</th>
<th>Default</th>
<th>With Override Prefix</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP (Code Address)</td>
<td>CS</td>
<td>Never</td>
</tr>
<tr>
<td>SP (Stack Address)</td>
<td>SS</td>
<td>Never</td>
</tr>
<tr>
<td>BP (Stack Address or Stack Marker)</td>
<td>SS</td>
<td>BP + DS or ES, or CS</td>
</tr>
<tr>
<td>SI or DI (not including strings)</td>
<td>DS</td>
<td>ES, SS, or CS</td>
</tr>
<tr>
<td>SI (Implicit Source Address for Strings)</td>
<td>DS</td>
<td>ES, SS, or CS</td>
</tr>
<tr>
<td>DI (Implicit Destination Address for Strings)</td>
<td>ES</td>
<td>Never</td>
</tr>
</tbody>
</table>
### Data Transfer

**MOV** = Move
Register/memory to/from register

<table>
<thead>
<tr>
<th>1 0 0 0 1 0 d w</th>
<th>mod</th>
<th>reg</th>
<th>r/m</th>
</tr>
</thead>
</table>

Immediate to register/memory

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

Immediate to register

| 1 0 1 1 w | reg | data | data if w=1 |

Memory to accumulator

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

Accumulator to memory

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

Register/memory to segment register

| 1 0 0 0 1 1 1 0 | mod | 0 | reg | r/m |

Segment register to register/memory

| 1 0 0 0 1 1 0 0 | mod | 0 | reg | r/m |

**PUSH** = Push
Register/memory

| 1 1 1 1 1 1 1 1 | mod | 1 1 0 | r/m |

Register

| 0 1 0 1 0 reg |

Segment register

| 0 0 0 | reg | 1 1 0 |

**POP** = Pop
Register/memory

| 1 0 0 0 1 1 1 | mod | 0 0 0 | r/m |

Register

| 0 1 0 1 1 reg |

Segment register

| 0 0 0 | reg | 1 1 1 |
**XCHG = Exchange**
Register/memory with register

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

Register with accumulator

| 1 0 0 1 0 reg |

**IN = Input to AL/AX from**
Fixed port

| 1 1 1 0 1 0 w | port |

Variable port (DX)

| 1 1 1 0 1 1 0 w |

**OUT = Output from AL/AX to**
Fixed port

| 1 1 1 0 1 0 1 1 1 0 w | port |

**Variable port (DX)**

| 1 1 1 0 0 1 1 0 w |

**XLAT = Translate byte to AL**

| 1 1 0 1 0 1 1 1 1 |

**LEA = Load EA to register**

| 1 0 0 0 1 1 0 1 | mod reg r/m |

**LDS = Load pointer to DS**

| 1 1 0 0 0 1 0 1 | mod reg r/m |

**LES = Load pointer to ES**

| 1 1 0 0 0 1 0 0 | mod reg r/m |

**LAHF = Load AH with flags**

| 1 0 0 1 1 1 1 1 |

**SAHF = Store AH into flags**

| 1 0 0 1 1 1 1 0 |

**PUSHF = Push flags**

| 1 0 0 1 1 1 0 0 |

**POPF = Pop flags**

| 1 0 0 1 1 1 0 1 |
### Arithmetic

**ADD** = Add
Register/memory with register to either

<table>
<thead>
<tr>
<th>D W</th>
<th>mod reg r/m</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0 0 0 0</td>
<td></td>
</tr>
</tbody>
</table>

Immediate to register/memory

<table>
<thead>
<tr>
<th>S W</th>
<th>mod 0 0 0 r/m</th>
<th>data</th>
<th>data if s:w=01</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Immediate to accumulator

<table>
<thead>
<tr>
<th>D W</th>
<th>data</th>
<th>data if w=1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0 1 0 0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**ADC** = Add with carry
Register/memory and register to either

<table>
<thead>
<tr>
<th>D W</th>
<th>mod reg r/m</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 1 0 0 0</td>
<td></td>
</tr>
</tbody>
</table>

Immediate to register/memory

<table>
<thead>
<tr>
<th>S W</th>
<th>mod 0 1 0 r/m</th>
<th>data</th>
<th>data if s:w=01</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Immediate to accumulator

<table>
<thead>
<tr>
<th>D W</th>
<th>data</th>
<th>data if w=1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 1 0 1 0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**INC** = Increment
Register/memory

<table>
<thead>
<tr>
<th>D W</th>
<th>mod 0 0 0 r/m</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 1 1 1 1 1</td>
<td></td>
</tr>
</tbody>
</table>

Register

<table>
<thead>
<tr>
<th>Reg</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 0 0 0</td>
</tr>
</tbody>
</table>

**AAA** = ASCII adjust for add

<table>
<thead>
<tr>
<th>D W</th>
<th>mod reg r/m</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 1 1 1 1 1</td>
<td></td>
</tr>
</tbody>
</table>

**DAA** = Decimal adjust for add

<table>
<thead>
<tr>
<th>D W</th>
<th>mod reg r/m</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 1 0 0 1 1</td>
<td></td>
</tr>
</tbody>
</table>

**SUB** = Subtract
Register/memory and register to either

<table>
<thead>
<tr>
<th>D W</th>
<th>mod reg r/m</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 1 0 1 0 0</td>
<td></td>
</tr>
</tbody>
</table>

Immediate from register/memory

<table>
<thead>
<tr>
<th>S W</th>
<th>mod 1 0 1 r/m</th>
<th>data</th>
<th>data if s:w=01</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Immediate from accumulator

<table>
<thead>
<tr>
<th>D W</th>
<th>data</th>
<th>data if w=1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 1 0 1 1 0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SBB = Subtract with borrow
Register/memory and register to either

\[
\begin{array}{c|c|c|c|c|c|c}
\text{mod} & \text{reg} & \text{r/m} \\
\hline
0 & 0 & 1 & 1 & 0 & d & w \\
\end{array}
\]

Immediate from register/memory

\[
\begin{array}{c|c|c|c|c|c|c}
\text{mod} & \text{r/m} & \text{data} \\
\hline
0 & 1 & 1 & 1 & 0 & w \\
\end{array}
\]

Immediate from accumulator

\[
\begin{array}{c|c|c|c|c|c|c}
\text{data} & \text{data if w=1} \\
\hline
0 & 0 & 1 & 1 & 0 & w \\
\end{array}
\]

DEC = Decrement
Register/memory

\[
\begin{array}{c|c|c|c|c|c|c}
\text{mod} & \text{r/m} \\
\hline
0 & 0 & 1 & 1 & d & w \\
\end{array}
\]

Register

\[
\begin{array}{c|c}
0 & 1 & 0 & 0 & 1 & \text{reg} \\
\end{array}
\]

NEG = Change sign

\[
\begin{array}{c|c|c|c|c|c|c}
\text{mod} & \text{r/m} \\
\hline
0 & 1 & 1 & 1 & 1 & 0 & w \\
\end{array}
\]

CMP = Compare
Register/memory and register

\[
\begin{array}{c|c|c|c|c|c|c}
\text{mod} & \text{r/m} \\
\hline
0 & 0 & 1 & 1 & 0 & d & w \\
\end{array}
\]

Immediate with register/memory

\[
\begin{array}{c|c|c|c|c|c|c}
\text{mod} & \text{r/m} & \text{data} & \text{data if s:w=01} \\
\hline
0 & 1 & 1 & 1 & 1 & 0 & w \\
\end{array}
\]

Immediate with accumulator

\[
\begin{array}{c|c|c|c|c|c|c}
\text{data} & \text{data if w=1} \\
\hline
0 & 0 & 1 & 1 & 1 & 0 & w \\
\end{array}
\]

AAS = ASCII adjust for subtract

\[
0 & 0 & 1 & 1 & 1 & 1 & 1
\]

DAS = Decimal adjust for subtract

\[
0 & 0 & 1 & 0 & 1 & 1 & 1
\]

MUL = Multiply (unsigned)

\[
\begin{array}{c|c|c|c|c|c|c}
\text{mod} & \text{r/m} \\
\hline
1 & 1 & 1 & 1 & 0 & 1 & 1 & w \\
\end{array}
\]

IMUL = Integer multiply (signed)

\[
\begin{array}{c|c|c|c|c|c|c}
\text{mod} & \text{r/m} \\
\hline
1 & 1 & 1 & 1 & 0 & 1 & 1 & w \\
\end{array}
\]

AAM = ASCII adjust for multiply

\[
\begin{array}{c|c|c|c|c|c|c}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
\end{array}
\]

DIV = Divide (unsigned)

\[
\begin{array}{c|c|c|c|c|c|c}
\text{mod} & \text{r/m} \\
\hline
1 & 1 & 1 & 1 & 0 & 1 & 1 & w \\
\end{array}
\]
**IDIV** = Integer divide (signed)  
1 1 1 1 0 1 1 w \mod 1 1 1 r/m

**AAD** = ASCII adjust for divide  
1 1 0 1 0 1 0 1 0 0 0 1 0 1 0

**CBW** = Convert byte to word  
1 0 0 1 1 0 0 0

**CWD** = Convert word to double word  
1 0 0 1 1 0 0 1

Grammar

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
<th>Format</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IDIV</strong></td>
<td>Integer divide (signed)</td>
<td>1 1 1 1 0 1 1 w \mod 1 1 1 r/m</td>
<td></td>
</tr>
<tr>
<td><strong>AAD</strong></td>
<td>ASCII adjust for divide</td>
<td>1 1 0 1 0 1 0 1 0 0 0 1 0 1 0</td>
<td></td>
</tr>
<tr>
<td><strong>CBW</strong></td>
<td>Convert byte to word</td>
<td>1 0 0 1 1 0 0 0</td>
<td></td>
</tr>
<tr>
<td><strong>CWD</strong></td>
<td>Convert word to double word</td>
<td>1 0 0 1 1 0 0 1</td>
<td></td>
</tr>
</tbody>
</table>

**Logic**

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
<th>Format</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NOT</strong></td>
<td>Invert</td>
<td>1 1 1 1 0 1 1 w \mod 0 1 0 r/m</td>
<td></td>
</tr>
<tr>
<td><strong>SHL/SAL</strong></td>
<td>Shift logical/arithmetic left</td>
<td>1 1 0 1 0 0 v w \mod 1 0 0 r/m</td>
<td></td>
</tr>
<tr>
<td><strong>SHR</strong></td>
<td>Shift logical right</td>
<td>1 1 0 1 0 0 v w \mod 1 0 1 r/m</td>
<td></td>
</tr>
<tr>
<td><strong>SAR</strong></td>
<td>Shift arithmetic right</td>
<td>1 1 0 1 0 0 v w \mod 1 1 1 r/m</td>
<td></td>
</tr>
<tr>
<td><strong>ROL</strong></td>
<td>Rotate left</td>
<td>1 1 0 1 0 0 v w \mod 0 0 0 r/m</td>
<td></td>
</tr>
<tr>
<td><strong>ROR</strong></td>
<td>Rotate right</td>
<td>1 1 0 1 0 0 v w \mod 0 0 1 r/m</td>
<td></td>
</tr>
<tr>
<td><strong>RCL</strong></td>
<td>Rotate through carry left</td>
<td>1 1 0 1 0 0 v w \mod 0 1 0 r/m</td>
<td></td>
</tr>
<tr>
<td><strong>RCR</strong></td>
<td>Rotate through carry right</td>
<td>1 1 0 1 0 0 v w \mod 0 1 1 r/m</td>
<td></td>
</tr>
<tr>
<td><strong>AND</strong></td>
<td>And</td>
<td>0 0 1 0 0 0 d w \mod reg r/m</td>
<td></td>
</tr>
</tbody>
</table>

Immediate to register/memory

<table>
<thead>
<tr>
<th>Format</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0 0 0 0 0 0 0 w \mod 1 0 0 r/m</td>
<td>data data if w=1</td>
</tr>
</tbody>
</table>

Immediate to accumulator

<table>
<thead>
<tr>
<th>Format</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 1 0 0 1 0 w</td>
<td>data data if w=1</td>
</tr>
</tbody>
</table>
### TEST

And function to flags, no result

<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0 0 0 0 1 0 w</td>
<td>mod reg r/m</td>
</tr>
</tbody>
</table>

### Immediate data and register/memory

<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 1 1 0 1 1 w</td>
<td>mod 0 0 0 r/m</td>
</tr>
</tbody>
</table>

### Immediate data and accumulator

<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0 1 0 1 0 0 w</td>
<td>data</td>
</tr>
</tbody>
</table>

### OR

OR = OR
Register/memory and register to either

<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0 1 0 d w</td>
<td>mod reg r/m</td>
</tr>
</tbody>
</table>

### Immediate to register/memory

<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0 0 0 0 0 0 0 w</td>
<td>mod 0 0 1 r/m</td>
</tr>
</tbody>
</table>

### Immediate to accumulator

<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0 1 1 0 w</td>
<td>data</td>
</tr>
</tbody>
</table>

### XOR

XOR = Exclusive or
Register/memory and register to either

<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 1 1 0 0 d w</td>
<td>mod reg r/m</td>
</tr>
</tbody>
</table>

### Immediate to register/memory

<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0 0 0 0 0 0 0 w</td>
<td>mod 1 1 0 r/m</td>
</tr>
</tbody>
</table>

### Immediate to accumulator

<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 1 1 0 1 0 w</td>
<td>data</td>
</tr>
</tbody>
</table>

### String Manipulation

### REP

REP = Repeat

<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 1 1 0 0 1 z</td>
<td></td>
</tr>
</tbody>
</table>

### MOVS

MOVS = Move String

<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0 1 0 0 1 0 w</td>
<td></td>
</tr>
</tbody>
</table>

### CMPS

CMPS = Compare String

<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0 1 0 0 1 1 w</td>
<td></td>
</tr>
</tbody>
</table>

### SCAS

SCAS = Scan String

<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0 1 0 1 1 1 w</td>
<td></td>
</tr>
</tbody>
</table>
**LODS** = Load String

\[ 1010110 \]

**STOS** = Store String

\[ 1010101 \]

---

**Control Transfer**

**CALL** = Call

Direct within segment

\[ 11101000 \] \[ \text{disp-low} \] \[ \text{disp-high} \]

Indirect within segment

\[ 11111111 \] \[ \text{mod 010 r/m} \]

Direct intersegment

\[ 1001101 \] \[ \text{offset-low} \] \[ \text{offset-high} \]

\[ \text{seg-low} \] \[ \text{seg-high} \]

Indirect intersegment

\[ 11111111 \] \[ \text{mod 011 r/m} \]

**JMP** = Unconditional Jump

Direct within segment

\[ 11101001 \] \[ \text{disp-low} \] \[ \text{disp-high} \]

Direct within segment-short

\[ 11101011 \] \[ \text{disp} \]

Indirect within segment

\[ 1111111111 \] \[ \text{mod 100 r/m} \]

Direct intersegment

\[ 11101010 \] \[ \text{offset-low} \] \[ \text{offset-high} \]

\[ \text{seg-low} \] \[ \text{seg-high} \]

Indirect intersegment

\[ 11111111 \] \[ \text{mod 101 r/m} \]
<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
<th>Machine Code</th>
<th>Disp.</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RET</strong></td>
<td>Return from CALL</td>
<td>11000011</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>JNZ</strong></td>
<td>Jump on not equal/not zero</td>
<td>0111100000</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>JAE</strong></td>
<td>Jump on above or equal/above or Below/parity even</td>
<td>0111101000</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>JNB</strong></td>
<td>Jump on Below/not above or equal</td>
<td>0111101100</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>JE</strong></td>
<td>Jump on equal/zero</td>
<td>0111100000</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>JBE</strong></td>
<td>Jump on below or equal/not above</td>
<td>0111101100</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>JLE</strong></td>
<td>Jump on less or equal/not greater</td>
<td>0111110000</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>JL</strong></td>
<td>Jump on less/not greater or equal</td>
<td>0111110000</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>JNG</strong></td>
<td>Jump on not greater/not less or equal</td>
<td>0111101111</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>JNGE</strong></td>
<td>Jump on not less or equal/less or equal</td>
<td>0111110111</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>JNP</strong></td>
<td>Jump on parity/parity even</td>
<td>0111101101</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>JAE</strong></td>
<td>Jump on above or equal/above or Below/parity even</td>
<td>0111101000</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>JNB</strong></td>
<td>Jump on Below/not above or equal</td>
<td>0111101100</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>JE</strong></td>
<td>Jump on equal/zero</td>
<td>0111100000</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>JBE</strong></td>
<td>Jump on below or equal/not above</td>
<td>0111101100</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>JLE</strong></td>
<td>Jump on less or equal/not greater</td>
<td>0111110000</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>JL</strong></td>
<td>Jump on less/not greater or equal</td>
<td>0111110000</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>JNG</strong></td>
<td>Jump on not greater/not less or equal</td>
<td>0111101111</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>JNGE</strong></td>
<td>Jump on not less or equal/less or equal</td>
<td>0111110111</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>JNP</strong></td>
<td>Jump on parity/parity even</td>
<td>0111101101</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>JAE</strong></td>
<td>Jump on above or equal/above or Below/parity even</td>
<td>0111101000</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>JNB</strong></td>
<td>Jump on Below/not above or equal</td>
<td>0111101100</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>JE</strong></td>
<td>Jump on equal/zero</td>
<td>0111100000</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>JBE</strong></td>
<td>Jump on below or equal/not above</td>
<td>0111101100</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>JLE</strong></td>
<td>Jump on less or equal/not greater</td>
<td>0111110000</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>JL</strong></td>
<td>Jump on less/not greater or equal</td>
<td>0111110000</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>JNG</strong></td>
<td>Jump on not greater/not less or equal</td>
<td>0111101111</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>JNGE</strong></td>
<td>Jump on not less or equal/less or equal</td>
<td>0111110111</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>JNP</strong></td>
<td>Jump on parity/parity even</td>
<td>0111101101</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>JAE</strong></td>
<td>Jump on above or equal/above or Below/parity even</td>
<td>0111101000</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>JNB</strong></td>
<td>Jump on Below/not above or equal</td>
<td>0111101100</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>JE</strong></td>
<td>Jump on equal/zero</td>
<td>0111100000</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>JBE</strong></td>
<td>Jump on below or equal/not above</td>
<td>0111101100</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>JLE</strong></td>
<td>Jump on less or equal/not greater</td>
<td>0111110000</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>JL</strong></td>
<td>Jump on less/not greater or equal</td>
<td>0111110000</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>JNG</strong></td>
<td>Jump on not greater/not less or equal</td>
<td>0111101111</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>JNGE</strong></td>
<td>Jump on not less or equal/less or equal</td>
<td>0111110111</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>JNP</strong></td>
<td>Jump on parity/parity even</td>
<td>0111101101</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**8088 Instruction Reference**
| Instruction | Description | Opcode | Disp | |---|---|---|---|
| JNLE/JG | Jump on not less or equal/greater | 0 1 1 1 1 1 1 1 | disp |
| JNB/JAE | Jump on not below/above or equal | 0 1 1 1 0 0 1 1 | disp |
| JNBE/JA | Jump on not below or equal/above | 0 1 1 1 0 1 1 1 | disp |
| JNP/JPO | Jump on not parity/parity odd | 0 1 1 1 1 0 1 1 | disp |
| JNO | Jump on not overflow | 0 1 1 1 1 0 0 1 | disp |
| JNS | Jump on not sign | 0 1 1 1 1 0 0 1 | disp |
| LOOP | Loop CX times | 1 1 1 0 0 0 1 0 | disp |
| LOOPZ/LOOPE | Loop while zero/equal | 1 1 1 0 0 0 0 1 | disp |
| LOOPNZ/LOOPNE | Loop while not zero/not equal | 1 1 1 0 0 0 0 0 | disp |
| JCXZ | Jump on CX zero | 1 1 1 0 0 0 1 1 | disp |
## 8088 Conditional Transfer Operations

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

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

### INT = Interrupt
Type specified

```
1 1 0 0 1 1 0 1 type
```

**Type 3**

```
1 1 0 0 1 1 0 0
```

### INTO = Interrupt on overflow

```
1 1 0 0 1 1 1 0
```

### IRET = Interrupt return

```
1 1 0 0 1 1 1 1
```

---

**B-14 8088 Instruction Reference**
### Processor Control

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLC</td>
<td>Clear carry</td>
<td><code>11111000</code></td>
</tr>
<tr>
<td>CMC</td>
<td>Complement carry</td>
<td><code>11110101</code></td>
</tr>
<tr>
<td>CLD</td>
<td>Clear direction</td>
<td><code>11111100</code></td>
</tr>
<tr>
<td>CLI</td>
<td>Clear interrupt</td>
<td><code>11111010</code></td>
</tr>
<tr>
<td>HLT</td>
<td>Halt</td>
<td><code>11110100</code></td>
</tr>
<tr>
<td>LOCK</td>
<td>Bus lock prefix</td>
<td><code>11111000</code></td>
</tr>
<tr>
<td>STC</td>
<td>Set carry</td>
<td><code>11111001</code></td>
</tr>
<tr>
<td>NOP</td>
<td>No operation</td>
<td><code>10010000</code></td>
</tr>
<tr>
<td>STD</td>
<td>Set direction</td>
<td><code>11111101</code></td>
</tr>
<tr>
<td>STI</td>
<td>Set interrupt</td>
<td><code>11111101</code></td>
</tr>
<tr>
<td>ESC</td>
<td>Escape (to external device)</td>
<td><code>11110011</code></td>
</tr>
</tbody>
</table>

### Footnotes:
- If \( d = 1 \) then "to"; if \( d = 0 \) then "from"
- If \( w = 1 \) then word instruction; if \( w = 0 \) then byte instruction
- If \( s:w = 01 \) then 16 bits of immediate data from the operand
- If \( s:w = 11 \) then an immediate data byte is sign extended to form the 16-bit operand
- If \( v = 0 \) then "count" = 1; if \( v = 1 \) then "count" in (CL)
- \( x \) = don't care
- \( z \) is used for some string primitives to compare with \( ZF \) FLAG
- AL = 8-bit accumulator
- AX = 16-bit accumulator
- CX = Count register
- DS = Data segment
- DX = Variable port register
- ES = Extra segment
- Above/below refers to unsigned value
- Greater = more positive;
- Less = less positive (more negative) signed values
<table>
<thead>
<tr>
<th>HI</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>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ADD b,f,r/m</td>
<td>ADD w,f,r/m</td>
<td>ADD b,t,r/m</td>
<td>ADD w,t,r/m</td>
<td>ADD b,ia w,ia</td>
<td>ADD w,ia</td>
<td>PUSH w,t,r/m</td>
<td>POP ES</td>
</tr>
<tr>
<td>1</td>
<td>ADC b,f,r/m</td>
<td>ADC w,f,r/m</td>
<td>ADC b,t,r/m</td>
<td>ADC w,t,r/m</td>
<td>ADC b,i w,i</td>
<td>ADC w,i</td>
<td>PUSH SS</td>
<td>POP</td>
</tr>
<tr>
<td>2</td>
<td>AND b,f,r/m</td>
<td>AND w,f,r/m</td>
<td>AND b,t,r/m</td>
<td>AND w,t,r/m</td>
<td>AND b,i w,i</td>
<td>SEG =ES</td>
<td>SEG =SS</td>
<td>DAA</td>
</tr>
<tr>
<td>3</td>
<td>XOR b,f,r/m</td>
<td>XOR w,f,r/m</td>
<td>XOR b,t,r/m</td>
<td>XOR w,t,r/m</td>
<td>XOR b,i w,i</td>
<td>SEG</td>
<td>SEG</td>
<td>AAA</td>
</tr>
<tr>
<td>4</td>
<td>INC AX</td>
<td>INC CX</td>
<td>INC DX</td>
<td>INC BX</td>
<td>INC SP</td>
<td>INC BP</td>
<td>INC SI</td>
<td>INC</td>
</tr>
<tr>
<td>5</td>
<td>PUSH AX</td>
<td>PUSH CX</td>
<td>PUSH DX</td>
<td>PUSH BX</td>
<td>PUSH SP</td>
<td>PUSH BP</td>
<td>PUSH SI</td>
<td>PUSH</td>
</tr>
<tr>
<td>6</td>
<td>JO JNO</td>
<td>JB/ JNAE</td>
<td>JNB/ JAE</td>
<td>JE/ JZ</td>
<td>JNE/ JNZ</td>
<td>JBE/ JNA</td>
<td>JNBE/ JA</td>
<td>JAE</td>
</tr>
<tr>
<td>7</td>
<td>Immed b,r/m</td>
<td>Immed w,r/m</td>
<td>Immed b,r/m</td>
<td>Immed w,r/m</td>
<td>TEST b,r/m</td>
<td>TEST w,r/m</td>
<td>XCHG b,r/m</td>
<td>XCHG w,r/m</td>
</tr>
<tr>
<td>8</td>
<td>NOP</td>
<td>XCHG CX</td>
<td>XCHG DX</td>
<td>XCHG BX</td>
<td>XCHG SP</td>
<td>XCHG BP</td>
<td>XCHG SI</td>
<td>XCHG</td>
</tr>
<tr>
<td>9</td>
<td>MOV m AL</td>
<td>MOV m AX</td>
<td>MOV m AX</td>
<td>MOV m AX</td>
<td>MOVS b</td>
<td>MOVS w</td>
<td>CMPS b</td>
<td>CMPS w</td>
</tr>
<tr>
<td>A</td>
<td>MOV i AL</td>
<td>MOV i CL</td>
<td>MOV i DL</td>
<td>MOV i BL</td>
<td>MOV i AH</td>
<td>MOV i CH</td>
<td>MOV i DH</td>
<td>MOV</td>
</tr>
<tr>
<td>B</td>
<td>RET (i+SP)</td>
<td>RET</td>
<td>LES</td>
<td>LDS</td>
<td>MOV b,i,r/m</td>
<td>MOV w,i,r/m</td>
<td>MOV</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>Shift b</td>
<td>Shift w</td>
<td>Shift b,v</td>
<td>Shift w,v</td>
<td>AAM</td>
<td>AAD</td>
<td>XLAT</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>LOOPNZ/ LOOPNE</td>
<td>LOOPZ/ LOOPE</td>
<td>LOOP</td>
<td>JCXZ</td>
<td>IN b</td>
<td>IN w</td>
<td>OUT b</td>
<td>OUT w</td>
</tr>
<tr>
<td>E</td>
<td>REP</td>
<td>REP</td>
<td>REP</td>
<td>HLT</td>
<td>CMC</td>
<td>Grp 1 b,r/m</td>
<td>Grp 1 w,r/m</td>
<td></td>
</tr>
</tbody>
</table>

b = byte operation  
m = memory  
d = direct  
r/m = EA is second byte  
f = from CPU reg  
si = short intrasegment  
i = immediate  
sr = segment register  
ia = immed. to accum.  
t = to CPU reg  
id = indirect  
v = variable  
is = immed. byte, sign ext.  
w = word operation  
l = long ie. intersegment  
z = zero
### 8088 Instruction Set Matrix

<table>
<thead>
<tr>
<th>LO</th>
<th>HI</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>0</td>
<td></td>
<td>OR</td>
<td>SBB</td>
<td>SUB</td>
<td>CMP</td>
<td>DEC</td>
<td>POP</td>
<td>AX</td>
<td>J S</td>
<td>MOV</td>
<td>CBW</td>
<td>TEST</td>
<td>MOV</td>
<td>RET</td>
<td>ESC</td>
<td>CALL</td>
<td>CLC</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>b,f,r/m</td>
<td>b,f,r/m</td>
<td>b,f,r/m</td>
<td>b,f,r/m</td>
<td>b,f,r/m</td>
<td>b,t,r/m</td>
<td>b,t,r/m</td>
<td>w,f,r/m</td>
<td>w,f,r/m</td>
<td>w,f,r/m</td>
<td>b,i</td>
<td>b,t,r/m</td>
<td>w,t,r/m</td>
<td>b,i</td>
<td>w,i</td>
<td>PUSH</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>OR</td>
<td>SBB</td>
<td>SUB</td>
<td>CMP</td>
<td>DEC</td>
<td>POP</td>
<td>AX</td>
<td>J S</td>
<td>MOV</td>
<td>CBW</td>
<td>TEST</td>
<td>MOV</td>
<td>RET</td>
<td>ESC</td>
<td>CALL</td>
<td>CLC</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>OR</td>
<td>SBB</td>
<td>SUB</td>
<td>CMP</td>
<td>DEC</td>
<td>POP</td>
<td>AX</td>
<td>J S</td>
<td>MOV</td>
<td>CBW</td>
<td>TEST</td>
<td>MOV</td>
<td>RET</td>
<td>ESC</td>
<td>CALL</td>
<td>CLC</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>OR</td>
<td>SBB</td>
<td>SUB</td>
<td>CMP</td>
<td>DEC</td>
<td>POP</td>
<td>AX</td>
<td>J S</td>
<td>MOV</td>
<td>CBW</td>
<td>TEST</td>
<td>MOV</td>
<td>RET</td>
<td>ESC</td>
<td>CALL</td>
<td>CLC</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>OR</td>
<td>SBB</td>
<td>SUB</td>
<td>CMP</td>
<td>DEC</td>
<td>POP</td>
<td>AX</td>
<td>J S</td>
<td>MOV</td>
<td>CBW</td>
<td>TEST</td>
<td>MOV</td>
<td>RET</td>
<td>ESC</td>
<td>CALL</td>
<td>CLC</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>OR</td>
<td>SBB</td>
<td>SUB</td>
<td>CMP</td>
<td>DEC</td>
<td>POP</td>
<td>AX</td>
<td>J S</td>
<td>MOV</td>
<td>CBW</td>
<td>TEST</td>
<td>MOV</td>
<td>RET</td>
<td>ESC</td>
<td>CALL</td>
<td>CLC</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>OR</td>
<td>SBB</td>
<td>SUB</td>
<td>CMP</td>
<td>DEC</td>
<td>POP</td>
<td>AX</td>
<td>J S</td>
<td>MOV</td>
<td>CBW</td>
<td>TEST</td>
<td>MOV</td>
<td>RET</td>
<td>ESC</td>
<td>CALL</td>
<td>CLC</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>OR</td>
<td>SBB</td>
<td>SUB</td>
<td>CMP</td>
<td>DEC</td>
<td>POP</td>
<td>AX</td>
<td>J S</td>
<td>MOV</td>
<td>CBW</td>
<td>TEST</td>
<td>MOV</td>
<td>RET</td>
<td>ESC</td>
<td>CALL</td>
<td>CLC</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>OR</td>
<td>SBB</td>
<td>SUB</td>
<td>CMP</td>
<td>DEC</td>
<td>POP</td>
<td>AX</td>
<td>J S</td>
<td>MOV</td>
<td>CBW</td>
<td>TEST</td>
<td>MOV</td>
<td>RET</td>
<td>ESC</td>
<td>CALL</td>
<td>CLC</td>
</tr>
</tbody>
</table>

**where:**

- **modo**: 000 001 010 011 100 101 110 111
- **Immed**: ADD OR ADC SBB AND SUB XOR CMP
- **Shift**: ROL ROR RCL RCR SHL/SAL SHR – SAR
- **Grp 1**: TEST – NOT NEG MUL IMUL DIV IDIV
- **Grp 2**: INC DEC CALL CALL JMP JMP PUSH –

---

**8088 Instruction Reference** B-17
<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Page</th>
<th>Mnemonic</th>
<th>Page</th>
<th>Mnemonic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAA</td>
<td>B-7</td>
<td>JG</td>
<td>B-13</td>
<td>MOV</td>
<td>B-5</td>
</tr>
<tr>
<td>AAD</td>
<td>B-9</td>
<td>JGE</td>
<td>B-12</td>
<td>MOVMS</td>
<td>B-10</td>
</tr>
<tr>
<td>AAM</td>
<td>B-8</td>
<td>JL</td>
<td>B-12</td>
<td>MUL</td>
<td>B-8</td>
</tr>
<tr>
<td>AAS</td>
<td>B-8</td>
<td>JLE</td>
<td>B-12</td>
<td>NEG</td>
<td>B-8</td>
</tr>
<tr>
<td>ADC</td>
<td>B-7</td>
<td>JMP</td>
<td>B-11</td>
<td>NOP</td>
<td>B-15</td>
</tr>
<tr>
<td>ADD</td>
<td>B-7</td>
<td>JNA</td>
<td>B-12</td>
<td>NOT</td>
<td>B-9</td>
</tr>
<tr>
<td>AND</td>
<td>B-9</td>
<td>JNAE</td>
<td>B-12</td>
<td>OR</td>
<td>B-10</td>
</tr>
<tr>
<td>CALL</td>
<td>B-11</td>
<td>JNB</td>
<td>B-13</td>
<td>OUT</td>
<td>B-6</td>
</tr>
<tr>
<td>CBW</td>
<td>B-9</td>
<td>JNBE</td>
<td>B-13</td>
<td>POP</td>
<td>B-5</td>
</tr>
<tr>
<td>CLC</td>
<td>B-15</td>
<td>JNE</td>
<td>B-12</td>
<td>POPF</td>
<td>B-6</td>
</tr>
<tr>
<td>CLD</td>
<td>B-15</td>
<td>JNG</td>
<td>B-12</td>
<td>PUSH</td>
<td>B-5</td>
</tr>
<tr>
<td>CLI</td>
<td>B-15</td>
<td>JNGE</td>
<td>B-12</td>
<td>PUSHF</td>
<td>B-6</td>
</tr>
<tr>
<td>CMC</td>
<td>B-15</td>
<td>JNL</td>
<td>B-12</td>
<td>RCL</td>
<td>B-9</td>
</tr>
<tr>
<td>CMP</td>
<td>B-8</td>
<td>JNLE</td>
<td>B-13</td>
<td>RCR</td>
<td>B-9</td>
</tr>
<tr>
<td>CMPS</td>
<td>B-10</td>
<td>JNO</td>
<td>B-13</td>
<td>REP</td>
<td>B-10</td>
</tr>
<tr>
<td>CWD</td>
<td>B-9</td>
<td>JNP</td>
<td>B-13</td>
<td>RET</td>
<td>B-12</td>
</tr>
<tr>
<td>DAA</td>
<td>B-7</td>
<td>JNS</td>
<td>B-13</td>
<td>ROL</td>
<td>B-9</td>
</tr>
<tr>
<td>DAS</td>
<td>B-8</td>
<td>JNZ</td>
<td>B-12</td>
<td>ROR</td>
<td>B-9</td>
</tr>
<tr>
<td>DEC</td>
<td>B-8</td>
<td>JO</td>
<td>B-12</td>
<td>SAHF</td>
<td>B-6</td>
</tr>
<tr>
<td>DIV</td>
<td>B-8</td>
<td>JP</td>
<td>B-12</td>
<td>SAL</td>
<td>B-9</td>
</tr>
<tr>
<td>ESC</td>
<td>B-15</td>
<td>JPE</td>
<td>B-12</td>
<td>SAR</td>
<td>B-9</td>
</tr>
<tr>
<td>HLT</td>
<td>B-15</td>
<td>JPO</td>
<td>B-13</td>
<td>SBB</td>
<td>B-8</td>
</tr>
<tr>
<td>IDIV</td>
<td>B-9</td>
<td>JS</td>
<td>B-12</td>
<td>SCAS</td>
<td>B-10</td>
</tr>
<tr>
<td>IMUL</td>
<td>B-8</td>
<td>JZ</td>
<td>B-12</td>
<td>SHL</td>
<td>B-9</td>
</tr>
<tr>
<td>IN</td>
<td>B-6</td>
<td>LAHF</td>
<td>B-6</td>
<td>SHR</td>
<td>B-9</td>
</tr>
<tr>
<td>INC</td>
<td>B-7</td>
<td>LDS</td>
<td>B-6</td>
<td>STC</td>
<td>B-15</td>
</tr>
<tr>
<td>INT</td>
<td>B-14</td>
<td>LEA</td>
<td>B-6</td>
<td>STD</td>
<td>B-15</td>
</tr>
<tr>
<td>INTO</td>
<td>B-14</td>
<td>LES</td>
<td>B-6</td>
<td>STI</td>
<td>B-15</td>
</tr>
<tr>
<td>IRET</td>
<td>B-14</td>
<td>LOCK</td>
<td>B-15</td>
<td>STOS</td>
<td>B-11</td>
</tr>
<tr>
<td>JA</td>
<td>B-13</td>
<td>LODS</td>
<td>B-11</td>
<td>SUB</td>
<td>B-7</td>
</tr>
<tr>
<td>JAE</td>
<td>B-13</td>
<td>LOOP</td>
<td>B-13</td>
<td>TEST</td>
<td>B-10</td>
</tr>
<tr>
<td>JB</td>
<td>B-12</td>
<td>LOOPE</td>
<td>B-13</td>
<td>WAIT</td>
<td>B-15</td>
</tr>
<tr>
<td>JBE</td>
<td>B-12</td>
<td>LOOPNE</td>
<td>B-13</td>
<td>XCHG</td>
<td>B-6</td>
</tr>
<tr>
<td>JCXZ</td>
<td>B-13</td>
<td>LOOPNZ</td>
<td>B-13</td>
<td>XLAT</td>
<td>B-6</td>
</tr>
<tr>
<td>JE</td>
<td>B-12</td>
<td>LOOPZ</td>
<td>B-13</td>
<td>XOR</td>
<td>B-10</td>
</tr>
</tbody>
</table>
## APPENDIX C: OF CHARACTERS, KEYSTROKES, AND COLOR

<table>
<thead>
<tr>
<th>Value</th>
<th>As Characters</th>
<th>Color/Graphics Attributes</th>
<th>IBM Monitor Adapter</th>
<th>IBM Monochrome Display Adapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hex</td>
<td>Dec</td>
<td>Symbol</td>
<td>Keystrokes</td>
<td>Modes</td>
</tr>
<tr>
<td>00</td>
<td>0</td>
<td>Blank (Null)</td>
<td>Ctrl 2</td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>1</td>
<td>😎</td>
<td>Ctrl A</td>
<td></td>
</tr>
<tr>
<td>02</td>
<td>2</td>
<td>😋</td>
<td>Ctrl B</td>
<td></td>
</tr>
<tr>
<td>03</td>
<td>3</td>
<td>😍</td>
<td>Ctrl C</td>
<td></td>
</tr>
<tr>
<td>04</td>
<td>4</td>
<td>🌟</td>
<td>Ctrl D</td>
<td></td>
</tr>
<tr>
<td>05</td>
<td>5</td>
<td>🌟</td>
<td>Ctrl E</td>
<td></td>
</tr>
<tr>
<td>06</td>
<td>6</td>
<td>🌟</td>
<td>Ctrl F</td>
<td></td>
</tr>
<tr>
<td>07</td>
<td>7</td>
<td>🌟</td>
<td>Ctrl G</td>
<td></td>
</tr>
<tr>
<td>08</td>
<td>8</td>
<td>🌟</td>
<td>Ctrl H, Backspace, Shift Backspace</td>
<td></td>
</tr>
<tr>
<td>09</td>
<td>9</td>
<td>🌟</td>
<td>Ctrl I</td>
<td></td>
</tr>
<tr>
<td>0A</td>
<td>10</td>
<td>🌟</td>
<td>Ctrl J, Ctrl L</td>
<td></td>
</tr>
<tr>
<td>0B</td>
<td>11</td>
<td>🌟</td>
<td>Ctrl K</td>
<td></td>
</tr>
<tr>
<td>0C</td>
<td>12</td>
<td>🌟</td>
<td>Ctrl L,</td>
<td></td>
</tr>
<tr>
<td>0D</td>
<td>13</td>
<td>🌟</td>
<td>Ctrl M, Shift M</td>
<td></td>
</tr>
<tr>
<td>0E</td>
<td>14</td>
<td>🌟</td>
<td>Ctrl N</td>
<td></td>
</tr>
<tr>
<td>0F</td>
<td>15</td>
<td>🌟</td>
<td>Ctrl O</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>16</td>
<td>🌟</td>
<td>Ctrl P</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>17</td>
<td>🌟</td>
<td>Ctrl Q</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>18</td>
<td>🌟</td>
<td>Ctrl R</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>19</td>
<td>🌟</td>
<td>Ctrl S</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>20</td>
<td>🌟</td>
<td>Ctrl T</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>21</td>
<td>🌟</td>
<td>Ctrl U</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>22</td>
<td>🌟</td>
<td>Ctrl V</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>23</td>
<td>🌟</td>
<td>Ctrl W</td>
<td></td>
</tr>
</tbody>
</table>

**Appendix C**

Of Characters, Keystrokes, and Colors
<table>
<thead>
<tr>
<th>Hex Dec</th>
<th>Symbol</th>
<th>Keystrokes</th>
<th>Modes</th>
<th>Background</th>
<th>FOREGROUND</th>
<th>IBM Monitor Adapter</th>
<th>IBM Monochrome Display Adapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>18 24</td>
<td>!</td>
<td>Ctrl X</td>
<td></td>
<td>Blue</td>
<td>Dark Grey</td>
<td>High Intensity</td>
<td></td>
</tr>
<tr>
<td>19 25</td>
<td>!</td>
<td>Ctrl Y</td>
<td></td>
<td>Blue</td>
<td>Light Blue</td>
<td>High Intensity Underline</td>
<td></td>
</tr>
<tr>
<td>1A 26</td>
<td>−</td>
<td>Ctrl Z</td>
<td></td>
<td>Blue</td>
<td>Light Green</td>
<td>High Intensity</td>
<td></td>
</tr>
<tr>
<td>1B 27</td>
<td>−</td>
<td>Ctrl [, Esc, Shift Esc, Ctrl Esc</td>
<td></td>
<td>Blue</td>
<td>Light Cyan</td>
<td>High Intensity</td>
<td></td>
</tr>
<tr>
<td>1C 28</td>
<td>\</td>
<td>Ctrl \</td>
<td></td>
<td>Blue</td>
<td>Light Red</td>
<td>High Intensity</td>
<td></td>
</tr>
<tr>
<td>1D 29</td>
<td>→</td>
<td>Ctrl ]</td>
<td></td>
<td>Blue</td>
<td>Light Magenta</td>
<td>High Intensity</td>
<td></td>
</tr>
<tr>
<td>1E 30</td>
<td>▲</td>
<td>Ctrl )</td>
<td></td>
<td>Blue</td>
<td>Yellow</td>
<td>High Intensity</td>
<td></td>
</tr>
<tr>
<td>1F 31</td>
<td>▼</td>
<td>Ctrl -</td>
<td></td>
<td>Blue</td>
<td>White</td>
<td>High Intensity</td>
<td></td>
</tr>
<tr>
<td>20 32</td>
<td>Blank Space</td>
<td>Space Bar, Shift, Space, Ctrl Space, Alt Space</td>
<td>Green</td>
<td>Black</td>
<td>Normal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21 33</td>
<td>!</td>
<td>!</td>
<td>Shift</td>
<td>Green</td>
<td>Blue</td>
<td>Underline</td>
<td></td>
</tr>
<tr>
<td>22 34</td>
<td>&quot;</td>
<td>&quot;</td>
<td>Shift</td>
<td>Green</td>
<td>Green</td>
<td>Normal</td>
<td></td>
</tr>
<tr>
<td>23 35</td>
<td>#</td>
<td>#</td>
<td>Shift</td>
<td>Green</td>
<td>Cyan</td>
<td>Normal</td>
<td></td>
</tr>
<tr>
<td>24 36</td>
<td>$</td>
<td>$</td>
<td>Shift</td>
<td>Green</td>
<td>Red</td>
<td>Normal</td>
<td></td>
</tr>
<tr>
<td>25 37</td>
<td>%</td>
<td>%</td>
<td>Shift</td>
<td>Green</td>
<td>Magenta</td>
<td>Normal</td>
<td></td>
</tr>
<tr>
<td>26 38</td>
<td>&amp;</td>
<td>&amp;</td>
<td>Shift</td>
<td>Green</td>
<td>Brown</td>
<td>Normal</td>
<td></td>
</tr>
<tr>
<td>27 39</td>
<td>'</td>
<td>'</td>
<td></td>
<td>Green</td>
<td>Light Grey</td>
<td>Normal</td>
<td></td>
</tr>
<tr>
<td>28 40</td>
<td>(</td>
<td>(</td>
<td>Shift</td>
<td>Green</td>
<td>Dark Grey</td>
<td>High Intensity</td>
<td></td>
</tr>
<tr>
<td>29 41</td>
<td>)</td>
<td>)</td>
<td>Shift</td>
<td>Green</td>
<td>Light Blue</td>
<td>High Intensity Underline</td>
<td></td>
</tr>
<tr>
<td>2A 42</td>
<td>&quot;</td>
<td>&quot; Note 1</td>
<td></td>
<td>Green</td>
<td>Light Green</td>
<td>High Intensity</td>
<td></td>
</tr>
<tr>
<td>2B 43</td>
<td>+</td>
<td>+</td>
<td>Shift</td>
<td>Green</td>
<td>Light Cyan</td>
<td>High Intensity</td>
<td></td>
</tr>
<tr>
<td>2C 44</td>
<td>'</td>
<td>'</td>
<td></td>
<td>Green</td>
<td>Light Red</td>
<td>High Intensity</td>
<td></td>
</tr>
<tr>
<td>2D 45</td>
<td>−</td>
<td>−</td>
<td></td>
<td>Green</td>
<td>Light Magenta</td>
<td>High Intensity</td>
<td></td>
</tr>
<tr>
<td>2E 46</td>
<td>.</td>
<td>Note 2</td>
<td></td>
<td>Green</td>
<td>Yellow</td>
<td>High Intensity</td>
<td></td>
</tr>
</tbody>
</table>

C-2 Of Characters, Keystrokes, and Colors
<table>
<thead>
<tr>
<th>Hex</th>
<th>Dec</th>
<th>Symbol</th>
<th>Keystrokes</th>
<th>Modes</th>
<th>Color/Graphics Monitor Adapter</th>
<th>IBM Monochrome Display Adapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>2F</td>
<td>47</td>
<td>/</td>
<td>/</td>
<td>Note 3</td>
<td>Green</td>
<td>High Intensity</td>
</tr>
<tr>
<td>30</td>
<td>48</td>
<td>0</td>
<td>0</td>
<td>Note 3</td>
<td>Cyan</td>
<td>Black</td>
</tr>
<tr>
<td>31</td>
<td>49</td>
<td>1</td>
<td>1</td>
<td>Note 3</td>
<td>Cyan</td>
<td>Blue</td>
</tr>
<tr>
<td>32</td>
<td>50</td>
<td>2</td>
<td>2</td>
<td>Note 3</td>
<td>Cyan</td>
<td>Green</td>
</tr>
<tr>
<td>33</td>
<td>51</td>
<td>3</td>
<td>3</td>
<td>Note 3</td>
<td>Cyan</td>
<td>Cyan</td>
</tr>
<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>Shift</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>Shift</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>
</tbody>
</table>

Of Characters, Keystrokes, and Colors   C-3
<table>
<thead>
<tr>
<th>Value</th>
<th>As Characters</th>
<th>As Text Attributes</th>
<th>Color/Graphics Monitor Adapter</th>
<th>IBM Monochrome Display Adapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hex Dec</td>
<td>Symbol</td>
<td>Keystrokes</td>
<td>Modes</td>
<td>Background</td>
</tr>
<tr>
<td>4B 75</td>
<td>K</td>
<td>K</td>
<td>Note 4</td>
<td>Red</td>
</tr>
<tr>
<td>4C 76</td>
<td>L</td>
<td>L</td>
<td>Note 4</td>
<td>Red</td>
</tr>
<tr>
<td>4D 77</td>
<td>M</td>
<td>M</td>
<td>Note 4</td>
<td>Red</td>
</tr>
<tr>
<td>4E 78</td>
<td>N</td>
<td>N</td>
<td>Note 4</td>
<td>Red</td>
</tr>
<tr>
<td>4F 79</td>
<td>O</td>
<td>O</td>
<td>Note 4</td>
<td>Red</td>
</tr>
<tr>
<td>50 80</td>
<td>P</td>
<td>P</td>
<td>Note 4</td>
<td>Magenta</td>
</tr>
<tr>
<td>51 81</td>
<td>Q</td>
<td>Q</td>
<td>Note 4</td>
<td>Magenta</td>
</tr>
<tr>
<td>52 82</td>
<td>R</td>
<td>R</td>
<td>Note 4</td>
<td>Magenta</td>
</tr>
<tr>
<td>53 83</td>
<td>S</td>
<td>S</td>
<td>Note 4</td>
<td>Magenta</td>
</tr>
<tr>
<td>54 84</td>
<td>T</td>
<td>T</td>
<td>Note 4</td>
<td>Magenta</td>
</tr>
<tr>
<td>55 85</td>
<td>U</td>
<td>U</td>
<td>Note 4</td>
<td>Magenta</td>
</tr>
<tr>
<td>56 86</td>
<td>V</td>
<td>V</td>
<td>Note 4</td>
<td>Magenta</td>
</tr>
<tr>
<td>57 87</td>
<td>W</td>
<td>W</td>
<td>Note 4</td>
<td>Magenta</td>
</tr>
<tr>
<td>58 88</td>
<td>X</td>
<td>X</td>
<td>Note 4</td>
<td>Magenta</td>
</tr>
<tr>
<td>59 89</td>
<td>Y</td>
<td>Y</td>
<td>Note 4</td>
<td>Magenta</td>
</tr>
<tr>
<td>5A 90</td>
<td>Z</td>
<td>Z</td>
<td>Note 4</td>
<td>Magenta</td>
</tr>
<tr>
<td>5B 91</td>
<td>[</td>
<td>[</td>
<td>Note 4</td>
<td>Magenta</td>
</tr>
<tr>
<td>5C 92</td>
<td>\</td>
<td>\</td>
<td>Note 4</td>
<td>Magenta</td>
</tr>
<tr>
<td>5D 93</td>
<td>]</td>
<td>]</td>
<td>Note 4</td>
<td>Magenta</td>
</tr>
<tr>
<td>5E 94</td>
<td>^</td>
<td>^</td>
<td>Shift</td>
<td>Magenta</td>
</tr>
<tr>
<td>5F 95</td>
<td>—</td>
<td>—</td>
<td>Shift</td>
<td>Magenta</td>
</tr>
<tr>
<td>60 96</td>
<td>\</td>
<td>\</td>
<td>Shift</td>
<td>Yellow</td>
</tr>
<tr>
<td>61 97</td>
<td>a</td>
<td>a</td>
<td>Note 5</td>
<td>Yellow</td>
</tr>
<tr>
<td>62 98</td>
<td>b</td>
<td>b</td>
<td>Note 5</td>
<td>Yellow</td>
</tr>
<tr>
<td>63 99</td>
<td>c</td>
<td>c</td>
<td>Note 5</td>
<td>Yellow</td>
</tr>
<tr>
<td>64 100</td>
<td>d</td>
<td>d</td>
<td>Note 5</td>
<td>Yellow</td>
</tr>
<tr>
<td>65 101</td>
<td>e</td>
<td>e</td>
<td>Note 5</td>
<td>Yellow</td>
</tr>
<tr>
<td>66 102</td>
<td>f</td>
<td>f</td>
<td>Note 5</td>
<td>Yellow</td>
</tr>
</tbody>
</table>

C-4 Of Characters, Keystrokes, and Colors
<table>
<thead>
<tr>
<th>Hex</th>
<th>Dec</th>
<th>Symbol</th>
<th>Keystrokes</th>
<th>Modes</th>
<th>Background</th>
<th>Foreground</th>
<th>IBM Monitor Adapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>67</td>
<td>103</td>
<td>g</td>
<td>g</td>
<td>Note 5</td>
<td>Yellow</td>
<td>Light Grey</td>
<td>Normal</td>
</tr>
<tr>
<td>68</td>
<td>104</td>
<td>h</td>
<td>h</td>
<td>Note 5</td>
<td>Yellow</td>
<td>Dark Grey</td>
<td>High Intensity</td>
</tr>
<tr>
<td>69</td>
<td>105</td>
<td>i</td>
<td>i</td>
<td>Note 5</td>
<td>Yellow</td>
<td>Light Blue</td>
<td>High Intensity Underline</td>
</tr>
<tr>
<td>6A</td>
<td>106</td>
<td>j</td>
<td>j</td>
<td>Note 5</td>
<td>Yellow</td>
<td>Light Green</td>
<td>High Intensity</td>
</tr>
<tr>
<td>6B</td>
<td>107</td>
<td>k</td>
<td>k</td>
<td>Note 5</td>
<td>Yellow</td>
<td>Light Cyan</td>
<td>High Intensity</td>
</tr>
<tr>
<td>6C</td>
<td>108</td>
<td>l</td>
<td>l</td>
<td>Note 5</td>
<td>Yellow</td>
<td>Light Red</td>
<td>High Intensity</td>
</tr>
<tr>
<td>6D</td>
<td>109</td>
<td>m</td>
<td>m</td>
<td>Note 5</td>
<td>Yellow</td>
<td>Light Magenta</td>
<td>High Intensity</td>
</tr>
<tr>
<td>6E</td>
<td>110</td>
<td>n</td>
<td>n</td>
<td>Note 5</td>
<td>Yellow</td>
<td>Yellow</td>
<td>High Intensity</td>
</tr>
<tr>
<td>6F</td>
<td>111</td>
<td>o</td>
<td>o</td>
<td>Note 5</td>
<td>Yellow</td>
<td>White</td>
<td>High Intensity</td>
</tr>
<tr>
<td>70</td>
<td>112</td>
<td>p</td>
<td>p</td>
<td>Note 5</td>
<td>White</td>
<td>Black</td>
<td>Reverse Video</td>
</tr>
<tr>
<td>71</td>
<td>113</td>
<td>q</td>
<td>q</td>
<td>Note 5</td>
<td>White</td>
<td>Blue</td>
<td>Underline</td>
</tr>
<tr>
<td>72</td>
<td>114</td>
<td>r</td>
<td>r</td>
<td>Note 5</td>
<td>White</td>
<td>Green</td>
<td>Normal</td>
</tr>
<tr>
<td>73</td>
<td>115</td>
<td>s</td>
<td>s</td>
<td>Note 5</td>
<td>White</td>
<td>Cyan</td>
<td>Normal</td>
</tr>
<tr>
<td>74</td>
<td>116</td>
<td>f</td>
<td>f</td>
<td>Note 5</td>
<td>White</td>
<td>Red</td>
<td>Normal</td>
</tr>
<tr>
<td>75</td>
<td>117</td>
<td>u</td>
<td>u</td>
<td>Note 5</td>
<td>White</td>
<td>Magenta</td>
<td>Normal</td>
</tr>
<tr>
<td>76</td>
<td>118</td>
<td>v</td>
<td>v</td>
<td>Note 5</td>
<td>White</td>
<td>Brown</td>
<td>Normal</td>
</tr>
<tr>
<td>77</td>
<td>119</td>
<td>w</td>
<td>w</td>
<td>Note 5</td>
<td>White</td>
<td>Light Grey</td>
<td>Normal</td>
</tr>
<tr>
<td>78</td>
<td>120</td>
<td>x</td>
<td>x</td>
<td>Note 5</td>
<td>White</td>
<td>Dark Grey</td>
<td>Reverse Video</td>
</tr>
<tr>
<td>79</td>
<td>121</td>
<td>y</td>
<td>y</td>
<td>Note 5</td>
<td>White</td>
<td>Light Blue</td>
<td>High Intensity Underline</td>
</tr>
<tr>
<td>7A</td>
<td>122</td>
<td>z</td>
<td>z</td>
<td>Note 5</td>
<td>White</td>
<td>Light Green</td>
<td>High Intensity</td>
</tr>
<tr>
<td>7B</td>
<td>123</td>
<td></td>
<td>Shift</td>
<td>White</td>
<td>Light Cyan</td>
<td>High Intensity</td>
<td></td>
</tr>
<tr>
<td>7C</td>
<td>124</td>
<td></td>
<td>Shift</td>
<td>White</td>
<td>Light Red</td>
<td>High Intensity</td>
<td></td>
</tr>
<tr>
<td>7D</td>
<td>125</td>
<td></td>
<td>Shift</td>
<td>White</td>
<td>Light Magenta</td>
<td>High Intensity</td>
<td></td>
</tr>
<tr>
<td>7E</td>
<td>126</td>
<td>~</td>
<td>~</td>
<td>Shift</td>
<td>White</td>
<td>Yellow</td>
<td>High Intensity</td>
</tr>
<tr>
<td>7F</td>
<td>127</td>
<td>Δ</td>
<td>Ctrl ~</td>
<td>White</td>
<td>White</td>
<td>High Intensity</td>
<td></td>
</tr>
<tr>
<td>Hex</td>
<td>Dec</td>
<td>Symbol</td>
<td>Keystrokes</td>
<td>Modes</td>
<td>Background</td>
<td>Foreground</td>
<td>IBM Monochrome Display Adapter</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>--------</td>
<td>------------</td>
<td>--------</td>
<td>------------</td>
<td>------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>80</td>
<td>128</td>
<td>Ç</td>
<td>Alt 128</td>
<td>Note 6</td>
<td>Black</td>
<td>Black</td>
<td>Non-Display</td>
</tr>
<tr>
<td>81</td>
<td>129</td>
<td>ü</td>
<td>Alt 129</td>
<td>Note 6</td>
<td>Black</td>
<td>Blue</td>
<td>Underline</td>
</tr>
<tr>
<td>82</td>
<td>130</td>
<td>é</td>
<td>Alt 130</td>
<td>Note 6</td>
<td>Black</td>
<td>Green</td>
<td>Normal</td>
</tr>
<tr>
<td>83</td>
<td>131</td>
<td>â</td>
<td>Alt 131</td>
<td>Note 6</td>
<td>Black</td>
<td>Cyan</td>
<td>Normal</td>
</tr>
<tr>
<td>84</td>
<td>132</td>
<td>ã</td>
<td>Alt 132</td>
<td>Note 6</td>
<td>Black</td>
<td>Red</td>
<td>Normal</td>
</tr>
<tr>
<td>85</td>
<td>133</td>
<td>à</td>
<td>Alt 133</td>
<td>Note 6</td>
<td>Black</td>
<td>Magenta</td>
<td>Normal</td>
</tr>
<tr>
<td>86</td>
<td>134</td>
<td>á</td>
<td>Alt 134</td>
<td>Note 6</td>
<td>Black</td>
<td>Brown</td>
<td>Normal</td>
</tr>
<tr>
<td>87</td>
<td>135</td>
<td>ç</td>
<td>Alt 135</td>
<td>Note 6</td>
<td>Black</td>
<td>Light Grey</td>
<td>Normal</td>
</tr>
<tr>
<td>88</td>
<td>136</td>
<td>è</td>
<td>Alt 136</td>
<td>Note 6</td>
<td>Black</td>
<td>Dark Grey</td>
<td>Non-Display</td>
</tr>
<tr>
<td>89</td>
<td>137</td>
<td>é</td>
<td>Alt 137</td>
<td>Note 6</td>
<td>Black</td>
<td>Light Blue</td>
<td>High Intensity Underline</td>
</tr>
<tr>
<td>8A</td>
<td>138</td>
<td>è</td>
<td>Alt 138</td>
<td>Note 6</td>
<td>Black</td>
<td>Light Green</td>
<td>High Intensity</td>
</tr>
<tr>
<td>8B</td>
<td>139</td>
<td>i</td>
<td>Alt 139</td>
<td>Note 6</td>
<td>Black</td>
<td>Light Cyan</td>
<td>High Intensity</td>
</tr>
<tr>
<td>8C</td>
<td>140</td>
<td>í</td>
<td>Alt 140</td>
<td>Note 6</td>
<td>Black</td>
<td>Light Red</td>
<td>High Intensity</td>
</tr>
<tr>
<td>8D</td>
<td>141</td>
<td>ì</td>
<td>Alt 141</td>
<td>Note 6</td>
<td>Black</td>
<td>Light Magenta</td>
<td>High Intensity</td>
</tr>
<tr>
<td>8E</td>
<td>142</td>
<td>A</td>
<td>Alt 142</td>
<td>Note 6</td>
<td>Black</td>
<td>Yellow</td>
<td>High Intensity</td>
</tr>
<tr>
<td>8F</td>
<td>143</td>
<td>Æ</td>
<td>Alt 143</td>
<td>Note 6</td>
<td>Black</td>
<td>White</td>
<td>High Intensity</td>
</tr>
<tr>
<td>90</td>
<td>144</td>
<td>Ê</td>
<td>Alt 144</td>
<td>Note 6</td>
<td>Blue</td>
<td>Black</td>
<td>Normal</td>
</tr>
<tr>
<td>91</td>
<td>145</td>
<td>æ</td>
<td>Alt 145</td>
<td>Note 6</td>
<td>Blue</td>
<td>Blue</td>
<td>Underline</td>
</tr>
<tr>
<td>92</td>
<td>146</td>
<td>Ä</td>
<td>Alt 146</td>
<td>Note 6</td>
<td>Blue</td>
<td>Green</td>
<td>Normal</td>
</tr>
<tr>
<td>93</td>
<td>147</td>
<td>ô</td>
<td>Alt 147</td>
<td>Note 6</td>
<td>Blue</td>
<td>Cyan</td>
<td>Normal</td>
</tr>
<tr>
<td>94</td>
<td>148</td>
<td>ò</td>
<td>Alt 148</td>
<td>Note 6</td>
<td>Blue</td>
<td>Red</td>
<td>Normal</td>
</tr>
<tr>
<td>95</td>
<td>149</td>
<td>õ</td>
<td>Alt 149</td>
<td>Note 6</td>
<td>Blue</td>
<td>Magenta</td>
<td>Normal</td>
</tr>
<tr>
<td>96</td>
<td>150</td>
<td>ù</td>
<td>Alt 150</td>
<td>Note 6</td>
<td>Blue</td>
<td>Brown</td>
<td>Normal</td>
</tr>
<tr>
<td>97</td>
<td>151</td>
<td>ü</td>
<td>Alt 151</td>
<td>Note 6</td>
<td>Blue</td>
<td>Light Grey</td>
<td>Normal</td>
</tr>
<tr>
<td>98</td>
<td>152</td>
<td>ý</td>
<td>Alt 152</td>
<td>Note 6</td>
<td>Blue</td>
<td>Dark Grey</td>
<td>High Intensity</td>
</tr>
<tr>
<td>99</td>
<td>153</td>
<td>ö</td>
<td>Alt 153</td>
<td>Note 6</td>
<td>Blue</td>
<td>Light Blue</td>
<td>High Intensity Underline</td>
</tr>
<tr>
<td>9A</td>
<td>154</td>
<td>û</td>
<td>Alt 154</td>
<td>Note 6</td>
<td>Blue</td>
<td>Light Green</td>
<td>High Intensity</td>
</tr>
</tbody>
</table>

**C-6** Of Characters, Keystrokes, and Colors
<table>
<thead>
<tr>
<th>Value</th>
<th>As Characters</th>
<th>As Text Attributes</th>
<th>IBM Monochrome Display Adapter</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Color/Graphics</td>
<td>Monitor Adapter</td>
</tr>
<tr>
<td>Hex</td>
<td>Dec</td>
<td>Symbol</td>
<td>Keystrokes</td>
</tr>
<tr>
<td>9B</td>
<td>155</td>
<td>€</td>
<td>Alt 155</td>
</tr>
<tr>
<td>9C</td>
<td>156</td>
<td>£</td>
<td>Alt 156</td>
</tr>
<tr>
<td>9D</td>
<td>157</td>
<td>¥</td>
<td>Alt 157</td>
</tr>
<tr>
<td>9E</td>
<td>158</td>
<td>Pt</td>
<td>Alt 158</td>
</tr>
<tr>
<td>9F</td>
<td>159</td>
<td>£</td>
<td>Alt 159</td>
</tr>
<tr>
<td>A0</td>
<td>160</td>
<td>á</td>
<td>Alt 160</td>
</tr>
<tr>
<td>A1</td>
<td>161</td>
<td>í</td>
<td>Alt 161</td>
</tr>
<tr>
<td>A2</td>
<td>162</td>
<td>ó</td>
<td>Alt 162</td>
</tr>
<tr>
<td>A3</td>
<td>163</td>
<td>ü</td>
<td>Alt 163</td>
</tr>
<tr>
<td>A4</td>
<td>164</td>
<td>ñ</td>
<td>Alt 164</td>
</tr>
<tr>
<td>A5</td>
<td>165</td>
<td>Ñ</td>
<td>Alt 165</td>
</tr>
<tr>
<td>A6</td>
<td>166</td>
<td>a</td>
<td>Alt 166</td>
</tr>
<tr>
<td>A7</td>
<td>167</td>
<td>o</td>
<td>Alt 167</td>
</tr>
<tr>
<td>A8</td>
<td>168</td>
<td>ñ</td>
<td>Alt 168</td>
</tr>
<tr>
<td>A9</td>
<td>169</td>
<td>r</td>
<td>Alt 169</td>
</tr>
<tr>
<td>AA</td>
<td>170</td>
<td>–</td>
<td>Alt 170</td>
</tr>
<tr>
<td>AB</td>
<td>171</td>
<td>½</td>
<td>Alt 171</td>
</tr>
<tr>
<td>AC</td>
<td>172</td>
<td>¼</td>
<td>Alt 172</td>
</tr>
<tr>
<td>AD</td>
<td>173</td>
<td>⅐</td>
<td>Alt 173</td>
</tr>
<tr>
<td>AE</td>
<td>174</td>
<td>«</td>
<td>Alt 174</td>
</tr>
<tr>
<td>AF</td>
<td>175</td>
<td>»</td>
<td>Alt 175</td>
</tr>
<tr>
<td>B0</td>
<td>176</td>
<td>…</td>
<td>Alt 176</td>
</tr>
<tr>
<td>B1</td>
<td>177</td>
<td>ⅈ</td>
<td>Alt 177</td>
</tr>
<tr>
<td>B2</td>
<td>178</td>
<td>ⅉ</td>
<td>Alt 178</td>
</tr>
<tr>
<td>B3</td>
<td>179</td>
<td></td>
<td>Alt 179</td>
</tr>
<tr>
<td>B4</td>
<td>180</td>
<td></td>
<td>Alt 180</td>
</tr>
<tr>
<td>B5</td>
<td>181</td>
<td></td>
<td>Alt 181</td>
</tr>
<tr>
<td>B6</td>
<td>182</td>
<td></td>
<td>Alt 182</td>
</tr>
<tr>
<td>Hex/Dec</td>
<td>As Characters</td>
<td>Keystrokes</td>
<td>Modes</td>
</tr>
<tr>
<td>---------</td>
<td>---------------</td>
<td>------------</td>
<td>-------</td>
</tr>
<tr>
<td>B7 183</td>
<td>Alt 183</td>
<td>Note 6</td>
<td>Cyan</td>
</tr>
<tr>
<td>B8 184</td>
<td>Alt 184</td>
<td>Note 6</td>
<td>Cyan</td>
</tr>
<tr>
<td>B9 185</td>
<td>Alt 185</td>
<td>Note 6</td>
<td>Cyan</td>
</tr>
<tr>
<td>BA 186</td>
<td>Alt 186</td>
<td>Note 6</td>
<td>Cyan</td>
</tr>
<tr>
<td>BB 187</td>
<td>Alt 187</td>
<td>Note 6</td>
<td>Cyan</td>
</tr>
<tr>
<td>BC 188</td>
<td>Alt 188</td>
<td>Note 6</td>
<td>Cyan</td>
</tr>
<tr>
<td>BD 189</td>
<td>Alt 189</td>
<td>Note 6</td>
<td>Cyan</td>
</tr>
<tr>
<td>BE 190</td>
<td>Alt 190</td>
<td>Note 6</td>
<td>Cyan</td>
</tr>
<tr>
<td>BF 191</td>
<td>Alt 191</td>
<td>Note 6</td>
<td>Cyan</td>
</tr>
<tr>
<td>C0 192</td>
<td>Alt 192</td>
<td>Note 6</td>
<td>Red</td>
</tr>
<tr>
<td>C1 193</td>
<td>Alt 193</td>
<td>Note 6</td>
<td>Red</td>
</tr>
<tr>
<td>C2 194</td>
<td>Alt 194</td>
<td>Note 6</td>
<td>Red</td>
</tr>
<tr>
<td>C3 195</td>
<td>Alt 195</td>
<td>Note 6</td>
<td>Red</td>
</tr>
<tr>
<td>C4 196</td>
<td>Alt 196</td>
<td>Note 6</td>
<td>Red</td>
</tr>
<tr>
<td>C5 197</td>
<td>Alt 197</td>
<td>Note 6</td>
<td>Red</td>
</tr>
<tr>
<td>C6 198</td>
<td>Alt 198</td>
<td>Note 6</td>
<td>Red</td>
</tr>
<tr>
<td>C7 199</td>
<td>Alt 199</td>
<td>Note 6</td>
<td>Red</td>
</tr>
<tr>
<td>C8 200</td>
<td>Alt 200</td>
<td>Note 6</td>
<td>Red</td>
</tr>
<tr>
<td>C9 201</td>
<td>Alt 201</td>
<td>Note 6</td>
<td>Red</td>
</tr>
<tr>
<td>CA 202</td>
<td>Alt 202</td>
<td>Note 6</td>
<td>Red</td>
</tr>
<tr>
<td>CB 203</td>
<td>Alt 203</td>
<td>Note 6</td>
<td>Red</td>
</tr>
<tr>
<td>CC 204</td>
<td>Alt 204</td>
<td>Note 6</td>
<td>Red</td>
</tr>
<tr>
<td>CD 205</td>
<td>Alt 205</td>
<td>Note 6</td>
<td>Red</td>
</tr>
<tr>
<td>CE 206</td>
<td>Alt 206</td>
<td>Note 6</td>
<td>Red</td>
</tr>
<tr>
<td>CF 207</td>
<td>Alt 207</td>
<td>Note 6</td>
<td>Red</td>
</tr>
<tr>
<td>DO 208</td>
<td>Alt 208</td>
<td>Note 6</td>
<td>Magenta</td>
</tr>
</tbody>
</table>

C-8 Of Characters, Keystrokes, and Colors
<table>
<thead>
<tr>
<th>Value</th>
<th>As Characters</th>
<th>Color/Graphics Monitor Adapter</th>
<th>IBM Monochrome Display Adapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hex</td>
<td>Symbol</td>
<td>Keystrokes</td>
<td>Modes</td>
</tr>
<tr>
<td>D1 209</td>
<td>Alt 209</td>
<td>Note 6</td>
<td>Magenta</td>
</tr>
<tr>
<td>D2 210</td>
<td>Alt 210</td>
<td>Note 6</td>
<td>Magenta</td>
</tr>
<tr>
<td>D3 211</td>
<td>Alt 211</td>
<td>Note 6</td>
<td>Magenta</td>
</tr>
<tr>
<td>D4 212</td>
<td>Alt 212</td>
<td>Note 6</td>
<td>Magenta</td>
</tr>
<tr>
<td>D5 213</td>
<td>Alt 213</td>
<td>Note 6</td>
<td>Magenta</td>
</tr>
<tr>
<td>D6 214</td>
<td>Alt 214</td>
<td>Note 6</td>
<td>Magenta</td>
</tr>
<tr>
<td>D7 215</td>
<td>Alt 215</td>
<td>Note 6</td>
<td>Magenta</td>
</tr>
<tr>
<td>D8 216</td>
<td>Alt 216</td>
<td>Note 6</td>
<td>Magenta</td>
</tr>
<tr>
<td>D9 217</td>
<td>Alt 217</td>
<td>Note 6</td>
<td>Magenta</td>
</tr>
<tr>
<td>DA 218</td>
<td>Alt 218</td>
<td>Note 6</td>
<td>Magenta</td>
</tr>
<tr>
<td>DB 219</td>
<td>Alt 219</td>
<td>Note 6</td>
<td>Magenta</td>
</tr>
<tr>
<td>DC 220</td>
<td>Alt 220</td>
<td>Note 6</td>
<td>Magenta</td>
</tr>
<tr>
<td>DD 221</td>
<td>Alt 221</td>
<td>Note 6</td>
<td>Magenta</td>
</tr>
<tr>
<td>DE 222</td>
<td>Alt 222</td>
<td>Note 6</td>
<td>Magenta</td>
</tr>
<tr>
<td>DF 223</td>
<td>Alt 223</td>
<td>Note 6</td>
<td>Magenta</td>
</tr>
<tr>
<td>E0 224</td>
<td>α</td>
<td>Alt 224</td>
<td>Note 6</td>
</tr>
<tr>
<td>E1 225</td>
<td>β</td>
<td>Alt 225</td>
<td>Note 6</td>
</tr>
<tr>
<td>E2 226</td>
<td>γ</td>
<td>Alt 226</td>
<td>Note 6</td>
</tr>
<tr>
<td>E3 227</td>
<td>π</td>
<td>Alt 227</td>
<td>Note 6</td>
</tr>
<tr>
<td>E4 228</td>
<td>Σ</td>
<td>Alt 228</td>
<td>Note 6</td>
</tr>
<tr>
<td>E5 229</td>
<td>σ</td>
<td>Alt 229</td>
<td>Note 6</td>
</tr>
<tr>
<td>E6 230</td>
<td>μ</td>
<td>Alt 230</td>
<td>Note 6</td>
</tr>
<tr>
<td>E7 231</td>
<td>τ</td>
<td>Alt 231</td>
<td>Note 6</td>
</tr>
<tr>
<td>E8 232</td>
<td>Φ</td>
<td>Alt 232</td>
<td>Note 6</td>
</tr>
<tr>
<td>E9 233</td>
<td>θ</td>
<td>Alt 233</td>
<td>Note 6</td>
</tr>
<tr>
<td>EA 234</td>
<td>Ω</td>
<td>Alt 234</td>
<td>Note 6</td>
</tr>
<tr>
<td>EB 235</td>
<td>δ</td>
<td>Alt 235</td>
<td>Note 6</td>
</tr>
<tr>
<td>Hex</td>
<td>Dec</td>
<td>Symbol</td>
<td>Keystrokes</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>--------</td>
<td>------------</td>
</tr>
<tr>
<td>EC</td>
<td>236</td>
<td>∞</td>
<td>Alt 236</td>
</tr>
<tr>
<td>ED</td>
<td>237</td>
<td>ɸ</td>
<td>Alt 237</td>
</tr>
<tr>
<td>EE</td>
<td>238</td>
<td>ε</td>
<td>Alt 238</td>
</tr>
<tr>
<td>EF</td>
<td>239</td>
<td>ʌ</td>
<td>Alt 239</td>
</tr>
<tr>
<td>F0</td>
<td>240</td>
<td>≡</td>
<td>Alt 240</td>
</tr>
<tr>
<td>F1</td>
<td>241</td>
<td>±</td>
<td>Alt 241</td>
</tr>
<tr>
<td>F2</td>
<td>242</td>
<td>≥</td>
<td>Alt 242</td>
</tr>
<tr>
<td>F3</td>
<td>243</td>
<td>≤</td>
<td>Alt 243</td>
</tr>
<tr>
<td>F4</td>
<td>244</td>
<td>⊥</td>
<td>Alt 244</td>
</tr>
<tr>
<td>F5</td>
<td>245</td>
<td>‖</td>
<td>Alt 245</td>
</tr>
<tr>
<td>F6</td>
<td>246</td>
<td>÷</td>
<td>Alt 246</td>
</tr>
<tr>
<td>F7</td>
<td>247</td>
<td>≈</td>
<td>Alt 247</td>
</tr>
<tr>
<td>F8</td>
<td>248</td>
<td>∅</td>
<td>Alt 248</td>
</tr>
<tr>
<td>F9</td>
<td>249</td>
<td>•</td>
<td>Alt 249</td>
</tr>
<tr>
<td>FA</td>
<td>250</td>
<td>•</td>
<td>Alt 250</td>
</tr>
<tr>
<td>FB</td>
<td>251</td>
<td>√</td>
<td>Alt 251</td>
</tr>
<tr>
<td>FC</td>
<td>252</td>
<td>η</td>
<td>Alt 252</td>
</tr>
<tr>
<td>FD</td>
<td>253</td>
<td>2</td>
<td>Alt 253</td>
</tr>
<tr>
<td>FE</td>
<td>254</td>
<td>□</td>
<td>Alt 254</td>
</tr>
<tr>
<td>FF</td>
<td>255</td>
<td>BLANK</td>
<td>Alt 255</td>
</tr>
</tbody>
</table>

C-10 Of Characters, Keystrokes, and Colors
NOTE 1 Asterisk (*) can easily be keyed using two methods: 1) hit the \textsc{Prt Sc} key or 2) in shift mode hit the \textbf{8} key.

NOTE 2 Period (.) can easily be keyed using two methods: 1) hit the \textbf{.} key or 2) in shift or Num Lock mode hit the \textbf{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) in shift or Num Lock mode hit the numeric keys in the 10-key pad portion of the keyboard.

NOTE 4 Upper case alphabetic characters (A—Z) can easily be keyed in two modes: 1) in shift mode the appropriate alphabetic key or 2) in Caps Lock mode hit the appropriate alphabetic key.

NOTE 5 Lower case alphabetic characters (a—z) can easily be keyed in two modes: 1) in “normal” mode hit the appropriate key or 2) in Caps Lock combined with shift mode hit the appropriate alphabetic key.

NOTE 6 The 3 digits after the Alt key must be typed from the numeric key pad (keys 71—73, 75—77, 79—82). Character codes 000 through 255 can be entered in this fashion. (With Caps Lock activated, character codes 97 through 122 will display upper case rather than lower case alphabetic characters.)
<table>
<thead>
<tr>
<th>DECIMAL VALUE</th>
<th>HEXA 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>@</td>
<td>P</td>
<td>'</td>
<td>p</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>A</td>
<td>Q</td>
<td>a</td>
<td>q</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>B</td>
<td>R</td>
<td>b</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>C</td>
<td>S</td>
<td>c</td>
<td>s</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>D</td>
<td>T</td>
<td>d</td>
<td>t</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>E</td>
<td>U</td>
<td>e</td>
<td>u</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>F</td>
<td>V</td>
<td>f</td>
<td>v</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>G</td>
<td>W</td>
<td>g</td>
<td>w</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>H</td>
<td>X</td>
<td>h</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>I</td>
<td>Y</td>
<td>i</td>
<td>y</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>A</td>
<td>J</td>
<td>Z</td>
<td>j</td>
<td>z</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>B</td>
<td>K</td>
<td>I</td>
<td>l</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>L</td>
<td>M</td>
<td>m</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>D</td>
<td>N</td>
<td>O</td>
<td>n</td>
<td>~</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>E</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DECIMAL</td>
<td>VALUE</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>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<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>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<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>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<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>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<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>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HEXA</th>
<th>128 144 160 176 192 208 224 240</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8 9 A B C D E F</td>
</tr>
</tbody>
</table>

- Of Characters, Keystrokes, and Colors  C-13
Notes:
APPENDIX D: LOGIC DIAGRAMS

System Board .................................................. D-2
Keyboard – Type 1 .............................................. D-12
Keyboard – Type 2 .............................................. D-14
Expansion Board ................................................ D-15
Extender Card .................................................. D-16
Receiver Card .................................................. D-19
Printer .......................................................... D-22
Printer Adapter ................................................ D-25
Monochrome Display Adapter ............................. D-26
Color/Graphics Monitor Adapter ........................... D-36
Color Display ................................................... D-42
Monochrome Display ......................................... D-44
5–1/4 Inch Diskette Drive Adapter ......................... D-45
5–1/4 Inch Diskette Drive – Type 1 ......................... D-49
5–1/4 Inch Diskette Drive – Type 2 ......................... D-52
Fixed Disk Drive Adapter .................................... D-54
Fixed Disk Drive – Type 1 ..................................... D-60
Fixed Disk Drive – Type 2 ..................................... D-63
32K Memory Expansion Option ............................ D-66
64K Memory Expansion Option ............................ D-69
64/256K Memory Expansion Option ...................... D-72
Game Control Adapter ....................................... D-76
Prototype Card ................................................ D-77
Asynchronous Communications Adapter ................ D-78
Binary Synchronous Communications Adapter ........... D-79
SDLC Communications Adapter ............................ D-81
System Board (Sheet 2 of 10)
System Board (Sheet 3 of 10)
System Board (Sheet 8 of 10)
System Board (Sheet 10 of 10)
Keyboard - Type 2 (Sheet 1 of 1)
Extender Card (Sheet 1 of 3)
Extender Card (Sheet 2 of 3)
Receiver Card (Sheet 2 of 3)
Logic Diagrams D-25

Printer Adapter (Sheet 1 of 1)
Monochrome Display Adapter (Sheet 1 of 10)
Monochrome Display Adapter (Sheet 4 of 10)
Monochrome Display Adapter (Sheet 5 of 10)
Logic Diagrams D-31

Monochrome Display Adapter (Sheet 6 of 10)
Monochrome Display Adapter (Sheet 10 of 10)
Color/Graphics Monitor Adapter (Sheet 1 of 6)
D-38 Logic Diagrams

Color/Graphics Monitor Adapter (Sheet 3 of 6)
Logic Diagrams D-39

Color/Graphics Monitor Adapter (Sheet 4 of 6)
Color/Graphics Monitor Adapter (Sheet 6 of 6)
DANGER
HAZARDOUS VOLTAGES
UP TO 450 VOLTS EXIST
ON THE PRINTED
CIRCUIT BOARDS

Color Display (Sheet 1 of 1)

D-42 Logic Diagrams
DANGER
HAZARDOUS VOLTAGES UP TO 450 VOLTS EXIST ON THE PRINTED CIRCUIT BOARDS
DANGER
HAZARDOUS VOLTAGES UP TO 450 VOLTS EXIST ON THE PRINTED CIRCUIT BOARDS

NOTES:
1. RESISTOR VALUES ARE IN OHM (1k = 1000, 1k = 1000, 000)
2. ALL RESISTORS ARE 1/4W EXCEPT WHERE OTHERWISE INDICATED
3. ALL CAPACITORS ARE 10Uf EXCEPT WHERE OTHERWISE INDICATED
4. CAPACITOR VALUES ARE '-f' UNLESS OTHERWISE INDICATED
5. AC WIRING INFORMATION
PHASE = BLACK/BROWN WIRE
NEUTRAL = WHITE/BLUE WIRE
GROUND = GREEN AND YELLOW WIRE
IMPORTANT: THE PHASE WIRE MUST GO TO THE FUSED SIDE OF TRANSFORMER.
NOTES:

1. SIGNALS ON DRIVE PINS 10 THRU 16 ARE SWAPPED BY THE DRIVE CABLE BETWEEN DRIVES 1 & 2 (AND 3 & 4) AS FOLLOWS:
   10 TO 16
   11 TO 15
   12 TO 14
   13 TO 13
   14 TO 12
   15 TO 11
   16 TO 10

2. ALL DRIVES ARE JUMPERED FOR MULTIPLEX OPERATION. HEAD LOAD WITH DRIVE SELECT AND DRIVE SELECT VIA INPUT PIN 12. TERMINATING R-PACS ARE LEFT IN DRIVES 1 & 3 ONLY.

3. DACH/FD SHOULD BE ADJACENT TO MODULES MC4017, 7418, 7414, 16 KZ DCC 8/1, MC4004, MC4022, 74LS00, A 74LS101 8.2K FABS SHOULD BE ADJACENT TO EACH PIN.

4. ALL SIGNAL LINES HIGHER THAN OR EQUAL TO 15VDC SHOULD BE KEPT TO THE SHORTEST POSSIBLE LENGTH. THIS IS A PRIMARY DESIGN GOAL.

5. MAKE NO CONNECTION TO UNUSABLE PINS OR TO THE VCC/CHARGE PUMP & DATA SEPARATOR MODULES.

6. ALL VOLTAGE AND GROUND CONNECTORS TO THE VCC/CHARGE PUMP AND ASSOCIATED DISCRETE COMPONENTS SHOULD BE SEPARATE FROM OTHER CIRCUITS AND THEN JOINED TO THE OTHER CIRCUITS AT ONE POINT.

5-1/4 Inch Diskette Drive Adapter (Sheet 1 of 4)
5-1/4 Inch Diskette Drive Adapter (Sheet 4 of 4)
5-1/4 Inch Diskette Drive - Type 1 (Sheet 1 of 3)
NOTES: UNLESS OTHERWISE SPECIFIED
1. ALL RESISTORS ARE IN OHMS, 1/4 W.
2. ALL CAPS ARE IN .001.
3. ALL DIODES ARE 1N4148.
4. ALL TRANSISTORS PNP ARE 2N4124 & PNP ARE 2N4125.

5-1/4 Inch Diskette Drive - Type 1 (Sheet 2 of 3)
5-1/4 Inch Diskette Drive - Type 1 (Sheet 3 of 3)

NOTES: UNLESS OTHERWISE SPECIFIED
1. RESISTORS ARE IN OHMS, ±1%, 1/4W.
2. 1% RESISTORS ARE 1/8W.
3. CAPACITORS ARE IN μF, ±20%, 35V.
5-1/4 Inch Diskette Drive - Type 2 (Sheet 1 of 2)
5-1/4 Inch Diskette Drive - Type 2 (Sheet 2 of 2)
Fixed Disk Drive Adapter (Sheet 1 of 6)
Fixed Disk Drive Adapter (Sheet 3 of 6)
Fixed Disk Drive Adapter (Sheet 4 of 6)
Fixed Disk Drive Adapter (Sheet 6 of 6)
Fixed Disk Drive - Type 1 (Sheet 3 of 3)
Logic Diagrams

Fixed Disk Drive - Type 2 (Sheet 1 of 3)

Sheet No.

NOTE:
1. SHEET TO SHEET CONNECTION IS AS FOLLOWS:

2. UNLESS OTHERWISE SPECIFIED ALL RESISTORS ARE 1/4W 5% VALUE IN OHMS
   CAPACITORS .01µF 5% VALUE IN µF

3. 11 IS A PROGRAMMABLE JUMPER SOCKET
32K Memory Expansion Option (Sheet 3 of 3)
64K Memory Expansion Option (Sheet 1 of 3)
64/256K Memory Expansion Option (Sheet 3 of 4)
64/256K Memory Expansion Option (Sheet 4 of 4)
Binary Synchronous Communications Adapter (Sheet 1 of 2)
SDLC Communications Adapter (Sheet 2 of 2)
APPENDIX E: SPECIFICATIONS

System Unit

Size:
- Length--19.6 in (500 mm)
- Depth--16.1 in (410 mm)
- Height--5.5 in (142 mm)

Weight:
- 32 lb (14.5 kb)

Power Cables:
- Length--6 ft (1.83 m)
- Size--18 AWG

Environment:
- Air Temperature
  - System ON, 60° to 90° F (15.6° to 32.2° C)
  - System OFF, 50° to 110° F (10° to 43° C)
- Humidity
  - System ON, 8% to 80%
  - System OFF, 20% to 80%

Heat Output:
- 717 BTU/hr

Noise Level:
- 49.5 dB(a) (System unit with monochrome display and expansion unit attached.)

Electrical:
- Nominal--120 Vac
- Minimum--104 Vac
- Maximum--127 Vac

Keyboard

Size:
- Length--19.6 in (500 mm)
- Depth--7.87 in (200 mm)
- Height--2.2 in (57 mm)

Weight:
- 6.5 lb (2.9 kg)
Color Display

Size:
- Length--15.4 in (392 mm)
- Depth--15.6 in (407 mm)
- Height--11.7 in (297 mm)

Weight:
- 26 lb (11.8 kg)

Heat Output:
- 240 BTU/hr

Power Cables:
- Length--6 ft (1.83 m)
- Size--18 AWG

Signal Cable:
- Length--5 ft (1.5 m)
- Size--22 AWG

Expansion Unit

Size:
- Length--19.6 in (500 mm)
- Depth--16.1 in (410 mm)
- Height--5.5 in (142 mm)

Weight:
- 33 lb (14.9 kg)

Power Cables:
- Length--6 ft (1.83 m)
- Size--18 AWG

Signal Cable:
- Length--3.28 ft (1 m)
- Size--22 AWG

Environment:
- Air Temperature
  - System ON, 60° to 90° F (15.6° to 32.2° C)
  - System OFF, 50° to 110° F (10° to 43° C)
- Humidity
  - System ON, 8% to 80%
  - System OFF, 20% to 80%

Heat Output:
- 717 BTU/hr

Electrical:
- Nominal--120 Vac
- Minimum--104 Vac
- Maximum--127 Vac

E-2 Specifications
Monochrome Display

Size:
   Length--14.9 in (380 mm)
   Depth--13.7 in (350 mm)
   Height--11 in (280 mm)

Weight:
   17.3 lb (7.9 kg)

Heat Output:
   325 BTU/hr

Power Cable:
   Length--3 ft (.914 m)
   Size--18 AWG

Signal Cable:
   Length--4 ft (1.22 m)
   Size--22 AWG

80 CPS Printers

Size:
   Length--15.7 in (400 mm)
   Depth--14.5 in (370 mm)
   Height--4.3 in (110 mm)

Weight:
   12.9 lb (5.9 kg)

Power Cable:
   Length--6 ft (1.83 mm)
   Size--18 AWG

Signal Cable:
   Length--6 ft (1.83 m)
   Size--22 AWG

Heat Output:
   341 BTU/hr (maximum)

Electrical:
   Nominal--120 Vac
   Minimum--104 Vac
   Maximum--127 Vac
Front View (Component Side)

Loc. Hole

0.125 ± .002 (3.175 ± .05)

Mounting Holes

0.125 ± .005 (3.175 ± .127)

Loc. Hole

0.125 ± .002 (3.175 ± .05)

Notes:
1. All Card Dimensions are ± .010 (.254) Tolerance (With Exceptions Indicated on Drawing or in Notes).
2. Max. Card Length is 13.15 (334.01) Smaller Length is Permissible.
3. Loc. and Mounting Holes are Non-Plated Thru. (Loc. 3X, Mtg. 2X).
4. 31 Gold Tabs Each Side, 0.100 ± .0005 (2.54 ± .0127) Center to Center, 0.06 ± .0005 (1.524 ± .0127) Width.
5. Numbers in Parentheses are in Millimeters. All Others are in Inches.
Information processing equipment used for communications is called data terminal equipment (DTE). Equipment used to connect the DTE to the communications line is called data communications equipment (DCE).

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

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

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

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

The CCITT V.24 interface standard is equivalent to the RS-232C standard; therefore, the descriptions of the EIA standards also apply to the CCITT standards.
The following is an illustration of data terminal equipment connected to an external modem using connections defined by the RS-232C interface standard:

<table>
<thead>
<tr>
<th>EIA/CCITT Line Number</th>
<th>Telephone Co. Lead Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protective Ground</td>
<td>AA/101</td>
</tr>
<tr>
<td>Signal Ground</td>
<td>AB/102</td>
</tr>
<tr>
<td>Transmitted Data</td>
<td>BA/103</td>
</tr>
<tr>
<td>Received Data</td>
<td>BB/104</td>
</tr>
<tr>
<td>Request to Send</td>
<td>CA/105</td>
</tr>
<tr>
<td>Clear to Send</td>
<td>CB/106</td>
</tr>
<tr>
<td>Data Set Ready</td>
<td>CC/107</td>
</tr>
<tr>
<td>Data Terminal Ready</td>
<td>CD/108.2</td>
</tr>
<tr>
<td>Connect Data Set to Line</td>
<td>**/108.1</td>
</tr>
<tr>
<td>Received Data Set to Line</td>
<td>CF/109</td>
</tr>
<tr>
<td>Speed Select</td>
<td>CH/111</td>
</tr>
<tr>
<td>Transmit Signal Element Timing</td>
<td>DB/114</td>
</tr>
<tr>
<td>Receive Signal Element Timing</td>
<td>DD/115</td>
</tr>
<tr>
<td>Select Standby</td>
<td>**/116</td>
</tr>
<tr>
<td>Ring Indicator</td>
<td>**/116</td>
</tr>
<tr>
<td>Test</td>
<td>**/116</td>
</tr>
</tbody>
</table>

*Not used when business machine clocking is used.

**Not standardized by EIA (Electronics Industry Association).

***Not standardized by CCITT
Establishing a Communications Link

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

Switched Timing Sequence

<table>
<thead>
<tr>
<th>Event</th>
<th>Time Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Terminal Ready</td>
<td></td>
</tr>
<tr>
<td>Data Set Ready</td>
<td></td>
</tr>
<tr>
<td>Request to Send</td>
<td></td>
</tr>
<tr>
<td>Clear to Send</td>
<td></td>
</tr>
<tr>
<td>Transmitted Data</td>
<td></td>
</tr>
</tbody>
</table>

Nonswitched Timing Sequence

<table>
<thead>
<tr>
<th>Event</th>
<th>Time Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Terminal Ready</td>
<td></td>
</tr>
<tr>
<td>Data Terminal Ready</td>
<td></td>
</tr>
<tr>
<td>Request to Send</td>
<td></td>
</tr>
<tr>
<td>Clear to Send</td>
<td></td>
</tr>
<tr>
<td>Transmitted Data</td>
<td></td>
</tr>
</tbody>
</table>

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

1. The terminals at both locations activate the 'data terminal ready' lines. Normally the 'data set ready' lines from the modems are active whenever the modems are powered on.

2. Terminal A activates the 'request to send' line, which causes the modem at terminal A to generate a carrier signal. Modem B detects the carrier, and activates the 'received line signal detector' line (sometimes called data carrier detect). Modem B also activates the 'receiver signal element timing' line (sometimes called receive clock) to send receive clock signals to the terminal. Some modems activate the clock signals whenever the modem is powered on.

3. After a specified delay, modem A activates the 'clear to send' line, which indicates to terminal A that the modem is ready to transmit data.

4. Terminal A serializes the data to be transmitted (through the serdes) and transmits the data one bit at a time (synchronized by the transmit clock) onto the 'transmitted data' line to the modem.

5. The modem modulates the carrier signal with the data and transmits it to the modem B.

6. Modem B demodulates the data from the carrier signal and sends it to terminal B on the 'received data' line.

7. Terminal B deserializes the data (through the serdes) using the receive clock signals (on the 'receiver signal element timing' line) from the modem.

8. After terminal A completes its transmission, it deactivates the 'request to send' line, which causes the modem to turn off the carrier and deactivate the 'clear to send' line.
Establishing a Link on a Nonswitched Multipoint Line

1. The control station serializes the address for the tributary or secondary station (AA) and sends its address to the modem on the ‘transmitted data’ line.

2. Since the ‘request to send’ line and, therefore, the modem carrier, is active continuously, the modem immediately modulates the carrier with the address, and, thus, the address is transmitted to all modems on the line.

3. All tributary modems, including the modem for station A, demodulate the address and send it to their terminals on the ‘received data’ line.

4. Only station A responds to the address; the other stations ignore the address and continue monitoring their ‘received data’ line. To respond to the poll, station A activates its ‘request to send’ line which causes the modem to begin transmitting a carrier signal.

5. The control station’s modem receives the carrier and activates the ‘received line signal detector’ line and the ‘receiver signal element timing’ line (to send clock signals to the control station). Some modems activate the clock signals as soon as they are powered on.

6. After a short delay to allow the control station modem to receive the carrier, the tributary modem activates the ‘clear to send’ line.

7. When station A detects the active ‘clear to send’ line, it transmits its response. (For this example, assume that station A has no data to send; therefore, it transmits an EOT.)

8. After transmitting the EOT, station A deactivates the ‘request to send’ line. This causes the modem to deactivate the carrier and the ‘clear to send’ line.

9. When the modem at the control station (host) detects the absence of the carrier, it deactivates the ‘received line signal detector’ line.

10. Tributary station A is now in receive mode waiting for the next poll or select transmission from the control station.
These lines are active continuously.
Establishing a Link on a Switched Point-To-Point Line

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

The sequence of the remaining steps to establish the data link is the same as the sequence required on a nonswitched point-to-point line. When the terminals have completed their transmission, they both deactivate the ‘data terminal ready’ line to disconnect the modems from the line.
Notes:
### APPENDIX G: SWITCH SETTINGS

<table>
<thead>
<tr>
<th>Settings</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Board Switch Settings</td>
<td>G-3</td>
</tr>
<tr>
<td>System Board Switch</td>
<td>G-3</td>
</tr>
<tr>
<td>Math Coprocessor Switch Setting</td>
<td>G-3</td>
</tr>
<tr>
<td>System Board Memory Switch Settings</td>
<td>G-4</td>
</tr>
<tr>
<td>Monitor Type Switch Settings</td>
<td>G-4</td>
</tr>
<tr>
<td>5-1/4” Diskette Drive Switch Settings</td>
<td>G-5</td>
</tr>
<tr>
<td>Extender Card Switch Settings</td>
<td>G-6</td>
</tr>
<tr>
<td>Memory Option Switch Settings</td>
<td>G-7</td>
</tr>
<tr>
<td>288K Total Memory</td>
<td>G-7</td>
</tr>
<tr>
<td>320K Total Memory</td>
<td>G-8</td>
</tr>
<tr>
<td>352K Total Memory</td>
<td>G-9</td>
</tr>
<tr>
<td>384K Total Memory</td>
<td>G-10</td>
</tr>
<tr>
<td>416K Total Memory</td>
<td>G-11</td>
</tr>
<tr>
<td>448K Total Memory</td>
<td>G-12</td>
</tr>
<tr>
<td>480K Total Memory</td>
<td>G-13</td>
</tr>
<tr>
<td>512K Total Memory</td>
<td>G-14</td>
</tr>
<tr>
<td>544K Total Memory</td>
<td>G-15</td>
</tr>
<tr>
<td>576K Total Memory</td>
<td>G-16</td>
</tr>
<tr>
<td>608K Total Memory</td>
<td>G-17</td>
</tr>
<tr>
<td>640K Total Memory</td>
<td>G-18</td>
</tr>
</tbody>
</table>
Switches in your system are set to reflect the addition of memory and other installed options. Switches are located on the system board, extender card, and memory expansion options.

The switches are dual inline pin (dip) switches that can be easily set with a ballpoint pen. Refer to the diagrams below to familiarize yourself with the different types of switches that may be used in your system.

Refer to the charts on the following pages to determine the correct switch settings for your system.

Note: Set a rocker switch by pressing down the rocker to the desired position.

G-2 Switch Settings
System Board Switch Settings

The switches on the system board are set as shown in the following figure. These settings are necessary for the system to address the attached components, and to specify the amount of memory installed on the system board.

<table>
<thead>
<tr>
<th>Position</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Normal operation, Off (set to On to loop POST)</td>
</tr>
<tr>
<td>2</td>
<td>Used for Math Coprocessor</td>
</tr>
<tr>
<td>3-4</td>
<td>Amount of memory on the system board</td>
</tr>
<tr>
<td>5-6</td>
<td>Type of monitor you are using</td>
</tr>
<tr>
<td>7-8</td>
<td>Number of 5-1/4 inch diskette drives attached</td>
</tr>
</tbody>
</table>

Math Coprocessor Switch Settings

The following figure shows the settings for position 2.

Math Coprocessor installed

Math Coprocessor not installed
System Board Memory Switch Settings

The following figure shows the settings for positions 3 and 4 for the amount of memory on the system board.

128K

192K

256K

Monitor Type Switch Settings

No Monitor

IBM Color Display or other color monitor in the 40x25 Color mode

IBM Color Display or other color monitor in the 80x25 Color mode

Note: The 80x25 color setting, when used with your television and other monitors, can cause loss of character quality.

IBM Monochrome Display or more than one monitor
5 1/4" Diskette Drive Switch Settings

1 DRIVE

2 DRIVES

3 DRIVES

4 DRIVES
## Extender Card Switch Settings

<table>
<thead>
<tr>
<th>System Memory</th>
<th>Memory Segment</th>
</tr>
</thead>
<tbody>
<tr>
<td>16K to 64K</td>
<td>1</td>
</tr>
<tr>
<td>96K to 128K</td>
<td>2</td>
</tr>
<tr>
<td>160K to 192K</td>
<td>3</td>
</tr>
<tr>
<td>224K to 256K</td>
<td>4</td>
</tr>
<tr>
<td>288K to 320K</td>
<td>5</td>
</tr>
<tr>
<td>352K to 384K</td>
<td>6</td>
</tr>
<tr>
<td>416K to 448K</td>
<td>7</td>
</tr>
<tr>
<td>480K to 512K</td>
<td>8</td>
</tr>
<tr>
<td>544K to 576K</td>
<td>9</td>
</tr>
<tr>
<td>608K to 640K</td>
<td>A</td>
</tr>
</tbody>
</table>

G-6  Switch Settings
# Memory Option Switch Settings

288K Total Memory  
32K + (256K on System Board)

<table>
<thead>
<tr>
<th>1 - 32K option</th>
<th>64/256K Option Card Switches</th>
<th>64K Option Card Switches</th>
<th>32K Option Card Switches</th>
</tr>
</thead>
</table>

![Card Switches Diagram]
320K Total Memory
64K + (256K on System Board)

<table>
<thead>
<tr>
<th>64/256K Option Card Switches</th>
<th>64K Option Card Switches</th>
<th>32K Option Card Switches</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - 64/256K option with 64K installed</td>
<td><img src="#" alt="Switch Settings" /></td>
<td><img src="#" alt="Switch Settings" /></td>
</tr>
<tr>
<td>1 - 64K option</td>
<td><img src="#" alt="Switch Settings" /></td>
<td><img src="#" alt="Switch Settings" /></td>
</tr>
<tr>
<td>2 - 32K options</td>
<td><img src="#" alt="Switch Settings" /></td>
<td><img src="#" alt="Switch Settings" /></td>
</tr>
</tbody>
</table>
### 352K Total Memory
96K + (256K on System Board)

<table>
<thead>
<tr>
<th>64/256K Option Card Switches</th>
<th>64K Option Card Switches</th>
<th>32K Option Card Switches</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - 64/256K option with 64K installed</td>
<td><img src="image1" alt="Card Switches" /></td>
<td><img src="image2" alt="Card Switches" /></td>
</tr>
<tr>
<td>1 - 32K option</td>
<td><img src="image3" alt="Card Switches" /></td>
<td><img src="image4" alt="Card Switches" /></td>
</tr>
<tr>
<td>1 - 64K option</td>
<td><img src="image5" alt="Card Switches" /></td>
<td><img src="image6" alt="Card Switches" /></td>
</tr>
<tr>
<td>1 - 32K option</td>
<td><img src="image7" alt="Card Switches" /></td>
<td><img src="image8" alt="Card Switches" /></td>
</tr>
<tr>
<td>3 - 32K options</td>
<td><img src="image9" alt="Card Switches" /></td>
<td><img src="image10" alt="Card Switches" /></td>
</tr>
</tbody>
</table>

Switch Settings

Appendix G-9
### 384K Total Memory
128K + (256K on System Board)

<table>
<thead>
<tr>
<th>64/256K Option Card Switches</th>
<th>64K Option Card Switches</th>
<th>32K Option Card Switches</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1 - 64/256K option with 64K option installed</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 - 64K option</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2 - 64K options</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>1 - 64/256K option with 64K installed</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 - 32K options</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>1 - 64K option</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 - 32K options</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>1 - 64/256K option with 128K installed</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Switch Settings</td>
<td>64/256K Option Card Switches</td>
<td>64K Option Card Switches</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>1 - 64/256K option with 64K installed</td>
<td><img src="image1" alt="Switch Settings" /></td>
<td><img src="image2" alt="Switch Settings" /></td>
</tr>
<tr>
<td>1 - 64K option</td>
<td><img src="image4" alt="Switch Settings" /></td>
<td><img src="image5" alt="Switch Settings" /></td>
</tr>
<tr>
<td>1 - 32K option</td>
<td><img src="image7" alt="Switch Settings" /></td>
<td><img src="image8" alt="Switch Settings" /></td>
</tr>
<tr>
<td>2 - 64K options</td>
<td><img src="image10" alt="Switch Settings" /></td>
<td><img src="image11" alt="Switch Settings" /></td>
</tr>
<tr>
<td>1 - 32K option</td>
<td><img src="image13" alt="Switch Settings" /></td>
<td><img src="image14" alt="Switch Settings" /></td>
</tr>
<tr>
<td>1 - 64/256K option with 128K installed</td>
<td><img src="image16" alt="Switch Settings" /></td>
<td><img src="image17" alt="Switch Settings" /></td>
</tr>
<tr>
<td>1 - 32K option</td>
<td><img src="image19" alt="Switch Settings" /></td>
<td><img src="image20" alt="Switch Settings" /></td>
</tr>
<tr>
<td>64/256K Option Card Switches</td>
<td>64K Option Card Switches</td>
<td>32K Option Card Switches</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>--------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>1 - 64/256K option with 192K installed</td>
<td><img src="image1" alt="Switches" /></td>
<td><img src="image2" alt="Switches" /></td>
</tr>
<tr>
<td>1 - 64/256K option with 128K installed 1 - 64K option</td>
<td><img src="image3" alt="Switches" /></td>
<td><img src="image4" alt="Switches" /></td>
</tr>
<tr>
<td>1 - 64/256K option with 64K installed 2 - 64K options</td>
<td><img src="image5" alt="Switches" /></td>
<td><img src="image6" alt="Switches" /></td>
</tr>
<tr>
<td>3 - 64K options</td>
<td><img src="image7" alt="Switches" /></td>
<td><img src="image8" alt="Switches" /></td>
</tr>
<tr>
<td>1 - 64/256K option with 128 installed 2 - 32K options</td>
<td><img src="image9" alt="Switches" /></td>
<td><img src="image10" alt="Switches" /></td>
</tr>
</tbody>
</table>
### Switch Settings

**480K Total Memory**

\[224K + (256K \text{ on System Board})\]

<table>
<thead>
<tr>
<th></th>
<th>64/256K Option Card Switches</th>
<th>64K Option Card Switches</th>
<th>32K Option Card Switches</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - 64/256K option with 192K installed</td>
<td><img src="image" alt="Switch Settings" /></td>
<td><img src="image" alt="Switch Settings" /></td>
<td><img src="image" alt="Switch Settings" /></td>
</tr>
<tr>
<td>1 - 32K option</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>64/256K Option Card Switches</th>
<th>64K Option Card Switches</th>
<th>32K Option Card Switches</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - 64/256K option with 128K installed</td>
<td><img src="image" alt="Switch Settings" /></td>
<td><img src="image" alt="Switch Settings" /></td>
<td><img src="image" alt="Switch Settings" /></td>
</tr>
<tr>
<td>1 - 64K option</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 - 32K option</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>512K Total Memory</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>256K + (256K on System Board)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>64/256K Option Card Switches</th>
<th>64K Option Card Switches</th>
<th>32K Option Card Switches</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - 64/256K option with 128K installed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 - 64K options</td>
<td></td>
<td></td>
</tr>
<tr>
<td><img src="image1" alt="Card Switches" /></td>
<td><img src="image2" alt="Card Switches" /></td>
<td><img src="image3" alt="Card Switches" /></td>
</tr>
<tr>
<td>1 - 64/256K option with 192K installed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 - 64K option</td>
<td></td>
<td></td>
</tr>
<tr>
<td><img src="image4" alt="Card Switches" /></td>
<td><img src="image5" alt="Card Switches" /></td>
<td><img src="image6" alt="Card Switches" /></td>
</tr>
<tr>
<td>1 - 64/256K option with 192K installed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 - 32K options</td>
<td></td>
<td></td>
</tr>
<tr>
<td><img src="image7" alt="Card Switches" /></td>
<td><img src="image8" alt="Card Switches" /></td>
<td><img src="image9" alt="Card Switches" /></td>
</tr>
<tr>
<td>1 - 64/256K option with 256K installed</td>
<td></td>
<td></td>
</tr>
<tr>
<td><img src="image10" alt="Card Switches" /></td>
<td><img src="image11" alt="Card Switches" /></td>
<td><img src="image12" alt="Card Switches" /></td>
</tr>
</tbody>
</table>
### 544K Total Memory
288K + (256K on System Board)

<table>
<thead>
<tr>
<th>1 - 64/256K option with 192K installed</th>
<th>1 - 64K option</th>
<th>1 - 32K option</th>
</tr>
</thead>
<tbody>
<tr>
<td>64/256K Option Card Switches</td>
<td>64K Option Card Switches</td>
<td>32K Option Card Switches</td>
</tr>
<tr>
<td><img src="image" alt="Switch Settings" /></td>
<td><img src="image" alt="Switch Settings" /></td>
<td><img src="image" alt="Switch Settings" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1 - 64/256K option with 256K installed</th>
<th>1 - 32K option</th>
</tr>
</thead>
<tbody>
<tr>
<td>64/256K Option Card Switches</td>
<td>32K Option Card Switches</td>
</tr>
<tr>
<td><img src="image" alt="Switch Settings" /></td>
<td><img src="image" alt="Switch Settings" /></td>
</tr>
</tbody>
</table>

---

Switch Settings
G-15

Appendix G
<table>
<thead>
<tr>
<th>Switch Settings</th>
<th>64/256K Option Card Switches</th>
<th>64K Option Card Switches</th>
<th>32K Option Card Switches</th>
</tr>
</thead>
</table>
| 1 - 64/256K option with 192K installed  
2 - 64K options                  | ![Switch Configuration]        | ![Switch Configuration]  | ![Switch Configuration]  |
| 1 - 64/256K option with 256K installed  
1 - 64/256K option with 64K installed | ![Switch Configuration]        | ![Switch Configuration]  | ![Switch Configuration]  |
| 1 - 64/256K option with 256K installed  
1 - 64K option                   | ![Switch Configuration]        | ![Switch Configuration]  | ![Switch Configuration]  |
| 1 - 64/256K option with 256K installed  
2 - 32K options                  | ![Switch Configuration]        | ![Switch Configuration]  | ![Switch Configuration]  |

576K Total Memory  
320K + (256K on System Board)
<table>
<thead>
<tr>
<th>Switch Settings</th>
<th>64/256K Option Card Switches</th>
<th>64K Option Card Switches</th>
<th>32K Option Card Switches</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - 64/256K option with 256K installed</td>
<td><img src="image1.png" alt="Switch Settings" /></td>
<td><img src="image2.png" alt="Switch Settings" /></td>
<td><img src="image3.png" alt="Switch Settings" /></td>
</tr>
<tr>
<td>1 - 64/256K option with 64K installed</td>
<td><img src="image4.png" alt="Switch Settings" /></td>
<td><img src="image5.png" alt="Switch Settings" /></td>
<td><img src="image6.png" alt="Switch Settings" /></td>
</tr>
<tr>
<td>1 - 32K option</td>
<td><img src="image7.png" alt="Switch Settings" /></td>
<td><img src="image8.png" alt="Switch Settings" /></td>
<td><img src="image9.png" alt="Switch Settings" /></td>
</tr>
</tbody>
</table>

608K Total Memory
352K + (256K on System Board)
### 640K Total Memory

384K + (256K on System Board)

<table>
<thead>
<tr>
<th>64/256K Option Card Switches</th>
<th>64K Option Card Switches</th>
<th>32K Option Card Switches</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1 - 64/256K option with 256K installed</strong></td>
<td><img src="image" alt="Switch Setting" /></td>
<td><img src="image" alt="Switch Setting" /></td>
</tr>
<tr>
<td><strong>1 - 64/256K option with 64K installed</strong></td>
<td><img src="image" alt="Switch Setting" /></td>
<td><img src="image" alt="Switch Setting" /></td>
</tr>
<tr>
<td><strong>1 - 64K option</strong></td>
<td><img src="image" alt="Switch Setting" /></td>
<td><img src="image" alt="Switch Setting" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>64/256K Option Card Switches</th>
<th>64K Option Card Switches</th>
<th>32K Option Card Switches</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1 - 64/256K option with 256K installed</strong></td>
<td><img src="image" alt="Switch Setting" /></td>
<td><img src="image" alt="Switch Setting" /></td>
</tr>
<tr>
<td><strong>2 - 64K options</strong></td>
<td><img src="image" alt="Switch Setting" /></td>
<td><img src="image" alt="Switch Setting" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>64/256K Option Card Switches</th>
<th>64K Option Card Switches</th>
<th>32K Option Card Switches</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1 - 64/256K option with 256K installed</strong></td>
<td><img src="image" alt="Switch Setting" /></td>
<td><img src="image" alt="Switch Setting" /></td>
</tr>
<tr>
<td><strong>1 - 64/256K option with 128K installed</strong></td>
<td><img src="image" alt="Switch Setting" /></td>
<td><img src="image" alt="Switch Setting" /></td>
</tr>
</tbody>
</table>
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 alphanumeric.

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.

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.

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

H-2 Glossary
BIOS: Basic input/output system.

bit: In binary notation, either of the characters 0 or 1.

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

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

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

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

bps: Bits per second.

BSC: Binary synchronous communications.

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

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

byte: (1) A 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 Telephony.

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: Cathode ray tube.

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.

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: Electronics Industries Association/Consultative Committee on International Telegraphy and Telephony.

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 changed 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 enabling 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: 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 in decimal notation.

K: When referring to storage capacity, 2 to the tenth power; 1,024 in decimal notation.

KB: Kilobyte; 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.

H-8 Glossary
M: Mega; 1,000,000 in decimal notation. When referring to storage capacity, 2 to the twentieth power; 1,048,576 in decimal notation.

mA: Milliampere.

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

main storage: A storage device in which the access time is effectively independent of the location of the data.

MB: Megabyte, 1,048,576 bytes.

mega (M): 10 to the sixth power, 1,000,000 in decimal notation. When referring to storage capacity, 2 to the twentieth power, 1,048,576 in decimal notation.

megabyte (MB): 1,048,576 bytes.

megahertz (MHz): A unit of measure of frequency. 1 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 (μs): 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 a “mpy” for “multiply.”

mode: (1) A method of operation; for example, the binary mode, the interpretive mode, the alphanumerical 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 interleaving the events of two or more activities, or 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-thousandth-millionth of a second.

nonconjunction: The dyadic boolean operation the result of which has the boolean value 0 if, and only if, each operand has the boolean value 1.

non-return-to-zero inverted (NRZI): A transmission encoding method in which the data terminal equipment changes the signal to the opposite state to send a binary 0 and leaves it in the same state to send a binary 1.
NOR: A logic operator having the property that if P is a statement, Q is a statement, R is a statement, ..., then the NOR of P, Q, R, ..., is true if all statements are false, false if at least one statement is true.

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

NRZI: Non-return-to-zero inverted.

ns: Nanosecond; one-thousandth-millionth 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: A logic operator having the property that if P is a statement, Q is a statement, R is a statement, ..., then the OR of P, Q, R, ..., is true if at least one statement is true, false if all statements are false.

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

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

overcurrent: A current of higher than specified strength.

overvoltage: A voltage of higher than specified value.

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

personal computer: A small home or business computer that has a processor and keyboard 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.

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

PROM: Programmable read-only memory.

propagation delay: The time necessary for a signal to travel from one point on a circuit to another.

radix: (1) In a radix numeration system, the positive integer by which the weight of the digit place is multiplied to obtain the weight of the digit place with the next higher weight; for example, in the decimal numeration system, the radix of each digit place is 10. (2) Another term for base.
radix numeration system: A positional representation system in which the ratio of the weight of any one digit place to the weight of the digit place with the next lower weight is a positive integer. The permissible values of the character in any digit place range from zero to one less than the radix of the digit place.

RAS: Row address strobe.

RGBI: Red-green-blue-intensity.

read-only memory (ROM): A storage device whose contents cannot be modified, except by a particular user, or when operating under particular conditions; for example, a storage device in which writing is prevented by a lockout.

read/write memory: A storage device whose contents can be modified.

red-green-blue-intensity (RGBI): The description of a direct-drive color monitor which accepts red, green, blue, and intensity signal inputs.

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

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

ROM: Read-only memory.

ROM/BIOS: The ROM resident basic input/output system, 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 standard 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.

SDLC: Synchronous Data Link Control.

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

serdes: Serializer/deserializer.

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

sink: A device or circuit into which current drains.

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

source: The origin of a signal or electrical energy.

source circuit: (1) Generator circuit. (2) Control with sink.

SS: Start-stop transmission.

start bit: Synonym for start signal.

start-of-text character (STX): A transmission control character that precedes a text and may be used to terminate the message heading.
start signal: (1) A signal to a receiving mechanism to get ready to receive data or perform a function. (2) In a start-stop system, a signal preceding a character or block that prepares the receiving device for the reception of the code elements. Synonymous with start bit.

start-stop (SS) transmission: Asynchronous 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 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 Data Link Control (SLDC): A protocol for the management of data transfer over a data communications link.

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: (1) The path or one of the set of paths, parallel to the reference edge on a data medium, associated with a single reading or writing component as the data medium moves past the component. (2) The portion of a moving data medium such as a drum, tape, or disk, that is accessible to a given reading head position.

transistor-transistor logic (TTL): A circuit in which the multiple-diode cluster of the diode-transistor logic circuit has been replaced by a multiple-emitter transistor.

TTL: Transistor-transistor logic.

TX Data: Transmit data. Associated with modem control. External connections of the RS-232C asynchronous communications adapter interface.

video: Computer data or graphics 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.
BIBLIOGRAPHY

This manual introduces the 8086 family of microcomputing components and serves as a reference in system design and implementation.

Intel Corporation. *8086/8087/8088 Macro Assembly Reference Manual for 8088/8085 Based Development System*
This manual describes the 8086/8087/8088 Macro Assembly Language, and is intended for use by persons who are familiar with assembly language.

Intel Corporation. *Component Data Catalog*
This book describes Intel components and their technical specifications.

Motorola, Inc. *The Complete Microcomputer Data Library.*
This book describes Motorola components and their technical specifications.

INDEX

A

A/N mode (alphanumeric mode) 1-123
A0-A19 (Address Bits 0 to 19), I/O channel 1-15
adapter card with ROM 2-10
adapter,
  asynchronous communication 1-215
  binary synchronous communication 1-245
  color/graphics monitor 1-123
  diskette drive 1-151
  fixed disk drive 1-179
  game control 1-203
  monochrome display and printer 1-115
  printer 1-113
  synchronous data link control 1-265
Address Bits 0 to 19 (A0-A19), I/O channel 1-15
address bits (asynchronous communication) 1-217
Address Enable (AEN), I/O channel 1-18
Address Latch Enable (ALE), I/O channel 1-15
address map, I/O 1-8
AEN (Address Enable), I/O channel 1-18
ALE (Address Latch Enable), I/O channel 1-15
all points addressable mode 1-129, 1-123
alphanumeric mode, 1-128
  high resolution 1-135
  low resolution 1-132
alt (keyboard extended code) 2-15
APA mode (all points addressable mode) 1-124
asynchronous communications adapter, 1-215
  adapter address jumper module 1-242
  address bits 1-217
  block diagram 1-216
  connector specifications 1-243
  current loop interface 1-219
  divisor latch least significant bit 1-229
  divisor latch most significant bit 1-230
  I/O decode 1-217
  INS8250 functional pin description 1-221
  INS8250 input signals 1-221
  INS8250 input/output signals 1-225
INS8250 output signals 1-224
interface descriptions 1-218
interface format jumper module 1-242
interrupt control functions 1-234
interrupt enable register 1-235
interrupt identification register 1-233
interrupts 1-218
line control register 1-227
line status register 1-231
modem control register 1-236
modem status register 1-238
modes of operation 1-216
programmable baud rate generator 1-229
programming considerations 1-226
receiver buffer register 1-240
reset functions 1-226
transmitter holding register 1-241
voltage interchange information 1-220
attributes, character
(see character attributes)

B

BASIC reserved interrupts 2-7
BASIC,
  DEF SEG 2-8
  reserved interrupt 2-7
  screen editor keyboard functions 2-20
  workspace variables 2-8
baud rate generator 1-231
bell (printer) 1-92
bibliography 1-1
binary synchronous communications adapter, 1-245
  8252A programming procedures 1-257
  8252A universal synchronous/asynchronous
    receiver/transmitter 1-241
  8253-5 programmable interval timer 1-251
  8255A-5 programmable peripheral interface 1-251
block diagram 1-246
command instruction format 1-258
connector information 1-257
data bus buffer 1-247
interface signal information 1-260
interrupt information 1-262
mode instruction definition 1-257
read/write control logic 1-247
receive 1-252
receiver buffer 1-249
receiver control 1-249
status read definition 1-259
transmit 1-251
transmitter buffer 1-248
transmitter control 1-249
typical programming sequence 1-253

BIOS,
fixed disk ROM A-84
memory map 2-9
parameter passing 2-3
software interrupt listing 2-4
system ROM A-2
use of 2-2
bisync communications
(see binary synchronous communications adapter)

block diagram
8252A universal synchronous/asyncronous receiver/transmitter 1-246
8273 SDLC protocol controller 1-267
asynchronous communications adapter 1-215
color/graphics monitor adapter 1-125
coprocessor 1-25
diskette drive adapter 1-151
diskette drive adapter 1-151
expansion board 1-71
extender card 1-74
fixed disk drive adapter 1-179
game control adapter 1-203
keyboard interface 1-67
monochrome display adapter 1-114
printer adapter 1-108
prototype card 1-118
receiver card 1-77
speaker drive system 1-20
synchronous data link control adapter 1-265
system 1-2
break (keyboard extended code) 2-14
BSC adapter
(see binary synchronous communications)
cable
  communications adapter  1-285
  expansion unit  1-171
  printer  1-81
cancel (printer)  1-93
cancel ignore paper end (printer)  1-95
cancel skip perforation (printer)  1-99
caps lock (keyboard extended code)  2-16
card dimensions and specifications  E-4
card selected  1-19
CARD SLCTD (card selected), I/O channel  1-19
card,
  dimensions and specifications  E-4
  extender  1-74
  prototype  1-209
  receiver  1-77
carriage return (printer)  1-92
CCITT,  F-1
  standards  F-1
character attributes
  color/graphics monitor adapter  1-130
  monochrome display adapter  1-130
character codes
  keyboard  2-11
character set,
  graphics printer (set 1)  1-103
  graphics printer (set 2)  1-105
  matrix printer  1-101
  quick reference  C-12
clear printer buffer (printer)  1-100
CLK (system clock), I/O channel  1-16
color display  1-149
  operating characteristics  1-150
  specifications  E-2
color select register  1-141
color/graphics monitor adapter  1-123
  6845 register description  1-136
  alphanumeric mode  1-127
  alphanumeric mode (high-resolution)  1-128
  alphanumeric mode (low-resolution)  1-128
  block diagram  1-126
  character attributes  1-130
  color-select register  1-141
  composite connector specifications  1-146
connectors,
   power supply (system unit) 1-21
consideration, programming
   (see programming considerations)
control byte, fixed disk drive adapter 1-186
control codes, printer 1-91
control/read/write logic 1-268
coprocessor,
   (see math coprocessor)
ctrl (keyboard extended code) 2-13
current loop interface 1-219

D

D0-D7 (data bits 0 to 7), I/O channel 1-16
DACK0-DACK3 (DMA Acknowledge 0 to 3), I/O channel 1-18
Data Bits 0 to 7 (D0-D7), I/O channel 1-16
data flow,
   system board 1-6
data register 1-185
data transfer mode register 1-282
DEF SEG (default segment workspace) 2-8
default workspace segment (DEF SEG) 2-8
diagram, block (see block diagram)
digital output register 1-153
diskette drive adapter 1-151
   adapter input 1-171
   adapter output 1-170
   block diagram 1-152
   command status register 0 1-164
   command status register 1 1-165
   command status register 2 1-166
   command status register 3 1-167
   command summary 1-158
connector specifications (external) 1-174
connector specifications (internal) 1-173
digital-output register 1-153
DPC registers 1-167
drive A and B interface 1-170
drive constants 1-168
FDC constants 1-168
floppy disk controller 1-154
functional description 1-153
programming considerations 1-156
programming summary 1-167
symbol descriptions 1-156
system I/O channel interface 1-168

J-6 Index
diskette drive, 1-175
  electrical specifications 1-176
  mechanical specifications 1-176
  switch settings G-1
diskettes 1-177
display adapter type switch settings G-1
display,
  color 1-149
  monochrome 1-121
divisor latch,
  least significant bit 1-229
  most significant bit 1-230
DMA Acknowledge 0 to 3 (DACK0-DACK3), 
  I/O channel 1-18
DMA Request 1 to 3 (DRQ1-DRQ3), I/O channel 1-18
DOS reserved interrupts 2-7
DOS,
  keyboard functions 2-20
  reserved interrupts 2-7
double strike (printer) 1-96
double strike off (printer) 1-96
double width (printer) 1-93
double width off (printer) 1-93
DPC registers 1-167
DRQ1-DRQ3 (DMA Request 1 to 3), I/O channel 1-18

E

EIA, F-1
  standards F-1
  emphasized (printer) 1-96
  emphasized off (printer) 1-96
  escape (printer) 1-93
  establishing a communications link F-3
expansion board, 1-71
  block diagram 1-72
expansion channel 1-73
expansion unit, 1-71
  cable 1-71
  expansion board 1-71
  expansion channel 1-73
  extender card 1-74
  interface information 1-79
  power supply 1-71
  receiver card 1-77
  specifications E-2
extender card, 1-74
block diagram 1-76
programming considerations 1-75
switch settings G-1

F

FABS (coprocessor) 1-36
FADD (coprocessor) 1-36
FBLD (coprocessor) 1-37
FBSTP (coprocessor) 1-37
FABS (coprocessor) 1-38
FCLEX/FNCLEX (coprocessor) 1-38
FCOM (coprocessor) 1-38
FCOMP (coprocessor) 1-39
FCOMP (coprocessor) 1-39
FDECSTP (coprocessor) 1-40
FDISI/FNDISI (coprocessor) 1-40
FDIV (coprocessor) 1-41
FDIVR (coprocessor) 1-42
FENI/FNENI (coprocessor) 1-43
FFREE (coprocessor) 1-43
FICOM (coprocessor) 1-43
FICOMP (coprocessor) 1-44
FILD (coprocessor) 1-44
FINCSTP (coprocessor) 1-44
FINIT/FNINIT (coprocessor) 1-45
FIST (coprocessor) 1-46
FISTP (coprocessor) 1-46
fixed disk controller 1-179
fixed disk drive 1-195
fixed disk drive adapter 1-179
block diagram 1-180
command summary 1-187
control byte 1-186
data register 1-185
fixed disk controller 1-179
interface specifications 1-193
programming considerations 1-181
programming summary 1-191
ROM BIOS listing A-84
sense bytes 1-181
status register 1-181
system I/O channel interface 1-192
fixed disk drive, 1-195
  electrical specifications 1-196
  mechanical specifications 1-196
fixed disk ROM BIOS A-84
FLD (coprocessor) 1-47
FLDCW (coprocessor) 1-47
FLDENV (coprocessor) 1-48
FLDLG2 (coprocessor) 1-48
FLDLN2 (coprocessor) 1-48
FLDL2E (coprocessor) 1-49
FLDL2T (coprocessor) 1-49
FLDPI (coprocessor) 1-49
FLDZ (coprocessor) 1-50
FLD1 (coprocessor) 1-50
floppy disk controller 1-154
form feed (printer) 1-92
FMUL 1-51
FNOP 1-52
FPTAN 1-52
FPREM 1-52
FPTAN 1-53
FRNDINT 1-53
FRSTOR 1-53
FSAVE/FNSAVE (coprocessor) 1-54
FSAVE (coprocessor) 1-54
FSQRT (coprocessor) 1-55
FST (coprocessor) 1-55
FSTCW/FNSTCW (coprocessor) 1-56
FSTENV/FNSTENV (coprocessor) 1-56
FSTP (coprocessor) 1-57
FSTSW/FNSTSW (coprocessor) 1-57
FSUB (coprocessor) 1-58
FSUBR (coprocessor) 1-59
FTST (coprocessor) 1-60
FWAIT (coprocessor) 1-60
FXAM (coprocessor) 1-61
FXCH (coprocessor) 1-62
FXTRACT (coprocessor) 1-62
FYL2X (coprocessor) 1-63
FYL2XP1 (coprocessor) 1-63
F2XM1 (coprocessor) 1-64
G

game control adapter, 1-203
  block diagram 1-203
  connector specifications 1-208
  functional description 1-204
  I/O channel description 1-205
  interface description 1-206
  joy stick schematic diagram 1-207

glossary, H-1

graphics mode, 1-123
  high resolution 1-124
  low resolution 1-123
  medium resolution 1-124

H

hardware interrupt listing 1-9
home head (printer) 1-95
horizontal tab (printer) 1-92

I

I/O address map 1-8
I/O bit map, 8255A 1-10
I/O CH CK (I/O Channel Check), I/O channel 1-17
I/O CH RDY (I/O Channel Ready), I/O channel 1-17
I/O Channel Check (I/O CH CK), I/O channel 1-17
I/O channel interface,
  diskette drive adapter 1-168
  fixed disk drive adapter 1-192
  prototype card 1-211
I/O Channel Ready (I/O CH RDY), I/O channel 1-17
I/O channel, 1-14
  -I/O Channel Check (I/O CH CK) 1-17
  -I/O Read Command (IOR) 1-17
  -I/O Write Command (IOW) 1-17
Address Bits 0 to 19 (A0-A19) 1-16
Address Enable (AEN) 1-18
Address Latch Enable (ALE) 1-16
Data Bits 0 to 7 (D0-D7) 1-16
description 1-16
diagram 1-15
DMA Request 1 to 3 (DRQ1-DRQ3) 1-18
I/O Channel Ready (I/O CH RDY) 1-17

J-10 Index
Interrupt Request 2 to 7 (IRQ2-IRQ7) 1-17
Memory Read Command (MEMR) 1-18
Memory Write Command (MEMW) 1-18
Oscillator (OSC) 1-16
Reset Drive (RESET DRV) 1-16
System Clock (CLK) 1-16
Terminal Count (T/C) 1-18
I/O Read Command (IOR), I/O channel 1-17
I/O Write Command (IOW), I/O channel 1-17
IBM 10MB Fixed Disk Drive 1-195
IBM 5-1/4" Diskette Drive 1-175
IBM 5-1/4" Diskette Drive Adapter 1-151
IBM 80 CPS Graphics Printer 1-81
IBM 80 CPS Matrix Printer 1-81
IBM 80 CPS Printers 1-81
IBM Asynchronous Communications Adapter 1-215
IBM Binary Synchronous Communications Adapter 1-245
IBM Color Display 1-149
IBM Color/Graphics Monitor Adapter 1-123
IBM Communications Adapter Cable 1-295
IBM Fixed Disk Drive Adapter 1-179
IBM Game Control Adapter 1-203
IBM Memory Expansion Options 1-197
IBM Monochrome Display and Printer Adapter 1-113
IBM Monochrome Display 1-121
IBM Personal Computer Math Coprocessor 1-25
IBM Printer Adapter 1-107
IBM Prototype Card 1-209
IBM Synchronous Data Link Controller Adapter 1-265
ignore paper end (printer) 1-94
INS8250,
(see National Semiconductor INS8250)
Intel 8088 microprocessor,
    arithmetic B-7
    conditional transfer operations B-14
    control transfer B-11
    data transfer B-5
    hardware interrupt listing 1-8
    instruction set index B-18
    instruction set matrix B-16
    logic B-9
    memory segmentation model B-4
    operand summary B-3
    processor control B-14
    register model B-2

Index J-11
second instruction byte summary  B-3
segment override prefix  B-4
software interrupt listing  2-4
string manipulation  B-11
use of segment override  B-4
Intel 8253-5 Programmable Interval Timer
   (see synchronous data link control communications adapter)
Intel 8255A Programmable Peripheral Interface
   I/O bit map  1-10
Intel 8255A-5 Programmable Peripheral Interface
   (see synchronous data link control communications adapter)
Intel 8273 SDLC Protocol Controller
   (see synchronous data link control communications adapter)
      block diagram  1-265
interrupt enable register  1-235
interrupt identification register  1-233
interrupt listing,
   8088 hardware  1-9
   8088 software  2-4
Interrupt Request 2 to 7 (IRQ2-IRQ7), I/O channel  1-17
interrupts,
   8088 hardware  1-9
   8088 software  2-4
   asynchronous communications adapter  1-215
   BASIC reserved  2-7
   DOS reserved  2-7
   special  2-5
IOR (I/O Read Command), I/O channel  1-17
IOW (I/O Write Command), I/O channel  1-17
IRQ2-IRQ7 (Interrupt Request 2 to 7), I/O channel  1-17

J

joy stick,
   positions  1-204
   schematic diagram  1-207
jumper module, asynchronous communications adapter  1-242

K

keyboard extended codes,
   alt  2-15
   break  2-16
   caps lock  2-16
   ctrl  2-15

J-12  Index
pause 2-17
print screen 2-17
scroll lock 2-16
shift 2-15
shift key priorities 2-16
shift states 2-15
system reset 2-16

keyboard 1-65
BASIC screen editor special functions 2-20
character codes 2-11
commonly used functions 2-18
diagram 1-68
DOS special functions 2-20
encoding 2-11
extended functions 2-14
interface block diagram 1-70
interface connector specifications 1-70
scan codes 1-69
specifications E-1

L
light pen connector specifications 1-147
line control register 1-227
line feed (printer) 1-92
line status register 1-223
logic diagrams D-1

M
math coprocessor 1-25
block diagram 1-29
control unit 1-29
control word 1-32
data types 1-26
exception pointers 1-33
FABS 1-36
FADD 1-36
FBLD 1-37
FBSTP 1-37
FCHS 1-38
FCLEX/FNCLEX 1-38
FCOM 1-38
FCOMP 1-39
FCOMPP 1-39
FDECSTP 1-40
FDISI/FNDISI 1-40
FDIV 1-41
FDIVR 1-42
FENI/FNENI 1-43
FFREE 1-43
FICOM 1-43
FICOMP 1-44
FILD 1-44
FINCSTP 1-44
FINIT/FNINIT 1-45
FIST 1-46
FISTP 1-46
FLD 1-47
FLDCW 1-47
FLDENV 1-48
FLDLG2 1-48
FLDLN2 1-48
FLDL2E 1-49
FLDL2T 1-49
FLDPI 1-49
FLDZ 1-50
FLD1 1-50
FMUL 1-51
FNOP 1-52
FPATAN 1-52
FPREM 1-52
FPTAN 1-53
FRNDCINT 1-53
FRSTOR 1-53
FSAVE/FNSAVE 1-54
FSCALE 1-54
FSQRT 1-55
FST 1-55
FSTCW/FNSTCW 1-56
FSTENV/FNSTENV 1-56
FSTP 1-57
FSTSW/FNSTSW 1-57
FSUB 1-58
FSUBR 1-59
FTST 1-60
FWAIT 1-60
FXAM 1-61
FXCH 1-62
FXTRACT 1-62
FYL2X 1-63
FYL2XP1 1-63
F2XM1 1-64
hardware interface 1-27
instruction set 1-35
interconnection 1-28
number system 1-34
programming interface 1-26
register stack 1-30
status word 1-31
tag word 1-33
memory expansion options, 1-197
  DIP module start address 1-201
memory module description 1-198
memory module pin configuration 1-199
memory option switch settings G-1
R/W memory operating characteristics 1-198
switch-configurable start address 1-200
memory locations,
  reserved 2-8
memory map,
  BIOS 2-9
  system 1-11
Memory Read Command (MEMR), I/O channel 1-18
memory switch settings, G-1
  extender card G-1
  memory options G-1
  system board G-1
Memory Write Command (MEMW), I/O channel 1-18
(MEMR) Memory Read Command, I/O channel 1-18
(MEMW) Memory Write Command, I/O channel 1-18
microprocessor (see Intel 8088 microprocessor)
mode control and status register 1-139
mode select register 1-141
modem control register 1-236
modem status register 1-238
monochrome display 1-121
monochrome display and printer adapter 1-113
monochrome display adapter 1-115
  6845 CRT control port 1-118
  6845 CRT status port 1-118
  block diagram 1-114
  character attributes 1-117
  connector specifications 1-119
  I/O address and bit map 1-117
  programming considerations 1-115
monochrome display, 1-121
  operating characteristics 1-122
  specifications E-3
Motorola 6845 CRT Controller,
(see color/graphics monitor adapter)
(see monochrome display adapter)

National Semiconductor INS8250 Asynchronous
(see asynchronous communications adapter)
  functional pin description  1-221
  input signals  1-221
  input/output signals  1-225
  output signals  1-224
null (printer)  1-92

one bit delay mode register  1-283
operating mode register  1-280
OSC (oscillator)  1-16
Oscillator (OSC), I/O channel  1-16

parameter passing (ROM BIOS)  2-3
pause (keyboard extended code)  2-17
power good signal  1-24
power supply  1-21
  connectors  1-23
  input requirements  1-22
  over-voltage/current protection  1-24
  pin assignments  1-23
  power good signal  1-24
  Vac output  1-22
  Vdc output  1-22
print screen (keyboard extended code)  2-17
printer adapter,  1-107
  block diagram  1-108
  connector specifications  1-111
  programming considerations  1-109
printer control codes,  1-91
  1/8-inch line feeding  1-94
  1920 bit-image graphics mode  1-101
  480 bit-image graphics mode  1-97
  7/72-inch line feeding  1-94
  960 bit-image graphics mode  1-99
960 bit-image graphics mode normal speed  1-100
bell  1-92
cancel  1-93
cancel ignore paper end  1-95
cancel skip perforation  1-99
carriage return  1-92
clear printer buffer  1-100
compressed  1-93
compressed off  1-93
double strike  1-96
double strike off  1-97
double width  1-99, 1-93
double width off  1-93
emphasized  1-96
emphasized off  1-96
escape  1-93
form feed  1-92
home head  1-91
horizontal tab  1-92
ignore paper end  1-94
line feed  1-92
null  1-92
printer deselected  1-93
printer selected  1-93
select character set 1  1-94
select character set 2  1-94
set horizontal tab stops  1-96
set lines per page  1-96
set skip perforation  1-99
set variable line feeding  1-97
set vertical tabs  1-95
starts variable line feeding  1-94
subscript/superscript  1-99
subscript/superscript off  1-99
underline  1-94
unidirectional printing  1-99
vertical tab  1-92
printer deselected (printer)  1-93
printer selected (printer)  1-93
printer,  1-81
  additional specifications  1-83
cable  1-81
connect pin assignment  1-87
control codes  1-91
graphic character set 1  1-103
graphic character set 2  1-105
interface signal descriptions  1-87
matrix character set 1-101
modes 1-90
parallel interface 1-86
parallel interface timing diagram 1-86
 specifications 1-82, E-3
switch locations 1-84
switch settings 1-84
processor (see Intel 8088 microprocessor)
programmable baud rate generator 1-229
programming considerations,
  asynchronous communications adapter 1-226
  binary synchronous communications adapter 1-253
  color/graphics monitor adapter 1-137
diskette drive adapter 1-151
extender card 1-74
fixed disk drive adapter 1-179
monochrome display adapter 1-115
printer adapter 1-109
receiver card 1-77
SDLC adapter 1-275
prototype card, 1-209
  block diagram 1-210
  external interface 1-213
  I/O channel interface 1-211
  layout 1-211
  system loading and power limitations 1-213

Q
quick reference, character set C-12

R
receiver buffer register 1-249
receiver card, 1-77
  block diagram 1-78
  programming considerations 1-77
register,

6845 description (color/graphic adapter) 1-139
color select (color/graphic adapter) 1-140
command status 0 (diskette drive adapter) 1-167
command status 1 (diskette drive adapter) 1-168
command status 2 (diskette drive adapter) 1-169
command status 3 (diskette drive adapter) 1-170
data (fixed disk drive adapter) 1-185
data transfer mode (SDLC) 1-272
digital output (diskette drive adapter) 1-153
DPC (diskette drive adapter) 1-167
interrupt enable (asynchronous communications) 1-235
interrupt identification (asynchronous communications) 1-235
line control (asynchronous communications) 1-227
line status (asynchronous communications) 1-231
mode control and status (color/graphics) 1-139
mode select (color/graphics) 1-141
modem control (asynchronous communications) 1-236
modem status (asynchronous communications) 1-238
one-bit delay mode (SDLC) 1-283
operating mode (SDLC) 1-280
receiver buffer (asynchronous communications) 1-240
serial I/O mode (SDLC) 1-282
status (color/graphics) 1-143
status (fixed disk drive adapter) 1-181
transmitter holding (asynchronous communications) 1-241

reserved interrupts,

BASIC and DOS 2-7
reserved memory locations 2-7
Reset Drive (RESET DRV), I/O channel 1-16
RESET DRV (Reset Drive), I/O channel 1-16
RF modulator connector specifications 1-147

ROM BIOS, 2-2

Fixed Disk A-84
System A-2

ROM, adapter cards with 2-10
RS-232C,

interface standards F-2
scan codes,
  keyboard 1-65
scroll lock (keyboard extended code) 2-14
SDLC (see synchronous data link control)
select character set 1 (printer) 1-94
select character set 2 (printer) 1-94
sense bytes, fixed disk drive adapter 1-181
serial I/O mode register 1-282
set horizontal tab stops (printer) 1-96
set lines per page (printer) 1-96
set skip perforation (printer) 1-99
set variable line feeding (printer) 1-95, 1-97
set vertical tabs (printer) 1-95
shift (keyboard extended code) 2-13
shift key priorities (keyboard code) 2-14
shift states (keyboard extended code) 2-13
software interrupt listing 2-4
speaker connector 1-20
speaker drive system 1-20
speaker interface 1-20
specifications,
  80 CPS printers E-3
color display E-2
expansion unit E-2
keyboard E-1
monochrome display E-3
printer 1-82
printer (additional) 1-83
system unit E-1
stack area 2-7
starts variable line feeding (printer) 1-94
status register, 1-137
  color/graphics monitor adapter 1-143
  fixed disk drive adapter 1-181
  synchronous data link control adapter 1-276
subscriber/superscript (printer) 1-99
subscriber/superscript off (printer) 1-99
switch settings, G-1
diskette drive G-1
display adapter type G-1
extender card G-1
memory options G-1
printer 1-84
system board  G-1
system board memory  G-1
synchronous data link control communications adapter,  1-265
  8253-5 interval timer control word  1-279
  8253-5 programmable interval timer  1-275
  8255A-5 port A assignments  1-274
  8255A-5 port B assignments  1-274
  8255A-5 port C assignments  1-275
  8255A-5 programmable peripheral interface  1-274
  8273 command phase flow chart  1-286
  8273 commands  1-285
  8273 control/read/write registers  1-269
  8273 data interfaces  1-270
  8273 elements of data transfer interface  1-270
  8273 mode register commands  1-283
  8273 modem control block  1-271
  8273 modem control port A  1-271
  8273 modem control port B  1-272
  8273 modem interface  1-271
  8273 protocol controller operations  1-266
  8273 protocol controller structure  1-267
  8273 register selection  1-268
  8273 SDLC protocol controller block diagram  1-257
  8273 transmit/receiver timing  1-283
  block diagram  1-265
  command phase  1-284
  connector specifications  1-293
  control/read/write logic  1-268
  data transfer mode register  1-282
  device addresses  1-291
  execution phase  1-287
  general receive  1-288
  initialization/configuration commands  1-280
  initializing the SDLC adapter  1-278
  interface information  1-292
  interrupt information  1-291
  one bit delay code register  1-283
  operating mode register  1-290
  partial byte received codes  1-290
  processor interface  1-268
  programming considerations  1-275
  protocol control module features  1-266
  protocol controller operations  1-266
  result code summary  1-290
  result phase  1-287
selective receive 1-289
serial data timing block 1-273
serial I/O mode register 1-282
status register format 1-276
transmit 1-288
system block diagram 1-2
system board, 1-3
  component diagram 1-13
data flow 1-6
  R/W memory operating characteristics 1-198
switch settings G-1
System Clock (CLK), I/O channel 1-16
system memory map 1-12
system reset (keyboard extended code) 2-16
system ROM BIOS A-2
system unit, 1-3
  I/O channel 1-14
  I/O channel diagram 1-15
  keyboard interface 1-67
  power supply 1-21
  speaker interface 1-20
  specifications E-1
  system board 1-3

T

T/C (Terminal Count), I/O channel 1-19
transmitter holding register 1-241

U

underline (printer) 1-94
unidirectional printer (printer) 1-99
V

Vac output,
  system unit  1-22
Vdc output,
  system unit  1-22
vectors with special meanings  2-5
vertical tab (printer)  1-92
voltage interchange,
  asynchronous communications adapter  1-215

Numerics

1/8 inch line feeding (printer)  1-94
1920 bit-image graphics mode (printer)  1-100
480 bit-image graphics mode (printer)  1-97
6845,
  (see color/graphics monitor adapter)
  (see monochrome display adapter)
7/72 inch line feeding (printer)  1-94
8088,
  (see Intel 8088 microprocessor)  1-4
  (see asynchronous communications adapter)
8250,
  (see synchronous data link control adapter)
8253-5,
  (see synchronous data link control adapter)
8255A  1-10
8255A-5,
  (see synchronous data link control adapter)
8273,
  (see synchronous data link control adapter)
960 bit-image graphics mode (printer)  1-99
960 bit-image graphics mode normal speed (printer)  1-100
Notes:
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:
IBM PERSONAL COMPUTER
SALES & SERVICE
P.O. BOX 1328-C
BOCA RATON, FLORIDA 33432
Reader's Comment Form

TECHNICAL REFERENCE 1502237

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