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.
Technical Reference
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
5-¼ Inch Diskette Drive Adapter ........................................ D-45
5-¼ Inch Diskette Drive – Type 1 .......................................... D-49
5-¼ 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
Notes:
Notes:
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
System Block Diagram

1-2 System Unit
IBM Personal Computer XT 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>2FO-2FF</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>3FB-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
<table>
<thead>
<tr>
<th>Hex Port Number</th>
<th>PA0</th>
<th>+Keyboard Scan Code</th>
<th>0</th>
<th>Diagnostic Outputs</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0060</td>
<td>1</td>
<td>I</td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>2</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>3</td>
<td>3</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>P</td>
<td>4</td>
<td>4</td>
<td>or</td>
<td></td>
</tr>
<tr>
<td></td>
<td>U</td>
<td>5</td>
<td>5</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>T</td>
<td>6</td>
<td>6</td>
<td></td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7</td>
<td>7</td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>0061</td>
<td>PB0</td>
<td>+Timer 2 Gate Speaker</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>+Speaker Data</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>U</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>T</td>
<td>3</td>
<td></td>
<td>Read High Switches</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>or Read Low Switches</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>P</td>
<td>4</td>
<td></td>
<td>–Enable RAM Parity</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Check</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>U</td>
<td>5</td>
<td></td>
<td>–Enable I/O Channel Check</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T</td>
<td>6</td>
<td></td>
<td>–Hold Keyboard Clock Low</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>7</td>
<td></td>
<td>–(Enable Keyboard)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>or + (Clear Keyboard)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0062</td>
<td>PC0</td>
<td>Loop on POST</td>
<td>Sw–1</td>
<td>Display</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>**Sw–5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>I</td>
<td>1</td>
<td>Sw–2</td>
<td>Display</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+Co-Processor Installed</td>
<td></td>
<td>**Sw–6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>2</td>
<td>*Sw–3</td>
<td>Or #5-1/4 Drives</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+Planar RAM Size 0</td>
<td></td>
<td>**Sw–7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>P</td>
<td>3</td>
<td>*Sw–4</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>U</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>T</td>
<td>5</td>
<td></td>
<td>+Timer Channel 2 Out</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>6</td>
<td></td>
<td>+I/O Channel Check</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>7</td>
<td></td>
<td>+RAM Parity Check</td>
<td></td>
</tr>
<tr>
<td>0063</td>
<td></td>
<td>Command/Mode Register Hex 99</td>
<td>Mode Register Value</td>
<td>7 6 5 4 3 2 1 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 0 0 1 1 0 0 1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>* Sw–4 Sw–3</th>
<th>Amount of Memory On System Board</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0</td>
<td>64K</td>
</tr>
<tr>
<td>0 1</td>
<td>128K</td>
</tr>
<tr>
<td>1 0</td>
<td>192K</td>
</tr>
<tr>
<td>1 1</td>
<td>256K</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>** Sw–6 Sw–5</th>
<th>Display at Power-Up Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0</td>
<td>Reserved</td>
</tr>
<tr>
<td>0 1</td>
<td>Color 40 X 25 (BW Mode)</td>
</tr>
<tr>
<td>1 0</td>
<td>Color 80 X 25 (BW Mode)</td>
</tr>
<tr>
<td>1 1</td>
<td>IBM Monochrome 80 X 25</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>*** Sw–8 Sw–7</th>
<th>Number of 5-1/4&quot; Drives In System</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0</td>
<td>1</td>
</tr>
<tr>
<td>0 1</td>
<td>2</td>
</tr>
<tr>
<td>1 0</td>
<td>3</td>
</tr>
<tr>
<td>1 1</td>
<td>4</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></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>128-256K Read/Write Memory on System Board</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>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decimal</td>
<td>Hex</td>
</tr>
<tr>
<td>640K</td>
<td>A0000</td>
</tr>
<tr>
<td>656K</td>
<td>A4000</td>
</tr>
<tr>
<td>672K</td>
<td>A8000</td>
</tr>
<tr>
<td>688K</td>
<td>AC000</td>
</tr>
<tr>
<td>704K</td>
<td>B0000 Monochrome</td>
</tr>
<tr>
<td>720K</td>
<td>B4000</td>
</tr>
<tr>
<td>736K</td>
<td>B8000 Color/Graphics</td>
</tr>
<tr>
<td>752K</td>
<td>BC000</td>
</tr>
<tr>
<td>768K</td>
<td>C0000</td>
</tr>
<tr>
<td>784K</td>
<td>C4000</td>
</tr>
<tr>
<td>800K</td>
<td>C8000 Fixed Disk Control</td>
</tr>
<tr>
<td>816K</td>
<td>CC000 192K Read Only Memory</td>
</tr>
<tr>
<td>832K</td>
<td>D0000 Expansion and Control</td>
</tr>
<tr>
<td>848K</td>
<td>D4000</td>
</tr>
<tr>
<td>864K</td>
<td>D8000</td>
</tr>
<tr>
<td>880K</td>
<td>DC000</td>
</tr>
<tr>
<td>896K</td>
<td>E0000</td>
</tr>
<tr>
<td>912K</td>
<td>E4000</td>
</tr>
<tr>
<td>928K</td>
<td>E8000</td>
</tr>
<tr>
<td>944K</td>
<td>EC000</td>
</tr>
<tr>
<td>960K</td>
<td>F0000 64K Base System ROM</td>
</tr>
<tr>
<td>976K</td>
<td>F4000 BIOS and BASIC</td>
</tr>
<tr>
<td>992K</td>
<td>F8000</td>
</tr>
<tr>
<td>1008K</td>
<td>FC000</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 Diagram
# 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 Vdc ±5%, located on 2 connector pins
−5 Vdc ±10%, located on 1 connector pin
+12 Vdc ±5%, located on 1 connector pin
−12 Vdc ±10%, located on 1 connector pin
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]

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>Voltage @ 50/60 Hz</th>
<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>
<td></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></td>
<td>Nominal</td>
<td>Minimum</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></td>
<td>Nominal</td>
<td>Minimum</td>
</tr>
<tr>
<td>120</td>
<td>0.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Vac Output

1-22   System Unit
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:
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>
IBM Personal Computer Math Coprocessor

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>$-32,768 \leq X \leq +32,767$</td>
</tr>
<tr>
<td>Short Integer</td>
<td>32</td>
<td>9</td>
<td>$-2 \times 10^9 \leq X \leq +2 \times 10^9$</td>
</tr>
<tr>
<td>Long Integer</td>
<td>64</td>
<td>18</td>
<td>$-9 \times 10^{18} \leq X \leq +9 \times 10^{18}$</td>
</tr>
<tr>
<td>Packed Decimal</td>
<td>80</td>
<td>18</td>
<td>$-99...99 \leq X \leq +99...99$ (18 digits)</td>
</tr>
<tr>
<td>Short Real*</td>
<td>32</td>
<td>6-7</td>
<td>$8.43 \times 10^{-37} \leq</td>
</tr>
<tr>
<td>Long Real*</td>
<td>64</td>
<td>15-16</td>
<td>$4.19 \times 10^{-307} \leq</td>
</tr>
<tr>
<td>Temporary Real</td>
<td>80</td>
<td>19</td>
<td>$3.4 \times 10^{-4932} \leq</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.

![Register Structure Diagram]

Register Structure
**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.

---

**Status Word Format**

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

---

Coprocessor 1-31
The coprocessor provides several options that are selected by loading a control word register.

**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 0)
- 10 = Round Up (toward +∞)
- 11 = Chop (Truncate Toward Zero)

(4) **Infinity Control:**
- 0 = Projective
- 1 = Affine
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>TAG(7)</th>
<th>TAG(6)</th>
<th>TAG(5)</th>
<th>TAG(4)</th>
<th>TAG(3)</th>
<th>TAG(2)</th>
<th>TAG(1)</th>
<th>TAG(0)</th>
</tr>
</thead>
</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 ±4.19×10^{-307} to ±1.67×10^{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 ±3.4×10^{-4932} to ±1.2×10^{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.

Coprocessor Number System
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></td>
<td><strong>Typical</strong></td>
</tr>
<tr>
<td>(no operands)</td>
<td>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></td>
<td><strong>Typical</strong></td>
</tr>
<tr>
<td>//ST,ST(1)/ST(1),ST</td>
<td>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></td>
<td><strong>Typical</strong></td>
</tr>
<tr>
<td>ST(1),ST</td>
<td>90</td>
</tr>
</tbody>
</table>

1-36 Coprocessor
FIADD

Exceptions: I, D, O, P

<table>
<thead>
<tr>
<th>Operands</th>
<th>Execution Clocks</th>
<th>Transfers</th>
<th>Bytes</th>
<th>Coding Example</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Typical</td>
<td>8088</td>
<td></td>
<td></td>
</tr>
<tr>
<td></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’.

<table>
<thead>
<tr>
<th>FBLD</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></td>
<td>Range</td>
</tr>
<tr>
<td>packed-decimal</td>
<td>300+EA</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>FBSTP</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></td>
<td>Range</td>
</tr>
<tr>
<td>packed-decimal</td>
<td>530+EA</td>
</tr>
</tbody>
</table>
FCHS

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

<table>
<thead>
<tr>
<th>FCHS (no operands)</th>
<th>Exceptions: 1</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: 1, 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>
</tbody>
</table>

1-38 Coprocessor
<table>
<thead>
<tr>
<th>C3</th>
<th>C0</th>
<th>Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>ST &gt; source</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>ST &lt; source</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>ST = source</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>ST ? source</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>
FDECSTP

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

<table>
<thead>
<tr>
<th>FDECSTP (no operands)</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>9</td>
</tr>
</tbody>
</table>

FDISI/FNDISI

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

<table>
<thead>
<tr>
<th>FDISI/FNDISI (no operands)</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>5</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>FDIV</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>198</td>
</tr>
<tr>
<td>short-real</td>
<td>220+EA</td>
</tr>
<tr>
<td>long-real</td>
<td>225+EA</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FDIVP</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>202</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FIDIV</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>236+EA</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>
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></td>
<td>Typical</td>
</tr>
<tr>
<td></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></td>
<td>Typical</td>
</tr>
<tr>
<td></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></td>
<td>Typical</td>
</tr>
<tr>
<td></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><strong>Typical</strong></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><strong>Typical</strong></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><strong>Typical</strong></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.

<table>
<thead>
<tr>
<th>FINIT/FNINIT (no operands)</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>5</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>

Coprocessor 1-45
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></td>
<td></td>
</tr>
<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></td>
<td></td>
</tr>
<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>Operands</th>
<th>Execution Clocks</th>
<th>Transfers</th>
<th>Bytes</th>
<th>Coding Example</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Typical</td>
<td>Range</td>
<td>8088</td>
<td></td>
</tr>
<tr>
<td>ST(i)</td>
<td>20</td>
<td>17-22</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>short-real</td>
<td>43+EA</td>
<td>38-56+EA</td>
<td>4</td>
<td>2-4</td>
</tr>
<tr>
<td>long-real</td>
<td>46+EA</td>
<td>40-60+EA</td>
<td>8</td>
<td>2-4</td>
</tr>
<tr>
<td>temp-real</td>
<td>57+EA</td>
<td>53-65+EA</td>
<td>10</td>
<td>2-4</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>Operands</th>
<th>Execution Clocks</th>
<th>Transfers</th>
<th>Bytes</th>
<th>Coding Example</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Typical</td>
<td>Range</td>
<td>8088</td>
<td></td>
</tr>
<tr>
<td>2-bytes</td>
<td>10+EA</td>
<td>7-14+EA</td>
<td>2</td>
<td>2-4</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>FLDENV</th>
<th>Exceptions: None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands</td>
<td>Execution Clocks</td>
</tr>
<tr>
<td>14-bytes</td>
<td>Typical</td>
</tr>
<tr>
<td></td>
<td>40+EA</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>FLDLG2 (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>21</td>
</tr>
</tbody>
</table>

FLDLN2

FLDLN2 (load log base e of 2) loads (pushes) the value of \(\log_e2\) onto the stack.

<table>
<thead>
<tr>
<th>FLDLN2 (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>20</td>
</tr>
</tbody>
</table>

1-48 Coprocessor
**FLDL2E**

FLDL2E (load log base 2 of e) loads (pushes) the value \( \log_2 e \) 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>Typical 18</td>
<td>Range 15-21</td>
<td>0</td>
<td>2</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>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 19</td>
<td>Range 16-22</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

**FLDPI**

FLDPI (load \( \pi \)) loads (pushes) \( \pi \) 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>Typical 19</td>
<td>Range 16-22</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>
FLDZ

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

<table>
<thead>
<tr>
<th>FLDZ (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>14</td>
</tr>
</tbody>
</table>

FLD1

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

<table>
<thead>
<tr>
<th>FLD1 (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>18</td>
</tr>
</tbody>
</table>
**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>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></td>
<td><strong>Typical</strong></td>
</tr>
<tr>
<td>//ST(i),ST/ST,ST(i)</td>
<td>97</td>
</tr>
<tr>
<td>//ST(i),ST/ST,ST(i)</td>
<td>138</td>
</tr>
<tr>
<td>short-real</td>
<td>118+EA</td>
</tr>
<tr>
<td>long-real</td>
<td>120+EA</td>
</tr>
<tr>
<td>long-real</td>
<td>161+EA</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>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></td>
<td><strong>Typical</strong></td>
</tr>
<tr>
<td>ST(i),ST</td>
<td>100</td>
</tr>
<tr>
<td>ST(i),ST</td>
<td>142</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>Exceptions: I, D, O, 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>130+EA</td>
</tr>
<tr>
<td>short-integer</td>
<td>136+EA</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>Operands</td>
<td>Execution Clocks</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 $\theta = \text{ARCTAN}(Y/X)$. $X$ 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>Operands</td>
<td>Execution Clocks</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>Operands</td>
<td>Execution Clocks</td>
</tr>
<tr>
<td></td>
<td>Typical</td>
</tr>
<tr>
<td>(no operands)</td>
<td>125</td>
</tr>
</tbody>
</table>

1-52  Coprocessor
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>(no operands)</td>
<td>Typical</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>(no operands)</td>
<td>Typical</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>94-bytes</td>
<td>Typical</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></td>
<td><strong>Typical</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, FSCALE provides rapid multiplication or division by integral powers of 2.

<table>
<thead>
<tr>
<th>FSCALE (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></td>
<td><strong>Typical</strong></td>
</tr>
<tr>
<td>(no operands)</td>
<td>35</td>
</tr>
</tbody>
</table>

1-54 Coprocessor
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><strong>Operands</strong></td>
<td><strong>Execution Clocks</strong></td>
</tr>
<tr>
<td>(no operands)</td>
<td>Typical</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><strong>Operands</strong></td>
<td><strong>Execution Clocks</strong></td>
</tr>
<tr>
<td>ST(i) short-real</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><strong>Operands</strong></td>
<td><strong>Execution Clocks</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Typical</strong></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><strong>Operands</strong></td>
<td><strong>Execution Clocks</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Typical</strong></td>
</tr>
<tr>
<td>14-bytes</td>
<td>45+EA</td>
</tr>
</tbody>
</table>
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>Operands</td>
<td>Execution Clocks</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>Operands</td>
<td>Execution Clocks</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>Exceptions: I, D, O, U, P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands</td>
<td>Execution Clocks</td>
</tr>
<tr>
<td>Typical</td>
<td>Range</td>
</tr>
<tr>
<td>//ST,ST(i)/ST(i),ST</td>
<td></td>
</tr>
<tr>
<td>short-real</td>
<td>85</td>
</tr>
<tr>
<td>long-real</td>
<td>105+EA</td>
</tr>
<tr>
<td></td>
<td>110+EA</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FSUBP</th>
<th>Exceptions: I, D, O, U, P</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),ST</td>
<td>90</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FISUB</th>
<th>Exceptions: I, D, O, P</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>120+EA</td>
</tr>
<tr>
<td>short-integer</td>
<td>125+EA</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><strong>Operands</strong></td>
<td><strong>Execution Clocks</strong></td>
</tr>
<tr>
<td>(no operands)</td>
<td>Typical</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 $\infty$)</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><strong>Operands</strong></td>
<td><strong>Execution Clocks</strong></td>
</tr>
<tr>
<td>(no operands)</td>
<td>Typical</td>
</tr>
<tr>
<td></td>
<td>3+5n</td>
</tr>
</tbody>
</table>
FXAM

FXAM (examine) reports the content of the top stack element as positive/negative and NAN/unnorma/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></td>
<td>Typical</td>
</tr>
<tr>
<td>(no operands)</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>

Coprocessor 1-61
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></td>
<td>Execution Clocks</td>
</tr>
<tr>
<td></td>
<td>Typical Range</td>
</tr>
<tr>
<td>//ST(i)</td>
<td>12 10-15</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></td>
<td>Execution Clocks</td>
</tr>
<tr>
<td></td>
<td>Typical Range</td>
</tr>
<tr>
<td>(no operands)</td>
<td>50 27-55</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.

\[ \log_2 X \]

### Exceptions: P (operands not checked)

<table>
<thead>
<tr>
<th>Operands</th>
<th>Execution Clocks</th>
<th>Transfers</th>
<th>Bytes</th>
<th>Coding Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>(no operands)</td>
<td>950</td>
<td>0</td>
<td>2</td>
<td>FYL2X</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.

### Exceptions: P (operands not checked)

<table>
<thead>
<tr>
<th>Operands</th>
<th>Execution Clocks</th>
<th>Transfers</th>
<th>Bytes</th>
<th>Coding Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>(no operands)</td>
<td>850</td>
<td>0</td>
<td>2</td>
<td>FYL2XP1</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>(no operands)</td>
<td>Typical</td>
</tr>
<tr>
<td></td>
<td>500</td>
</tr>
</tbody>
</table>
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
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
## Keyboard Interface Connector Specifications

<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></td>
<td><strong>Power Supply Voltages</strong></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>
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 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>A0 - 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 FFFFF (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 FXXXXX. 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 assignments]

<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
Notes:
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</td>
<td>9 x 9</td>
</tr>
<tr>
<td>Matrix:</td>
<td>Full 96-character ASCII with descenders plus 9 international characters/symbols.</td>
</tr>
<tr>
<td>Character Set:</td>
<td>See “Additional Printer Specifications”</td>
</tr>
<tr>
<td>Graphic Character:</td>
<td></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>Adjustable sprocket pin feed</td>
</tr>
<tr>
<td>Paper Feed:</td>
<td></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>Parallel 8-bit Data and Control Lines</td>
</tr>
<tr>
<td>Standard:</td>
<td></td>
</tr>
<tr>
<td>(10) Inked Ribbon:</td>
<td>Black</td>
</tr>
<tr>
<td>Color:</td>
<td>Cartridge</td>
</tr>
<tr>
<td>Type:</td>
<td>3 million characters</td>
</tr>
<tr>
<td>Life Expectancy:</td>
<td></td>
</tr>
<tr>
<td>(11) Environmental Conditions</td>
<td>41 to 95°F (5 to 35°C)</td>
</tr>
<tr>
<td>Operating Temperature Range:</td>
<td></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
64 block characters.
IBM 80 CPS Graphics Printer
(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:
\[
\begin{array}{|c|c|c|}
\hline
 & \text{Characters per inch} & \text{Maximum characters per line} \\
\hline
\text{Subscript:} & 10 & 80 \\
\text{Superscript:} & 10 & 80 \\
\hline
\end{array}
\]

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.

<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 (Graphic Pattern Select)</td>
<td>N.A.</td>
<td>Graphic Patterns Select</td>
<td>Off</td>
</tr>
<tr>
<td>1-8</td>
<td>SLCT IN Signal Fixed Internally</td>
<td>Fixed</td>
<td>Not Fixed</td>
<td>On</td>
</tr>
</tbody>
</table>

Functions and Conditions of DIP Switch 1 (Matrix)
### 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>
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
| Connector Pin Assignment and Descriptions of Interface Signals (Part 1 of 3) |

<table>
<thead>
<tr>
<th>Signal</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 &quot;high&quot; 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 &quot;low&quot; level, the paper is automatically fed one line after printing. (The signal level can be fixed to &quot;low&quot; 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>0V</td>
<td></td>
<td>Logic GND level.</td>
</tr>
<tr>
<td>17</td>
<td></td>
<td>CHASSIS-</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>&quot;Twisted-Pair Return&quot; 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 &quot;low&quot; the printer controller is reset to its initial state and the print buffer is cleared. This signal is normally at &quot;high&quot; 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)
<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 “low” when the printer is in “Paper End” state, “Offline” state and “Error” 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 “low”. (Internal fixing can be carried out with DIP SW 1-8. The condition at the time of shipment is set “low” 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</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</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></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>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</td>
</tr>
<tr>
<td></td>
<td>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).</td>
</tr>
<tr>
<td></td>
<td>Example: LPRINT CHR$ (0);</td>
</tr>
<tr>
<td>BEL</td>
<td>Bell</td>
</tr>
<tr>
<td></td>
<td>Sounds the printer buzzer for 1 second.</td>
</tr>
<tr>
<td></td>
<td>Example: LPRINT CHR$ (7);</td>
</tr>
<tr>
<td>HT</td>
<td>Horizontal Tab</td>
</tr>
<tr>
<td></td>
<td>Tabs to the next horizontal tap stop. Tab stops are set with ESC D.</td>
</tr>
<tr>
<td></td>
<td>No tab stops are set when the printer is powered on. (Graphics Printer sets a tab stop every 8 columns when powered on.)</td>
</tr>
<tr>
<td></td>
<td>Example: LPRINT CHR$ (9);</td>
</tr>
<tr>
<td>LF</td>
<td>Line Feed</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>Example: LPRINT CHR$ (10);</td>
</tr>
<tr>
<td>VT</td>
<td>Vertical Tab</td>
</tr>
<tr>
<td></td>
<td>Spaces 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.)</td>
</tr>
<tr>
<td></td>
<td>Example: LPRINT CHR$ (11);</td>
</tr>
<tr>
<td>FF</td>
<td>Form Feed</td>
</tr>
<tr>
<td></td>
<td>Advances the paper to the top of the next page.</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>Example: LPRINT CHR$ (12);</td>
</tr>
<tr>
<td>CR</td>
<td>Carriage Return</td>
</tr>
<tr>
<td></td>
<td>Ends the line that the printer is on and prints the data remaining in the printer buffer. (No Line Feed operation takes place.)</td>
</tr>
<tr>
<td></td>
<td>Note: IBM Personal Computer BASIC adds a Line Feed unless 128 is added [for example, CHR$ (141)].</td>
</tr>
<tr>
<td></td>
<td>Example: LPRINT CHR$ (13);</td>
</tr>
<tr>
<td>Printer Code</td>
<td>Printer Function</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------</td>
</tr>
<tr>
<td>SO</td>
<td><strong>Shift Out (Double Width)</strong>&lt;br&gt;Changes the printer to the Double Width print mode.&lt;br&gt;<em>Note:</em> A Carriage Return, Line Feed or DC4 cancels Double Width print mode.&lt;br&gt;Example:&lt;br&gt;LPRINT CHR$(14);</td>
</tr>
<tr>
<td>SI</td>
<td><strong>Shift In (Compressed)</strong>&lt;br&gt;Changes the printer to the Compressed Character print mode.&lt;br&gt;Example:&lt;br&gt;LPRINT CHR$(15);</td>
</tr>
<tr>
<td>DC1</td>
<td><strong>Device Control 1 (Printer Selected)</strong>&lt;br&gt;(Graphics Printer ignores DC1)&lt;br&gt;Printer accepts data from the system unit. Printer DIP switch 1-8 must be set to the Off position.&lt;br&gt;Example:&lt;br&gt;LPRINT CHR$(17);</td>
</tr>
<tr>
<td>DC2</td>
<td><strong>Device Control 2 (Compressed Off)</strong>&lt;br&gt;Stops printing in the Compressed print mode.&lt;br&gt;Example:&lt;br&gt;LPRINT CHR$(18);</td>
</tr>
<tr>
<td>DC3</td>
<td><strong>Device Control 3 (Printer Deselected)</strong>&lt;br&gt;(Graphics Printer ignores DC3)&lt;br&gt;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.&lt;br&gt;Example:&lt;br&gt;LPRINT CHR$(19);</td>
</tr>
<tr>
<td>DC4</td>
<td><strong>Device Control 4 (Double Width Off)</strong>&lt;br&gt;Stops printing in the Double Width print mode.&lt;br&gt;Example:&lt;br&gt;LPRINT CHR$(20);</td>
</tr>
<tr>
<td>CAN</td>
<td><strong>Cancel</strong>&lt;br&gt;Clears the printer buffer. Control codes, except SO, remain in effect.&lt;br&gt;Example:&lt;br&gt;LPRINT CHR$(24);</td>
</tr>
<tr>
<td>ESC</td>
<td><strong>Escape</strong>&lt;br&gt;Lets the printer know that the next data sent is a printer command.&lt;br&gt;(See the following list of commands.)&lt;br&gt;Example:&lt;br&gt;LPRINT CHR$(27);</td>
</tr>
<tr>
<td>Printer Code</td>
<td>Printer Function</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------</td>
</tr>
</tbody>
</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 n/72-inch. The example below tells the printer to set line feeding to 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 24/72-inch. Any increment between 1/72 and 85/72 may be used.&lt;br&gt;Example:&lt;br&gt;LPRINT CHR$(27);CHR$(65);CHR$(24);CHR$(27);CHR$(50);</td>
</tr>
<tr>
<td>ESC B</td>
<td><strong>Escape B (Set Vertical Tabs)</strong>&lt;br&gt;Format: ESC B;n₁;n₂;...nₖ;NUL;&lt;br&gt;(Graphics Printer ignores ESC B)&lt;br&gt;Sets vertical tab stop positions. Up to 64 vertical tab stop positions are recognized by the printer. The n’s, in the format above, are used to indicate tab stop positions. Tab stop numbers must be received in ascending numeric order. The tab stop numbers will not become valid until the NUL code is entered. Once vertical tab stops are established, they will be valid until new tab stops are specified. (If the printer is reset or powered Off, set tab stops are cleared.) If no tab stop is set, the Vertical Tab command behaves as a Line Feed command. ESC B followed only by NUL will cancel tab stops. The form length must be set by the ESC C command prior to setting tabs.&lt;br&gt;Example:&lt;br&gt;LPRINT CHR$(27);CHR$(66);CHR$(10);CHR$(20);CHR$(40);CHR$(0);</td>
</tr>
<tr>
<td>Printer Code</td>
<td>Printer Function</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>ESC C</td>
<td><strong>Escape C (Set Lines per Page)</strong>&lt;br&gt;Format: ESC C;n;&lt;br&gt;Sets the page length. The ESC C command must have a value following it to specify the length of page desired. (Maximum form length for the printer is 127 lines.)&lt;br&gt;The example below sets the page length to 55 lines. The printer defaults to 66 lines per page when powered on or reset.&lt;br&gt;Example:&lt;br&gt;LPRINT CHR$(27);CHR$(67);CHR$(55);</td>
</tr>
<tr>
<td>ESC D</td>
<td><strong>Escape D (Set Horizontal Tab Stops)</strong>&lt;br&gt;Format: ESC D;n₁;n₂;...nₖ;NUL;&lt;br&gt;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$(0), the NUL code. They must also be in ascending numeric order as shown. Tab stops can be set between 1 and 80. When in the Compressed print mode, tab stops can be set up to 132.&lt;br&gt;The maximum number of tabs that can be set is 112. The Graphics Printer can have a maximum of 28 tab stops. The HT (CHR$(9)) is used to execute a tab operation.&lt;br&gt;Example:&lt;br&gt;LPRINT CHR$(27);CHR$(68);CHR$(10)CHR$(20)CHR$(40);CHR$(0);</td>
</tr>
<tr>
<td>ESC E</td>
<td><strong>Escape E (Emphasized)</strong>&lt;br&gt;Changes the printer to the Emphasized print mode. The speed of the printer is reduced to half speed during the Emphasized print mode.&lt;br&gt;Example:&lt;br&gt;LPRINT CHR$(27);CHR$(69);</td>
</tr>
<tr>
<td>ESC F</td>
<td><strong>Escape F (Emphasized Off)</strong>&lt;br&gt;Stops printing in the Emphasized print mode.&lt;br&gt;Example:&lt;br&gt;LPRINT CHR$(27);CHR$(70);</td>
</tr>
<tr>
<td>ESC G</td>
<td><strong>Escape G (Double Strike)</strong>&lt;br&gt;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.&lt;br&gt;Example:&lt;br&gt;LPRINT CHR$(27);CHR$(71);</td>
</tr>
</tbody>
</table>

1-96 Printers
<table>
<thead>
<tr>
<th>Printer Code</th>
<th>Printer Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESC H</td>
<td><strong>Escape H (Double Strike Off)</strong>&lt;br&gt;Stops printing in the Double Strike mode.&lt;br&gt;Example:&lt;br&gt;LPRINT CHR$(27);CHR$(72);</td>
</tr>
<tr>
<td>ESC J</td>
<td><strong>Escape J (Set Variable Line Feeding)</strong>&lt;br&gt;Format: ESC J;n;&lt;br&gt;(Graphics Printer only)&lt;br&gt;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.&lt;br&gt;Example:&lt;br&gt;LPRINT CHR$(27);CHR$(74);CHR$(50);</td>
</tr>
</tbody>
</table>
| ESC K        | **Escape K (480 Bit-Image Graphics Mode)**<br>Format ESC K;n₁;n₂;v₁;v₂;...vₖ;<br>(Graphics Printer only)<br>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.<br>**Note:** Assign values to n₁ and n₂ as follows:<br>n₁ represents values from 0 - 255.<br>n₂ represents values from 0 - 1 x 256.<br>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>12</td>
<td>11</td>
</tr>
<tr>
<td>10</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</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>2</td>
<td>2</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</td>
<td>Length of data</td>
<td>Bit-image data</td>
<td>Text</td>
<td>Length of data</td>
<td>Bit-image data</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

480 bit-image dot positions

Example:

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$(FIX (NDOTS/256));
11 'SEND NDOTS NUMBER OF BIT IMAGE BYTES
12 FOR I=1 TO NDOTS/12 'NUMBER OF SLASHES TO PRINT USING GRAPHICS
13 PRINT #1,SLASH$;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>
<tbody>
<tr>
<td><strong>ESC L</strong></td>
<td><strong>Escape L (960 Bit-Image Graphics Mode)</strong></td>
</tr>
<tr>
<td></td>
<td>Format: ESC L;n₁;n₂;v₁;v₂;...vₖ.</td>
</tr>
<tr>
<td></td>
<td>(Graphics Printer only)</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td><strong>ESC N</strong></td>
<td><strong>Escape N (Set Skip Perforation)</strong></td>
</tr>
<tr>
<td></td>
<td>Format ESC N;n;</td>
</tr>
<tr>
<td></td>
<td>(Graphics Printer only)</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>Example: CHR$(27);CHR$(78);CHR$(12);</td>
</tr>
<tr>
<td><strong>ESC O</strong></td>
<td><strong>Escape O (Cancel Skip Perforation)</strong></td>
</tr>
<tr>
<td></td>
<td>(Graphics Printer only)</td>
</tr>
<tr>
<td></td>
<td>Cancels the Skip Perforation function.</td>
</tr>
<tr>
<td></td>
<td>Example: LPRINT CHR$(27);CHR$(79);</td>
</tr>
<tr>
<td><strong>ESC S</strong></td>
<td><strong>Escape S (Subscript/Superscript)</strong></td>
</tr>
<tr>
<td></td>
<td>Format: ESC S;n;</td>
</tr>
<tr>
<td></td>
<td>(Graphics Printer only)</td>
</tr>
<tr>
<td></td>
<td>Changes the printer to the Subscript print mode when ESC S is followed by a 1, as in the example below. When ESC S is followed by a 0 (zero), the printer will print in the Superscript print mode.</td>
</tr>
<tr>
<td></td>
<td>Example: LPRINT CHR$(27);CHR$(83);CHR$(1);</td>
</tr>
<tr>
<td><strong>ESC T</strong></td>
<td><strong>Escape T (Subscript/Superscript Off)</strong></td>
</tr>
<tr>
<td></td>
<td>(Graphics Printer only)</td>
</tr>
<tr>
<td></td>
<td>The printer stops printing in the Subscript or Superscript print mode.</td>
</tr>
<tr>
<td></td>
<td>Example: LPRINT CHR$(27);CHR$(84);</td>
</tr>
<tr>
<td><strong>ESC U</strong></td>
<td><strong>Escape U (Unidirectional Printing)</strong></td>
</tr>
<tr>
<td></td>
<td>Format: ESC U;n;</td>
</tr>
<tr>
<td></td>
<td>(Graphics Printer only)</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>Example: LPRINT CHR$(27);CHR$(85);CHR$(1);</td>
</tr>
<tr>
<td>Printer Code</td>
<td>Printer Function</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td><strong>ESC W</strong></td>
<td><strong>Escape W (Double Width)</strong></td>
</tr>
<tr>
<td></td>
<td>Format: ESC W;n;</td>
</tr>
<tr>
<td></td>
<td>(Graphics Printer only)</td>
</tr>
<tr>
<td></td>
<td>Changes the printer to the Double Width print mode when ESC W is followed by a 1. This mode is not canceled by a line-feed operation and must be canceled with ESC W followed by a 0 (zero).</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>LPRINT CHR$(27);CHR$(87);CHR$(1);</td>
</tr>
<tr>
<td><strong>ESC Y</strong></td>
<td><strong>Escape Y (960 Bit-Image Graphics Mode Normal Speed)</strong></td>
</tr>
<tr>
<td></td>
<td>Format: ESC Y n₁;n₂;v₁;v₂;...vₖ;</td>
</tr>
<tr>
<td></td>
<td>(Graphics Printer only)</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td><strong>ESC Z</strong></td>
<td><strong>Escape Z (1920 Bit-Image Graphics Mode)</strong></td>
</tr>
<tr>
<td></td>
<td>Format: ESC Z;n₁;n₂;v₁;v₂;...vₖ;</td>
</tr>
<tr>
<td></td>
<td>(Graphics Printer only)</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td><strong>DEL</strong></td>
<td><strong>Delete (Clear Printer Buffer)</strong></td>
</tr>
<tr>
<td></td>
<td>(Graphics Printer ignores DEL)</td>
</tr>
<tr>
<td></td>
<td>Clears the printer buffer. Control codes, except SO, still remain in effect. DIP switch 1-5 must be in the Off position.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>LPRINT CHR$(127);</td>
</tr>
</tbody>
</table>
Matrix Printer Character Set (Part 1 of 2)
Matrix Printer Character Set (Part 2 of 2)
<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BEL</td>
<td></td>
<td>HT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
</tr>
<tr>
<td>LF</td>
<td>VT</td>
<td>FF</td>
<td>CR</td>
<td>SO</td>
<td>SI</td>
<td>DC2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
<td>29</td>
</tr>
<tr>
<td>DC4</td>
<td>CAN</td>
<td>ESC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>31</td>
<td>32</td>
<td>33</td>
<td>34</td>
<td>35</td>
<td>36</td>
<td>37</td>
<td>38</td>
<td>39</td>
</tr>
<tr>
<td>SP</td>
<td>!</td>
<td>&quot;</td>
<td>#</td>
<td>$</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>O</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>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>~</td>
<td>NUL</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>140</td>
<td>141</td>
<td>142</td>
<td>143</td>
<td>144</td>
<td>145</td>
<td>146</td>
<td>147</td>
<td>148</td>
<td>149</td>
</tr>
<tr>
<td>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></td>
<td></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>CAN</td>
<td>ESC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>á</td>
<td>í</td>
<td>ó</td>
<td>ú</td>
<td>ñ</td>
<td>Ñ</td>
<td>a</td>
<td>o</td>
<td>è</td>
<td>è</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>( \frac{1}{2} )</td>
<td>( \frac{1}{4} )</td>
<td>i</td>
<td>≪</td>
<td>≫</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>( \alpha )</td>
<td>( \beta )</td>
<td>( \Gamma )</td>
<td>( \Pi )</td>
<td>( \Sigma )</td>
<td>( \sigma )</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>( \mu )</td>
<td>( \tau )</td>
<td>( \phi )</td>
<td>( \Theta )</td>
<td>( \Omega )</td>
<td>( \delta )</td>
<td>( \infty )</td>
<td>( \emptyset )</td>
<td>( \epsilon )</td>
<td>( \cap )</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>( \equiv )</td>
<td>( \pm )</td>
<td>( \geq )</td>
<td>( \leq )</td>
<td>( \int )</td>
<td>( \div )</td>
<td>( \approx )</td>
<td>( \circ )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>250</td>
<td>251</td>
<td>252</td>
<td>253</td>
<td>254</td>
<td>255</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- ( \sqrt[n]{\cdot} )</td>
<td>2</td>
<td>SP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Graphics Printer Character Set 1 (Part 2 of 2)
<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>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>Ø</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>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>51</td>
<td>52</td>
<td>53</td>
<td>54</td>
<td>55</td>
<td>56</td>
<td>57</td>
<td>58</td>
<td>59</td>
</tr>
<tr>
<td>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>
<tr>
<td>~</td>
<td>ç</td>
<td>ü</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)
Graphics Printer Character Set 2 (Part 2 of 2)

1-106 Printers
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.
Printer Adapter Block Diagram
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 Bit 6 Bit 5 Bit 4 Bit 3 Bit 2 Bit 1 Bit 0</td>
<td>Pin 9 Pin 8 Pin 7 Pin 6 Pin 5 Pin 4 Pin 3 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.
IBM Monochrome Display & Printer Adapter

<table>
<thead>
<tr>
<th>Output to address hex 3BE</th>
<th>Printer Adapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 4</td>
<td>Bit 3</td>
</tr>
<tr>
<td>IRQ</td>
<td>Bit 2</td>
</tr>
<tr>
<td>Enable</td>
<td>Bit 1</td>
</tr>
<tr>
<td></td>
<td>Bit 0</td>
</tr>
<tr>
<td>Pin 17</td>
<td>Pin 16</td>
</tr>
<tr>
<td></td>
<td>Pin 14</td>
</tr>
<tr>
<td></td>
<td>Pin 1</td>
</tr>
</tbody>
</table>

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.

IBM Monochrome Display & Printer Adapter

<table>
<thead>
<tr>
<th>Input from address Hex 3BC</th>
<th>Printer Adapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input from address hex 378</td>
<td></td>
</tr>
</tbody>
</table>

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>

<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>

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></td>
<td></td>
<td>IRQ Enable</td>
<td>Pin 17</td>
<td>Pin 16</td>
<td>Pin 14</td>
<td>Pin 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Por=0</td>
<td>Por=1</td>
<td>Por=0</td>
<td>Por=1</td>
<td>Por=1</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).

<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
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>02</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.

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.

**I/O Address and Bit Map**
<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

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Ground</td>
<td>1</td>
</tr>
<tr>
<td>Ground</td>
<td>2</td>
</tr>
<tr>
<td>Not Used</td>
<td>3</td>
</tr>
<tr>
<td>Not Used</td>
<td>4</td>
</tr>
<tr>
<td>Not Used</td>
<td>5</td>
</tr>
<tr>
<td>+Intensity</td>
<td>6</td>
</tr>
<tr>
<td>+Video</td>
<td>7</td>
</tr>
<tr>
<td>+Horizontal</td>
<td>8</td>
</tr>
<tr>
<td>-Vertical</td>
<td>9</td>
</tr>
</tbody>
</table>

**IBM Monochrome Display**

**IBM Monochrome Display and Printer Adapter**

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

### Connector Specifications
Notes:

1-120  Monochrome Adapter
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 (P 39).
- 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.
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 Function</th>
<th>Attribute Byte</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>7 6 5 4 3 2 1 0</td>
<td></td>
</tr>
<tr>
<td>B R G B FG Background</td>
<td>B = Blinking Foreground (Character)</td>
<td></td>
</tr>
<tr>
<td>B = Blinking Foreground (Character)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7 6 5 4 3 2 1 0</td>
<td>I R G B Foreground</td>
<td></td>
</tr>
<tr>
<td>I = Highlighted Foreground (Character)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I = Highlighted Foreground (Character)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1-128  Color Graphics Adapter
The attribute byte definitions are:

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>R</td>
<td>G</td>
<td>B</td>
<td>I</td>
<td>R</td>
<td>G</td>
<td>B</td>
</tr>
</tbody>
</table>

- 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></td>
<td>Background Color</td>
<td>Character Color</td>
</tr>
<tr>
<td>B R G B I R G B</td>
<td>Black</td>
<td>White</td>
</tr>
<tr>
<td>B 0 0 0 1 1 1 1</td>
<td>Black</td>
<td>White</td>
</tr>
<tr>
<td>B 1 1 1 1 0 0 0</td>
<td>White</td>
<td>Black</td>
</tr>
<tr>
<td>B 0 0 0 1 0 0 0</td>
<td>Black</td>
<td>Black</td>
</tr>
<tr>
<td>B 1 1 1 1 1 1 1</td>
<td>White</td>
<td>White</td>
</tr>
</tbody>
</table>

1-130 Color Graphics Adapter
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
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 5 4 3 2 1 0</th>
<th>C1 C0</th>
<th>C1 C0</th>
<th>C1 C0</th>
<th>C1 C0</th>
</tr>
</thead>
<tbody>
<tr>
<td>First Display PEL</td>
<td>Second Display PEL</td>
<td>Third Display PEL</td>
<td>Fourth Display PEL</td>
<td></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.

1-134  Color Graphics Adapter
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:

```
<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>
First Display PEL
Second Display PEL
Third Display PEL
Fourth Display PEL
Fifth Display PEL
Sixth Display PEL
Seventh Display PEL
Eighth Display PEL
```
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)

```
AB
X
```

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 Alphanumeric</th>
<th>80 by 25 Alphanumeric</th>
<th>Graphic Modes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>R0</td>
<td>Horizontal Total</td>
<td>Character</td>
<td>Write Only</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 Only</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 Only</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 Only</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 Only</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 Only</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 Only</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 Only</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 Only</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 Only</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 Only</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 Only</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 Only</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 Only</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*
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>Description</th>
</tr>
</thead>
</table>
| 0   | Selects B (Blue) Border Color in 40 x 25 Alphanumeric Mode  
     | Selects B (Blue) Background Color in 320 x 200 Graphics Mode  
     | Selects B (Blue) Foreground Color in 640 x 200 Graphics Mode |
| 1   | Selects G (Green) Border Color in 40 x 25 Alphanumeric Mode  
     | Selects G (Green) Background Color in 320 x 200 Graphics Mode  
     | Selects G (Green) Foreground Color in 640 x 200 Graphics Mode |
| 2   | Selects R (Red) Border Color in 40 x 25 Alphanumeric Mode  
     | Selects R (Red) Background Color in 320 x 200 Graphics Mode  
     | Selects R (Red) Foreground Color in 640 x 200 Graphics Mode |
| 3   | Selects I (Intensified) Border Color in 40 x 25 Alphanumeric Mode  
     | Selects I (Intensified) Background Color in 320 x 200 Graphics Mode  
     | Selects I (Intensified) Foreground Color in 640 x 200 Graphics Mode |
| 4   | Selects Alternate, Intensified Set of Colors in Graphics Mode  
     | Selects Background Colors in the Alphanumeric Mode |
| 5   | Selects Active Color Set in 320 x 200 Graphics Mode |
| 6   | Not Used |
| 7   | Not Used |

**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.

1-140 Color Graphics Adapter
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</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>80 x 25 Alphanumeric Mode</td>
</tr>
<tr>
<td>1</td>
<td>Graphics Select</td>
</tr>
<tr>
<td>2</td>
<td>Black/White Select</td>
</tr>
<tr>
<td>3</td>
<td>Enable Video Signal</td>
</tr>
<tr>
<td>4</td>
<td>High-Resolution (640 x 200) Black/White Mode</td>
</tr>
<tr>
<td>5</td>
<td>Change Background Intensity to Blink Bit</td>
</tr>
<tr>
<td>6</td>
<td>Not Used</td>
</tr>
<tr>
<td>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>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>z</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>z</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>z</td>
<td></td>
</tr>
</tbody>
</table>

- 40 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 = 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</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Display Enable</td>
</tr>
<tr>
<td>1</td>
<td>Light-Pen Trigger Set</td>
</tr>
<tr>
<td>2</td>
<td>Light-Pen Switch Made</td>
</tr>
<tr>
<td>3</td>
<td>Vertical Sync</td>
</tr>
<tr>
<td>4</td>
<td>Not Used</td>
</tr>
<tr>
<td>5</td>
<td>Not Used</td>
</tr>
<tr>
<td>6</td>
<td>Not Used</td>
</tr>
<tr>
<td>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.
At Standard TTL Levels

IBM Color Display or other Direct-Drive Monitor

1. Ground
2. Ground
3. Red
4. Green
5. Blue
6. Intensity
7. Reserved
8. Horizontal Drive
9. Vertical Drive

Color/Graphics Direct-Drive Adapter

Composite Phono Jack Hookup to Monitor

1. Composite Video Signal of Approximately 1.5 Volts
2. Peak to Peak Amplitude
3. Chassis Ground

Color/Graphics Composite Jack

Connector Specifications (Part 1 of 2)
Connector Specifications (Part 2 of 2)
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 1</th>
<th>Bit 0</th>
<th>Drive</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0 (A)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1 (B)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>2 (C)</td>
</tr>
<tr>
<td>1</td>
<td>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 = &quot;1&quot;, then transfer is from FDC data register to the processor. If DIO = &quot;0&quot;, 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 &quot;ready&quot; and &quot;direction&quot; to the processor.</td>
</tr>
</tbody>
</table>
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:

**Command Phase**

The FDC receives all information required to perform a particular operation from the processor.

**Execution Phase**

The FDC performs the operation it was instructed to do.

**Result Phase**

After completion of the operation, status and other housekeeping information is made available to the processor.
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)
<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)
## 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>Data Bus</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>D7 D6 D5 D4 D3 D2 D1 D0</td>
<td></td>
</tr>
<tr>
<td><strong>Read Data</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<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 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><strong>Execution</strong></td>
<td></td>
<td></td>
<td>Data transfer between the FDD and main system.</td>
</tr>
<tr>
<td><strong>Result</strong></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><strong>Read Deleted Data</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Command</td>
<td>W</td>
<td>MT MF SK 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>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><strong>Execution</strong></td>
<td></td>
<td></td>
<td>Data transfer between the FDD and main system.</td>
</tr>
<tr>
<td><strong>Result</strong></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>
</tbody>
</table>

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>Command</td>
<td>W</td>
<td>MT MF 0 0 0 1 0 1</td>
<td>Write Data</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>X X X X X X HD US1 US0</td>
<td>Command Codes</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>C H R N</td>
<td>Sector ID information to command execution.</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>EOT GPL DTL</td>
<td>Data transfer between the main system and FDD.</td>
</tr>
<tr>
<td>Execution</td>
<td></td>
<td></td>
<td></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>Sector ID information after command execution.</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>ST 2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>C H R N</td>
<td></td>
</tr>
</tbody>
</table>

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

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

| **Scan High or Equal** |     |    |    |    |    |    |    |    |    | Command Codes                |
|                       | W   | MT | MF | SK | 1  | 1  | 1  | 0  | 1  |                              |
|                       | W   | X  | X  | X  | X  | X  | X  | HD | US1 | US0                          |
|                       | W   |     |    |    | C  | 1  | 0  | 0  | 1  | Sector ID information        |
|                       | W   |     |    |    | H  |    |    |    |    | prior to command execution. |
|                       | W   |     |    |    | R  |    |    |    |    |                              |
|                       | W   |     |    |    | N  |    |    |    |    |                              |
|                       | W   |     |    |    | EOT|    |    |    |    |                              |
|                       | W   |     |    |    | GPL|    |    |    |    |                              |
|                       | W   |     |    |    | STP|    |    |    |    | Data compared between the FDD |
|                       |     |    |    |    |    |    |    |    |    | and main system.             |
|                       |     |    |    |    |    |    |    |    |    |                              |
| **Result**           | R   | ST | 0  |    |    |    |    |    |    | Status information           |
|                       | R   | ST | 1  |    |    |    |    |    |    | after command execution.    |
|                       | R   | ST | 2  |    |    |    |    |    |    | Sector ID information        |
|                       | R   |    | C  |    |    |    |    |    |    | after command execution.    |
|                       | R   |    | H  |    |    |    |    |    |    |                              |
|                       | R   |    | R  |    |    |    |    |    |    |                              |
|                       | R   |    | N  |    |    |    |    |    |    |                              |

1-162 Diskette Adapter
<table>
<thead>
<tr>
<th>Phase</th>
<th>R/W</th>
<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>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command</td>
<td>W</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td>Command Codes</td>
</tr>
<tr>
<td>Execution</td>
<td>W</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>US1</td>
<td>US0</td>
<td></td>
</tr>
<tr>
<td>No Result Phase</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Head retracted to track 0</td>
</tr>
<tr>
<td>Recalibrate</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Command Codes</td>
</tr>
<tr>
<td>Result</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Head retracted to track 0</td>
</tr>
</tbody>
</table>

| Command | W   | 0  | 0  | 0  | 0  | 1  | 0  | 0  |    | Command Codes                                    |
| Result   | R   |    |    |    |    |    |    |    |    | Status information at the end of seek operation about the FDC |
|          | R   |    |    |    |    |    |    |    |    |                                                  |
| Sense Interrupt Status |     |    |    |    |    |    |    |    |    | Command Codes                                    |
|          |     |    |    |    |    |    |    |    |    | Status information at the end of seek operation about the FDC |

| Command | W   | 0  | 0  | 0  | 0  | 0  | 1  | 1  |    | Command Codes                                    |
| No Result | W  |    |    |    |    |    |    |    |    |                                                  |
| Phase     | W   |    |    |    |    |    |    |    |    | Command Codes                                    |
| Specifying | W  |    |    |    |    |    |    |    |    |                                                  |
| W         | W   | SRT|     |     | HUT|     |     |     |     | Command Codes                                    |
| W         | W   | HLT|     |     |     |     |     |     |     |                                                  |
| W         | W   |    |     |     |     |     |     |     |     | Command Codes                                    |

| Command | W   | 0  | 0  | 0  | 0  | 1  | 0  | 0  |    | Command Codes                                    |
| Result   | W   | X  | X  | X  | X  | X  | HD US1 US0 | ST 3 | |
|          | R   |    |    |    |    |    |    |    |    |                                                  |
| Sense Drive Status |     |    |    |    |    |    |    |    |    | Command Codes                                    |
|          |     |    |    |    |    |    |    |    |    | Status information about FDD.                    |

| Command | W   | 0  | 0  | 0  | 0  | 1  | 1  | 1  |    | Command Codes                                    |
| Execution | W  |    |    |    |    |    |    |    |    |                                                  |
| No Result | W  |    |    |    |    |    |    |    |    | Command Codes                                    |
| Phase     | W   |    |    |    |    |    |    |    |    |                                                  |
| Seek      | W   |    |    |    |    |    |    |    |    | Command Codes                                    |
| W         | W   | X  | X  | X  | X  | HD US1 US0 | NCN |    |                                               |
| W         | W   |    |    |    |    |    |    |    |    | Command Codes                                    |
| No Result | W   |    |    |    |    |    |    |    |    | Command Codes                                    |
| Phase     | W   |    |    |    |    |    |    |    |    |                                               |
| Invalid   | W   |    |    |    |    |    |    |    |    | Command Codes                                    |
| Invalid Codes |     |    |    |    |    |    |    |    |    |                                                  |

| Command | W   |    |    |    |    |    |    |    |    | Invalid command codes (NoOp - FDC goes into standby state). |
| Result   | R   |    |    |    |    |    |    |    |    | ST 0 = 80.                                        |

Diskette Adapter  1-163
<table>
<thead>
<tr>
<th>Bit No.</th>
<th>Name</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
</table>
| D7     | Interrupt Code | IC | D7 = 0 and D6 = 0  
Normal termination of command (NT). Command was completed and properly executed. |
|        |       |        | D7 = 0 and D6 = 1  
Abnormal termination of command (AT). Execution of command was started, but was not successfully completed. |
|        |       |        | D7 = 1 and D6 = 0  
Invalid command issue (IC). Command that was issued was never started. |
|        |       |        | D7 = 1 and D6 = 1  
Abnormal termination because, during command execution, the ready signal from FDD changed state. |
| D5     | Seek End | SE | When the FDC completes the seek command, this flag is set to 1 (high). |
| D4     | Equipment Check | EC | If a fault signal is received from the FDD, or if the track 0 signal fails to occur after 77 step pulses (recalibrate command), then this flag is set. |
| D3     | Not Ready | NR | When the FDD is in the not-ready state and a read or write command is issued, this flag is set. If a read or write command is issued to side 1 of a single-sided drive, then this flag is set. |
| D2     | Head Address | HD | This flag is used to indicate the state of the head at interrupt. |
| D1     | Unit Select 1 | US 1 | These flags are used to indicate a drive unit number at interrupt. |
| D0     | Unit Select 0 | US 0 | |

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. During execution of the read ID command, if the FDC cannot read the ID field without an error, then this flag is set. During the execution of the read a cylinder command, if the starting sector cannot be found, then 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 0</th>
<th>Drive</th>
<th>00: DR #A</th>
<th>10: DR #C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Select</td>
<td>01: DR #B</td>
<td>11: DR #D</td>
</tr>
<tr>
<td>2</td>
<td>Not FDC Reset</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Enable INT &amp; DMA Requests</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Drive A Motor Enable</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Drive B Motor Enable</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Drive C Motor Enable</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Drive D Motor Enable</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:

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

All adapter outputs are driven by 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

- Index
  The selected drive supplies one pulse per diskette revolution on this line.

- Write Protect
  The selected drive makes this line active if a write-protected diskette is mounted in the drive.

- Track 0
  The selected drive makes this line active if the read/write head is over track 0.

- Read Data
  The selected drive supplies a pulse on this line for each flux change encountered on the diskette.
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)
## Rear Panel

![Rear Panel Diagram](image)

## Connector Specifications (Part 2 of 2)

<table>
<thead>
<tr>
<th>Pin</th>
<th>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>

---

**1-174 Diskette Adapter**
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 (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 (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**
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.

Recording Medium
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>0</td>
<td>e</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></td>
<td>Cylinder Low</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</td>
<td>5 4 3 2 1 0</td>
<td></td>
</tr>
<tr>
<td>0 0</td>
<td>0 0 0 0</td>
<td>The controller did not detect any error during the execution of the previous operation.</td>
</tr>
<tr>
<td>0 0</td>
<td>0 0 0 1</td>
<td>The controller did not detect an index signal from the drive.</td>
</tr>
<tr>
<td>0 0</td>
<td>0 0 1 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>
</tr>
<tr>
<td>0 0</td>
<td>0 0 1 1</td>
<td>The controller detected a write fault from the drive during the last operation.</td>
</tr>
<tr>
<td>0 0</td>
<td>0 1 0 0</td>
<td>After the controller selected the drive, the drive did not respond with a ready signal.</td>
</tr>
<tr>
<td>0 0</td>
<td>0 1 0 1</td>
<td>Not used.</td>
</tr>
<tr>
<td>0 0</td>
<td>0 1 1 0</td>
<td>After stepping the maximum number of cylinders, the controller did not receive the track 00 signal from the drive.</td>
</tr>
<tr>
<td>0 0</td>
<td>0 1 1 1</td>
<td>Not used.</td>
</tr>
<tr>
<td>0 0</td>
<td>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>
</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 0 0 0 0</td>
<td>ID Read Error: The controller detected an ECC error in the target ID field on the disk.</td>
<td></td>
</tr>
<tr>
<td>0 1 0 0 0 1</td>
<td>Data Error: The controller detected an uncorrectable ECC error in the target sector during a read operation.</td>
<td></td>
</tr>
<tr>
<td>0 1 0 0 1 0</td>
<td>Address Mark: The controller did not detect the target address mark (AM) on the disk.</td>
<td></td>
</tr>
<tr>
<td>0 1 0 0 1 1</td>
<td>Not used.</td>
<td></td>
</tr>
<tr>
<td>0 1 0 1 0 0</td>
<td>Sector Not Found: The controller found the correct cylinder and head, but not the target sector.</td>
<td></td>
</tr>
<tr>
<td>0 1 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>
<td></td>
</tr>
<tr>
<td>0 1 0 1 1 0</td>
<td>Not used.</td>
<td></td>
</tr>
<tr>
<td>0 1 0 1 1 1</td>
<td>Not used.</td>
<td></td>
</tr>
<tr>
<td>0 1 1 0 0 0</td>
<td>Correctable Data Error: The controller detected a correctable ECC error in the target field.</td>
<td></td>
</tr>
<tr>
<td>0 1 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>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>Error Type</td>
<td>Error Code</td>
</tr>
<tr>
<td>-----------------------------------------------------------------------------</td>
<td>------------</td>
<td>------------</td>
</tr>
<tr>
<td>Invalid Command: The controller has received an invalid command from the system unit.</td>
<td>1 0 0 0 0</td>
<td>0 0 0 0 0</td>
</tr>
<tr>
<td>Illegal Disk Address: The controller detected an address that is beyond the maximum range.</td>
<td>1 0 0 0 1</td>
<td>0 0 0 1 0</td>
</tr>
<tr>
<td>RAM Error: The controller detected a data error during the RAM sector-buffer diagnostic test.</td>
<td>1 1 0 0 0 0</td>
<td>0 0 0 0 0</td>
</tr>
<tr>
<td>Program Memory Checksum Error: During this internal diagnostic test, the controller detected a program-memory checksum error.</td>
<td>1 1 0 0 0 1</td>
<td>0 0 0 1 0</td>
</tr>
<tr>
<td>ECC Polynominal Error: During the controller's internal diagnostic tests, the hardware ECC generator failed its test.</td>
<td>1 1 0 0 1 0</td>
<td>0 0 1 0 0</td>
</tr>
</tbody>
</table>

1-184 Fixed Disk Adapter
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 0 d</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 6 5 4 3 2 1 0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>r a 0 0 0 s s s</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>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td>This drive is not specified and defaults to 3 milliseconds per step.</td>
</tr>
<tr>
<td>0 0 1</td>
<td>N/A</td>
</tr>
<tr>
<td>0 1 0</td>
<td>N/A</td>
</tr>
<tr>
<td>0 1 1</td>
<td>N/A</td>
</tr>
<tr>
<td>1 0 0</td>
<td>200 microseconds per step.</td>
</tr>
<tr>
<td>1 0 1</td>
<td>70 microseconds per step (specified by BIOS).</td>
</tr>
<tr>
<td>1 1 0</td>
<td>3 milliseconds per step.</td>
</tr>
<tr>
<td>1 1 1</td>
<td>3 milliseconds per step.</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</td>
<td>d = drive (0 or 1)</td>
</tr>
<tr>
<td>Ready</td>
<td>Byte 0 0 0 0 0 0 0 0</td>
<td>x = don’t care</td>
</tr>
<tr>
<td>(Class 0,</td>
<td>Byte 1 0 0 d x x x x x</td>
<td>Bytes 2, 3, 4, 5 = don’t care</td>
</tr>
<tr>
<td>Opcode 00)</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</td>
<td>d = drive (0 or 1)</td>
</tr>
<tr>
<td>(Class 0,</td>
<td>Byte 0 0 0 0 0 0 0 1</td>
<td>x = don’t care</td>
</tr>
<tr>
<td>Opcode 01)</td>
<td>Byte 1 0 0 d x x x x x</td>
<td>r = retries</td>
</tr>
<tr>
<td></td>
<td>Byte 5 r 0 0 0 0 s s s</td>
<td>s = Step Option</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bytes 2, 3, 4 = don’t care</td>
</tr>
<tr>
<td>Reserved</td>
<td>Bit 7 6 5 4 3 2 1 0</td>
<td></td>
</tr>
<tr>
<td>(Class 0,</td>
<td>Byte 0 0 0 0 0 0 0 1</td>
<td>This Opcode is not used.</td>
</tr>
<tr>
<td>Opcode 02)</td>
<td>Byte 1 0 0 d x x x x x</td>
<td></td>
</tr>
<tr>
<td>Request Sense</td>
<td>Bit 7 6 5 4 3 2 1 0</td>
<td>d = drive (0 or 1)</td>
</tr>
<tr>
<td>Status</td>
<td>Byte 0 0 0 0 0 0 0 1</td>
<td>x = don’t care</td>
</tr>
<tr>
<td>(Class 0,</td>
<td>Byte 1 0 0 d x x x x x</td>
<td>Bytes 2, 3, 4, 5 = don’t care</td>
</tr>
<tr>
<td>Opcode 03)</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</td>
<td>d = drive (0 or 1)</td>
</tr>
<tr>
<td>(Class 0,</td>
<td>Byte 0 0 0 0 0 1 0 0</td>
<td>r = retries</td>
</tr>
<tr>
<td>Opcode 04)</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</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 0 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</td>
<td>Interleave: 1 to 16 for 512-byte sectors</td>
</tr>
<tr>
<td>(Class 0,</td>
<td>Byte 0 0 0 0 0 1 0 1</td>
<td></td>
</tr>
<tr>
<td>Opcode 05)</td>
<td>Byte 1 0 0 d Head Number</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 2 ch Sector Number</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>
</tbody>
</table>

Fixed Disk Adapter 1-187
<table>
<thead>
<tr>
<th>Command</th>
<th>Data Control Block</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Format Track</strong>&lt;br&gt;(Class 0, Opcode 06)</td>
<td>Bit: 7 6 5 4 3 2 1 0&lt;br&gt;Byte 0: 0 0 0 0 1 1 0&lt;br&gt;Byte 1: 0 0 d Head Number&lt;br&gt;Byte 2: ch 0 0 0 0 0 0&lt;br&gt;Byte 3: Cylinder Low&lt;br&gt;Byte 4: 0 0 0 Interleave&lt;br&gt;Byte 5: r 0 0 0 0 s s s</td>
<td>d = drive (0 or 1)&lt;br&gt;r = retries&lt;br&gt;s = step option&lt;br&gt;ch = cylinder high&lt;br&gt;Interleave: 1 to 16 for 512-byte sectors</td>
</tr>
<tr>
<td><strong>Format Bad Track</strong>&lt;br&gt;(Class 0, Opcode 07)</td>
<td>Bit: 7 6 5 4 3 2 1 0&lt;br&gt;Byte 0: 0 0 0 0 1 1 1&lt;br&gt;Byte 1: 0 0 d Head Number&lt;br&gt;Byte 2: ch 0 0 0 0 0 0&lt;br&gt;Byte 3: Cylinder Low&lt;br&gt;Byte 4: 0 0 0 Interleave&lt;br&gt;Byte 5: r 0 0 0 0 s s s</td>
<td>d = drive (0 or 1)&lt;br&gt;r = retries&lt;br&gt;s = step option&lt;br&gt;ch = cylinder high&lt;br&gt;Interleave: 1 to 16 for 512-byte sectors</td>
</tr>
<tr>
<td><strong>Read</strong>&lt;br&gt;(Class 0, Opcode 08)</td>
<td>Bit: 7 6 5 4 3 2 1 0&lt;br&gt;Byte 0: 0 0 0 0 1 0 0 0&lt;br&gt;Byte 1: 0 0 d Head Number&lt;br&gt;Byte 2: ch Sector Number&lt;br&gt;Byte 3: Cylinder Low&lt;br&gt;Byte 4: Sector Number&lt;br&gt;Byte 5: r a 0 0 0 s s s</td>
<td>d = drive (0 or 1)&lt;br&gt;r = retries&lt;br&gt;a = retry option on&lt;br&gt;data ECC error&lt;br&gt;s = step option&lt;br&gt;ch = cylinder high</td>
</tr>
<tr>
<td><strong>Reserved</strong>&lt;br&gt;(Class 0, Opcode 09)</td>
<td></td>
<td>This Opcode is not used</td>
</tr>
<tr>
<td><strong>Write</strong>&lt;br&gt;(Class 0, Opcode 0A)</td>
<td>Bit: 7 6 5 4 3 2 1 0&lt;br&gt;Byte 0: 0 0 0 0 1 0 1 0&lt;br&gt;Byte 1: 0 0 d Head Number&lt;br&gt;Byte 2: ch Sector Number&lt;br&gt;Byte 3: Cylinder Low&lt;br&gt;Byte 4: Block Count&lt;br&gt;Byte 5: r 0 0 0 0 s s s</td>
<td>d = drive (0 or 1)&lt;br&gt;r = retries&lt;br&gt;s = step option&lt;br&gt;ch = cylinder high</td>
</tr>
<tr>
<td><strong>Seek</strong>&lt;br&gt;(Class 0, Opcode 0B)</td>
<td>Bit: 7 6 5 4 3 2 1 0&lt;br&gt;Byte 0: 0 0 0 0 1 0 1 1&lt;br&gt;Byte 1: 0 0 d Head Number&lt;br&gt;Byte 2: ch 0 0 0 0 0 0&lt;br&gt;Byte 3: Cylinder Low&lt;br&gt;Byte 4: x x x x x x x&lt;br&gt;Byte 5: r 0 0 0 0 s s s</td>
<td>d = drive (0 or 1)&lt;br&gt;r = retries&lt;br&gt;s = step option&lt;br&gt;x = don't care&lt;br&gt;ch = cylinder high</td>
</tr>
</tbody>
</table>

1-188  Fixed Disk Adapter
<table>
<thead>
<tr>
<th>Command</th>
<th>Data Control Block</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialize Drive</td>
<td>Bit</td>
<td>Bytes 1, 2, 3, 4, 5 = don't care</td>
</tr>
<tr>
<td>Characteristics*</td>
<td>Byte 0</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</td>
<td>Bytes 1, 2, 3, 4, 5 = don't care</td>
</tr>
<tr>
<td>(Class 0, Opcode 0D)</td>
<td>Byte 0</td>
<td></td>
</tr>
<tr>
<td>Read Data from Sector Buffer</td>
<td>Bit</td>
<td>Bytes 1, 2, 3, 4, 5 = don't care</td>
</tr>
<tr>
<td>(Class 0, Opcode 0E)</td>
<td>Byte 0</td>
<td></td>
</tr>
<tr>
<td>Write Data to Sector Buffer</td>
<td>Bit</td>
<td>Bytes 1, 2, 3, 4, 5 = don't care</td>
</tr>
<tr>
<td>(Class 0, Opcode 0F)</td>
<td>Byte 0</td>
<td></td>
</tr>
<tr>
<td>RAM Diagnostic</td>
<td>Bit</td>
<td>Bytes 1, 2, 3, 4, 5 = don't care</td>
</tr>
<tr>
<td>(Class 7, Opcode 00)</td>
<td>Byte 0</td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td>Bit</td>
<td>This Opcode is not used</td>
</tr>
<tr>
<td>(Class 7, Opcode 01)</td>
<td>Byte 0</td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td>Bit</td>
<td>This Opcode is not used</td>
</tr>
<tr>
<td>(Class 7, Opcode 02)</td>
<td>Byte 0</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 Diagnostic</td>
<td></td>
<td>d = drive (0 or 1)</td>
</tr>
<tr>
<td>(Class 7, Opcode 03)</td>
<td></td>
<td>s = step option</td>
</tr>
<tr>
<td></td>
<td>Bit</td>
<td>r = retries</td>
</tr>
<tr>
<td></td>
<td>Byte 0</td>
<td>x = don’t care</td>
</tr>
<tr>
<td></td>
<td>Byte 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 5</td>
<td></td>
</tr>
<tr>
<td>Controller Internal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Diagnostics</td>
<td></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*</td>
<td></td>
<td>d = (0 or 1)</td>
</tr>
<tr>
<td>(Class 7, Opcode 05)</td>
<td></td>
<td>s = step option</td>
</tr>
<tr>
<td></td>
<td>Bit</td>
<td>r = retries</td>
</tr>
<tr>
<td></td>
<td>Byte 0</td>
<td>ch = cylinder high</td>
</tr>
<tr>
<td></td>
<td>Byte 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 5</td>
<td></td>
</tr>
<tr>
<td>Write Long**</td>
<td></td>
<td>d = drive (0 or 1)</td>
</tr>
<tr>
<td>(Class 7, Opcode 06)</td>
<td></td>
<td>s = step option</td>
</tr>
<tr>
<td></td>
<td>Bit</td>
<td>r = retries</td>
</tr>
<tr>
<td></td>
<td>Byte 0</td>
<td>ch = cylinder high</td>
</tr>
<tr>
<td></td>
<td>Byte 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 5</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></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></td>
<td>321</td>
<td>Controller reset.</td>
</tr>
<tr>
<td>Read</td>
<td>322</td>
<td>Reserved.</td>
</tr>
<tr>
<td></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></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).

I-192  Fixed Disk Adapter
## Disk Drive Connector J1

<table>
<thead>
<tr>
<th>Signal Description</th>
<th>Pin Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ground - 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 2^0</td>
<td>14</td>
</tr>
<tr>
<td>Head Select 2^1</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)

**Pin Number 1** Position 5 has No Pin (for Cable Orientation)

## Disk Drive Connector J2 or J3

<table>
<thead>
<tr>
<th>Signal Description</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 Read Data</td>
<td>15</td>
</tr>
<tr>
<td>-MFM Read Data</td>
<td>17</td>
</tr>
<tr>
<td>Ground</td>
<td>19</td>
</tr>
</tbody>
</table>

**Pin Number 2** Position 5 has No Pin (for Cable Orientation)

*Fixed Disk Adapter Interface Specifications*
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>Specification</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Media</td>
<td>Rigid media disk</td>
</tr>
<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 10⁴ bits read</td>
</tr>
<tr>
<td>Hard Read Errors</td>
<td>1 per 10¹² bits read</td>
</tr>
<tr>
<td>Seek Errors</td>
<td>1 per 10⁶ 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 ±5% 1.8 A (4.5 A maximum)</td>
</tr>
<tr>
<td></td>
<td>+5 Vdc ±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**
IBM Memory Expansion Options

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 or 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>Memory</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

Memory Expansion Options  1-199
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.”
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

64K \( \frac{\text{xxxK}}{\text{}} \)

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 joy
stick 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 joy stick 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.
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-204  Game Control Adapter
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 CH CK: 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>

1-206 Game Control Adapter
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</td>
<td>C</td>
<td>B</td>
<td>A</td>
<td>D</td>
<td>C</td>
<td>B</td>
<td>A</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>

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.

Joy Stick Schematic Diagram
### 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>+5 Vdc</td>
<td>9</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>

### External Devices

- Button 6
- Position 2
- Ground
- Position 3
- Button 7
- +5 Vdc

### Game Control Adapter

- 15-Pin D-Shell Connector

### 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

Bit 0-7 Data Bus

I/O Read/Write
Memory Read/Write
Spare-E18
Address Bit 0
Address Bit 2
Address Bit 3
Address Bit 9
Address Enable

Command and Address Buffer

Address Buffer

Transceiver

Buffered Address Lines

Data Select Logic

Bus Direction

Buffered Data Bus

Prototype Card Block Diagram
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 hole 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

1-212 Prototype Card
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.

Component Side
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.
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

<table>
<thead>
<tr>
<th>Hex Address 3F8 to 3FF and 2F8 to 2FF</th>
</tr>
</thead>
<tbody>
<tr>
<td>A9</td>
</tr>
<tr>
<td>----</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></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:

```
<table>
<thead>
<tr>
<th>D0</th>
<th>D1</th>
<th>D2</th>
<th>D3</th>
<th>D4</th>
<th>D5</th>
<th>D6</th>
<th>D7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start Bit</td>
<td>Data Marking Bit</td>
<td>Data Bit</td>
<td>Parity Bit</td>
<td>Stop Bit</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

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 Vdc

On Function

+3 Vdc

0 Vdc

Invalid Levels

-3 Vdc

Off Function

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

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

For interface control circuits, the function is “on” when the voltage is more positive than \(+3\) Vdc with respect to signal ground and is “off” when the voltage is more negative than \(-3\) Vdc with respect to signal ground.
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>0</td>
<td>1</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>0</td>
<td>1</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, OUT 1, OUT 2, 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, 3-7 Permanent</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>SOUT</td>
<td>Master Reset</td>
<td>Bits 4-7 - Input Signal</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

1-226 Asynchronous Adapter
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>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.

![Hex Address 3F8 DLAB = 1 Diagram]

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

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</th>
<th>Percent Error Difference Between Desired and Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(Decimal) (Hex)</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>2304 (900)</td>
<td>—</td>
</tr>
<tr>
<td>75</td>
<td>1536 (600)</td>
<td>—</td>
</tr>
<tr>
<td>110</td>
<td>1047 (417)</td>
<td>0.026</td>
</tr>
<tr>
<td>134.5</td>
<td>857 (359)</td>
<td>0.058</td>
</tr>
<tr>
<td>150</td>
<td>768 (300)</td>
<td>—</td>
</tr>
<tr>
<td>300</td>
<td>384 (180)</td>
<td>—</td>
</tr>
<tr>
<td>600</td>
<td>192 (0C0)</td>
<td>—</td>
</tr>
<tr>
<td>1200</td>
<td>96 (060)</td>
<td>—</td>
</tr>
<tr>
<td>1800</td>
<td>64 (040)</td>
<td>—</td>
</tr>
<tr>
<td>2000</td>
<td>58 (03A)</td>
<td>0.69</td>
</tr>
<tr>
<td>2400</td>
<td>48 (030)</td>
<td>—</td>
</tr>
<tr>
<td>3600</td>
<td>32 (020)</td>
<td>—</td>
</tr>
<tr>
<td>4800</td>
<td>24 (018)</td>
<td>—</td>
</tr>
<tr>
<td>7200</td>
<td>16 (010)</td>
<td>—</td>
</tr>
<tr>
<td>9600</td>
<td>12 (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:

<table>
<thead>
<tr>
<th>Hex Address 3FD</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>Data Ready (DR)</td>
<td>Overrun Error (OR)</td>
<td>Parity Error (PE)</td>
<td>Framing Error (FE)</td>
<td>Break Interrupt (BI)</td>
<td>Transmitter Holding Register Empty (THRE)</td>
<td>Tx Shift Register Empty (TSRE)</td>
<td>= 0</td>
</tr>
</tbody>
</table>

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.

**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.
<table>
<thead>
<tr>
<th>Interrupt ID Register</th>
<th>Interrupt Set and Reset Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 2 Bit 1 Bit 0</td>
<td>Priority Level</td>
</tr>
<tr>
<td>0 0 1</td>
<td>Highest</td>
</tr>
<tr>
<td>1 1 0</td>
<td>Second</td>
</tr>
<tr>
<td>1 0 0</td>
<td>Third</td>
</tr>
<tr>
<td>0 0 0</td>
<td>Fourth</td>
</tr>
</tbody>
</table>

**Interrupt Control Functions**

I-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:

![Interrupt Enable Register (IER)](image)

**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 3FC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 7 6 5 4 3 2 1 0</td>
</tr>
<tr>
<td>DTR   RTS   Out 1 Out 2 Loop = 0 = 0 = 0</td>
</tr>
</tbody>
</table>

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 (\(\text{OUT}_1\)) signal, which is an auxiliary user-designated output. Bit 2 affects the \(\text{OUT}_1\) output in a manner identical to that described above for bit 0.

Bit 3: This bit controls the output 2 (\(\text{OUT}_2\)) signal, which is an auxiliary user-designated output. Bit 3 affects the \(\text{OUT}_2\) 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, \(\text{OUT}_1\), and \(\text{OUT}_2\)) 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>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>7</td>
<td>Delta Clear to Send (DCTS)</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>Delta Data Set Ready (DDSR)</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>Trailing Edge Ring Indicator (TERI)</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>Delta Rx Line Signal Detect (DRLSD)</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Clear to Send (CTS)</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Data Set Ready (DSR)</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Ring Indicator (RI)</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>Receive Line Signal Detect (RLSD)</td>
</tr>
</tbody>
</table>

Modem Status Register (MSR)
Bit 0: This bit is the delta clear to send (DCTS) indicator. Bit 0 indicates that the $\overline{CTS}$ 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 $\overline{DRS}$ 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 $\overline{RI}$ 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 $\overline{RLSD}$ 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 ($\overline{CTS}$) 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 ($\overline{DSR}$) 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 ($\overline{RI}$) 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 ($\overline{RLSD}$) 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:

Hex Address 3F8  DLAB = 0  Read Only

<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>
<th>Data Bit 0</th>
<th>Data Bit 1</th>
<th>Data Bit 2</th>
<th>Data Bit 3</th>
<th>Data Bit 4</th>
<th>Data Bit 5</th>
<th>Data Bit 6</th>
<th>Data Bit 7</th>
</tr>
</thead>
</table>

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

Module Position for Primary Asynchronous Adapter

Module Position for Alternate Asynchronous Adapter

Asynchronous Communications Adapter

Current Loop Interface Dot Down

Shunt Module Socket

Voltage Interface Dot Up

1-242 Asynchronous Adapter
**Description** | **Pin**
--- | ---
NC | 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
+Transmit Current Loop Data | 9
NC | 10
-Transmit Current Loop Data | 11
NC | 12
NC | 13
NC | 14
NC | 15
NC | 16
NC | 17
+Receive Current Loop Data | 18
NC | 19
Data Terminal Ready | 20
NC | 21
Ring Indicator | 22
NC | 23
NC | 24
-Receive Current Loop Return | 25

**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**
Notes:
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:

8251A Block Diagram

1-246  BSC Adapter
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.
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.

1-252  BSC Adapter
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 D6 D5 D4 D3 D2 D1 D0</td>
<td>SC1 SC0 RL1 RL0 M2 M1 M0 BCD</td>
</tr>
</tbody>
</table>

**Definition of Control**

**SC — Select Counter:**

<table>
<thead>
<tr>
<th>SC1</th>
<th>SC0</th>
<th>SC0 - Select Counter</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>RL1 RL0 - Counter Latching operation, Read/Load</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>
<th>Terminal Count Interrupt</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Mode 0</td>
</tr>
</tbody>
</table>

**BCD:**

<table>
<thead>
<tr>
<th>BCD</th>
<th>Binary Counter 16-bits</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 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.

```
<table>
<thead>
<tr>
<th></th>
<th>Mode Instruction 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>3A9</td>
<td>C/D = 1</td>
</tr>
<tr>
<td>3A9</td>
<td>C/D = 1</td>
</tr>
<tr>
<td>3A9</td>
<td>C/D = 1</td>
</tr>
<tr>
<td>3A9</td>
<td>C/D = 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>SYNC Character 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>3A9</td>
<td>C/D = 1</td>
</tr>
<tr>
<td>3A9</td>
<td>C/D = 1</td>
</tr>
<tr>
<td>3A9</td>
<td>C/D = 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>SYNC Character 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>3A9</td>
<td>C/D = 1</td>
</tr>
<tr>
<td>3A9</td>
<td>C/D = 1</td>
</tr>
<tr>
<td>3A9</td>
<td>C/D = 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Command Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>3A9</td>
<td>C/D = 1</td>
</tr>
<tr>
<td>3A9</td>
<td>C/D = 1</td>
</tr>
<tr>
<td>3A9</td>
<td>C/D = 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>3A8</td>
<td>C/D = 0</td>
</tr>
<tr>
<td>3A9</td>
<td>C/D = 1</td>
</tr>
<tr>
<td>3A8</td>
<td>C/D = 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Command Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>3A9</td>
<td>C/D = 1</td>
</tr>
<tr>
<td>3A9</td>
<td>C/D = 1</td>
</tr>
<tr>
<td>3A9</td>
<td>C/D = 1</td>
</tr>
</tbody>
</table>

Typical Data Block

1-256  BSC Adapter
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>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Hex 389 for Alternate BSC</td>
</tr>
<tr>
<td>Bit 7 6 5 4 3 2 1 0</td>
<td></td>
</tr>
<tr>
<td>Bit 0</td>
<td>Not used; always 0</td>
</tr>
<tr>
<td>Bit 1</td>
<td>Not used; always 0</td>
</tr>
<tr>
<td>Bit 2 and Bit 3</td>
<td>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.</td>
</tr>
<tr>
<td>Bit 4</td>
<td>In the synchronous mode, parity is enabled from this bit. A 1 on this bit sets parity enable.</td>
</tr>
<tr>
<td>Bit 5</td>
<td>The parity generation/check is set from this bit. For BSC, even parity is used by having bit 5 = 1.</td>
</tr>
<tr>
<td>Bit 6</td>
<td>External synchronization is set by this bit. A 1 on this bit establishes synchronization detection as an input.</td>
</tr>
<tr>
<td>Bit 7</td>
<td>This bit establishes the mode of character synchronization. A 0 is set on this bit to give double character synchronization.</td>
</tr>
</tbody>
</table>
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>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>Transmit Enable</td>
</tr>
<tr>
<td></td>
<td>Data Terminal Ready</td>
</tr>
<tr>
<td></td>
<td>Receive Enable</td>
</tr>
<tr>
<td></td>
<td>Send Break Character</td>
</tr>
<tr>
<td></td>
<td>Error Reset</td>
</tr>
<tr>
<td></td>
<td>Request to Send</td>
</tr>
<tr>
<td></td>
<td>Internal Reset</td>
</tr>
<tr>
<td></td>
<td>Enter Hunt Mode</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.
<table>
<thead>
<tr>
<th>Bit 0</th>
<th>See the Note in the preceding figure.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 1</td>
<td>An output on this bit means a character is ready to be received by the computer’s 8088 microprocessor.</td>
</tr>
<tr>
<td>Bit 2</td>
<td>A 1 on this bit indicates the 8251A has no characters to transmit.</td>
</tr>
<tr>
<td>Bit 3</td>
<td>The Parity Error bit sets a flag when errors are detected. It is reset by the error reset in the command instruction.</td>
</tr>
<tr>
<td>Bit 4</td>
<td>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.</td>
</tr>
<tr>
<td>Bit 5</td>
<td>Not used</td>
</tr>
<tr>
<td>Bit 6</td>
<td>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.</td>
</tr>
<tr>
<td>Bit 7</td>
<td>The Data Set Ready bit is a one bit inverting input. It is used to check modem conditions, such as data-set ready.</td>
</tr>
</tbody>
</table>

**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.
### Interface Voltage Levels

#### Driver

<table>
<thead>
<tr>
<th>Voltage Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+15 Vdc</td>
<td>Active/Data = 0</td>
</tr>
<tr>
<td>+5 Vdc</td>
<td>Invalid Level</td>
</tr>
<tr>
<td>+5 Vdc</td>
<td>Invalid Level</td>
</tr>
<tr>
<td>-5 Vdc</td>
<td>Inactive/Data = 1</td>
</tr>
<tr>
<td>-5 Vdc</td>
<td>Inactive/Data = 1</td>
</tr>
<tr>
<td>-15 Vdc</td>
<td>Inactive/Data = 1</td>
</tr>
</tbody>
</table>

#### Receiver

<table>
<thead>
<tr>
<th>Voltage Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+25 Vdc</td>
<td>Active/Data = 0</td>
</tr>
<tr>
<td>+3 Vdc</td>
<td>Invalid Level</td>
</tr>
<tr>
<td>+3 Vdc</td>
<td>Invalid Level</td>
</tr>
<tr>
<td>-3 Vdc</td>
<td>Inactive/Data = 1</td>
</tr>
<tr>
<td>-3 Vdc</td>
<td>Inactive/Data = 1</td>
</tr>
<tr>
<td>-25 Vdc</td>
<td>Inactive/Data = 1</td>
</tr>
</tbody>
</table>

**EIA RS232C/CCITT V24-V28 Signal 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>3A5</td>
<td>385</td>
<td>8253</td>
<td>Counter 0 MSB</td>
</tr>
<tr>
<td>3A6</td>
<td>386</td>
<td>8253</td>
<td>Counter 1 LSB</td>
</tr>
<tr>
<td>3A7</td>
<td>387</td>
<td>8253</td>
<td>Counter 1 MSB</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>
</tbody>
</table>

Device Address Summary
Rear Panel

25-Pin D-Shell Connector

Pin

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

Signal Name — Description

No Connection
Transmitted Data
Received Data
Request to Send
Clear to Send
Data Set Ready
Signal Ground
Received Line Signal Detector
No Connection
No Connection
Select Standby*
No Connection
No Connection
Transmitter Signal Element Timing
No Connection
Receiver Signal Element Timing
Test (IBM Modems Only)*
No Connection
Data Terminal Ready
No Connection
Ring Indicator
Data Signal Rate Selector
No Connection
Test Indicate (IBM Modems Only)*

*Not standardized by EIA (Electronics Industry Association).

Binary Synchronous Communications Adapter

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  Commands and/or parameters for the required operation are issued by the processor.

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  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.
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 A1</td>
<td>CS WR RD</td>
<td></td>
</tr>
<tr>
<td>0 0</td>
<td>0 0 1</td>
<td>Command</td>
</tr>
<tr>
<td>0 0</td>
<td>0 1 0</td>
<td>Status</td>
</tr>
<tr>
<td>0 1</td>
<td>0 0 1</td>
<td>Parameter</td>
</tr>
<tr>
<td>0 1</td>
<td>0 1 0</td>
<td>Result</td>
</tr>
<tr>
<td>1 0</td>
<td>0 0 1</td>
<td>Reset</td>
</tr>
<tr>
<td>1 0</td>
<td>0 1 0</td>
<td>Txl/R</td>
</tr>
<tr>
<td>1 1</td>
<td>0 0 1</td>
<td>None</td>
</tr>
<tr>
<td>1 1</td>
<td>0 1 0</td>
<td>Rxl/R</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 (Result)</td>
<td>Commands that execute immediately produce a result byte in this register, to be read by the processor.</td>
</tr>
<tr>
<td>Transmit Interrupt Results (TxI/R)</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>Receiver Interrupt Results (Rx/I/R)</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>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 (DACK 1 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.
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>Bit PA</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>PA0 Clear to Send</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA1 Carrier Detect</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA2 Data Set Ready</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA3 CTS Change</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA4 DSR Change</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 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></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></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.
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>Hex Address 380</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>0 = Ring Indicator is on from Interface</td>
</tr>
<tr>
<td>6</td>
<td>0 = Data Carrier Detect is on from Interface</td>
</tr>
<tr>
<td>5</td>
<td>Oscillating = Transmit Clock Active</td>
</tr>
<tr>
<td>4</td>
<td>0 = Clear to Send is on from Interface</td>
</tr>
<tr>
<td>3</td>
<td>Oscillating = Receive Clock Active</td>
</tr>
<tr>
<td>2</td>
<td>1 = Modem Status Changed</td>
</tr>
<tr>
<td>1</td>
<td>1 = Timer 2 Output Active</td>
</tr>
<tr>
<td>0</td>
<td>1 = Timer 1 Output Active</td>
</tr>
</tbody>
</table>

*Port A is defined as an input port

### 8255A-5 Port B Assignments*

<table>
<thead>
<tr>
<th>Bit</th>
<th>Hex Address 381</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>0 = Turn On Data Signal Rate Select at Modem Interface</td>
</tr>
<tr>
<td>6</td>
<td>0 = Turn On Select Standby at Modem Interface</td>
</tr>
<tr>
<td>5</td>
<td>0 = Turn On Test</td>
</tr>
<tr>
<td>4</td>
<td>1 = Reset Modem Status Changed Logic</td>
</tr>
<tr>
<td>3</td>
<td>1 = Reset 8273</td>
</tr>
<tr>
<td>2</td>
<td>1 = Gate Timer 2</td>
</tr>
<tr>
<td>1</td>
<td>1 = Gate Timer 1</td>
</tr>
<tr>
<td>0</td>
<td>1 = Enable Level 4 Interrupt</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>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>TxIRA = TxINT Result Available</td>
</tr>
<tr>
<td>6</td>
<td>RxIRA = RxINT Result Available</td>
</tr>
<tr>
<td>5</td>
<td>TxINT = Tx Interrupt</td>
</tr>
<tr>
<td>4</td>
<td>RxINT = Rx Interrupt</td>
</tr>
<tr>
<td>3</td>
<td>CRBF = Command Result Buffer Full</td>
</tr>
<tr>
<td>2</td>
<td>CPBF = Command Parameter Buffer Full</td>
</tr>
<tr>
<td>1</td>
<td>CBF = Command Buffer Full</td>
</tr>
<tr>
<td>0</td>
<td>CBSY = 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>
</tr>
</thead>
<tbody>
<tr>
<td>SC1</td>
<td>SC0</td>
<td>RL1</td>
<td>RL0</td>
<td>M2</td>
<td>M1</td>
<td>M0</td>
<td>BCD</td>
</tr>
</tbody>
</table>

### Definitions of Control

**SC - Select Counter:**

<table>
<thead>
<tr>
<th>SC1</th>
<th>SC0</th>
<th>Control</th>
<th>Counter</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Select 0</td>
<td>Counter 0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Select 1</td>
<td>Counter 1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Select 2</td>
<td>Counter 2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Illegal</td>
<td></td>
</tr>
</tbody>
</table>

**RL - Read/Load:**

<table>
<thead>
<tr>
<th>RL1</th>
<th>RL0</th>
<th>Definition</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:**

<table>
<thead>
<tr>
<th>BCD</th>
<th>Definition</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>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 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>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>1 = NRZI Mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 = Clock Loopback</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 = Data Loopback</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Not Used</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Not Used</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Not Used</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Not Used</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Not Used</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>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>1 = Interrupt Data Transfers</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Not Used</td>
</tr>
</tbody>
</table>
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**

<table>
<thead>
<tr>
<th>8273 One-Bit Delay Mode Register Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit</td>
</tr>
<tr>
<td>-----</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

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>B</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>385</td>
<td>8253</td>
<td>Counter 0 MSB</td>
<td>Square Wave Generator</td>
</tr>
<tr>
<td>386</td>
<td>8253</td>
<td>Counter 1 LSB</td>
<td>Inactivity Time-Outs</td>
</tr>
<tr>
<td>387</td>
<td>8253</td>
<td>Counter 1 MSB</td>
<td>Inactivity Time-Outs</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).

---

Drivers

<table>
<thead>
<tr>
<th>+15 Vdc</th>
<th>+5 Vdc</th>
<th>-5 Vdc</th>
<th>-15 Vdc</th>
</tr>
</thead>
</table>

Active Level: Data = 0

Invalid Level

Inactive Level: Data = 1

Receivers

<p>| +25 Vdc | +3 Vdc | -3 Vdc | -25 Vdc |</p>
<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**
IBM Communications Adapter Cable

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.

<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>NC</td>
<td></td>
<td>NC</td>
</tr>
<tr>
<td>NC</td>
<td></td>
<td>NC</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></td>
<td>NC</td>
</tr>
<tr>
<td>15</td>
<td>Transmitter Signal Element Timing</td>
<td>15</td>
</tr>
<tr>
<td>NC</td>
<td></td>
<td>NC</td>
</tr>
<tr>
<td>17</td>
<td>Receiver Signal Element Timing</td>
<td>NC</td>
</tr>
<tr>
<td>18</td>
<td>Test</td>
<td>18</td>
</tr>
<tr>
<td>NC</td>
<td></td>
<td>NC</td>
</tr>
<tr>
<td>20</td>
<td>Data Terminal Ready</td>
<td>20</td>
</tr>
<tr>
<td>NC</td>
<td></td>
<td>NC</td>
</tr>
<tr>
<td>22</td>
<td>Ring Indicator</td>
<td>22</td>
</tr>
<tr>
<td>23</td>
<td>Data Signal Rate Selector</td>
<td>23</td>
</tr>
<tr>
<td>NC</td>
<td></td>
<td>NC</td>
</tr>
<tr>
<td>25</td>
<td>Test Indicate</td>
<td>25</td>
</tr>
</tbody>
</table>

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

To read the time of day:

```
MOV AH,0 ;function is to read time of day.
INT 1AH ;read the timer.
```

Generally, the BIOS routines save all registers except for AX and the flags. Other registers are modified on return only if they are returning a value to the caller. The exact register usage 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>DISKETTE_IO</td>
</tr>
<tr>
<td>50-53</td>
<td>14</td>
<td>Communications</td>
<td>RS232_IO</td>
</tr>
<tr>
<td>54-57</td>
<td>15</td>
<td>Cassette</td>
<td>CASSETTE_IO</td>
</tr>
<tr>
<td>58-5B</td>
<td>16</td>
<td>Keyboard</td>
<td>KEYBOARD_IO</td>
</tr>
<tr>
<td>5C-5F</td>
<td>17</td>
<td>Printer</td>
<td>PRINTER_IO</td>
</tr>
<tr>
<td>60-63</td>
<td>18</td>
<td>Resident BASIC</td>
<td>F600:0000</td>
</tr>
<tr>
<td>64-67</td>
<td>19</td>
<td>Bootstrap</td>
<td>BOOT_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

2-4  ROM BIOS
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></td>
<td>Reserved</td>
</tr>
<tr>
<td>4F0-4FF</td>
<td></td>
<td>Reserved as Intra-Application Communication Area for any application</td>
</tr>
<tr>
<td>500-5FF</td>
<td>DOS</td>
<td>Print Screen Status Flag Store</td>
</tr>
<tr>
<td>500</td>
<td></td>
<td>0-Print Screen Not Active or Successful</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Print Screen Operation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1-Print Screen In Progress</td>
</tr>
<tr>
<td></td>
<td></td>
<td>255-Error Encountered during Print Screen Operation</td>
</tr>
<tr>
<td>504</td>
<td>DOS</td>
<td>Single Drive Mode Status Byte</td>
</tr>
<tr>
<td>510-511</td>
<td>BASIC</td>
<td>BASIC's Segment Address Store</td>
</tr>
<tr>
<td>512-515</td>
<td>BASIC</td>
<td>Clock Interrupt Vector Segment: Offset Store</td>
</tr>
<tr>
<td>516-519</td>
<td>BASIC</td>
<td>Break Key Interrupt Vector Segment: Offset Store</td>
</tr>
<tr>
<td>51A-51D</td>
<td>BASIC</td>
<td>Disk Error Interrupt Vector Segment: Offset Store</td>
</tr>
</tbody>
</table>

Reserved Memory Locations

If you do DEF SEG (Default workspace segment):

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

Example

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

```
100 Print Hex 64 Hex 00
```

BASIC Workspace Variables
### BIOS Memory Map

<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 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.
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>!</td>
<td>1</td>
<td>-1</td>
<td>Note 1</td>
</tr>
<tr>
<td>3</td>
<td>@</td>
<td>Nul (000)</td>
<td>Note 1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>#</td>
<td>-1</td>
<td>Note 1</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>$</td>
<td>-1</td>
<td>Note 1</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>%</td>
<td>-1</td>
<td>Note 1</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>^</td>
<td>RS(030)</td>
<td>Note 1</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>&amp;</td>
<td>-1</td>
<td>Note 1</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>*</td>
<td>-1</td>
<td>Note 1</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>(</td>
<td>-1</td>
<td>Note 1</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>)</td>
<td>-1</td>
<td>Note 1</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>-</td>
<td>US(031)</td>
<td>Note 1</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>+</td>
<td>-1</td>
<td>Note 1</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Backspace (008)</td>
<td>Backspace (008)</td>
<td>Del (127)</td>
<td>Note 1</td>
</tr>
<tr>
<td>15</td>
<td>(009)</td>
<td>(Note 1)</td>
<td>-1</td>
<td>Note 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>&quot;</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>41</td>
<td>&quot;</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></td>
<td>FS (028)</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>59</td>
<td>Nul (Note 1)</td>
<td>Nul (Note 1)</td>
<td>Nul (Note 1)</td>
<td>Nul (Note 1)</td>
</tr>
<tr>
<td>60</td>
<td>Nul (Note 1)</td>
<td>Nul (Note 1)</td>
<td>Nul (Note 1)</td>
<td>Nul (Note 1)</td>
</tr>
<tr>
<td>61</td>
<td>Nul (Note 1)</td>
<td>Nul (Note 1)</td>
<td>Nul (Note 1)</td>
<td>Nul (Note 1)</td>
</tr>
<tr>
<td>62</td>
<td>Nul (Note 1)</td>
<td>Nul (Note 1)</td>
<td>Nul (Note 1)</td>
<td>Nul (Note 1)</td>
</tr>
<tr>
<td>63</td>
<td>Nul (Note 1)</td>
<td>Nul (Note 1)</td>
<td>Nul (Note 1)</td>
<td>Nul (Note 1)</td>
</tr>
<tr>
<td>64</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>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>71</td>
<td>7</td>
<td>Home (Note 1)</td>
<td>-1 Clear Screen</td>
<td>1. Refer to &quot;Extended Codes&quot; in this section.</td>
</tr>
<tr>
<td>72</td>
<td>8</td>
<td>↑ (Note 1)</td>
<td>-1 -1</td>
<td>2. Refer to &quot;Special Handling&quot; in this section.</td>
</tr>
<tr>
<td>73</td>
<td>9</td>
<td>Page Up (Note 1)</td>
<td>-1 Top of Text and Home</td>
<td></td>
</tr>
<tr>
<td>74</td>
<td>-</td>
<td>---------------------------</td>
<td>-1 Reverse Word (Note 1)</td>
<td></td>
</tr>
<tr>
<td>75</td>
<td>4</td>
<td>← (Note 1)</td>
<td>-1 -1</td>
<td></td>
</tr>
<tr>
<td>76</td>
<td>5</td>
<td>-1</td>
<td>-1 Advance Word (Note 1)</td>
<td></td>
</tr>
<tr>
<td>77</td>
<td>6</td>
<td>← (Note 1)</td>
<td>-1 -1</td>
<td></td>
</tr>
<tr>
<td>78</td>
<td>+</td>
<td>+</td>
<td>-1 -1</td>
<td></td>
</tr>
<tr>
<td>79</td>
<td>1</td>
<td>End (Note 1)</td>
<td>-1 Erase to EOL (Note 1)</td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>2</td>
<td>↓ (Note 1)</td>
<td>-1 -1</td>
<td></td>
</tr>
<tr>
<td>81</td>
<td>3</td>
<td>Page Down (Note 1)</td>
<td>-1 Erase to EOS (Note 1)</td>
<td></td>
</tr>
<tr>
<td>82</td>
<td>0</td>
<td>Ins</td>
<td>-1 -1</td>
<td></td>
</tr>
<tr>
<td>83</td>
<td></td>
<td>Del (Notes 1,2)</td>
<td>Note 2 Note 2</td>
<td></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.

2-16 Keyboard Encoding
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</td>
<td>End</td>
<td>Editors; word processors</td>
</tr>
<tr>
<td>Place cursor at end of line</td>
<td></td>
<td></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>Destructive backspace</td>
<td>← Key 14</td>
<td>Text, command entry</td>
</tr>
<tr>
<td>Tab forward</td>
<td>→</td>
<td>Text entry</td>
</tr>
<tr>
<td>Tab reverse</td>
<td>←</td>
<td>Text entry</td>
</tr>
<tr>
<td>Clear screen and home</td>
<td>Ctrl 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 Num Lock</td>
<td>Stop list, stop program, and so on</td>
</tr>
<tr>
<td></td>
<td></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 Break</td>
</tr>
<tr>
<td>Backspace</td>
<td>Ctrl</td>
</tr>
<tr>
<td>Line feed</td>
<td>Esc</td>
</tr>
<tr>
<td>Cancel line</td>
<td></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>System ROM BIOS</th>
<th>Page</th>
<th>Line Number</th>
</tr>
</thead>
<tbody>
<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

<table>
<thead>
<tr>
<th>Fixed Disk ROM BIOS</th>
<th>Page</th>
<th>Line Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed Disk I/O Interface</td>
<td>A-84</td>
<td>1</td>
</tr>
<tr>
<td>Boot Strap Loader</td>
<td>A-89</td>
<td>399</td>
</tr>
</tbody>
</table>
TITLE(IBM PERSONAL COMPUTER XT)

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.

---

ABSOLUTE SEGMENT AT 0

0000  40 STEM_LOC0 LABEL BYTE
0000  41 ORG 2'4
0000  42 BKPTR LABEL WORD
0000  43 ORG 5'4
0000  44 TSTPẬR LABEL WORD
0000  45 ORG 0'4
0000  46 INT_ADDR LABEL WORD
0000  47 INT_PTR LABEL DWORD
0000  48 ORG 0'4
0000  49 VIDEO_INT LABEL WORD
0000  50 ORG 10'H
0000  51 PARM_PTR LABEL DWORD ; POINTER TO VIDEO PARMS
0000  52 ORG 10'H
0000  53 BASIC_PTR LABEL WORD ; ENTRY POINT FOR CASSETTE BASIC
0000  54 ORG 01'H
0000  55 DISK_POINTER LABEL DWORD
0000  56 ORG 01'H
0000  57 EXT_PTR LABEL DWORD ; LOCATION OF POINTER
0000  58 ORG 40'H
0000  59 DATA_AREA LABEL BYTE ; ABSOLUTE LOCATION OF DATA SEGMENT
0000  60 DATAWORD LABEL WORD
0000  61 ORG 050'H
0000  62 MFG_TEST_PTR LABEL FAR
0000  63 ORG 7C0'H
0000  64 BOOT_LOC0 LABEL FAR
0000  65 ABS0 ENDS

---

STACK SEGMENT AT 30H

0000 (128)
     DW  128 DUP(?)

0100 TOS LABEL WORD

---

STACK ENDS

---

ROM BIOS DATA AREAS

A-2 System BIOS
**LOC OBJ**

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>78</td>
<td>--------------</td>
</tr>
<tr>
<td>79</td>
<td>--------------</td>
</tr>
<tr>
<td>80</td>
<td>DATA SEGMENT AT 40H</td>
</tr>
<tr>
<td>81</td>
<td>RS232_BASE DW 4 DUP(?) ; ADDRESSES OF RS232 ADAPTERS</td>
</tr>
<tr>
<td>82</td>
<td>PRINTER_BASE DW 4 DUP(?) ; ADDRESSES OF PRINTERS</td>
</tr>
<tr>
<td>83</td>
<td>EQUIP_FLAG DW ? ; INSTALLED HARDWARE</td>
</tr>
<tr>
<td>84</td>
<td>MFG_TST DB ? ; INITIALIZATION FLAG</td>
</tr>
<tr>
<td>85</td>
<td>MEMORY_SZ DW ? ; MEMORY SIZE IN X BYTES</td>
</tr>
<tr>
<td>86</td>
<td>MFG_ERR_FLAG DB ? ; SCRATCHPAD FOR MANUFACTURING</td>
</tr>
<tr>
<td>87</td>
<td>DB ? ; ERROR CODES</td>
</tr>
<tr>
<td>88</td>
<td>--------------</td>
</tr>
<tr>
<td>89</td>
<td>KEYBOARD DATA AREAS</td>
</tr>
<tr>
<td>90</td>
<td>--------------</td>
</tr>
<tr>
<td>91</td>
<td>--------------</td>
</tr>
<tr>
<td>92</td>
<td>--------------</td>
</tr>
<tr>
<td>93</td>
<td>KB_FLAG DB ? ;</td>
</tr>
<tr>
<td>94</td>
<td>--------------</td>
</tr>
<tr>
<td>95</td>
<td>SHIFT FLAG EQUATES WITHIN KB_FLAG</td>
</tr>
<tr>
<td>96</td>
<td>--------------</td>
</tr>
<tr>
<td>97</td>
<td>INS_STATE EQU 80H ; INSERT STATE IS ACTIVE</td>
</tr>
<tr>
<td>98</td>
<td>CAPS_STATE EQU 40H ; CAPS LOCK STATE HAS BEEN TOGGLED</td>
</tr>
<tr>
<td>99</td>
<td>MNH_STATE EQU 20H ; MHN LOCK STATE HAS BEEN TOGGLED</td>
</tr>
<tr>
<td>100</td>
<td>SCROLL_STATE EQU 10H ; SCROLL LOCK STATE HAS BEEN TOGGLED</td>
</tr>
<tr>
<td>101</td>
<td>ALB_SHIFT EQU 00H ; ALTERNATE SHIFT KEY DEPRESSED</td>
</tr>
<tr>
<td>102</td>
<td>CTL_SHIFT EQU 04H ; CONTROL SHIFT KEY DEPRESSED</td>
</tr>
<tr>
<td>103</td>
<td>LEFT_SHIFT EQU 02H ; LEFT SHIFT KEY DEPRESSED</td>
</tr>
<tr>
<td>104</td>
<td>RIGHT_SHIFT EQU 01H ; RIGHT SHIFT KEY DEPRESSED</td>
</tr>
<tr>
<td>105</td>
<td>--------------</td>
</tr>
<tr>
<td>106</td>
<td>KB_FLAG_1 DB ? ; SECOND BYTE OF KEYBOARD STATUS</td>
</tr>
<tr>
<td>107</td>
<td>--------------</td>
</tr>
<tr>
<td>108</td>
<td>INS_SHIFT EQU 80H ; INSERT KEY IS DEPRESSED</td>
</tr>
<tr>
<td>109</td>
<td>CAPS_SHIFT EQU 40H ; CAPS LOCK KEY IS DEPRESSED</td>
</tr>
<tr>
<td>110</td>
<td>MNH_SHIFT EQU 20H ; MHN LOCK KEY IS DEPRESSED</td>
</tr>
<tr>
<td>111</td>
<td>SCROLL_SHIFT EQU 10H ; SCROLL LOCK KEY IS DEPRESSED</td>
</tr>
<tr>
<td>112</td>
<td>HOLD_STATE EQU 00H ; SUSPEND KEY HAS BEEN TOGGLED</td>
</tr>
<tr>
<td>113</td>
<td>--------------</td>
</tr>
<tr>
<td>114</td>
<td>ALT_INPUT DB ? ; STORAGE FOR ALTERNATE KEYPAD ENTRY</td>
</tr>
<tr>
<td>115</td>
<td>BUFFER_HEAD DW ? ; POINTER TO HEAD OF KEYBOARD BUFFER</td>
</tr>
<tr>
<td>116</td>
<td>BUFFER_TAIL DW ? ; POINTER TO TAIL OF KEYBOARD BUFFER</td>
</tr>
<tr>
<td>117</td>
<td>KB_BUFFER DW 16 DUP(?) ; ROOM FOR IS ENTRIES</td>
</tr>
<tr>
<td>118</td>
<td>--------------</td>
</tr>
<tr>
<td>119</td>
<td>--------------</td>
</tr>
<tr>
<td>120</td>
<td>HEAD = TAIL INDICATES THAT THE BUFFER IS EMPTY</td>
</tr>
<tr>
<td>121</td>
<td>--------------</td>
</tr>
<tr>
<td>122</td>
<td>MHN_KEY EQU 69 ; SCAN CODE FOR NUMBER LOCK</td>
</tr>
<tr>
<td>123</td>
<td>SCROLL_KEY EQU 70 ; SCAN CODE FOR SCROLL LOCK</td>
</tr>
<tr>
<td>124</td>
<td>ALT_KEY EQU 96 ; ALTERNATE SHIFT KEY SCAN CODE</td>
</tr>
<tr>
<td>125</td>
<td>CTL_KEY EQU 29 ; SCAN CODE FOR CONTROL KEY</td>
</tr>
<tr>
<td>126</td>
<td>CAPS_KEY EQU 58 ; SCAN CODE FOR SHIFT LOCK</td>
</tr>
<tr>
<td>127</td>
<td>LEFT_KEY EQU 42 ; SCAN CODE FOR LEFT SHIFT</td>
</tr>
<tr>
<td>128</td>
<td>RIGHT_KEY EQU 54 ; SCAN CODE FOR RIGHT SHIFT</td>
</tr>
<tr>
<td>129</td>
<td>INS_KEY EQU 02 ; SCAN CODE FOR INSERT KEY</td>
</tr>
<tr>
<td>130</td>
<td>DEL_KEY EQU 03 ; SCAN CODE FOR DELETE KEY</td>
</tr>
<tr>
<td>131</td>
<td>--------------</td>
</tr>
<tr>
<td>132</td>
<td>--------------</td>
</tr>
<tr>
<td>133</td>
<td>DISKETTE DATA AREAS</td>
</tr>
<tr>
<td>134</td>
<td>--------------</td>
</tr>
<tr>
<td>135</td>
<td>SEEK_STATUS DB ? ; DRIVE RECALIBRATION STATUS</td>
</tr>
<tr>
<td>136</td>
<td>BIT 3-0 = DRIVE 3-0 NEEDS RECAL</td>
</tr>
<tr>
<td>137</td>
<td>BEFORE NEXT SEEK IF BIT IS = 0</td>
</tr>
<tr>
<td>138</td>
<td>--------------</td>
</tr>
<tr>
<td>139</td>
<td>INT_FLAG EQU 080H ; INTERRUPT OCCURRENCE FLAG</td>
</tr>
<tr>
<td>140</td>
<td>MOTOR_STATUS DB ? ; MOTOR STATUS</td>
</tr>
<tr>
<td>141</td>
<td>--------------</td>
</tr>
<tr>
<td>142</td>
<td>--------------</td>
</tr>
<tr>
<td>143</td>
<td>--------------</td>
</tr>
<tr>
<td>144</td>
<td>--------------</td>
</tr>
<tr>
<td>145</td>
<td>--------------</td>
</tr>
<tr>
<td>146</td>
<td>MOTOR_COUNT DB ? ; TIME OUT COUNTER FOR DRIVE TURN OFF</td>
</tr>
<tr>
<td>147</td>
<td>MOTOR_WAIT EQU 37 ; 2 SECS OF COUNTS FOR MOTOR TURN OFF</td>
</tr>
</tbody>
</table>
LOC OBJ  LINE SOURCE

0091 ??  149 DISKETTE_STATUS DB ?  ; RETURN CODE STATUS BYTE
0090 150 TIME_OUT EQU 60H  ; ATTACHMENT FAILED TO RESPOND
0090 151 BAD_SEEK EQU 40H  ; SEEK OPERATION FAILED
0091 152 BAD_SEC EQU 20H  ; NEC CONTROLLER HAS FAILED
0091 153 BAD_CRC EQU 10H  ; BAD CRC ON DISKETTE READ
0091 154 DMA_BOUNDARY EQU 0FH  ; ATTEMPT TO DMA ACROSS 64K BOUNDARY
0091 155 BAD_DMA EQU 00H  ; DMA OVERRUN ON OPERATION
0091 156 RECORD_NOT_CMD EQU 04H  ; REQUESTED SECTOR NOT FOUND
0091 157 WRITE_PROTECT EQU 03H  ; WRITE ATTEMPTED ON WRITE PROT DISK
0091 158 BAD_ADDR MARK EQU 02H  ; ADDRESS MARK NOT FOUND
0091 159 BAD_CMD EQU 01H  ; BAD COMMAND PASSED TO DISKETTE I/O
0091 160 ??

0091 161 NEC_STATUS DB ? DUP(?)  ; STATUS BYTES FROM NEC

0094 ??  161 NEC_STATUS DB ? DUP(?)  ; STATUS BYTES FROM NEC

0094 ??  166 CRT_MODE DB ?  ; CURRENT CRT MODE
0094 ??  167 CRT_COL DB ?  ; NUMBER OF COLUMNS ON SCREEN
0094 ??  168 CRT_LEN DB ?  ; LENGTH OF REGN IN BYTES
0094 ??  169 CRT_START DW ?  ; STARTING ADDRESS IN REGEN BUFFER
0094 (6) 170 CURSOR_POSN DW 0 DUP(?)  ; CURSOR FOR EACH OF UP TO 8 PAGES

0095 ??  171 CURSOR_MODE DW ?  ; CURRENT CURSOR MODE SETTING
0095 ??  172 ACTIVE_PAGE DB ?  ; CURRENT PAGE BEING DISPLAYED
0095 ??  173 ADDR_64K DW ?  ; BASE ADDRESS FOR ACTIVE DISPLAY CARD
0095 ??  174 CRT_MODE_SET DB ?  ; CURRENT SETTING OF THE 32B REGISTER
0095 ??  175 CRT_PALETTE DB ?  ; CURRENT PALETTE SETTING COLOR CARD
0095 ??  176 ??

0096 ??  177 ----- -------------------------------
0096 ??  178 ; POST DATA AREA :
0096 ??  179 ----- -------------------------------

0096 ??  180 IO_ROM_INST DW ?  ; PMTR TO OPTIONAL I/O ROM INIT ROUTINE
0096 ??  181 IO_ROM_SEG DW ?  ; PMTR TO IO ROM SEGMENT
0096 ??  182 INTR_FLAG DB ?  ; FLAG TO INDICATE AN INTERRUPT HAPEND

0096 ??  183 ??

0096 ??  184 ----- -------------------------------
0096 ??  185 ; TIMER DATA AREA :
0096 ??  186 ----- -------------------------------

0096 ??  187 TIMER_LOW DW ?  ; LOW WORD OF TIMER COUNT
0096 ??  188 TIMER_HIGH DW ?  ; HIGH WORD OF TIMER COUNT
0096 ??  189 TIMER_RES EQU 01H  ; TIMER HAS ROLLED OVER SINCE LAST READ
0096 ??  190 COUNTS_SEC EQU 16
0096 ??  191 COUNTS_MIN EQU 1092
0096 ??  192 COUNTS_HOUR EQU 65543
0096 ??  193 COUNTS_DAY EQU 1573040 = 18006BH

0096 ??  194 ??

0096 ??  195 ----- -------------------------------
0096 ??  196 ; SYSTEM DATA AREA :
0096 ??  197 ----- -------------------------------

0096 ??  198 BIOS_BREAK DB ?  ; BIT 7:1 IF BREAK KEY HAS BEEN HIT
0096 ??  199 RESET_FLAG DB ?  ; WORD=1254H IF KEYBOARD RESET UNDERWAY

0096 ??  200 ----- -------------------------------
0096 ??  201 ; FIXED DISK DATA AREAS :
0096 ??  202 ----- -------------------------------

0096 ??  203 ??

0096 ??  204 ??

0096 ??  205 ----- -------------------------------
0096 ??  206 ; PRINTER AND RS232 TIME-OUT VARIABLES :
0096 ??  207 ----- -------------------------------

0096 ??  208 ??

0096 ??  209 RS232_TIME_OUT DB 4 DUP(?)

0096 ??  209 ??

0097 (4) 209 ??

0097 (4) 209 ??

0097 (4) 209 ??

0097 ??  210 ----- -------------------------------
0097 ??  211 ; ADDITIONAL KEYBOARD DATA AREA :
0097 ??  212 ----- -------------------------------

0097 ??  213 ??

0097 ??  214 ??

0097 ??  215 DATA_ENDS

0097 ??  216 ??

0097 ??  217 EXTRA_DATA_AREA :

0097 ??  218 ??

A-4 System BIOS
SOURCE

XXDATA SEGMENT AT SOH

STATUS_BYTE

DB 221

XXDATA ENDS

VIDEO_RAM SEGMENT AT OF00H

REGEN LABEL BYTE

REGEN LABEL NRND

DB 16384 DUP(?)

)

VIDEO_RAM ENDS

E000 (57344

DB 57344 DUP(?)

) FILL LOWEST 56K

COPYRIGHT NOTICE

IBM 1981'

LOWEST 56K

INITIAL RELIABILITY TESTS -- PHASE 1 :

DATA DEFINITIONS

RETURN ADDRESS FOR DUMMY STACK

RETURN ADDRESS FOR MEMORY SIZE

System BIOS  A-5
A-6  System BIOS
; DESCRIPTION

; A CHECKSUM IS DONE FOR THE BK

; A BIOS MODULE CONTAINING ROM AND...

; BIOS.

;i----------------------------------------------------------------------------------

EDAE 369 C10:

370 i ZERO IN AL ALREADY

371 OUT 0AH,AL i DISABLE NMI INTERRUPTS

372 OUT 08H,AL i INITIALIZE DMA PAGE REG.

373 MOV DX,320BH i DISABLE COLOR VIDEO

374 OUT DX,AL i DISABLE B/W VIDEO, EN HIGH RES

375 INC AL i SETUP 8K FOR B,A+OUT, C3IN

376 MOV DL,08BH i Set Initial DMA Controls

377 OUT DX,AL ; 8237 DMA CONTROLLER, VERIFY THAT TIMER 1 FUNCTIONS OK.

378 MOV AL,69H i SET 6255 FOR B,A+OUT, C3IN

379 OUT CMD_PORT,AL i ENABLE PARITY CHECKERS AND

380 MOV AL,16100101B ; ENABLE THE 8237 DATA CONTROLLER. VERIFY THAT TIMER 1 FUNCTIONS OK.

381 OUT PORT_B,AL i SET UP DATA SEG TO POINT TO ROM ADDRESS

382 OUT PORT_B,AL ; PULL KB CLOCK HI, TRISTATE, KEYBOARD INPUTS ENABLE HIGH

383 MOV AX,CS i INITIALIZE AND START DMA FOR MEMORY

384 MOV DS,AX i SET UP DATA SEG TO POINT TO

385 MOV AL,04 i DEFINE FLAG TO INC.

386 MOV AL,04 i SETUP STARTING ADDR

387 MOV AX,0FH i SETUP RETURN ADDRESS

388 MOV AX,0FH i DISABLE DMA CONTROLLER

389 MOV AL,04 ; DISABLE DMA CONTROLLER

390 MOV AL,04 i VERIFY THAT TIMER 1 FUNCTIONS OK

391 MOV AX,0FH i VERIFY THAT TIMER 1 FUNCTIONS OK

392 MOV AX,0FH i VERIFY THAT TIMER 1 FUNCTIONS OK

393 MOV AL,04 i VERIFY THAT TIMER 1 FUNCTIONS OK

394 MOV AL,04 i VERIFY THAT TIMER 1 FUNCTIONS OK

395 MOV AL,04 i VERIFY THAT TIMER 1 FUNCTIONS OK

396 MOV AL,04 i VERIFY THAT TIMER 1 FUNCTIONS OK

397 MOV AX,0FH i VERIFY THAT TIMER 1 FUNCTIONS OK

398 MOV AX,0FH i VERIFY THAT TIMER 1 FUNCTIONS OK

399 MOV AX,0FH i VERIFY THAT TIMER 1 FUNCTIONS OK

400 MOV AX,0FH i VERIFY THAT TIMER 1 FUNCTIONS OK

401 MOV AX,0FH i VERIFY THAT TIMER 1 FUNCTIONS OK

402 MOV AX,0FH i VERIFY THAT TIMER 1 FUNCTIONS OK

403 MOV AX,0FH i VERIFY THAT TIMER 1 FUNCTIONS OK

404 MOV AX,0FH i VERIFY THAT TIMER 1 FUNCTIONS OK

405 MOV AX,0FH i VERIFY THAT TIMER 1 FUNCTIONS OK

406 MOV AX,0FH i VERIFY THAT TIMER 1 FUNCTIONS OK

407 MOV AX,0FH i VERIFY THAT TIMER 1 FUNCTIONS OK

408 MOV AX,0FH i VERIFY THAT TIMER 1 FUNCTIONS OK

409 MOV AX,0FH i VERIFY THAT TIMER 1 FUNCTIONS OK

410 MOV AX,0FH i VERIFY THAT TIMER 1 FUNCTIONS OK

411 MOV AX,0FH i VERIFY THAT TIMER 1 FUNCTIONS OK

412 MOV AX,0FH i VERIFY THAT TIMER 1 FUNCTIONS OK

413 MOV AX,0FH i VERIFY THAT TIMER 1 FUNCTIONS OK

414 MOV AX,0FH i VERIFY THAT TIMER 1 FUNCTIONS OK

415 MOV AX,0FH i VERIFY THAT TIMER 1 FUNCTIONS OK

416 MOV AX,0FH i VERIFY THAT TIMER 1 FUNCTIONS OK

417 MOV AX,0FH i VERIFY THAT TIMER 1 FUNCTIONS OK

418 MOV AX,0FH i VERIFY THAT TIMER 1 FUNCTIONS OK

419 MOV AX,0FH i VERIFY THAT TIMER 1 FUNCTIONS OK

420 MOV AX,0FH i VERIFY THAT TIMER 1 FUNCTIONS OK

421 MOV AX,0FH i VERIFY THAT TIMER 1 FUNCTIONS OK

422 MOV AX,0FH i VERIFY THAT TIMER 1 FUNCTIONS OK

423 MOV AX,0FH i VERIFY THAT TIMER 1 FUNCTIONS OK

424 MOV AX,0FH i VERIFY THAT TIMER 1 FUNCTIONS OK

425 MOV AX,0FH i VERIFY THAT TIMER 1 FUNCTIONS OK

426 MOV AX,0FH i VERIFY THAT TIMER 1 FUNCTIONS OK

427 MOV AX,0FH i VERIFY THAT TIMER 1 FUNCTIONS OK

428 MOV AX,0FH i VERIFY THAT TIMER 1 FUNCTIONS OK

429 MOV AX,0FH i VERIFY THAT TIMER 1 FUNCTIONS OK

430 MOV AX,0FH i VERIFY THAT TIMER 1 FUNCTIONS OK

431 MOV AX,0FH i VERIFY THAT TIMER 1 FUNCTIONS OK

432 MOV AX,0FH i VERIFY THAT TIMER 1 FUNCTIONS OK

433 MOV AX,0FH i VERIFY THAT TIMER 1 FUNCTIONS OK

434 MOV AX,0FH i VERIFY THAT TIMER 1 FUNCTIONS OK

435 MOV AX,0FH i VERIFY THAT TIMER 1 FUNCTIONS OK

436 MOV AX,0FH i VERIFY THAT TIMER 1 FUNCTIONS OK

437 MOV AX,0FH i VERIFY THAT TIMER 1 FUNCTIONS OK

438 MOV AX,0FH i VERIFY THAT TIMER 1 FUNCTIONS OK

439 MOV AX,0FH i VERIFY THAT TIMER 1 FUNCTIONS OK

440 MOV AX,0FH i VERIFY THAT TIMER 1 FUNCTIONS OK

;i----------------------------------------------------------------------------------
LOC OBJ   LINE   SOURCE

441 LOOP CX4 ; TIMER_LOOP

442 HLT ; HALT SYSTEM

443

444 ;----- INITIALIZE TIMER 1 TO REFRESH MEMORY

445

446 C15: MOV AH,03H ; <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>

447 OUT PORT_A,AL ; <<<<<<<<<<CHECKPOINT 3:>>>>>>>>

448 ; WRAP_DMA_REG

449 OUT DMA+0DH,AL ; SEND MASTER CLEAR TO DMA

450

451 ;----- WRAP DNA CHANNELS ADDRESS AND COUNT REGISTERS

452

453 MOV AL,0FFH ; WRITE PATTERN FF TO ALL REGS

454 C16: MOV BL,AL ; SAVE PATTERN FOR COMPARE

455 MOV BH,AL ;

456 MOV CX,B ; SETUP LOOP CNT

457 MOV DX,DATA ; SETUP I/O PORT ADDR OF REG

458 OUT DX,AL ; WRITE PATTERN TO REG. LSB

459 PUSH AX ; SATISFY 8237 I/O TIMINGS

460 OUT DX,AL ; MSB OF 16 BIT REG

461 MOV AL,01H ; AL TO ANOTHER REG BEFORE RD

462 IN AL,DX ; READ 16-BIT DMA CH REG. LSB

463 MOV AH,AL ; SAVE LSB OF 16-BIT REG

464 IN AL,DX ; READ MSB OF DMA CH REG

465 CMP BX,AX ; PATTERN READ AS WRITTEN?

466 JE C1B ; YES - CHECK NEXT REG

467 ; HLT ; NO - HALT THE SYSTEM

468 C18: ; NEXT_DMA_CH

469 INC DX ; SET I/O PORT TO NEXT CH REG

470 LOOP C17 ; WRITE PATTERN TO NEXT REG

471 INC AL ; SET PATTERN TO 0

472 JZ C16 ; WRITE TO CHANNEL REGS

473

474 ;----- INITIALIZE AND START DMA FOR MEMORY REFRESH.

475

476 MOV DS,DX ; SET UP ABSO INTO DS AND ES

477 MOV ES,DX ;

478 ASSUME DS:ABS0,ES:ABS0 ;

479 MOV AL,0FFH ; SET CNT OF 64K FOR REFRESH

480 OUT DMA+1AL ;

481 PUSH AX ;

482 OUT DMA+1AL ;

483 MOV AL,05H ; SET DMA MODE,CH 0,RD,ADJINT

484 MOV DMA+0BH,AL ; WRITE DMA MODE REG

485 MOV AL,0 ; ENABLE DMA CONTROLLER

486 MOV CH,AL ; SET COUNT HIGH=0D

487 OUT DMA+0AL ; SETUP DMA COMMAND REG

488 PUSH AX ;

489 MOV DMA+0+AL,AL ; ENABLE DMA CH 0

490 MOV AL,10 ; START TIMER 1

491 OUT TIMER+1AL,AL ;

492 MOV AL,61H ; SET MODE FOR CHANNEL 1

493 OUT DMA+0BH,AL ;

494 PUSH AX ;

495 IN AL,DMA+0B ; GET DMA STATUS

496 AND AL,00010000B ; IS TIMER REQUEST THERE?

497 JZ C1B ; (IT SHOULDN'T BE)

498 JZ C1B ;

499 C18: MOV AL,02H ; SET MODE FOR CHANNEL 2

500 OUT DMA+0DH,AL ;

501 MOV AL,03H ; SET MODE FOR CHANNEL 3

502 OUT DMA+0DH,AL ;

503

504 ; BASE 16K READ/WRITE STORAGE TEST :

505 ; DESCRIPTION :

506 ; WRITE/READ/VERIFY DATA PATTERNS :

507 ; AA,55,FF,01, AND 00 TO 1ST 32K OF :

508 ; STORAGE, VERIFY STORAGE ADDRESSABILITY :

509 ;----------------------------------------

510

511 ;----- DETERMINE MEMORY SIZE AND FILL MEMORY WITH DATA

512

513 MOV DX,0213H ; ENABLE I/O EXPANSION BOX

514 MOV AL,01H ;

515 OUT DX,AL ;

516

517 MOV BX,DATA_WOR[OFFSET RESET_FLAG] ; SAVE 'RESET_FLAG' IN BX

A-8 System BIOS
Appendix A

System BIOS A-9
SET MODE FOR BIW CARD

SET MODE FOR 40X25

A-to

LOC OBJ

LOC OBJ

LINE

SOURCE

0300 891000 593 MOV CX,16

0300 A5 594 Dia: MOVSW ; MOVE VECTOR TABLE TO RAM

0300 47 595 INC DI ; SKIP SEGMENT POINTER

0300 62FB 596 INC DI

0300 601 LOOP 03A

0300 598 |-----------------------------------------------|

0300 599 | DETERMINE CONFIGURATION AND MFG. MODE |

0300 600 |-----------------------------------------------|

0300 601

0300 602 IF

0300 603 POP DS ; RECOVER DATA SEG

0300 604 PUSH DS ; RECOVER DATA SEG

0300 605 IN AL, PORT_C ; GET SWITCH INFO

0300 606 AND AL, 0000011B ; ISOLATE SWITCHES

0300 607 MOV AH, AL ; SAVE

0300 608 MOV AL, 10101010B ; ENABLE OTHER BANK OF RAM.

0300 609 OUT PORT_B, AL

0300 60A NOP

0300 60B IN AL, PORT_C

0300 60C MOV CL, 4

0300 60D ROL AL, CL ; ROTATE TO HIGH Nibble

0300 60E AND AL, 11111000B ; ISOLATE

0300 60F 0AC4 MOV AL, 0AH ; COMBINE WITH OTHER BANK

0300 610 0AE4 SUB AH, AH

0300 611 A1004 MOV DATA_ADDR[OFFSET EQUIP_FLAG1].AX ; SAVE SWITCH INFO

0300 612 B099 MOV AL, 99H

0300 613 E63 MOV AL, 99H

0300 614 0516 CALL KBD_RESET ; SEE IF MFG. JUMPER IN

0300 615 0F80AA CMP BL, 0AAH ; KEYBOARD PRESENT?

0300 616 7418 JE E6

0300 617 0F8065 CMP BL, 065H ; LOAD MFG. TEST REQUEST?

0300 618 7503 JNE D3B

0300 619 EYEFDD JMP MFG_BOOT ; GO TO BOOTSTRAP IF SO

0300 61A E036 D3B: MOV AL, 36H

0300 61B E661 OUT PORT_B, AL

0300 61C 90 NOP

0300 61D 90 IN AL, PORT_A

0300 61E 660 AND AL, 00FH ; WAS DATA LINE GROUNDED

0300 61F 7504 JNZ E6

0300 620 FE061204 DATA_AREA[OFFSET MFG_TEST] ; SET MANUFACTURING TEST FLAG

0300 621 |-------------------------------| |

0300 622 | INITIALIZE AND START CRT CONTROLLER (64x45) |

0300 623 |-------------------------------| |

0300 624 | TEST VIDEO READ/WRITE STORAGE. |

0300 625 |-------------------------------| |

0300 626 | DESCRIPTION |

0300 627 |-------------------------------| |

0300 628 | RESET THE VIDEO ENABLE SIGNAL. |

0300 629 |-------------------------------| |

0300 62A | SELECT ALPHANUMERIC MODE, 40 * 25, B & W. |

0300 62B |-------------------------------| |

0300 62C | READ/WRITE DATA PATTERNS TO STG. CHECK STG |

0300 62D |-------------------------------| |

0300 62E | ADDRESSABILITY. |

0300 62F |-------------------------------| |

0300 630 | ERROR <= 1 LONG AND 2 SHORT BEEPS |

0300 631 |-------------------------------| |

0300 632 | E6:

0300 633 |-------------------------------| |

0300 634 |-------------------------------| |

0300 635 |-------------------------------| |

0300 636 |-------------------------------| |

0300 637 |-------------------------------| |

0300 638 |-------------------------------| |

0300 639 |-------------------------------| |

0300 63A |-------------------------------| |

0300 63B |-------------------------------| |

0300 63C |-------------------------------| |

0300 63D |-------------------------------| |

0300 63E |-------------------------------| |

0300 63F |-------------------------------| |
E274 6A40 670 E8: XOR AL, AL SET_MODE:
E278 50 671 PUSH AX
E279 2A4E 672 SUB AL, AL INITIALIZE TO ALPHANUMERIC MD
E27B CD10 673 INT 10H CALL_VIDEO
E27D 56 674 POP AX RESTORE_VIDEO_SENSE_SHS_IN_AL
E27E 50 675 PUSH AX RESAVE_VALUE
E27F 880080 676 MOV BX, 0080H B/4_VIDEO_RAM_ADDR_B/4_CD
E282 B8803 677 MOV DX, 3B8H MODE_REG FOR B/W
E285 B90008 678 MOV CX, 2048 RAM_MODE_CNT FOR B/W_CD
E288 BD01 679 MOV AH, 1 SET_MODE_FOR_B/W_CARD
E28A 80FC30 680 CMP AH, 30H B/W_VIDEO_CARD_ATTACHED?
E28D 7409 681 JE E9 YES - GO_TEST_VIDEO_STG
E28F B788 682 MOV DH, 0B8H B/4_VIDEO_RAM_ADDR_COLOR_CD
E291 B8803 683 MOV DX, 3D8H MODE_REG_FOR_COLOR_CD
E294 D200 684 MOV CH, 2B0H RAM_MODE_CRT_FOR_COLOR_CD
E296 FFC8 685 MOV AL DEC AROUND MODE soll
E298 EE 686 E9: ; SETUP_VIDEO_STG:
E299 EE 687 OUT DX, AL DISABLE_VIDEO_FOR_COLOR_CD
E29F B1372043412 688 CMP DATA_WORDDOSET_RESET_FLAG, 123AH ; POD INIT BY KBD_RESET?
E2AF 8E3 689 MOV ES, BX POINT ES_TO_VIDEO_RAM_STG
E2A1 7407 690 JE E10 YES - SKIP_VIDEO_RAM_TEST
E2A3 6E0B 691 MOV DS, BX POINT_DS_TO_VIDEO_RAM_STG
E2A5 E0C703 692 ASSUME DS: NOTHING, ES: NOTHING
E2A6 7546 693 CALL STGTST_CNT ; GO_TEST_VIDEO_B/W_STG
E2AB 750A 694 JNC E17 ; B/W_STG_FAILURE - KEEP_SPK
E2AA 702 695 -------------------------
E2A4 50 703 POP AX GET_VIDEO_SENSE_SHS (AH)
E2A5 50 704 PUSH AX
E2A8 B400 705 MOV AH, 0 ENABLE_VIDEO_AND_SET_MODE
E2A9 CD10 706 INT 10H VIDEO
E2B0 B0070 707 MOV AX, 7000H MRT_BLANKS_IN_REVERSE_VIDEO
E2B1 709 708
E2B3 8B11 711 JMP SHORT E10A
E2C3 712 ORG 04C3h
E2C3 E99915 713 JMP NE1_INT
E2C6 714
E2C6 2BFF 715 E10A: SUB DI, DI SETUP_STARTING_LOC
E2C8 B92B00 716 MOV CX, 40h NO_OF_BLANKS_TO_DISPLAY
E2CD F3 717 REP STOSW WRITE_VIDEO_STORAGE
E2CC AD 718
E2CD 719 -------------------------
E2CD 56 720 POP AX GET_VIDEO_SENSE_SH_INFO
E2CF 80FC30 721 PUSH AX
E2DF BABA03 722 MOV AH, 30H B/W_VIDEO_CARD_ATTACHED?
E2E0 7403 723 MOV DX, DX3AH SETUP_ADDR_OF_BW_STATUS_PORT
E2E7 7A03 724 JE E11 YES - GO_TEST_LINES
E2E8 BABA03 725 MOV DX, DX3AH COLOR_CARD_ATTACHED
E2EA 732 E11: 726 MOV AH, 0 LINE_TEST:
E2EA B400 733 MOV DX, 00H
E2ED 734 E12: 735 SUB CX, CX OF_LOOP_CNT:
E2EE 735 736 SUB CX, CX
E2EE B6C9 737 E13: IN AL,DX READ_CRT_STATUS_PORT
E2EE 736 737 IN AL, AX
E2EF 22C4 738 IN AL, AH READ_CRT_STATUS_PORT
E2E1 7504 739 JNZ E14 ; CHECK_VIDEO/HORIZ_LINE
E2F3 E2F9 740 LOOP E13 LOOP_TILL_ON OR_TIMEOUT
E2FE E009 741 JMP SHORT E17 ; GO_PRINT_ERROR_MSG
E2F7 742 E14: 743 SUB CX, CX ; WRITE_VIDEO_STORAGE
E2F9 744 E15: IN AL, DX ; READ_CRT_STATUS_PORT
E2FE 745

Appendix A

System BIOS A-11
LOC OBJ | LINE | SOURCE
--- | --- | ---
E2F4 22C0 | 746 | AND AL,AL ; CHECK VIDEO/HORZ LINE
E2EC 7411 | 747 | JZ E16 ; ITS ON - CHECK NEXT LINE
E2EF E2F9 | 748 | LOOP E15 ; LOOP IF OFF TILL IT GOES ON
E2F0 | 749 | E17: ; CRT_ERR:
E2F0 IF | 750 | POP DS
E2F1 IE | 751 | PUSH DS
E2F2 C606150006 | 752 | MOV DS:MF_ERR_FLAG,06H ; <<<<<<<<CRT ERR CHKPT. 06<<<<<<
E2F7 BA0201 | 753 | MOV DX,102H
E2FA E8B016 | 754 | CALL ERR_BEEP ; GO BEEP SPEAKER
E2FB E806 | 755 | JMP SHORT E18
E2FF | 756 | E16: ; NOT_LINE:
E2F0 DI03 | 757 | MOV CL,3 ; GET NEXT BIT TO CHECK
E301 D2EC | 758 | SHR AH,CL
E303 7507 | 759 | JNZ E12 ; GO CHECK HORIZONTAL LINE
E305 58 | 760 | E18: ; DISPLAY_CURSOR:
E305 58 | 761 | POP AX ; GET VIDEO SENSE SMS (AH)
E306 B400 | 762 | MOV AH,0 ; SET MODE AND DISPLAY CURSOR
E306 C010 | 763 | INT 10H ; CALL VIDEO I/O PROCEDURE
E306 | 764 | E18: ;
E30A BA00C3 | 765 | MOV DX,0C000H
E30D | 766 | E18A: ;
E30D 8600 | 767 | MOV DS,DX
E30F 2808 | 768 | SUB BX,BX
E311 0B07 | 769 | MOV AX,[BX+1] ; GET FIRST 2 LOCATIONS
E313 53 | 770 | PUSH BX
E314 58 | 771 | POP BX ; LET BUS SETTLE
E315 3055AA | 772 | CMP AX,0A45H ; PRESENT?
E318 7505 | 773 | JNZ E100 ; NOT GO LOOK FOR OTHER MODULES
E31A E83616 | 774 | CALL ROM_CHECK ; GO SCAN MODULE
E31D E804 | 775 | JMP SHORT E1BC
E31F | 776 | E18B: ;
E31F 0C20000 | 777 | ADD DX,0000H ; POINT TO NEXT 2K BLOCK
E323 | 778 | E18C: ;
E323 B1A00C8 | 779 | CMP DX,0C400H ; TOP OF VIDEO ROM AREA YET?
E327 7C44 | 780 | JL E16A ; GO SCAN FOR ANOTHER MODULE
781 | ;-----------------------------------------------
782 ; 0159 INTERRUPT CONTROLLER TEST :
783 ; DESCRIPTION :
784 ; READ/WRITE THE INTERRUPT MASK REGISTER (IMR) :
785 ; WITH ALL ONES AND ZEROS, ENABLE SYSTEM :
786 ; INTERRUPTS. MASK DEVICE INTERRUPTS OFF. CHECK :
787 ; FOR NOT INTERRUPTS (UNEXPECTED). :
788 | ;-----------------------------------------------
789 | ASSUME DS:AB50
E329 IF | 790 | C21: ; POP DS
791 | ;----- TEST THE IMR REGISTER
793 | ;----- TEST THE IMR REGISTER
E32A C606150405 | 794 | C21A: MOV DATA_AREA[OFFSET MF_ERR_FLAG],00H
795 | ; <<<<<<<<><<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
796 | ; <<<<<<<<CHECKPOINT 5<<<<<<<<<<<
E32F B000 | 797 | MOV AL,0 ; SET IMR TO ZERO
E331 E621 | 798 | MOV OUT INTADD,AL
E333 E621 | 799 | IN AL,INTADD ; READ IMR
E335 0A0C | 800 | OR AL,AL ; IMR = 0?
E337 751B | 801 | JNZ D6 ; GO TO ERROR ROUTINE IF NOT 0
E339 B0FF | 802 | MOV AL,OFFH ; DISABLE DEVICE INTERRUPTS
E33B E621 | 803 | MOV OUT INTADD,AL ; WRITE TO IMR
E33D E621 | 804 | IN AL,INTADD ; READ IMR
E33F 0401 | 805 | ADD AL,1 ; ALL IMR BIT ON?
E341 7511 | 806 | JNZ D6 ; NO - GO TO ERR ROUTINE
807 ;----- CHECK FOR NOT INTERRUPTS
808 | ;----- CHECK FOR NOT INTERRUPTS
809 | ;----- INTERRUPTS ARE MASKED OFF. CHECK THAT NO INTERRUPTS OCCUR.
810 ;----- INTERRUPTS ARE MASKED OFF. CHECK THAT NO INTERRUPTS OCCUR.
E343 A26804 | 812 | MOV DATA_AREA[OFFSET INTF_FLAG],AL ; CLEAR INTERRUPT FLAG
E344 FB | 813 | STI ; ENABLE EXTERNAL INTERRUPTS
E347 2BC9 | 814 | SUB CX,CX ; WAIT 1 SEC FOR ANY INTRS THAT
E349 | 815 | D4: ;
E349 E2FE | 816 | LOOP D4 ; MIGHT OCCUR
E34B | 817 | D5: ;
E34B E2FE | 818 | LOOP D5
E350 00360D0400 | 819 | CMP DATA_AREA[OFFSET INTF_FLAG],00H ; DID ANY INTERRUPTS OCCUR?
E352 7409 | 820 | JZ D7 ; NO - GO TO NEXT TEST
E354 | 821 | D6: ;
E354 BFFFF90 | 822 | MOV SI,OFFSET E0 ; DISPLAY 101 ERROR
--- | --- | ---
A-12 System BIOS
E350 EHE16 823 CALL E_MSG
E35B FA 824 CLI
E35C F4 825 HLT ; HALT THE SYSTEM

E35D 07:
E35E C06650402 033 MOV DATA_AREA[OFFSET MFR_ERR_FLAG], BH
E35F 834 ; <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
E360 B0E 835 ; <<<<<<TIMER CHECKPOINT (2)>>>>>
E361 E12 836 MOV AL,0FEH ; MASK ALL INTRS EXCEPT LVL 0
E362 E621 837 OUT 004H,AL ; WRITE the 8253 IMR
E362 E620 838 MOV AL,000010000B ; SEL TIM 0, LSB, MODE 0, BINARY
E363 E143 839 OUT TIPCTL,AL ; WRITE TIMER CONTROL MODE REG
E364 B91600 840 MOV CX,16H ; SET PGM LOOP CNT
E365 D0C1 841 MOV AL,CL ; SET TIMER 0 CNT REG
E366 E040 842 OUT TIMER0,AL ; WRITE TIMER 0 CNT REG

E371 F06660401 843 DB: TEST DATA_AREA[OFFSET INTR_FLAG], DH
E372 844 ; DID TIMER 0 INTERRUPT OCCUR?
E373 7504 845 JNZ D9 ; YES - CHECK TIMER OP FOR SLOW TIME
E374 E2F 846 LOOP D8 ; WAIT FOR INTR FOR SPECIFIED TIME
E375 E008 847 JNP D6 ; TIMER 0 INTR D2DN T OCCUR - ERR
E376 E099 848 MOV AL,12H ; SET PGM LOOP CNT
E377 E0FF 851 MOV AL,OFFH ; WRITE TIMER 0 CNT REG
E378 E629 852 OUT TIMER0,AL ; WRITE TIMER 0 CNT REG
E379 E056 853 MOV DATA_AREA[OFFSET INTR_FLAG], O ; RESET INTR RECEIVED FLAG
E37A 854 MOV AL,OFFH ; REENABLE TIMER 0 INTERRUPTS
E37B E621 855 OUT 004H,AL
E37C E056 856 DIO: TEST DATA_AREA[OFFSET INTR_FLAG], DH ; DID TIMER 0 INTERRUPT OCCUR?
E37D 75C2 857 JNZ D6 ; YES - TIMER CNTING TOO FAST, ERR
E37E E2F 859 LOOP D10 ; WAIT FOR INTR FOR SPECIFIED TIME

E37F 860 ;----- SETUP TIMER 0 TO MODE 3
E380 862 E0FF 863 MOV AL,OFFH ; DISABLE ALL DEVICE INTERRUPTS
E381 864 E621 865 MOV DATA_AREA, AL ; INITIALIZE
E382 866 E620 867 MOV OUT ADDR,AL
E383 868 E643 869 MOV TIMER+,AL
E384 86A 870 MOV AL,0
E385 E640 871 OUT TIMER-,AL
E386 872 MOV AL,0
E387 873 OUT TIMER+,AL
E388 874 MOV AL,0
E389 875 OUT TIMER-,AL

E390 876 ;----- KEYBOARD TEST
E391 877 E3A 878 JS CONTROL_switch
E392 879 JS CONTROL_KEY
E393 880 JS CONTROL_KEY
E394 881 JS CONTROL_KEY
E395 882 JS CONTROL_KEY
E396 883 JS CONTROL_KEY
E397 884 JS CONTROL_KEY
E398 885 JS CONTROL_KEY
E399 886 JS CONTROL_KEY
E400 887 JS CONTROL_KEY
E401 888 JS CONTROL_KEY
E402 889 JS CONTROL_KEY
E403 890 JS CONTROL_KEY
E404 891 JS CONTROL_KEY

E405 892 ;----- CHECK FOR STUCK KEYS
E406 893 E3C8 894 JS CONTROL_KEY
E407 895 JS CONTROL_KEY
E408 896 JS CONTROL_KEY
E409 897 JS CONTROL_KEY
E410 898 JS CONTROL_KEY
E411 899 JS CONTROL_KEY

Appendix A

System BIOS A-13
E3C OBJ | LINE | SOURCE
---|---|---
E3CE E3FE | 900 | LOOP F5 ; DELAY FOR A WHILE
E3CE E400 | 901 | IN AL,KBD_IN ; CHECK FOR STUCK KEYS
E3D0 3C00 | 902 | CMP AL,0 ; SCAN CODE = 0?
E3D0 740A | 903 | JE F7 ; YES - CONTINUE TESTING
E3D4 E00415 | 904 | CALL XPCBYTE ; CONVERT AND PRINT
E3D7 BE4ECE90 | 905 | F6: MOV SI,OFFSET F1 ; GET MSG ADDR
E3D8 E0CB15 | 906 | CALL E_MSG ; PRINT MSG ON SCREEN
E3D8 E0CB15 | 907 | F6: MOV SI,OFFSETVECTOR_TABLE ; SETUP HARDWARE INT. VECTOR TABLE
E3D8 E0CB15 | 908 | --------------------------------------------
E3DE | 911 | F7:
E3DE 1E | 912 | PUSH DS ; SETUP_INT_TABLE:
E3DF 20C0 | 913 | SUB AX,AX
E3E1 00C0 | 914 | MOV ES,AX
E3E3 D0000 | 915 | MOV CX,0B ; GET VECTOR CNT
E3E6 0E | 916 | PUSH CS ; SETUP DS SEG REG
E3E7 1F | 917 | POP DS
E3E8 B53FE0 | 918 | MOV SI,OFFSET VECTOR_TABLE
E3EC B5200 | 919 | MOV DI,OFFSET_INT_PTR
E3EF 20F7A | 920 | MOV AX,9004
E3F0 47 | 921 | INC DI ; SKIP OVER SEGMENT
E3F1 47 | 922 | INC DI
E3F2 E3FB | 923 | LOOP F7A
E3F4 1F | 924 | POP DS
E3F6 20 | 925 | --------------------------------------------
E3F5 C7060000FF8 | 927 | MOV MNI_PTR,OFFSET MNI_INT ; MNI INTERRUPT
E3FB C7061000FF | 930 | MOV INTS_PTR,OFFSETPRINT_SCREEN ; PRINT SCREEN
E401 C7062000FF6 | 931 | MOV BASIC_PTR+2,0F600H ; SEGMENT FOR CASSETTE BASIC
E402 C70640 | 932 | SET UP OTHER INTERRUPTS AS NECESSARY
E407 8031200401 | 933 | CMP DATA_AREA,OFFSET HFG_TST1,01H ; HFG. TEST MODE?
E40C C7050A | 934 | JNZ EXP.IO
E40E C7070003CF | 937 | MOV WORD_PTR,OFFSETBLINK_INT ; SETUP TIMER INT TO BLINK LED
E414 B0FE | 938 | MOV AL,OFEH ; ENABLE TIMER INTERRUPT
E416 E621 | 939 | OUT INTA01,AL
E417 50 | 940 | --------------------------------------------
E418 0003H | 941 | EXP.IO: ; (CARD WAS ENABLED EARLIER)
E41B DA1002 | 942 | MOV DX,00210H ; CONTROL PORT ADDRESS
E41B D05555 | 943 | MOV AX,5555H ; SET DATA PATTERN
E41E EE | 944 | OUT DX,AL
E41F B001 | 945 | MOV AL,01H ; MAKE AL DIFFERENT
E421 EC | 946 | IN AL,DX ; RECOVER DATA
E422 3AC9 | 947 | CMP AL,AL ; REPLY?
E424 7544 | 948 | JNE E19 ; NO RESPONSE, GO TO NEXT TEST
E426 F700 | 949 | NDT AX ; MAKE DATA=AAAA
E428 EE | 950 | OUT DX,AL
E429 B001 | 951 | MOV AL,01H
E42B EC | 952 | IN AL,DX ; RECOVER DATA
E42C 3ACA | 953 | CMP AL,AL
E42E 753A | 954 | JNE E19
E42E 753A | 955 | --------------------------------------------
E430 | 956 | EXP2: ; (CARD WAS ENABLED EARLIER)
E430 B00100 | 957 | MOV BX,0001H
E433 1A1502 | 958 | MOV DX,0015H ; LOAD HI ADDR. REG ADDRESS
E436 E91000 | 959 | MOV CX,0016 ; GO ACROSS 16 BITS
E439 | 960 | --------------------------------------------
E439 2E8007 | 961 | MOV CS:[BX],AL ; WRITE ADDRESS F0000+DX
E43C 90 | 962 | NOP
E43D EC | 963 | IN AL,DX ; READ ADDR. HIGH
E43E 3AC7 | 964 | CMP AL,AL
E440 7521 | 965 | JNE EXP_ERR ; GO ERROR IF MISCOMPARE
E442 42 | 966 | INC DX
E442 42 | 967 | --------------------------------------------

A-14 System BIOS
Appendix A

System BIOS A-15
Appendix A

System BIOS

A-17
LOC OBJ | LINE | SOURCE
---|---|---
ES87 B0414 | 1200 | MOV AX,1414H ; DEFAULT=20
ES8A AB | 1209 | STOSW
ES8B AB | 1210 | STOSW
ES8C B00101 | 1211 | MOV AX,0101H ; RS232_DEFAULT=01
ES8F AB | 1212 | STOSW
ESC A AB | 1213 | STOSW
ESC1 F43 | 1214 | IN AL,INTA01
ESC2 F4C | 1215 | AND AL,0FCH ; ENABLE TIMER AND KB INTS
ESC5 E621 | 1216 | OUT INTA01,AL
ESC7 E3F000 | 1217 | CHK BP,0000H ; CHECK FOR BP= NON-ZERO
ESC8 | 1218 | ; ERROR HAPPENS
ESCA 7419 | 1219 | JE FISA_0 ; CONTINUE IF NO ERROR
ESCC BA0200 | 1220 | MOV DX,2 ; 2 SHORT BEEPS (ERROR)
ESCF E80614 | 1221 | CALL ERR_BEEP
ESD0 BE09E890 | 1222 | MOV SI,OFFSET F3D ; LOAD ERROR MSG
ESD6 E0F113 | 1223 | CALL P_MSG
ESD9 | 1224 | ERR_WAIT: ;
ESD0 B400 | 1225 | MOV AH,00 ;
ESD0 CD16 | 1226 | INT 16H ; WAIT FOR 'FI' KEY
ESD0 D8FC | 1227 | CMP AH,5BH
ESD0 75F7 | 1228 | JNE ERR_WAIT ;
ESD2 E0E090 | 1229 | JMP FISA ; BYPASS ERROR
ESD5 | 1230 | FISA_0: ;
ESD5 03E120001 | 1231 | CMP MFG_TST,1 ; MFG MODE
ESD6 7A46 | 1232 | JE FISA ; BYPASS BEEP
ESDA BA0100 | 1233 | MOV DX,1 ; 1 SHORT BEEP (NO ERRORS)
ESEF E0613 | 1234 | CALL ERR_BEEP
ESF2 A0100 | 1235 | FISA: MOV AL,BYTE PTR EQUIP_FLAG ; GET SWITCHES
ESF5 2401 | 1236 | AND AL,00000001B ; 'LOOP POST' SWITCH ON
ESF7 7503 | 1237 | JNZ FISB ; CONTINUE WITH BRING-UP
ESF9 E9F9FA | 1238 | JMP START
ESFC 2AE0 | 1239 | FIS0: SUB AL,AL
ESFE A04900 | 1240 | MOV AL,CRT_MODE
E601 CD10 | 1241 | INT 10H ; CLEAR SCREEN
E603 | 1242 | FISC: ;
E603 BBADW990 | 1243 | MOV BP,OFFSET F4 ; PTR_SRC_TBL
E607 BE0000 | 1244 | MOV SI,0
E60A | 1245 | F16: ; PTR_BASE:
E60A 0EB85600 | 1246 | MOV DX,CS:[BP] ; GET PRINTER BASE ADDR
E60C 0DA4 | 1247 | MOV AL,AAAH ; WRITE DATA TO PORT A
E610 EE | 1248 | OUT DX,AL
E611 IE | 1249 | PUSH DS ; BUS SETTLING
E612 EC | 1250 | IN AL,DX ; READ PORT A
E613 IF | 1251 | POP DS
E614 7C5A | 1252 | CMP AL,AAAH ; DATA PATTERN SAME
E616 7505 | 1253 | JNE F17 ; NO - CHECK NEXT PTR CD
E618 9F6408 | 1254 | MOV PRINTER_BASE:[SI],DX ; YES - STORE PTR BASE ADDR
E618 46 | 1255 | INC SI ; INCREMENT TO NEXT WORD
E61C 46 | 1256 | INC SI
E61D | 1257 | F17: ;
E61D 45 | 1258 | INC BP ; POINT TO NEXT BASE ADDR
E61E 45 | 1259 | INC BP
E61F 8F0AF9 | 1260 | CMP BP,OFFSET F4 ; ALL POSSIBLE ADDRS CHECKED?
E623 75E9 | 1261 | JNE F16 ; PTR_BASE
E625 BA0000 | 1262 | MOV DX,0 ; POINTER TO RS232 TABLE
E625 BAF0A3 | 1263 | MOV DX,3F9H ; CHECK IF RS232 CD 1 ATTACH?
E62B EC | 1264 | IN AL,DX ; READ INTR ID REG
E62C 80F8 | 1265 | TEST AL,OF0H
E62E 5706 | 1266 | JNZ F10
E630 C707F003 | 1267 | MOV RS232_BASE[BX+1],3F9H ; SETUP RS232 CD #1 ADDR
E634 43 | 1268 | INC BX
E635 43 | 1269 | INC BX
E636 | 1270 | F16: ;
E636 BAF0A2 | 1271 | MOV DX,3F9H ; CHECK IF RS232 CD 2 ATTACH
E639 EC | 1272 | IN AL,DX ; READ INTERRUPT ID REG
E63A 80F8 | 1273 | TEST AL,OF0H
E63C 7506 | 1274 | JNZ F19 ; BASE_END
E63E C707F002 | 1275 | MOV RS232_BASE[1],2F9H ; SETUP RS232 CD #2
E642 43 | 1276 | INC BX
E643 43 | 1277 | INC BX
E643 | 1278 | ;----- SETUP EQUIP FLAG TO INDICATE NUMBER OF PRINTERS AND RS232 CARDS
E644 | 1279 | i-----
E644 | 1280 | BASE_END;
E644 8BC6 | 1282 | MOV AX,SI ; SI HAS #2 NUMBER OF RS232
E646 B103 | 1283 | MOV CL,3 ; SHIFT COUNT
E648 DC68 | 1284 | ROR AL,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
E655 90  1089   NOP
E654 90  1090   NOP
E655 90  1091   NOP
E656 A0F  1092   TEST AL,0FH
E658 7505  1093   JNZ F20 ; NO_GAME_CARD
E65F B001110010  1094   OR BYTE PTR EQUIP_FLAG+1,AL
E65F 1283  1095   F20: ; NO_GAME_CARD:
E661 E66F  1096 ;------ ENABLE NMI INTERRUPTS
E663 E661  1299   IN AL,PORT_B ; RESET CHECK ENAB
E666 OC30  1300   OR AL,30H
E66B 24CF  1302   AND AL,0FH
E667 E661  1303   OUT PORT_B,AL
E668 B050  1304   MOV AL,ADH ; ENABLE NMI INTERRUPTS
E669 E640  1305   OUT GA0H,AL
E66D E66F  1306   F21: ; LOAD_BOOT_STRAP:
E66F CD19  1307   INT 19H ; GO TO THE BOOT LOADER
E670 2019  1308

******************************************************************************

E66F STGSTST_CNT PROC NEAR
E66F FC  1325   CLO ; SET DIF FLAG TO INCREMENT
E670 2BFF  1326   SUB DI,DI ; SET DI-OFFSET 0 REL TO ES REG
E672 2BC0  1327   SUB AX,AX ; SETUP FOR 0-FFFF PATTERN TEST
E676 2C24  1328   C2_1:
E677 B005  1329   MOV [DI],AL ; ON FIRST BYTE
E676 B005  1330   MOV AL,[DI]
E678 32C4  1331   XOR AL,AL ; O.K.?
E67A 7540  1332   JNZ C7 ; GO ERROR IF NOT
E67C E8C4  1333   JNC AH
E67E B0C4  1334   MOV AL,AH
E680 75F2  1335   JNZ C2_1 ; LOOP TILL WRAP THROUGH FF
E682 8009  1336   MOV BX,CX
E684 D1E3  1337   SHL BX,1 ; CONVERT TO A BYTE COUNT
E686 0A4AAA  1338   MOV AX,0AAA
E689 B55FF  1339   MOV DX,OFFSH ; SETUP OTHER DATA PATTERNS TO USE
E68C F3  1340   REP STOSW ; FILL STORAGE LOCATIONS IN BLOCK
E68D A0  1341
E68E E661  1342   IN AL,PORT_B ; TOGGLE PARITY CHECK LATCHES
E690 0C30  1343   OR AL,0011000B
E692 E661  1344   OUT PORT_B,AL
E694 90  1345   NOP
E695 24CF  1346   AND AL,1100111B
E697 E661  1347   OUT PORT_B,AL
E699 1348   C3: DEC DI ; POINT TO LAST BYTE JUST WRITEN
E69A FD  1349   STD ; SET DIF FLAG TO GO BACKWARDS
E69B 1350  1350  C4: ; INITIALIZE DESTINATION POINTER
E69D B0F7  1351   MOV SI,DI
E69D B0CD  1352   MOV CX,BX
E69F C5:  1353   SETUP BYTE COUNT FOR LOOP
E69F AC  1354   LOOPSB
E6AD 32C4  1355   XOR AL,AH ; READ OLD TEST BYTE FROM STORAGE [SI]E6AD 32
E6A2 7525  1356   JNE C7 ; DATA READ AS EXPECTED?
E6A4 BAC2  1357   MOV AL,DL ; NO - GO TO ERROR ROUTINE
E6A6 AA  1358   STOSB ; WRITE INTO LOC JUST READ [DI+1]
E6AB E2F6  1359   LOOP CS ; DECREMENT BYTE COUNT AND LOOP CX
E6A9 22E4  1360   AND AH,AH ; ENDING ZERO PATTERN WRITTEN TO STG?
E6AB 7416  1362   JZ O6X ; YES - RETURN TO CALLER WITH AL=0

******************************************************************************

Appendix A
LOC OBJ   LINE   SOURCE
E6AD B4E0 1363 MOV AH,AL ; SETUP NEW VALUE FOR COMPAR
E6AF 06F2 1364 XCHG DH,DL ; MOVE NEXT DATA PATTERN TO DL
E6B1 22E4 1365 AND AH,AH ; READING ZERO PATTERN THIS PASS?
E6B3 7504 1366 JNZ C6 ; CONTINUE TEST SEQUENCE TILL ZERO DATA
E6B5 8A04 1367 MOV DL,AL ; ELSE SET ZERO FOR END READ PATTERN
E6B7 E8E0 1368 JMP C3 ; AND MAKE FINAL BACKWARDS PASS
E6B9 1369 C6: ;
E6BA FC 1370 CLD ; SET DIR FLAG TO GO FORWARD
E6B4 47 1371 INC DI ; SET POINTER TO BEGIN LOCATION
E6B8 740E 1372 JZ C4 ; READ/WRITE FORWARD IN STG
E6B0 4F 1373 DEC DI ; ADJUST POINTER
E6B6 B100 1374 MOV DX,00000H ; SETUP 01 FOR PARITY BIT AND 00 FOR END
E6C1 E006 1375 JMP C3 ; READ/WRITE BACKWARD IN STG
E6C3 1376 C6X: ;
E6C3 E662 1377 IN AL,PORT_C ; DID A PARITY ERROR OCCUR?
E6C5 2400 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: ;
E6C9 FC 1381 CLD ; SET DIRECTION FLAG TO INC
E6CA C3 1382 RET
E6C3 1383 STGSTST_CNT ENDP
E6C8 1384 ;
E6CB 1385 PRINT ADDRESS AND ERROR MESSAGE FOR ROM CHECKSUM ERRORS ;
E6CA 1386 i---------------------------------------------------------------------
E6CB 1387 ROM_ERR PROC NEAR
E6CC 52 1388 PUSH DX ; SAVE POINTER
E6CD 50 1389 PUSH AX
E6CE 80 1390 MOV DS,0S ; GET ADDRESS POINTERS
E6D4 2A08361500 1391 MOV ES:SEG_ERR_FLAGS,DX ; <<<>>><<<>>><<<>>><<<>>><<<>>><<<
E6D4 01 1392 MOV ES:SEG_ERR_FLAGSAGAIN,DX ; <<<>>><<<>>><<<>>><<<>>><<<>>><<<
E6D4 8FAC0B 1393 CMP DX,0C000H ; CRT CARD IN ERROR?
E6D5 7C0D 1394 JL ROM_ERR_BEEP ; GIVE CRT CARD FAIL BEEP
E6D6 E6FDB8 1395 CALL PTR_REG ; PRINT SEGMENT IN ERROR
E6D7 0E0F900 1396 MOV SI,OFFSET F3A ; DISPLAY ERROR MSG
E6E1 E8C12 1397 CALL E_MSG
E6E4 1398 RET
E6E5 5A 1399 POP AX
E6E5 4A 1400 POP DX
E6E6 C3 1401 RET
E6E7 1402 ROM_ERR_BEEP:
E6E7 B42201 1403 MOV DX,0102H ; BEEP 1 LONG, 2 SHORT
E6EA E0EB12 1404 CALL ERR_BEEP
E6EB EDFS 1405 JMP SHORT ROM_ERR_END
E6EC 1406 ;
E6F0 1407 ;
E6F2 1408 ;
E6F2 1409 BOOT STRAP LOADER ;
E6F2 FB 1410 ;
E6F3 28C0 1411 STI ; ENABLE INTERRUPTS
E6F3 E6E5 1412 SUB AX,AX ; ESTABLISH ADDRESSING
E6F6 E8E0 1413 MOV DS,AX
E6F7 1414 ;
E6F7 C7067B00C7EF 1415 --- RESET THE DISK PARAMETER TABLE VECTOR
E6F0 BDE7AD0 1416 MOV WORD PTR_DISK_POINTER,OFFSET_DISK_BASE
E6F9 E8E0 1417 MOV WORD PTR_DISK_POINTER+2,CS
E6F9 1418 ;
E6F9 E8BF 1419 --- LOAD SYSTEM FROM DISKETTE -- CK HAS RETRY COUNT
E6F9 1420 ;
E6FA B90400 1421 MOV CX,4 ; SET RETRY COUNT
E6FB 1422 HI: ; IPL_SYSTEM
E6FC 51 1423 PUSH CX ; SAVE RETRY COUNT
E6FD B400 1424 MOV AH,0 ; RESTART THE DISKETTE SYSTEM
E6FE CD13 1425 INT 13H ; DISKETTE_ID
E6FF 720F 1426 JC HZ ; IF ERROR, TRY AGAIN
E6F0 B40102 1427 MOV AX,201H ; READ IN THE SINGLE SECTOR
E6F0 2002 1428 SUB DX,DX ; TO THE BOOT LOCATION

A-20 System BIOS
IPL WAS SUCCESSFUL

E710 B007C 1440 MOV ES,BX
E712 B007C 1441 MOV BX,OFFSET BOOT_LOC0
E715 B001D 1442 ; DRIVE 0, HEAD 0
E715 B0113 1443 MOV CX,1 ; SECTOR 1, TRACK 0
E716 C013 1444 INT 13H ; DISKETTE IO
E717 E71A 1445 H2: ; DO IT FOR RETRY TIMES
E718 E71A 1446 POP CX ; RECOVER RETRY COUNT
E719 E719 1447 JNC H4 ; CF SET BY UNSUCCESSFUL READ
E71D 1448 LOOP H1
E71E E725 1449 ;----- UNABLE TO IPL FROM THE DISKETTE
E71F E726 1450 ;----- IPL Was SUCCESSFUL
E721 E721 1451
E721 1452 M3: ; GO TO RESIDENT BASIC
E721 C018 1453 INT 1BH
E721 1454 ;----- INT 14
E721 E726 1455
E721 E726 1456 ;----- INT 16
E721 1457
E721 E72C 1458 JMP BOOT_LOC0
E721 1459 BOOTSTRAP ENDP
E721 1460
E721 E72C 1461 ;------- INT 16-------------------------------
E721 E72C 1462 ; RS232_TO
E721 E72C 1463 ; THIS ROUTINE PROVIDES BYTE STREAM I/O TO THE COMMUNICATIONS
E721 E72C 1464 ; PORT ACCORDING TO THE PARAMETERS:
E721 E72C 1465 ; (AH)=D INITIALIZE THE COMMUNICATIONS PORT
E721 E72C 1466 ; (AL) HAS PARAMETERS FOR INITIALIZATION
E721 E72C 1467 ;
E721 E72C 1468 ; 7 6 5 4 3 2 1 0
E721 E72C 1469 ;------ BAUD RATE -- -PARITY-- STOPBIT =-WORD LENGTH--
E721 E72C 1470 ; 00 = 110 X0 = NONE 6 - 1 10 - 7 BITS
E721 E72C 1471 ; 001 - 150 01 = ODD 1 - 2 11 - 8 BITS
E721 E72C 1472 ; 010 - 300 11 = EVEN
E721 E72C 1473 ; 011 - 600
E721 E72C 1474 ; 100 - 1200
E721 E72C 1475 ; 101 - 2400
E721 E72C 1476 ; 110 - 4800
E721 E72C 1477 ; 111 - 9600
E721 E72C 1478 ;
E721 E72C 1479 ; ON RETURN, CONDITIONS SET AS IN CALL TO COMM STATUS (AH=3)
E721 E72C 1480 ; (AH)=3 SEND THE CHARACTER IN (AL) OVER THE COMM LINE
E721 E72C 1481 ; (AL) REGISTER IS PRESERVED
E721 E72C 1482 ; ON EXIT, BIT 7 OF AH IS SET IF THE ROUTINE WAS UNABLE
E721 E72C 1483 ; TO TRANSMIT THE BYTE OF DATA OVER THE LINE.
E721 E72C 1484 ; IF BIT 7 OF AH IS NOT SET, THE REMAINDER OF AH
E721 E72C 1485 ; IS SET AS IN A STATUS REQUEST, REFLECTING THE
E721 E72C 1486 ; CURRENT STATUS OF THE LINE.
E721 E72C 1487 ; (AH)=2 RECEIVE A CHARACTER IN (AL) FROM COMM LINE BEFORE
E721 E72C 1488 ; RETURNING TO CALLER
E721 E72C 1489 ; ON EXIT, AH HAS THE CURRENT LINE STATUS, AS SET BY THE
E721 E72C 1490 ; THE STATUS ROUTINE, EXCEPT THAT THE ONLY BITS
E721 E72C 1491 ; LEFT ON ARE THE ERROR BITS (7,4,3,2,1)
E721 E72C 1492 ; IF AH HAS BIT 7 ON (TIME OUT) THE REMAINING
E721 E72C 1493 ; BITS ARE NOT PREDICTABLE.
E721 E72C 1494 ; THUS, AH IS NON ZERO ONLY WHEN AN ERROR
E721 E72C 1495 ; OCCURRED.
E721 E72C 1496 ; (AH)=3 RETURN THE COMM PORT STATUS IN (AX)
E721 E72C 1497 ; AH CONTAINS THE LINE STATUS
E721 E72C 1498 ; BIT 7 = TIME OUT
E721 E72C 1499 ; BIT 6 = TRANS SHIFT REGISTER EMPTY
E721 E72C 1500 ; BIT 5 = TRAN HOLDING REGISTER EMPTY
E721 E72C 1501 ; BIT 4 = BREAK DETECT
E721 E72C 1502 ; BIT 3 = FRAMING ERROR
E721 E72C 1503 ; BIT 2 = PARITY ERROR
E721 E72C 1504 ; BIT 1 = OVERRUN ERROR
E721 E72C 1505 ; BIT 0 = DATA READY
E721 E72C 1506 ; AL CONTAINS THE MODE STATUS
E721 E72C 1507 ; BIT 7 = RECEIVED LINE SIGNAL DETECT
E721 E72C 1508 ; BIT 6 = RING INDICATOR
E721 E72C 1509 ; BIT 5 = DATA SET READY
E721 E72C 1510 ; BIT 4 = CLEAR TO SEND
E721 E72C 1511 ; BIT 3 = DELTA RECEIVE LINE SIGNAL DETECT
E721 E72C 1512 ; BIT 2 = TRAILING EDGE RING DETECTOR
E721 E72C 1513 ; BIT 1 = DELTA DATA SET READY
E721 E72C 1514 ; BIT 0 = DELTA CLEAR TO SEND
E721 E72C 1515 ;
E721 E72C 1516 ; (DX) = PARAMETER INDICATING WHICH RS232 CARD (0,1 ALLOWED)
VECTOR TO APPROPRIATE ROUTINE

ASSUME CS:CODE,DS:DATA

ORG 0E720H

DL 1047 ; 110 BAUD
DH 768 ; TEST FOR 0 BASE ADDRESS
DI 300

; DATA AREA RS232_BASE CONTAINS THE BASE ADDRESS OF THE RS250 ON THE
; CARD LOCATION 400H CONTAINS UP TO 4 RS232 ADDRESSES POSSIBLE
; DATA AREA LABEL RS232_TIMOUT (BYTE) CONTAINS OUTER LOOP COUNT
; VALUE FOR TIMEOUT (DEFAULT=1)

; OUTPUT

; AX MODIFIED ACCORDING TO PARMS OF CALL

; ALL OTHERS UNCHANGED

;-----------------------------
E729 A5 LABEL WORD ; TABLE OF INIT VALUES

A729 1704 DW 1047 ; 110 BAUD
A729 1530 DW 768 ; TEST FOR 0 BASE ADDRESS
A729 0001 DW 300

E72C COOH DW 192 ; 600
E731 4000 DW 96 ; 1200
E733 3000 DW 48 ; 2400
E735 1000 DW 24 ; 4800
E737 0000 DW 12 ; 9600

E739 RS232_TO PROC FAR

;----- VECTOR TO APPROPRIATE ROUTINE

E73A 1E Push DS ; SAVE SEGMENT
E73C 56 Push SI
E73D 57 Push DI
E73E 51 Push CX
E73F 53 Push BX

E740 8BF2 MOV SI,DX ; RS232 VALUE TO SI
E742 8BF0 MOV DI,DX
E744 8166 SHL SI,1 ; WORD OFFSET
E746 E8013 CALL DD5
E749 0015 MOV DX,RS232_BASE[SI] ; GET BASE ADDRESS
E74B 0005 OR DX,DX
E74D 7013 JC A3 ; RETURN
E74F 0AE4 OR AH,AH ; TEST FOR (AH)=0
E751 7416 JNZ JS ; COMMUN INIT
E753 FEEC DEC AH ; TEST FOR (AH)=1
E755 7469 JS JE ; SEND AL
E757 FEEA DEC AH ; TEST FOR (AH)=2
E759 746A JZ A12 ; RECEIVE INTO AL
E75B 1562 A2:
E75D FEEC DEC AH ; TEST FOR (AH)=3
E75F 7503 JNZ A3
E761 84300 JMP A16 ; COMMUNICATION STATUS
E762 1566 A3: ; RETURN FROM RS232
E764 5B POP BX
E766 59 POP CX
E768 5F POP DI
E76A 5E POP SI
E76C 5A POP DX
E76E 1F POP DS
E770 CF IRET ; RETURN TO CALLER, NO ACTION

;----- INITIALIZE THE COMMUNICATIONS PORT

E772 1575 ; RETURN TO CALLER, NO ACTION

E774 1575 ; INITIALIZE THE COMMUNICATIONS PORT

E776 1576 ; INITIALIZE THE COMMUNICATIONS PORT

E778 1577 A4: ; SAVE INIT PARMS IN AH
E77A 3C203 ADD DX,3 ; POINT TO 8250 CONTROL REGISTER
E77E 8000 MOV AL,60H
E780 EE OUT DX,AL ; SET DLAB=1

;----- DETERMINE BAUD RATE DIVISOR

E782 1583 ; DETERMINE BAUD RATE DIVISOR

E784 1584 ; DETERMINE BAUD RATE DIVISOR

E787 0167 MOV AH,AL ; GET PARMS TO DL
E787 664 MOVL CL,4
E787 0CCH MOVL DL,CL
E787 0E00 ADD DX,DX
E788 0F07 MOV DI,OFFSET A1 ; BASE OF TABLE
E78A 03FA ADD DI,DX ; PUT INTO INDEX REGISTER
E78C 4805 MOV DX,RS232_BASE[SI] ; POINT TO HIGH ORDER OF DIVISOR
E78E 42 MOV DX
E78F 28AASO MOV AL,CS:[DI]+1 ; GET HIGH ORDER OF DIVISOR

A-22 System BIOS
E797 EE 1594 OUT DX,AL ; SET MS OF DIV TO 0
E798 4A 1595 DEC DX
E799 2EA05 1596 MOV AL,OC:IDII ; GET LOW ORDER OF DIVISOR
E79C EE 1597 OUT DX,AL ; SET LOW OF DIVISOR
E79D OC203 1598 ADD DX,3
E79E 8AC4 1599 MOV AL,AH ; GET PARMS BACK
E79F 24F 1600 AND AL,OIFN ; STRIP OFF THE BAUD BITS
E7A1 EE 1601 OUT DX,AL ; LINE CONTROL TO 0 BITS
E7A3 4A 1602 DEC DX
E7A4 4A 1603 DEC DX
E7A5 B000 1604 MOV AL,0
E7A6 EE 1605 OUT DX,AL ; INTERRUPT ENABLES ALL OFF
E7A7 EB49 1606 JMP SHORT A18 ; COM_STATUS
E7A8 7400 1607
E7A9 1608 ;---- SEND CHARACTER IN (AL) OVER COMMO LINE
E7AC 1609
E7A1 1610 A5: PUSH AX ; SAVE CHAR TO SEND
E7A2 1611 MOV AL,CL ; MODM CONTROL REGISTER
E7A3 B003 1612 MOV AL,3 ; DTR AND RTS
E7A4 EE 1613 OUT DX,AL ; DATA TERMINAL READY, REQUEST TO SEND
E7A5 42 1614 INC DX ; MODMEM STATUS REGISTER
E7A6 42 1615 INC DX
E7A7 B730 1616 MOV BH,30H ; DATA SET READY & CLEAR TO SEND
E7A8 E6000 1617 CALL WAIT_FOR_STATUS ; ARE BOTH TRUE
E7AA 7400 1618 JE A9 ; YES, READY TO TRANSMIT CHAR
E7AB 1619
E7AC 1620 A7: POP CX ; LOAD DATA BYTE
E7AF 8CC00 1621 MOV AL,CL ; RELOAD DATA BYTE
E7B0 EB8E 1622 JMP A3 ; RETURN
E7B1 1623
E7B2 1624 OR AH,60H ; INDICATE TIME OUT
E7B3 EE 1625 JMP A3 ; RETURN
E7B4 1626 A9: MOV DL,0 ; CLEAR_TO_SEND
E7B5 4A 1627 DEC DX ; LINE STATUS REGISTER
E7B6 1628 A10: MOV BH,20H ; WAIT_SEND
E7B7 B720 1629 MOV BH,20H ; IS TRANSMITTER READY
E7B8 E6000 1630 CALL WAIT_FOR_STATUS ; TEST FOR TRANSMITTER READY
E7B9 79F0 1631 JNZ A7 ; RETURN WITH TIME OUT SET
E7BA 1632 A11: MOV DL,0 ; CLEARToSend
E7BB 83EA05 1633 SUB DX,5 ; DATA PORT
E7BC 1634 POP CX ; RECOVER IN CX TEMPORARILY
E7BD 1635 MOV AL,CL ; MOVE CHAR TO AL FOR OUT, STATUS IN AH
E7BE EE 1636 OUT DX,AL ; OUTPUT CHARACTER
E7BF EB4D 1637 JMP A5 ; RETURN
E7C0 1638
E7C1 1639 ;---- RECEIVE CHARACTER FROM COMMO LINE
E7C2 1640
E7C3 BOC04 1641 A12: ADD DX,4 ; MODMEM CONTROL REGISTER
E7C4 B001 1642 MOV AL,1 ; DATA TERMINAL READY
E7C5 EE 1643 OUT DX,AL ; MODMEM CONTROL REGISTER
E7C6 42 1644 INC DX
E7C7 42 1645 INC DX
E7C8 1646 A13: MOV BH,20H ; WAIT_DSR
E7C9 B720 1647 MOV BH,20H ; DATA SET READY
E7CA E6000 1648 CALL WAIT_FOR_STATUS ; TEST FOR DSR
E7CB 750B 1649 JNZ A8 ; RETURN WITH ERROR
E7CD 1650 A15: MOV DL,0 ; WAIT_DSR_END
E7CE 4A 1651 DEC DX ; LINE STATUS REGISTER
E7CF 1652 A16: MOV BH,1 ; WAIT_RECV
E7D0 B071 1653 MOV BH,1 ; RECEIVE BUFFER FULL
E7D1 E0100 1654 CALL WAIT_FOR_STATUS ; TEST FOR REC. BUFF. FULL
E7D2 7003 1655 JNZ A8 ; SET TIME OUT ERROR
E7D3 1656
E7D4 B041E 1657 A17: MOV AH,0011110B ; TEST FOR ERR CONDITIONS ON RECCHAR
E7D5 014 1658 MOV DX,RS232_BASE[I] ; DATA PORT
E7D6 EE 1659 IN AL,DX ; GET CHARACTER FROM LINE
E7D7 EVDFF 1660 JMP A3 ; RETURN
E7D8 1661
E7D9 1662 ;---- COMDO PORT STATUS ROUTINE
E7DA 1663
E7DB 1664 A18: MOV DX,RS232_BASE[I] ; CONTROL PORT
E7DC 0B4 1665 ADD DX,5 ; CONTROL PORT
E7DD EE 1666 MOV AH,AL ; GET LINE CONTROL STATUS
E7DE 8A0D 1667 IN AL,DX ; PUT IN AH FOR RETURN
E7DF 42 1668 INC DX ; POINT TO MODMEM STATUS REGISTER
E7E0 BB14 1669 MOV AH,AL ; GET LINE CONTROL STATUS

System BIOS  A-23
; GET MODE CONTROL STATUS
JMP A3 ; RETURN

; WAIT FOR STATUS ROUTINE
; ENTRY:
; EXIT:
; ZERO FLAG ON = STATUS FOUND
; ZERO FLAG OFF = TIMEOUT.
; AH=LAST STATUS READ

BIOS 8A507C WAIT_FOR_STATUS PROC NEAR
MOV BL, RS232_TIM_OUT(1) ; LOAD OUTER LOOP COUNT

E7F6 WAIT_FOR_STATUS_PROC NEAR
E7F5 MOV BL, RS232_TIM_OUT(1) ; LOAD OUTER LOOP COUNT
E7F4 SUB CX,CX
EO01 IE RET

E804 4552524FS22E20 \E806 28524553554045 \E807 201D20265122 \E808 2044655929 \E825 00 \E824 0A

1705
1706 \E82E 0E
1707 \E802 8E
1708 \E82C 00
1709 \E82B 0A

INT 16 -------------------------------------------------------------
KEYBOARD I/O PROC FAR
STI ; INTERRUPTS BACK ON
PUSH CS ; CODE ,OS:
PUSH BX ; SAVE CURRENT OS
PUSH AX ; SAVE CURRENT DS
PUSH BX ; SAVE BX TEMPORARILY
CALL DDS ; AH=0
JZ ASCII_READ ; AH=1
DEC AX ; AH=2
JZ ASCII_STATUS ; AH=3
JZ K1 ; ASCII_READ
JMP SHORT INTIO_END ; EXIT

; READ THE NEXT ASCII CHARACTER STRUCK FROM THE KEYBOARD
; RETURN THE RESULT IN (AL), SCAN CODE IN (AH)
; SET THE Z FLAG TO INDICATE IF AN ASCII CHARACTER IS AVAILABLE
; IF ZF=0 -- NO CODE AVAILABLE
; IF ZF=1 -- CODE IS AVAILABLE
; THE NEXT CHARACTER IN THE BUFFER TO BE READ IS IN AX.
; THE ENTRY REMAINS IN THE BUFFER
; THE BIT SETTINGS FOR THIS CODE ARE INDICATED IN THE EQUATES FOR KB_FLAG
; ALL REGISTERS PRESERVED

; READ THE KEY TO FIGURE OUT WHAT TO DO
ASSUME CS:CODE,DS:DATA
LOC OBJ | LINE | SOURCE
--- | --- | ---
E642 FB | 1743 | STI ; INTERRUPTS BACK ON DURING LOOP
E643 90 | 1744 | NOP ; ALLOW AN INTERRUPT TO OCCUR
E644 FA | 1745 | CLI ; INTERRUPTS BACK OFF
E645 88 IE1A00 | 1746 | MOV BX,BUFFER_HEAD ; GET POINTER TO HEAD OF BUFFER
E646 30 1EC00 | 1747 | CMP BX,BUFFER_TAIL ; TEST END OF BUFFER
E647 74 F3 | 1748 | JZ K1 ; LOOP UNTIL SOMETHING IN BUFFER
E648 0B 07 | 1749 | MOV AX,[BX] ; GET SCAN CODE AND ASCII CODE
E649 E8 IE00 | 1750 | CALL K4 ; MOVE POINTER TO NEXT POSITION
E64A 89 IE1A00 | 1751 | MOV BUFFER_HEAD,BX ; STORE VALUE IN VARIABLE
E64B E8 IE14 | 1752 | JMP SHORT INT10_END ; RETURN

1753

E65A | 1754 | ;----- ASCII STATUS
E65B FA | 1755 | ; INTERRUPTS OFF
E65C 88 IE1A00 | 1756 | MOV BX,BUFFER_HEAD ; GET HEAD POINTER
E65D 3B IE0C00 | 1757 | CMP BX,BUFFER_TAIL ; IF EQUAL (Z=1) THEN NOTHING THERE
E65E 0B 07 | 1758 | MOV AX,[BX]
E65F F0 | 1759 | STI ; INTERRUPTS BACK ON
E660 58 | 1760 | POP BX ; RECOVER REGISTER
E661 1F | 1761 | POP DS ; RECOVER SEGMENT
E662 CA 0200 | 1762 | RET 2 ; THROW AWAY FLAGS

1763

E663 | 1764 | ;----- SHIFT STATUS
E664 | 1765 | K2:
E665 FA | 1766 | CLI ; INTERRUPTS OFF
E666 88 IE1A00 | 1767 | MOV BX,BUFFER_HEAD ; GET HEAD POINTER
E667 3B IE0C00 | 1768 | CMP BX,BUFFER_TAIL ; IF EQUAL (Z=1) THEN NOTHING THERE
E668 0B 07 | 1769 | MOV AX,[BX]
E669 F0 | 1770 | STI ; INTERRUPTS BACK ON
E66A 58 | 1771 | POP BX ; RECOVER REGISTER
E66B 1F | 1772 | POP DS ; RECOVER SEGMENT
E66C CA 0200 | 1773 | RET 2 ; THROW AWAY FLAGS

1774

E66D | 1775 | ;----- INCREMENT A BUFFER POINTER
E66E 77 | 1776 | K4:
E66F 43 | 1777 | INC BX ; MOVE TO NEXT WORD IN LIST
E670 63 | 1778 | INC BX
E671 3B IE0200 | 1779 | CMP BX,BUFFER_END ; AT END OF BUFFER?
E672 75 04 | 1780 | JNE K5 ; NO, CONTINUE
E673 88 IE0000 | 1781 | MOV BX,BUFFER_START ; YES, RESET TO BUFFER BEGINNING
E674 | 1782 | K5:
E675 | 1783 | RET

1784

E676 K3 | 1785 | KEND ; At End Of Buffer?
E677 | 1786 | ;----- TABLE OF SHIFT KEYS AND MASK VALUES
E678 | 1787 | K6:
E679 K6 | 1788 | LABEL BYTE
E67A 52 | 1789 | DB INS_KEY ; INSERT KEY
E67B 3A | 1790 | DB CAPS_KEY,NM_KEY,SCROLL_KEY,ALT_KEY,CTL_KEY
E67C 45 | 1791 | ;----- SCAN CODE TABLES
E67D | 1792 | 1801

1804 ;----- SCAN CODE TABLES
E680 1B | 1805 | KB DB 27,-1,0,-1,-1,-1,30,-1
E681 FF | 1806
E682 00 | 1807
E683 FF | 1808

System BIOS A-25
A-26 System BIOS
LOC OBJ    LINE    SOURCE

EOE 76
EOF FF
EO0 FF  1818 DB -1
EO1 1819 ;----- LC TABLE
EOE1 1820 K10 LABEL BYTE
EOE1 1821 DB 01H,'1234567890-','=','08H,09H
EOE2 31323334353637
EOE3 00
EOE 09
EOF 177657677975 1822 DB 'qwertuyio1',0DH,-1,'asdfghjkl';027H
EOF 66
EOF 67
EOF 68
EOF 69
EOF 70
EOF 71
EOF 72
EOF 73
EOF 74
EOF 75
EOF 76
EOF 77
EOF 78
EOF 79
EOF 80
EOF 81
EOF 82
EOF 83
EOF 84
EOF 85
EOF 86
EOF 87
EOF 88
EOF 89
EOF 90
EOF 91
EOF 92
EOF 93
EOF 94
EOF 95
EOF 96
EOF 97
EOF 98
EOF 99
EOF10 00
EOF10 01
EOF10 02
EOF10 03
EOF10 04
EOF10 05
EOF10 06
EOF10 07
EOF10 08
EOF10 09
EOF10 10
EOF10 11
EOF10 12
EOF10 13
EOF10 14
EOF10 15
EOF10 16
EOF10 17
EOF10 18
EOF10 19
EOF10 20
EOF10 21
EOF10 22
EOF10 23
EOF10 24
EOF10 25
EOF10 26
EOF10 27
EOF10 28
EOF10 29
EOF10 30
EOF10 31
EOF10 32
EOF10 33
EOF10 34
EOF10 35
EOF10 36
EOF10 37
EOF10 38
EOF10 39
EOF10 40
EOF10 41
EOF10 42
EOF10 43
EOF10 44
EOF10 45
EOF10 46
EOF10 47
EOF10 48
EOF10 49
EOF10 50
EOF10 51
EOF10 52
EOF10 53
EOF10 54
EOF10 55
EOF10 56
EOF10 57
EOF10 58
EOF10 59
EOF10 60
EOF10 61
EOF10 62
EOF10 63
EOF10 64
EOF10 65
EOF10 66
EOF10 67
EOF10 68
EOF10 69
EOF10 70
EOF10 71
EOF10 72
EOF10 73
EOF10 74
EOF10 75
EOF10 76
EOF10 77
EOF10 78
EOF10 79
EOF10 80
EOF10 81
EOF10 82
EOF10 83
EOF10 84
EOF10 85
EOF10 86
EOF10 87
EOF10 88
EOF10 89
EOF10 90
EOF10 91
EOF10 92
EOF10 93
EOF10 94
EOF10 95
EOF10 96
EOF10 97
EOF10 98
EOF10 99
EOF10 A0
EOF10 A1
EOF10 A2
EOF10 A3
EOF10 A4
EOF10 A5
EOF10 A6
EOF10 A7
EOF10 A8
EOF10 A9
EOF10 AA
EOF10 AB
EOF10 AC
EOF10 AD
EOF10 AE
EOF10 AF
EOF10 B0
EOF10 B1
EOF10 B2
EOF10 B3
EOF10 B4
EOF10 B5
EOF10 B6
EOF10 B7
EOF10 B8
EOF10 B9
EOF10 BA
EOF10 BB
EOF10 BC
EOF10 BD
EOF10 BE
EOF10 BF
EOF10 C0
EOF10 C1
EOF10 C2
EOF10 C3
EOF10 C4
EOF10 C5
EOF10 C6
EOF10 C7
EOF10 C8
EOF10 C9
EOF10 CA
EOF10 CB
EOF10 CC
EOF10 CD
EOF10 CE
EOF10 CF
EOF10 D0
EOF10 D1
EOF10 D2
EOF10 D3
EOF10 D4
EOF10 D5
EOF10 D6
EOF10 D7
EOF10 D8
EOF10 D9
EOF10 DA
EOF10 DB
EOF10 DC
EOF10 DD
EOF10 DE
EOF10 DF
EOF10 E0
EOF10 E1
EOF10 E2
EOF10 E3
EOF10 E4
EOF10 E5
EOF10 E6
EOF10 E7
EOF10 E8
EOF10 E9
EOF10 EA
EOF10 EB
EOF10 EC
EOF10 ED
EOF10 EE
EOF10 EF
EOF10 F0
EOF10 F1
EOF10 F2
EOF10 F3
EOF10 F4
EOF10 F5
EOF10 F6
EOF10 F7
EOF10 F8
EOF10 F9
EOF10 FA
EOF10 FB
EOF10 FC
EOF10 FD
EOF10 FE
EOF10 FF

System BIOS  A-27
LOC OBJ  |  LINE  |  SOURCE
---|---|---
| E969 3735392043536 | 1040 | DB '789-456-1230.'
| E976 | 1041 | ----- BASE CASE TABLE
| K15 | 1042 | LABEL BYTE
| E976 47 | 1043 | DB 71,72,73,-1.75,-1.77
| E977 4B | E978 49 | E979 FF | E97A 4B | E97B FF | E97C 4D | E97D FF | 1044 | DB -1.79,00,01.02.03
| E97E 4F | E97F 50 | E980 51 | E981 52 | E982 53 | 1045 | ----- KEYBOARD INTERRUPT ROUTINE
| E987 1046 | ORG 0E987H
| 1049 | KB_INT PROC FAR
| E987 FB | 1050 | STI ; ALLOW FURTHER INTERRUPTS
| E980 50 | 1051 | PUSH AX
| E989 53 | 1052 | PUSH BX
| E98A 51 | 1053 | PUSH CX
| E98B 52 | 1054 | PUSH DX
| E98C 56 | 1055 | PUSH SI
| E98D 57 | 1056 | PUSH DI
| E98E 1E | 1057 | PUSH DS
| E98F 06 | 1058 | PUSH ES
| E990 FC | 1059 | CALL DOS
| E991 8ECS10 | 1060 | CALL DOS
| E994 8E60 | 1061 | IN AL,KB_DATA ; READ IN THE CHARACTER
| E996 50 | 1062 | PUSH AX ; SAVE IT
| E997 8E61 | 1063 | IN AL,KB_CTL ; GET THE CONTROL PORT
| E999 8AE9 | 1064 | MOV AH,AL ; SAVE VALUE
| E99B 8E60 | 1065 | OR AL,80H ; RESET BIT FOR KEYBOARD
| E99D 8E61 | 1066 | OUT KB_CTL,AL
| E99F 8E60 | 1067 | XOR AL,AL ; GET BACK ORIGINAL CONTROL
| E9A1 8E61 | 1068 | OUT KB_CTL,AL ; KB HAS BEEN RESET
| E9A3 50 | 1069 | POP AX ; RECOVER SCAN CODE
| E9A4 8AE0 | 1070 | MOV AH,AL ; SAVE SCAN CODE IN AN ALSO
| E9A9 1071 | ----- TEST FOR OVERFLOW SCAN CODE FROM KEYBOARD
| FC | 1074 | CMP AL,OFFH ; IS THIS AN OVERRUN CHAR
| E9AA E97A02 | 1075 | JNZ K16 ; NO, TEST FOR SHIFT KEY
| E9A7 1076 | JMP K62 ; BUFFER_FULLSCREEN
| E977 | 1078 | ----- TEST FOR SHIFT KEYS
| E9AD 1079 | K16: ; TEST_SHIFT
| E9AD 247F | 1080 | AND AL,07FH ; TURN OFF THE BREAK BIT
| E9AF 0E | 1082 | PUSH CS
| E9B0 67 | 1083 | POP ES
| E9B1 8EFE8 | 1084 | MOV D1.OFFSET K6 ; ESTABLISH ADDRESS OF SHIFT TABLE
| E9B4 890000 | 1085 | MOV CX,K6L ; LENGTH
| E9B5 1F | 1086 | REPNE ASCB ; LOOK THROUGH THE TABLE FOR A MATCH
| E98E AE | 1087 | MOV AL,AH ; RECOVER SCAN CODE
| E9BB 7403 | 1088 | JE K17 ; JUMP IF MATCH FOUND
| E9BD E98500 | 1089 | JMP K25 ; IF NO MATCH, THEN SHIFT NOT FOUND
| E990 | 1090 | ----- SHIFT KEY FOUND
| E99C | 1091 | ----- SHIFT MAKE FOUND, DETERMINE SET OR TOGGLE
| E9CD | 1092 | ----- PLAIN SHIFT KEY, SET SHIFT ON

A-28 System BIOS
TEST FOR HOLD STATE

LOC OBJ   LINE   SOURCE
1904 E9D2 00517000  OR KB_FLAG, AH ; TURN ON SHIFT BIT
1906 E9D6 000000  JMP K26 ; INTERRUPT_RETURN
1907 E9D9 1910 K10:  ; SHIFT-TOGGLE
1911 E9D9 F66170004  TEST KB_FLAG, CTL_SHIFT ; CHECK CTL SHIFT STATE
1912 E9DE 765C  ; JMP IF CTL STATE
e9d2 3552  ; CMP AL, INS_KEY ; CHECK FOR INSERT KEY
1914 E9E2 7622  ; JUM IF NOT INSERT KEY
1915 E9E4 F66170006  TEST KB_FLAG, ALT_SHIFT ; CHECK FOR ALTERNATE SHIFT
1916 E9E8 755A  ; JUMP IF ALTERNATE SHIFT
1917 E9F1 E061700020  TEST KB_FLAG, HMP_STATE ; CHECK FOR BASE STATE
1918 E9F9 7600  ; JUMP IF NUM LOCK IS ON
1919 E9FF F66170003  TEST KB_FLAG, LEFT_SHIFT+, RIGHT_SHIFT
1920 E9F5 7400  ; JUMP IF BASE STATE
1921 E9F6 1922 K20:  ; NUMERIC ZERO, NOT INSERT KEY
1922 E9F6 083052  MDV AX, $030H ; PUT OUT AN ASCII ZERO
1923 E9F7 E9F601  JMP K57 ; BUFFER_FILL
1924 E9F8 1925 K21:  ; RIGHT DE NUMERIC
1925 E9F9 F66170003  TEST KB_FLAG, LEFT_SHIFT+, RIGHT SHIFT
1926 EA00 74F3  ; JUMP NUMERIC, NOT INSERT
1927 EA03 1929 K22:  ; SHIFT TOGGLE KEY HT1 PROCESS IT
1928 EA07 80461800  TEST AH, KB_FLAG_1 ; IS KEY ALREADY DEPRESSED
1930 EA0A 755D  ; JUMP IF KEY ALREADY DEPRESSED
1931 EA0D 00610000  OR KB_FLAG_1, AH ; INDICATE THAT THE KEY IS DEPRESSED
1932 EA10 30261700  XOR KB_FLAG, AH ; TOGGLE THE SHIFT STATE
1933 EA14 3552  ; CMP AL, INS_KEY ; TEST FOR 1ST MAKE OF INSERT KEY
1934 EA16 755E  ; JUMP IF NOT INSERT KEY
1935 EA18 800052  MDV AX, INS_KEY+40H ; SET SCAN CODE INTO AH, 0 INTO AL
1936 EA1B FW701  ; PUT INTO OUTPUT BUFFER
1937 EA20 1939 K23:  ; BREAK-SHIFT-FOUND
1940 EA21 800F1C  ; BREAK-SHIFT-FOUND
1942 EA23 731A  ; CMP AH, SCROLL_SHIFT ; IS THIS A TOGGLE KEY
1943 EA25 F604  ; JAE K24 ; YES, HANDLE BREAK TOGGLE
1944 EA28 20261700  NOT AH ; INVERT MASK
1945 EA2B 3C00 AND KB_FLAG, AH ; TURN OFF SHIFT BIT
1946 EA2F 755D  ; CMP AL, ALT_KEY+60H ; IS THIS ALTERNATE SHIFT RELEASE
1947 EA32 752C  ; JNE K26 ; INTERRUPT_RETURN
1948 EA3F 1949 K24:  ; BREAK-TOGGLE
1950 EA43 E8F0  ; NOT AH ; INVERT MASK
1951 EA46 20261800  AND KB_FLAG_1, AH ; INDICATE NO LONGER DEPRESSED
1952 EA49 E814  ; JMP SHORT K26 ; INTERRUPT_RETURN
1953 EA4C 1956 K25:  ; NO-SHIFT-FOUND
1955 EA4E 3C00  ; CMP AL, 0H ; TEST FOR BREAK KEY
1957 EA50 752C  ; JNE K26 ; NOTHING FOR BREAK CHAR FROM HERE ON
1958 EA53 7510  ; JAE K26
1959 EA56 F66180006  TEST KB_FLAG_1, HOLD_STATE ; ARE WE IN HOLD STATE
1960 EA5A 7417  ; JZ K26 ; BRANCH AROUND TEST IF NOT
1961 EA5C 7345  ; JMP HMP_KEY
1962 EA5E 7405  ; JE K26 ; CAN'T END HOLD ON HMP_LOCK
1963 EA5F 802618007  AND KB_FLAG_1, HMP_STATE ; TURN OFF THE HOLD STATE BIT
1964 EA5F 1963 K26:  ; INTERRUPT_RETURN
1965 EA63 FA  ; TURN OFF INTERRUPTS
1967 EA6A 8020  ; MDV AL, EDI ; END OF INTERRUPT COMMAND
1969 EA6C 6626  ; OUT $020H, AL ; SEND COMMAND TO INT CONTROL PORT
1970 EA76 1973 K27:  ; INTERRUPT_RETURN-HD_EDI
1971 EA7C 07  ; POP E5
1972 EA7E 1977 K1F  ; POP DS
1973 EA80 6F  ; POP DI
1974 EA83 61E  ; POP SI

System BIOS A-29
EA62 5A 1981 POP DX
EA63 59 1982 POP CX
EA64 5B 1983 POP BX
EA65 5B 1984 POP AX ; RESTORE STATE
EA66 CF 1985 INRET ; RETURN, INTERRUPTS BACK ON
1986 ; WITH FLAG CHANGE
1987
1988 ; ----- NOT IN HOLD STATE, TEST FOR SPECIAL CHARs
1989
EA67 1990 K28: ; NO-HOLD-STATE
EA67 F60617000D 1991 TEST KB_FLAG,ALT_SHIFT ; ARE WE IN ALTERNATE SHIFT
EA6C 7605 1992 JNZ K29 ; JUMP IF ALTERNATE SHIFT
EA6E 99100 1993 JMP K30 ; JUMP IF NOT ALTERNATE
1994
1995 ; ----- TEST FOR RESET KEY SEQUENCE (CTRL ALT DEL)
1996
EA71 1997 K29: ; TEST-RESET
EA71 F606170004 1998 TEST KB_FLAG,CTRL_SHIFT ; ARE WE IN CONTROL SHIFT ALSO
EA7E 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 ; ----- CTRL-ALT-DEL HAS BEEN FOUND. DO I/O CLEANUP
2003
EA7C C70620003412 2005 MOV RESET_FLAG,1234H ; SET FLAG FOR RESET FUNCTION
EA82 EA9E000F0 2006 JMP RESET ; JUMP TO POWER-ON DIAGNOSTICS
2007 ; ----- ALT-INPUT-TABLE
2008
EA87 2009 K30 LABEL BYTE
EA87 52 2010 DB 82,79,80,81,75,76,77
EA88 4F 2011 DB 16,17,18,19,20,21,22,23 ; A-Z TYPEWRITER CHARs
EA89 5F 2012 DB 24,25,26,27,28,29,30
EA8A 51 2013 DB 31,32,33,34,35
EA8B 4B 2014 DB 36,37,38,39,40
EA8C 4C 2015 DB 41,42,43,44,45
EA8D 4D 2016 DB 46,47,48,49,50
EA8E 47 2017 DB 51,52,53,54,55
EA8F 4B 2018 ; ----- IN ALTERNATE SHIFT. RESET NOT FOUND
EA90 49 2019 ; ----- SUPER-SHIFT-TABLE
EA91 10 2020 DB 16,17,18,19,20,21,22,23 ; A-Z TYPEWRITER CHARs
EA92 11 2021 DB 24,25,26,27,28,29,30
EA93 12 2022 DB 31,32,33,34,35
EA94 13 2023 DB 36,37,38,39,40
EA95 14 2024 DB 41,42,43,44,45
EA96 15 2025 DB 46,47,48,49,50
EA97 16 2026 DB 51,52,53,54,55
EA98 17 2027 ; ----- IN ALTERNATE SHIFT. RESET NOT FOUND
EA99 18 2028 ; ----- IN ALTERNATE SHIFT. RESET NOT FOUND
EA9A 19 2029 ; ----- IN ALTERNATE SHIFT. RESET NOT FOUND
EA9B 1E 2030 ; ----- IN ALTERNATE SHIFT. RESET NOT FOUND
EA9C 1F 2031 ; ----- IN ALTERNATE SHIFT. RESET NOT FOUND
EA9D 20 2032 ; ----- IN ALTERNATE SHIFT. RESET NOT FOUND
EA9E 21 2033 ; ----- IN ALTERNATE SHIFT. RESET NOT FOUND
EA9F 22 2034 ; ----- IN ALTERNATE SHIFT. RESET NOT FOUND
EA9A 23 2035 ; ----- IN ALTERNATE SHIFT. RESET NOT FOUND
EA9B 24 2036 ; ----- IN ALTERNATE SHIFT. RESET NOT FOUND
EA9C 25 2037 ; ----- IN ALTERNATE SHIFT. RESET NOT FOUND
EA9D 26 2038 ; ----- IN ALTERNATE SHIFT. RESET NOT FOUND
EA9E 27 2039 ; ----- IN ALTERNATE SHIFT. RESET NOT FOUND
EA9F 28 2040 ; ----- IN ALTERNATE SHIFT. RESET NOT FOUND
EA9A 29 2041 ; ----- IN ALTERNATE SHIFT. RESET NOT FOUND
EA9B 30 2042 ; ----- IN ALTERNATE SHIFT. RESET NOT FOUND
EA9C 31 2043 ; ----- IN ALTERNATE SHIFT. RESET NOT FOUND
EA9D 32 2044 ; ----- IN ALTERNATE SHIFT. RESET NOT FOUND
2045
EA9E 2046 K31: ; NO-RESET
EA9F 3C39 2047 CMP AL,57 ; TEST FOR SPACE KEY
EAAD 7505 2048 JNE K32 ; NOT THERE
EAAF B020 2049 MOV AL, ' ' ; SET SPACE CHAR
EAB0 E92101 2050 JMP K57 ; BUFFER_FILL
EABB 2051 ; ----- LOOK FOR KEY PAD ENTRY
EABC 2052
EABD 2053

A-30 System BIOS
<table>
<thead>
<tr>
<th>LOC OBJ</th>
<th>LINE</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>EAD6</td>
<td>2028</td>
<td>MOV DI,OFFSET K30</td>
</tr>
<tr>
<td>EAD6</td>
<td>2029</td>
<td>MOV CX,10</td>
</tr>
<tr>
<td>EAD8</td>
<td>2030</td>
<td>REPNE SCASB</td>
</tr>
<tr>
<td>EADB</td>
<td>2031</td>
<td>JNE K33</td>
</tr>
<tr>
<td>EABB</td>
<td>2032</td>
<td>SUB DI,OFFSET K30+1</td>
</tr>
<tr>
<td>EAC2</td>
<td>2033</td>
<td>MOV AL,ALT_INPUT</td>
</tr>
<tr>
<td>EAC5</td>
<td>2034</td>
<td>MOV AH,10</td>
</tr>
<tr>
<td>EAC7</td>
<td>2035</td>
<td>ADD AX,DI</td>
</tr>
<tr>
<td>EAC9</td>
<td>2036</td>
<td>MOV AL,ALT_INPUT,AL</td>
</tr>
<tr>
<td>EACF</td>
<td>2037</td>
<td>JMP K26</td>
</tr>
<tr>
<td>EABE</td>
<td>2038</td>
<td>MOV AL,0</td>
</tr>
<tr>
<td>EABC</td>
<td>2039</td>
<td>MOVL AL,0</td>
</tr>
<tr>
<td>EABF</td>
<td>2040</td>
<td>CMP AL,SCROLL_KEY</td>
</tr>
<tr>
<td>EABE</td>
<td>2041</td>
<td>CMP AL,0</td>
</tr>
<tr>
<td>EAF1</td>
<td>2042</td>
<td>CMP AL,59</td>
</tr>
<tr>
<td>EAF3</td>
<td>2043</td>
<td>CMP AL,14</td>
</tr>
<tr>
<td>EAF5</td>
<td>2044</td>
<td>JNE K37</td>
</tr>
<tr>
<td>EAF6</td>
<td>2045</td>
<td>CMP AL,71</td>
</tr>
<tr>
<td>EAF7</td>
<td>2046</td>
<td>JAE K36</td>
</tr>
<tr>
<td>EAF9</td>
<td>2047</td>
<td>MOV BX,OFFSET K13</td>
</tr>
<tr>
<td>EAF0</td>
<td>2048</td>
<td>JMP K57</td>
</tr>
<tr>
<td>EAF1</td>
<td>2049</td>
<td>CMP AL,E</td>
</tr>
<tr>
<td>EAF3</td>
<td>2050</td>
<td>MOV AL,0</td>
</tr>
<tr>
<td>EAF5</td>
<td>2051</td>
<td>JMP K57</td>
</tr>
<tr>
<td>EAF7</td>
<td>2052</td>
<td>CMP AL,0</td>
</tr>
<tr>
<td>EAF9</td>
<td>2053</td>
<td>CMP AL,E</td>
</tr>
<tr>
<td>EAF1</td>
<td>2054</td>
<td>CMP AL,59</td>
</tr>
<tr>
<td>EAF3</td>
<td>2055</td>
<td>CMP AL,14</td>
</tr>
<tr>
<td>EAF5</td>
<td>2056</td>
<td>JNE K37</td>
</tr>
<tr>
<td>EAF6</td>
<td>2057</td>
<td>CMP AL,71</td>
</tr>
<tr>
<td>EAF7</td>
<td>2058</td>
<td>JAE K36</td>
</tr>
<tr>
<td>EAF9</td>
<td>2059</td>
<td>MOV AL,0</td>
</tr>
<tr>
<td>EAF0</td>
<td>2060</td>
<td>JMP K57</td>
</tr>
<tr>
<td>EAF1</td>
<td>2061</td>
<td>CMP AL,E</td>
</tr>
<tr>
<td>EAF3</td>
<td>2062</td>
<td>CMP AL,59</td>
</tr>
<tr>
<td>EAF5</td>
<td>2063</td>
<td>CMP AL,0</td>
</tr>
<tr>
<td>EAF7</td>
<td>2064</td>
<td>CMP AL,0</td>
</tr>
<tr>
<td>EAF9</td>
<td>2065</td>
<td>CMP AL,E</td>
</tr>
<tr>
<td>EAF1</td>
<td>2066</td>
<td>CMP AL,59</td>
</tr>
<tr>
<td>EAF3</td>
<td>2067</td>
<td>CMP AL,14</td>
</tr>
<tr>
<td>EAF5</td>
<td>2068</td>
<td>JNE K37</td>
</tr>
<tr>
<td>EAF6</td>
<td>2069</td>
<td>CMP AL,71</td>
</tr>
<tr>
<td>EAF7</td>
<td>2070</td>
<td>JAE K36</td>
</tr>
<tr>
<td>EAF9</td>
<td>2071</td>
<td>MOV BX,OFFSET K13</td>
</tr>
<tr>
<td>EAF0</td>
<td>2072</td>
<td>JMP K57</td>
</tr>
<tr>
<td>EAF1</td>
<td>2073</td>
<td>CMP AL,E</td>
</tr>
<tr>
<td>EAF3</td>
<td>2074</td>
<td>CMP AL,59</td>
</tr>
<tr>
<td>EAF5</td>
<td>2075</td>
<td>JNE K37</td>
</tr>
<tr>
<td>EAF7</td>
<td>2076</td>
<td>CMP AL,71</td>
</tr>
<tr>
<td>EAF9</td>
<td>2077</td>
<td>JAE K36</td>
</tr>
<tr>
<td>EAF0</td>
<td>2078</td>
<td>JMP K57</td>
</tr>
</tbody>
</table>

Appendix A

System BIOS  A-31
i----- TEST CASE KEY SS

EB44 3C57
EB44 B00E6
EB57 3C5B
EB59 7276
EB5B K43:
EB5B BBC6EB
EB5E EB9C60

i----- SET UP TO TRANSLATE CONTROL SHIFT

EB54
EB5A B00E6
EB5B
EB5B
EB5C
EB5C
EB5C

i----- NOT IN CONTROL SHIFT

EB61
EB61 3C47
EB63 732C
EB65 F606170003
EB66 745A
EB66
EB66

i----- UPPER CASE, HANDLE SPECIAL CASES

EB6C 3C6F
EB6E 7305
EB70 B00D0F
EB73 EB60
EB75
EB75 3C37
EB77 7509

i----- ISSUE INTERRUPT TO INDICATE PRINT SCREEN FUNCTION

EB79 B020
EB7A E620
EB7B E605
EB7C 7206
EB7D 7206
EB7E 8055E9
EB7F E919
EB7F
EB7F
EB7F

i----- KEYPAD KEYS, MUST TEST NUM LOCK FOR DETERMINATION

EB81
EB82
EB87 F606170020
EB89 7520
EB89
EB8A 3C3B
EB8B 740C
EB8C 8810E9
EB8D E840

i----- BASE CASE FOR KEYPAD

EB91
EB92
EB95 F606170003
EB96 7520
EB96
EB96
EB96

A-32 System BIOS
LOC OBJ  |  LINE | SOURCE
---------|-------|--------
EB49 EB76E9 | 2100 | MOV BX,OFFSET K16 ; BASE CASE TABLE
EBAC EB71 | 2101 | JMP SHORT K64 ; CONVERT TO PSEUDO SCAN
EDAE | 2102 | K50: ; NOT-SHIFT
EBAE B2D04A | 2103 | MOV AX,74H+256*'-.' ; HMINUS
EB1B EB22 | 2104 | JMP SHORT K57 ; BUFFER_FILL
EB83 | 2105 | K51: ; ALMOST-HNUM-STATE
EB80 B2D04E | 2106 | MOV AX,74H+256*'+反对' ; PLUS
EB66 EB1D | 2107 | JMP SHORT K57 ; BUFFER_FILL
2108
2109 ; ------ MIGHT BE HNUM LOCK, TEST SHIFT STATUS
EB88 | 2110 | K52: ; ALMOST-HNUM-STATE
EB80 F406170003 | 2111 | TEST KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT ; CONVERT TO LOWER CASE
EBBD 7E00 | 2112 | JNZ K54 ; SHIFTED TEMP OUT OF HNUM STATE
EBBF | 2113 | K53: ; REALLY-HNUM-STATE
EBBF 2C46 | 2114 | SUB AL,70 ; CONVERT ORIGIN
EB1C B669E9 | 2115 | MOV BX,OFFSET K14 ; NUM STATE TABLE
EBC4 EB08 | 2116 | JMP SHORT K56 ; TRANSLATE_CHAR
2117
2118 ; ------ PLAIN OLD LOWER CASE
EBC6 | 2119 | K54: ; NOT-SHIFT
EBCC 3C3B | 2120 | CMP AL,59 ; TEST FOR FUNCTION KEYS
EB1C 7D04 | 2121 | JB K55 ; NOT-LOWER-FUNCTION
EBCA B000 | 2122 | MOV AL,0 ; SCAN CODE IN AN ALREADY
EBCC EB07 | 2123 | JMP SHORT K57 ; BUFFER_FILL
EBC8 932B | 2124 | MOV AX,74H+256*'+反对' ; PLUS
EB1E BBE1EB | 2125 | MOV BX,OFFSET K10 ; LC TABLE
EBDC | 2126 | XOR AL,AL ; CONVERT TO PSEUDO SCAN
EBDF | 2127 | K56: ; XOR-TO-UPPER
EBE1 EB1C | 2128 | DEC AL ; CONVERT ORIGIN
EBD3 2D17 | 2129 | XLAT CS:K11 ; CONVERT THE SCAN CODE TO ASCII
EBD7 741F | 2130 | CMP AL,-1 ; PUT CHARACTER INTO BUFFER
EBD9 0DFF | 2131 | JE K59 ; YES, DO NOTHING WITH IT
EBDC 741A | 2132 | CMP AH,-1 ; LOOK FOR -1 PSEUDO SCAN
EBDC 741A | 2133 | JE K59 ; NEAR_INTERRUPT_RETURN
EBDE | 2134 | XOR AL,AL ; HANDLE THE CAPS LOCK PROBLEM
EBDD F406170040 | 2135 | XOR AL,AL ; BUFFER-FILL-NOTTEST
EBE3 7420 | 2136 | TEST KB_FLAG,CAPS_STATE ; ARE WE IN CAPS LOCK STATE
EBE3 7420 | 2137 | JZ K61 ; SKIP IF NOT
2138
2139 ; ------ IN CAPS LOCK STATE
EBE5 F40617003 | 2140 | TEST KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT ; TEST FOR SHIFT STATE
EBEA 740F | 2141 | JZ K60 ; IF NOT SHIFT, CONVERT LOWER TO UPPER
EBE3 2D17 | 2142 | XOR AL,AL ; CONVERT ANY UPPER CASE TO LOWER CASE
EBE3 2D17 | 2143 | XOR AL,AL ; FIND OUT IF ALPHABETIC
EBE3 7215 | 2144 | JB K61 ; NOT-CAPS_STATE
EBF0 3C3A | 2145 | CMP AL,'Z' ; NOT-CAPS_STATE
EBF2 7711 | 2146 | JA K61 ; NOT-CAPS_STATE
EBF4 0420 | 2147 | ADD AL,'A'-'A' ; CONVERT TO LOWER CASE
EBF6 EB00 | 2148 | JMP SHORT K61 ; NOT-CAPS_STATE
EBF8 | 2149 | XOR AL,AL ; NEAR_INTERRUPT_RETURN
EBF8 E9EFE | 2150 | JMP K26 ; INTERRUPT_RETURN
EBF8 | 2151 | XOR AL,AL ; CONVERT ANY LOWER CASE TO UPPER CASE
EBF8 | 2152 | XOR AL,AL ; LOWER-TO-UPPER
EBF8 3C3A | 2153 | CMP AL,'A' ; FIND OUT IF ALPHABETIC
EBF8 7206 | 2154 | JB K61 ; NOT-CAPS_STATE
EBFF 3C7A | 2155 | CMP AL,'Z' ; NOT-CAPS_STATE
EC01 7702 | 2156 | JA K61 ; NOT-CAPS_STATE
EC03 2C20 | 2157 | SUB AL,'A'-'A' ; CONVERT TO UPPER CASE
EC05 | 2158 | XOR AL,AL ; NOT-CAPS-STATE
EC05 0E1EEOO | 2159 | MOV BX,BUFFER_TAIL ; GET THE END POINTER TO THE BUFFER
EC09 08F3 | 2160 | MOV SI,BX ; SAVE THE VALUE
EC0B E063FC | 2161 | CALL K4 ; ADVANCE THE TAIL
LOC OBJ  LINE  SOURCE

EC0 3B1E1A00  2257  CMP BX,BUFFER_HEAD  ; HAS THE BUFFER WRAPPED AROUND
EC1 7613  2258  JE K62  ; BUFFER_FULL_BEEP
EC1 8904  2259  MOV IS1IAX  ; STORE THE VALUE
EC1 691E1C00  2260  MOV BUFFER_TAIL,BX  ; MOVE THE POINTER UP
EC14 E93CFE  2261  JMP K26  ; INTERRUPT_RETURN

2262  ; ------ TRANSLATE SCAN FOR PSEUDO SCAN CODES

2264  ;

EC1D  2265  ; K63:  ; TRANSLATE_SCAN
EC1D 2C30  2266  ; SUB AL,59  ; CONVERT ORIGIN TO FUNCTION KEYS
EC1F  2267  ; K64:  ; TRANSLATE-SCAN-ORGD
EC1F 2E87  2268  ; XLAT CS:K9  ; CTL TABLE SCAN
EC21 8AE0  2269  ; MOV AH,AL  ; PUT VALUE INTO AH
EC23 B900  2270  ; MOV AL,0  ; ZERO ASCII CODE
EC26 EBAE  2271  ; JMP K57  ; PUT IT INTO THE BUFFER

2272  ; KB_INT ENDP

2274  ; ------ BUFFER IS FULL, SOUND THE BEEPER

EC27  2275  ; K65:  ; BUFFER-FULL-BEEP
EC27 D020  2276  ; MOV AL,EOI  ; END OF INTERRUPT COMMAND
EC29 E620  2277  ; OUT 20H,AL  ; SEND COMMAND TO INT CONTROL PORT
EC2B B90000  2278  ; MOV BX,D0H  ; NUMBER OF CYCLES FOR 1/12 SECOND TONE
EC2E E661  2279  ; IN AL,KB_CTL  ; GET CONTROL INFORMATION
EC30 50  2280  ; PUSH AX  ; SAVE
EC31  2281  ; K66:  ; DEEP-CYCLE
EC31 24FC  2282  ; AND AL,0FH  ; TURN OFF TIMER AND SPEAKER DATA
EC33 E661  2283  ; OUT KB_CTL,AL  ; OUTPUT TO CONTROL
EC35 B94000  2284  ; MOV CX,40H  ; HALF CYCLE TIME FOR TONE
EC3B  2285  ; K67:  ; DEEP-CYCLE
EC3B 2EFE  2286  ; LOOP K66  ; SPEAKER OFF
EC3A EC02  2287  ; OR AL,2  ; TURN ON SPEAKER BIT
EC3C E661  2288  ; OUT KB_CTL,AL  ; OUTPUT TO CONTROL
EC3E B94000  2289  ; MOV CX,40H  ; SET UP COUNT
EC41  2290  ; K68:  ; ANOTHER CYCLE
EC41 EEFE  2291  ; LOOP K67  ; ANOTHER HALF CYCLE
EC44 4B  2292  ; DEC BX  ; TOTAL TIME COUNT
EC46 76EB  2293  ; JNZ K65  ; DO ANOTHER CYCLE
EC46 56  2294  ; POP AX  ; RECOVER CONTROL
EC47 E661  2295  ; OUT KB_CTL,AL  ; OUTPUT THE CONTROL
EC49 E12FE  2296  ; JMP K27

2297  ; KB_INT ENDP

2298  ; KB_INT ENDP

2299  ; KB_INT ENDP

229A  ; KB_INT ENDP

229B  ; KB_INT ENDP

229C  ; KB_INT ENDP

229D  ; KB_INT ENDP

229E  ; KB_INT ENDP

229F  ; KB_INT ENDP

2300  ; F1 DB ' 301',13,10  ; KEYBOARD ERROR
EC50 0D  2301  ; F3 DB ' 601',13,10  ; DISKETTE ERROR
EC51 0A  2302  ; F3 DB ' 601',13,10  ; DISKETTE ERROR
EC52 365031  2303  ; F3 DB ' 601',13,10  ; DISKETTE ERROR

2304  ; --- INT 13 -------------------------------

2305  ; DISKETTE I/O

2306  ; --- INT 13 -------------------------------

2307  ; DISKETTE I/O

2308  ; --- INT 13 -------------------------------

2309  ; DISKETTE I/O

2310  ; --- INT 13 -------------------------------

2311  ; DISKETTE I/O

2312  ; --- INT 13 -------------------------------

2313  ; DISKETTE I/O

2314  ; --- INT 13 -------------------------------

2315  ; DISKETTE I/O

2316  ; --- INT 13 -------------------------------

2317  ; DISKETTE I/O

2318  ; --- INT 13 -------------------------------

2319  ; DISKETTE I/O

2320  ; --- INT 13 -------------------------------

2321  ; DISKETTE I/O

2322  ; --- INT 13 -------------------------------

2323  ; DISKETTE I/O

2324  ; --- INT 13 -------------------------------

2325  ; DISKETTE I/O

2326  ; --- INT 13 -------------------------------

2327  ; DISKETTE I/O

2328  ; --- INT 13 -------------------------------

2329  ; DISKETTE I/O

2330  ; --- INT 13 -------------------------------

A-34  System BIOS
LaC OBJ
LINE
SOURCE

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  :
2341  ; 00 = STATUS OF OPERATION :
2342  ; STATUS BITS ARE DEFINED IN THE EQUATES FOR :
2343  ; DISKETTE_STATUS VARIABLE IN THE DATA SEGMENT OF THIS :
2344  ; MODULE. :
2345  ; CY = 0 SUCCESSFUL OPERATION (AH=0 ON RETURN)
2346  ; CY = 1 FAILED OPERATION (AH HAS ERROR Reason) :
2347  ; FOR READ/WRITE/VERIFY :
2348  ; 01 = NUMBER OF SECTORS ACTUALLY READ :
2349  ; 02 = NUMBER OF SECTORS ACTUALLY READ:
2350  ; **** AL MAY NOT BE CORRECT If TIME OUT ERROR Occurs :
2351  ; NOTE: IF AN ERROR IS REPORTED BY THE DISKETTE CODE, :
2352  ; IS TAKEN, SO THAT THREE RETRIES ARE REQUIRED ON READS :
2353  ; TO ENSURE THAT THE PROBLEM IS NOT DUE TO MOTOR :
2354  ; START-UP.

2355  ;------------------------------------------------------------------------
2356  ; RESUME CODE,OS: DATA.ES: DATA

Appendix A

System BIOS A-35
A-36 System BIOS
DISKETTE FORMAT

2404 \---------- DISKETTE FORMAT

2405

2406 ED18 DISK_FORMAT PROC NEAR

2407 ED18 0003F0000

2408 ED1B 004A CALL MOTOR_STATUS,80H 1 INDICATE WRITE OPERATION

2409 ED1F 0060 CALL DMA_SETUP 1 SET UP THE DMA

2410 ED22 0D40 CALL DMA_SETUP 1 ESTABLISH THE FORMAT COMMAND

2411 ED26 ED26 JMP SHORT RH_OPN 1 DO THE OPERATION

2412 ED26 2492 J10: 1 CONTINUATION OF RH_OPN FOR HFT

2413 ED26 BB7070 2493 MOV BX,7 1 GET THE

2414 ED29 E80001 CALL GET_PARM 1 BYTES/SECTOR VALUE TO NEC

2415 ED2C BB9009 2494 MOV BX,9 1 GET THE

2416 ED2F E83A01 2495 CALL GET_PARM 1 SECTORS/TRACK VALUE TO NEC

2417 ED32 BBF009 2496 MOV BX,15 1 GET THE

2418 ED35 EB3401 2497 CALL GET_PARM 1 GAP LENGTH VALUE TO NEC

2419 ED38 BB1009 2498 MOV BX,17 1 GET THE FILLER BYTE

2420 ED3B E84B00 2499 JMP J16 1 TO THE CONTROLLER

2420 2500 DISK_FORMAT ENDP

2421 2502

2422 2503 \---------- DISKETTE WRITE ROUTINE

2423 2504

2424 ED3E DISK_WRITE PROC NEAR

2425 ED3E 0003F0000

2426 ED43 004A CALL MOTOR_STATUS,80H 1 INDICATE WRITE OPERATION

2427 ED45 E80001 CALL DMA_SETUP 1 DMA WRITE COMMAND

2428 ED48 B4C5 2509 MOV AH,0DH 1 NEC COMMAND TO WRITE TO DISKETTE

2429 2510 DISK_WRITE ENDP

2430 2511

2431 2512 \---------- ALLOW WRITE ROUTINE TO FALL INTO RH_OPN

2432 2513

2433 2514 \-------------------------------------------

2434 ED44 7308 2515 RH_OPN PROC NEAR

2434 ED44 CB00 2516 MOV DISKETTE_STATUS,DMA_BOUNDARY 1 SET ERROR

2435 ED4C C66410009

2436 ED51 B000 2521 MOV AL,0 1 NO SECTORS TRANSFERRED

2437 ED53 C3 2522 RET 1 RETURN TO MAIN ROUTINE

2438 ED54 2523 J11: 1 DO_RH_OPN

2439 ED54 50 2524 PUSH AX 1 SAVE THE COMMAND

2439 2525

2440 2526 \---------- TURN ON THE MOTOR AND SELECT THE DRIVE

2440 2527

2441 ED55 51 2528 PUSH CX 1 SAVE THE T/S PARMS

2442 ED56 0ACA 2529 MOV CL,CL 1 GET DRIVE NUMBER AS SHIFT COUNT

2443 ED58 BD01 2530 MOV AL,1 1 MASK FOR DETERMINING MOTOR BIT

2444 ED5A D2E0 2531 SAL AL,CL 1 SHIFT THE MASK BIT

2445 ED5C FA 2532 CLI 1 NO INTERRUPTS WHILE DETERMINING

2446 ED5D C6644000FF

2447 ED5E 84003F00 2534 MOV MOTOR_COUNT,OFFH 1 SET LARGE COUNT DURING OPERATION

2448 ED62 7931 2535 TEST AL,MOTOR_STATUS 1 TEST THAT MOTOR FOR OPERATING

2449 ED66 7831 2536 JNZ J14 1 IF RUNNING, SKIP THE WAIT

2450 ED68 80263F000F 2537 AND MOTOR_STATUS,OFFH 1 TURN OFF ALL MOTOR BITS

2451 ED6D 0003F00 2538 OR MOTOR_STATUS,AL 1 TURN ON THE CURRENT MOTOR

2452 ED71 FD 2539 STI 1 INTERRUPTS BACK ON

2453 ED72 D010 2540 MOV AL,10H 1 MASK BIT

2454 ED74 D2E0 2541 SAL AL,CL 1 DEVELOP BIT MASK FOR MOTOR ENABLE

2455 ED76 0ACA 2542 OR AL,CL 1 GET DRIVE SELECT BITS IN

2456 ED7B 0C0C 2543 OR AL,OH 1 MD RESET, ENABLE DMA/INT

2457 ED7A 55 2544 PUSH DX 1 SAVE REG

2458 ED7B 0AF03 2545 MOV DX,03F0H 1 CONTROL PORT ADDRESS

2459 ED7E EE 2546 OUT DX,AL 1 RECOVER REGISTERS

2459 2547

2459 2548 \---------- WAIT FOR MOTOR IF WRITE OPERATION

2459 2549

2459 2550 ED80 F663F0000

2459 2551 TEST MOTOR_STATUS,80H 1 IS THIS A WRITE

2459 ED85 7412 2552 JZ J14 1 HD, CONTINUE WITHOUT WAIT

2459 ED87 BB1400 2553 MOV BX,2B 1 GET THE MOTOR WAIT

2459 ED8A ED0F00 2554 CALL GET_PARM 1 PARAMETER

2459 ED8D 0A46 2555 OR AX,AX 1 TEST FOR NO WAIT

2459 ED8F 2556 J12: 1 TEST_WAIT_TIME

2459 ED8F 7408 2557 JE J14 1 EXIT WITH TIME EXPIRED

2459 ED91 2BCC 2558 J5 SUB CX,CX 1 SET UP 1/2 SECOND LOOP TIME

2459 ED93 2559 J13: 1 LOOP J13 1 WAIT FOR THE REQUIRED TIME

APPENDIX A

System BIOS A-37
E095 FECC 2561 DEC AL ; DECREMENT TIME VALUE
E097 EBF6 2562 JMP J12 ; ARE WE DONE YET
E099 JS 2563 J14: ; MOTOR_RUNNING
E099 FB 2564 STI ; INTERRUPTS BACK ON FOR BYPASS WAIT
E09A 59 2565 POP CX
E09C 2 2566
E09E 58 2567 ------ DO THE SEEK OPERATION
E09F 47 Fe 2568 CALL SEEK ; MOVE TO CORRECT TRACK
E0A0 56 2569
E0A2 58 2570 POP AX ; RECOVER COMMAHO
E0A4 8AFe 2571 HOV BH.AH ; SAVE COMMAND IN 8H
E0A7 B600 2572 MOV DH,0 ; SET NO SECTORS READ IN CASE OF ERROR
E0A9 72.48 2573 JC J17 ; IF ERROR, THEN EXIT AFTER MOTOR OFF
E0A9 D903 2574 MOV SI.OFFSET J17 ; DUMMY RETURN ON STACK FOR NEC_OUTPUT
E0B0 00E4 2575 SAL AL,1 ; IF ERROR, THEN EXIT AFTER MOTOR OFF
E0B2 0578 2576 ,----- SEND OUT THE PARAMETERS TO THE CONTROLLER
E0B4 B903 2577
E0B6 880700 2578 MOV AX,EX.7 ; BYTES/SECTOR PARM FROM BLOCK
E0B9 E86COO 2579 CALL GET_PARM , TO THE NEC
E0BE 880600 2580 MOV AX,EX.11 ; GAP LENGTH PARM FROM BLOCK
E0C3 E88600 2581 CALL GET_PARM , TO THE NEC
E0C6 5E 2582 POP S1 ; CAN NOW DISCARD THAT DUMMY
E0C8 E84301 2583 CALL WAIT_INT ; WAIT FOR THE INTERRUPT
E0E0 7245 2584 JC J21 ; LOOK FOR ERROR
E0E2 FC 2585 CLD ; SET THE CORRECT DIRECTION
E0E4 B64200 2586 MOV SI.OFFSET NEC_STATUS ; POINT TO STATUS FIELD
E0E6 AC 2587 LODS NEC_STATUS ; GET ST
E0E8 72C0 2588 MOV AL,AL.0CH ; TEST FOR NORMAL TERMINATION
E0EB 743B 2589 JC J22 ; 0PM_OK
E0EC 3C40 2590 CMP AL,AL.040H ; TEST FOR ABNORMAL TERMINATION
E0F0 7259 2591 JNZ J18 ; NOT ABNORMAL, BAD NEC
E0F2 00E4 2592 MOV SI.OFFSET NEC_STATUS ; GET ST
E0F4 D0E0 2593 SAL AL,1 ; TEST FOR EOT FOUND
E0F7 72D4 2594 MOV AH,RECORD_NOT_FOUND ; TEST FOR CRC ERROR
E0F9 7224 2595 JC J19 ; RWFAIL
E0FB D0E0 2596 SAL AL,1 ; TEST FOR EOT FOUND
E0FD D0E0 2597 SAL AL,1 ; TEST FOR CRC ERROR

A-38 System BIOS
<table>
<thead>
<tr>
<th>LOC OBJ</th>
<th>LINE</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0410</td>
<td>2630</td>
<td>MOV AH,BAD_CRC</td>
</tr>
<tr>
<td>721C</td>
<td>2639</td>
<td>JC J19</td>
</tr>
<tr>
<td>0000</td>
<td>2640</td>
<td>SAL AL,1</td>
</tr>
<tr>
<td>0440</td>
<td>2641</td>
<td>MOV AH,BAD_DMA</td>
</tr>
<tr>
<td>7216</td>
<td>2642</td>
<td>JC J19</td>
</tr>
<tr>
<td>0000</td>
<td>2643</td>
<td>SAL AL,1</td>
</tr>
<tr>
<td>0000</td>
<td>2644</td>
<td>SAL AL,1</td>
</tr>
<tr>
<td>0446</td>
<td>2645</td>
<td>MOV AH,RECORD_NOT_FOUND</td>
</tr>
<tr>
<td>720E</td>
<td>2646</td>
<td>JC J19</td>
</tr>
<tr>
<td>0000</td>
<td>2647</td>
<td>SAL AL,1</td>
</tr>
<tr>
<td>0403</td>
<td>2648</td>
<td>MOV AH,WRITE_PROTECT</td>
</tr>
<tr>
<td>7208</td>
<td>2649</td>
<td>JC J19</td>
</tr>
<tr>
<td>0000</td>
<td>2650</td>
<td>SAL AL,1</td>
</tr>
<tr>
<td>0000</td>
<td>2651</td>
<td>SAL AL,1</td>
</tr>
<tr>
<td>0000</td>
<td>2652</td>
<td>SAL AL,1</td>
</tr>
<tr>
<td>0401</td>
<td>2653</td>
<td>MOV AH,BAD_ADDR_MARK</td>
</tr>
<tr>
<td>7202</td>
<td>2654</td>
<td>JC J19</td>
</tr>
<tr>
<td>0000</td>
<td>2655</td>
<td>MOV AH,BAD_NEC</td>
</tr>
<tr>
<td>0410</td>
<td>2656</td>
<td>J18: MOV AH,BAD_NEC</td>
</tr>
<tr>
<td>0410</td>
<td>2657</td>
<td>MOV AH,BAD_NEC</td>
</tr>
<tr>
<td>0410</td>
<td>2658</td>
<td>J19: MOV AH,BAD_NEC</td>
</tr>
<tr>
<td>0000</td>
<td>2659</td>
<td>OR DX,DISKETTE_STATUS,AN</td>
</tr>
<tr>
<td>0000</td>
<td>2660</td>
<td>CALL H_M_TRANS,NUM_TRANSFERRED</td>
</tr>
<tr>
<td>0403</td>
<td>2661</td>
<td>J20: MOV AH,ERR</td>
</tr>
<tr>
<td>0403</td>
<td>2662</td>
<td>RET</td>
</tr>
<tr>
<td>0000</td>
<td>2663</td>
<td>RET</td>
</tr>
<tr>
<td>0000</td>
<td>2664</td>
<td>CALL RESULTS</td>
</tr>
<tr>
<td>0000</td>
<td>2665</td>
<td>RET</td>
</tr>
<tr>
<td>0000</td>
<td>2666</td>
<td>RET</td>
</tr>
<tr>
<td>0000</td>
<td>2667</td>
<td>RET</td>
</tr>
<tr>
<td>0000</td>
<td>2668</td>
<td>RET</td>
</tr>
<tr>
<td>0000</td>
<td>2669</td>
<td>J21: MOV AH,BAD_NEC</td>
</tr>
<tr>
<td>0000</td>
<td>2670</td>
<td>MOV AH,BAD_NEC</td>
</tr>
<tr>
<td>0000</td>
<td>2671</td>
<td>MOV AH,BAD_NEC</td>
</tr>
<tr>
<td>0000</td>
<td>2672</td>
<td>MOV AH,BAD_NEC</td>
</tr>
<tr>
<td>0000</td>
<td>2673</td>
<td>MOV AH,BAD_NEC</td>
</tr>
<tr>
<td>0000</td>
<td>2674</td>
<td>MOV AH,BAD_NEC</td>
</tr>
<tr>
<td>0000</td>
<td>2675</td>
<td>MOV AH,BAD_NEC</td>
</tr>
<tr>
<td>0000</td>
<td>2676</td>
<td>MOV AH,BAD_NEC</td>
</tr>
<tr>
<td>0000</td>
<td>2677</td>
<td>MOV AH,BAD_NEC</td>
</tr>
<tr>
<td>0000</td>
<td>2678</td>
<td>MOV AH,BAD_NEC</td>
</tr>
<tr>
<td>0000</td>
<td>2679</td>
<td>MOV AH,BAD_NEC</td>
</tr>
<tr>
<td>0000</td>
<td>2680</td>
<td>MOV AH,BAD_NEC</td>
</tr>
<tr>
<td>0000</td>
<td>2681</td>
<td>MOV AH,BAD_NEC</td>
</tr>
<tr>
<td>0000</td>
<td>2682</td>
<td>MOV AH,BAD_NEC</td>
</tr>
<tr>
<td>0000</td>
<td>2683</td>
<td>MOV AH,BAD_NEC</td>
</tr>
<tr>
<td>0000</td>
<td>2684</td>
<td>MOV AH,BAD_NEC</td>
</tr>
<tr>
<td>0000</td>
<td>2685</td>
<td>MOV AH,BAD_NEC</td>
</tr>
<tr>
<td>0000</td>
<td>2686</td>
<td>MOV AH,BAD_NEC</td>
</tr>
<tr>
<td>0000</td>
<td>2687</td>
<td>MOV AH,BAD_NEC</td>
</tr>
<tr>
<td>0000</td>
<td>2688</td>
<td>MOV AH,BAD_NEC</td>
</tr>
<tr>
<td>0000</td>
<td>2689</td>
<td>MOV AH,BAD_NEC</td>
</tr>
<tr>
<td>0000</td>
<td>2690</td>
<td>MOV AH,BAD_NEC</td>
</tr>
<tr>
<td>0000</td>
<td>2691</td>
<td>MOV AH,BAD_NEC</td>
</tr>
<tr>
<td>0000</td>
<td>2692</td>
<td>MOV AH,BAD_NEC</td>
</tr>
<tr>
<td>0000</td>
<td>2693</td>
<td>MOV AH,BAD_NEC</td>
</tr>
<tr>
<td>0000</td>
<td>2694</td>
<td>MOV AH,BAD_NEC</td>
</tr>
<tr>
<td>0000</td>
<td>2695</td>
<td>MOV AH,BAD_NEC</td>
</tr>
<tr>
<td>0000</td>
<td>2696</td>
<td>MOV AH,BAD_NEC</td>
</tr>
<tr>
<td>0000</td>
<td>2697</td>
<td>MOV AH,BAD_NEC</td>
</tr>
<tr>
<td>0000</td>
<td>2698</td>
<td>MOV AH,BAD_NEC</td>
</tr>
<tr>
<td>0000</td>
<td>2699</td>
<td>MOV AH,BAD_NEC</td>
</tr>
<tr>
<td>0000</td>
<td>2700</td>
<td>MOV AH,BAD_NEC</td>
</tr>
<tr>
<td>0000</td>
<td>2701</td>
<td>MOV AH,BAD_NEC</td>
</tr>
<tr>
<td>0000</td>
<td>2702</td>
<td>MOV AH,BAD_NEC</td>
</tr>
<tr>
<td>0000</td>
<td>2703</td>
<td>MOV AH,BAD_NEC</td>
</tr>
<tr>
<td>0000</td>
<td>2704</td>
<td>MOV AH,BAD_NEC</td>
</tr>
<tr>
<td>0000</td>
<td>2705</td>
<td>MOV AH,BAD_NEC</td>
</tr>
<tr>
<td>0000</td>
<td>2706</td>
<td>MOV AH,BAD_NEC</td>
</tr>
<tr>
<td>0000</td>
<td>2707</td>
<td>MOV AH,BAD_NEC</td>
</tr>
<tr>
<td>0000</td>
<td>2708</td>
<td>MOV AH,BAD_NEC</td>
</tr>
<tr>
<td>0000</td>
<td>2709</td>
<td>MOV AH,BAD_NEC</td>
</tr>
<tr>
<td>0000</td>
<td>2710</td>
<td>MOV AH,BAD_NEC</td>
</tr>
<tr>
<td>0000</td>
<td>2711</td>
<td>MOV AH,BAD_NEC</td>
</tr>
<tr>
<td>0000</td>
<td>2712</td>
<td>MOV AH,BAD_NEC</td>
</tr>
<tr>
<td>0000</td>
<td>2713</td>
<td>MOV AH,BAD_NEC</td>
</tr>
<tr>
<td>0000</td>
<td>2714</td>
<td>MOV AH,BAD_NEC</td>
</tr>
</tbody>
</table>

**Note:** The above code snippet is a system BIOS routine. It appears to be a part of the NEC diskette controller setup, handling various statuses and errors during operations. The routine includes commands to set diskette status, handle errors, and manage resources such as REGISTERS and DISKETTE_STATUS.
DISKETTE_STATUS SET ACCORDINGLY

E698 82F1
2715	JMP JZ4 ; ERROR CONDITION
E698 82F2
2716	JZ7: ; OUTPUT
E698 8AC4
2717	MOV AL,OH ; GET BYTE TO OUTPUT
E698 B2F0
2718	MOV DL,0FH ; DATA PORT (SFR)
E699 8E
2719	OUT DX,AL ; OUTPUT THE BYTE
E69A 89
2720	POP CX ; RECOVER REGISTERS
E69A C3
2721	POP DX
E69B C5
2722	RET ; CY = 0 FROM TEST INSTRUCTION
E69B C7
2723

; GET_PARTITION

E69C 1E
2724

E69C 2C0
2725

E69F 82D0
2726

E701 C5367800
2727

E705 D1E8
2728

E717 8A20
2729

E71F 1F
2730

E72A 72C5
2731

E72C 3
2732

E734 2:755 DRIVE RESET
2733

E735 2:759
2734

E737 1
2735

E739 1
2736

E741 E
2737

E752 2:756
2738

E753 1
2739

E755 1
2740

E757 1
2741

E758 1
2742

E759 1
2743

E760 1
2744

E761 1
2745

E762 1
2746

E763 1
2747

E764 2:757
2748

E765 1
2749

E766 1
2750

E767 1
2751

E768 1
2752

E769 1
2753

E76A 1
2754

E76B 1
2755

E76C 1
2756

E76D 1
2757

E76E 1
2758

E76F 1
2759

E770 1
2760

E771 1
2761

E772 1
2762

E773 1
2763

E774 1
2764

E775 1
2765

E776 1
2766

E777 1
2767

E778 1
2768

E779 1
2769

E77A 1
2770

E77B 1
2771

E77C 1
2772

E77D 1
2773

E77E 1
2774

E77F 1
2775

E780 1
2776

E781 1
2777

E782 1
2778

E783 1
2779

E784 1
2780


SEEK

E785 8001
2781

E786 51
2782

E787 8A4A
2783

E788 D2C0
2784

E789 59
2785

E78A 06063E00
2786

E78B 7513
2787

E78C 80063E00
2788

E78D 8A07
2789

E78E 8AB2FF
2790

E78F 8AEE
2791

E790 8A8FF
2792

E791 8E9FF
2793

E792 8AE2
2794

E793 8E9FF
2795

E794 8AE2
2796

E795 8E9FF
2797

E796 8AE2
2798

E797 8E9FF
2799

E798 8AE2
2800

E799 8E7600
2801

E79A 7229
2802

E79B 8A07
2803

E79C 8A07
2804

E79D 8A07
2805

E79E 8A07
2806

E79F 8A07
2807

E7A0 8E5000
2808

LOC_OBJ
LINE
SOURCE
Appendix A

System BIOS A-41
A-42  System BIOS

LOC OBJ     LINE       SOURCE

EF11 C3     2559       RET                          ; RETURN TO CALLER.
2570       RET                          ; CF L SET BY ABOVE IF ERROR
2571       DM A SETUP ENDP
2572
2573       ; CHK_STAT_2
2574       ; THIS ROUTINE HANDLES THE INTERRUPT RECEIVED AFTER A
2575       ; RECALIBRATE, SEEK, OR RESET TO THE ADAPTER.
2576       ; THE INTERRUPT IS WAITED FOR. THE INTERRUPT STATUS SENSED,
2577       ; AND THE RESULT RETURNED TO THE CALLER.
2578       ; INPUT
2579       ; NONE
2580       ; OUTPUT
2581       ; CY = 0 SUCCESS
2582       ; CY = 1 FAILURE -- ERROR IS IN DISKETTE_STATUS
2583       ; (AX) DESTROYED
2584
EF12       2065       CHK_STAT_2 PROC NEAR
2066       CALL WAIT_INT                 ; WAIT FOR THE INTERRUPT
2067       JC JS4                      ; IF ERROR, RETURN IT
2068       MOV AH,08H                 ; SENSE INTERRUPT STATUS COMMAND
2069       CALL NEC_OUTPUT
2070       CALL RESULTS                ; READ IN THE RESULTS
2071       JC JS4                      ; CHK2_RETURN
2072       MOV AL,NEC_STATUS          ; GET THE FIRST STATUS BYTE
2073       AND AL,060H                ; ISOLATE THE BITS
2074       CMP AL,060H                ; TEST FOR CORRECT VALUE
2075       JZ JS5                    ; IF ERROR, GO MARK IT
2076       CLC                        ; GOOD RETURN
2077       JK806410040
2078       JS4:                        ; RETURN TO CALLER
2079       JS5:                        ; CHK2_ERROR
2080       JS6: DISKETTE_STATUS,BAD_SEEK
2081       JS7:                        ; ERROR RETURN CODE
2082       JS8:                        ; ERROR RETURN CODE
2083       JS9:                        ; ERROR RETURN CODE
2084
EF13       2093       CHK_STAT_2 ENDP
2094
2095       ; WAIT_INT
2096       ; THIS ROUTINE WAITS FOR AN INTERRUPT TO OCCUR. A TIME OUT
2097       ; ROUTINE TAKES PLACE DURING THE WAIT, SO THAT AN ERROR MAY BE
2098       ; RETURNED IF THE DRIVE IS NOT READY.
2099       ; INPUT
2100       ; NONE
2101       ; OUTPUT
2102       ; CY = 0 SUCCESS
2103       ; CY = 1 FAILURE -- DISKETTE_STATUS IS SET ACCORDINGLY
2104       ; (AX) DESTROYED
2105
EF13       2169       WAIT_INT PROC NEAR
2170       STI                          ; TURN ON INTERRUPTS, JUST IN CASE
2171       PUSH BX
2172       PUSH CX
2173       MOV BL,2
2174       XOR CX,CX
2175       POP CX
2176       MOV AL,30H
2177       POPF
2178       TEST SEEK_STATUS,INT_FLAG  ; TEST FOR INTERRUPT OCCURRING
2179       JZ JS7                      ; TEST INTERRUPT IS NOT SET
2180       LOOP JS6                      ; COUNT DOWN WHILE WAITING
2181       DEC BL                      ; SECOND LEVEL COUNTER
2182       JNZ JS6                      ; SECOND INTERRUPT IS NOT SET
2183       OR DISKETTE_STATUS,TIME_OUT ; NOTHING HAPPENED
2184       JZ JS7                      ; DISKETTE_STATUS,TIME_OUT
2185       STC                          ; ERROR RETURN
2186       JS7:                        ; RETURN TO CALLER
2187       JS8:                        ; SENSE INTERRUPT
2188       JS9:                        ; CHK2_RETURN
2189       JS10:                       ; CHK2_ERROR
2190       JS11:                       ; DISKETTE_STATUS,BAD_SEEK
2191       JS12:                       ; ERROR RETURN CODE
2192       JS13:                       ; ERROR RETURN CODE
2193       JS14:                       ; ERROR RETURN CODE
2194       JS15:                       ; ERROR RETURN CODE
2195
EF14       2299       STA                          ; FROM TEST INST
2299       BRK                          ; FROM TEST INST
2299
EF15       2327       DISK_INT ENDP
2328
2329       ; DISK_INT
2330       ; THIS ROUTINE HANDLES THE DISKETTE INTERRUPT
2331       ; INPUT
2332       ; NONE
2333       ; OUTPUT
2334       ; THE INTERRUPT FLAG IS SET IS SEEK_STATUS
2335
EF15       2366       DISK_INT ENDP
2367
2368       ; DISK_INT
2369       ; THIS ROUTINE HANDLES THE DISKETTE INTERRUPT
2370       ; INPUT
2371       ; NONE
2372       ; OUTPUT
2373       ; THE INTERRUPT FLAG IS SET IS SEEK_STATUS
2374
EF15       2396       DISK_INT ENDP
2397

<table>
<thead>
<tr>
<th>LOC OBJ</th>
<th>LINE</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>EF57</td>
<td>2947</td>
<td>ORG</td>
</tr>
<tr>
<td>EF57</td>
<td>2948</td>
<td>DISK_INT</td>
</tr>
<tr>
<td>EF57</td>
<td>2949</td>
<td>STI</td>
</tr>
<tr>
<td>EF58</td>
<td>2950</td>
<td>PUSH DS</td>
</tr>
<tr>
<td>EF59</td>
<td>2951</td>
<td>PUSH AX</td>
</tr>
<tr>
<td>EF6A</td>
<td>2952</td>
<td>CALL</td>
</tr>
<tr>
<td>EF6B</td>
<td>2953</td>
<td>OR</td>
</tr>
<tr>
<td>EF6C</td>
<td>2954</td>
<td>MOV AL,0H</td>
</tr>
<tr>
<td>EF6E</td>
<td>2955</td>
<td>OUT 20H,AL</td>
</tr>
<tr>
<td>EF6F</td>
<td>2956</td>
<td>PDP AX</td>
</tr>
<tr>
<td>EF70</td>
<td>2957</td>
<td>POP DS</td>
</tr>
<tr>
<td>EF70</td>
<td>2958</td>
<td>IRET</td>
</tr>
</tbody>
</table>
| EF72    | 2959 | DISK_INT | END

| EF69    | 2960 | RESULTS | PROC NEAR |
| EF6A    | 2962 | ; RESULTS |
| EF6B    | 2963 | ; TO SAY FOLLOWING AN INTERRUPT |
| EF6D    | 2964 | ; INPUT |
| EF6E    | 2965 | ; NONE |
| EF6F    | 2966 | ; OUTPUT |
| EF70    | 2967 | ; CY = 0 SUCCESSFUL TRANSFER |
| EF71    | 2968 | ; CY = 1 FAILURE -- TIME OUT IN WAITING FOR STATUS |
| EF72    | 2969 | ; NEC_STATUS AREA HAS STATUS BYTE LOADED INTO IT |
| EF75    | 2970 | ; BTS DESTROYED |

| EF72    | 2971 | RESULTS_PROC | NEAR |
| EF72    | 2972 | RESULTS | PROC | NEAR |
| EF74    | 2973 | CALL | DOS |
| EF75    | 2974 | MOV DX,OFFSET NEC_STATUS | ; POINTER TO DATA AREA |
| EF76    | 2975 | PUSH CX | ; SAVE COUNTER |
| EF77    | 2976 | PUSH DX | |
| EF78    | 2977 | PUSH BX | |
| EF79    | 2978 | MOV BL,7 | ; MAX STATUS BYTES |
| EF80    | 2979 | J38: | INPUT_LOOP |
| EF81    | 2980 | XOR CX,CX | ; COUNTER |
| EF82    | 2981 | MOV DX,OFFSET NEC_STATUS | ; POINTER TO DATA AREA |
| EF83    | 2982 | J39: | WAIT FOR MASTER |
| EF84    | 2983 | IN AL,DX | ; GET STATUS |
| EF85    | 2984 | TEST AL,00H | ; MASTER READY |
| EF86    | 2985 | JNZ J40A | ; TEST_DIR |
| EF87    | 2986 | LOOP J39 | ; WAIT_MASTER |
| EF91    | 2987 | OR DISKETTE_STATUS.TIME_OUT | |
| EF92    | 2988 | J40: | RESULTS_ERROR |
| EF93    | 2989 | STC | ; SET ERROR RETURN |
| EF94    | 2990 | POP DX | |
| EF95    | 2991 | POP CX | |
| EF96    | 2992 | RET | |
| EF97    | 2993 | J43: | TEST THE DIRECTION BIT |
| EF98    | 2994 | J40A: | |
| EF99    | 2995 | IN AL,DX | ; GET STATUS REG AGAIN |
| EF9A    | 2996 | TEST AL,00H | ; TEST DIRECTION BIT |
| EF9B    | 2997 | JNZ J42 | ; OK TO READ STATUS |
| EF9C    | 2998 | DEC BL | ; INDEX THE BYTE |
| EF9D    | 2999 | J41: | NEC_FAIL |
| EF9E    | 3000 | OR DISKETTE_STATUS.BAD_NEC | |
| EF9F    | 3001 | JMP J40 | ; RESULTS_ERROR |
| EF9A    | 3002 | J40: | RESULTS_ERROR |
| EF9B    | 3003 | J42: | |
| EF9C    | 3004 | J43: | |
| EF9D    | 3005 | LOOP J43 | ; DATASTAT |
| EF9E    | 3006 | J44: | |
| EF9F    | 3007 | RET | |
| EOF0    | 3008 | J40A: | |
| EOF1    | 3009 | J42: | |
| EOF2    | 3010 | J43: | |
| EOF3    | 3011 | MOV DX,OFFSET NEC_STATUS | ; POINTER TO DATA AREA |
| EOF4    | 3012 | MOV DX,OFFSET NEC_STATUS | ; POINTER TO DATA AREA |
| EOF5    | 3013 | MOV [DI],AL | ; STORE THE BYTE |
| EOF6    | 3014 | MOV DI,0 | ; INCREMENT THE POINTER |
| EOF7    | 3015 | MOV CX,DX | ; LOOP TO KILL TIME FOR NEC |
| EOF8    | 3016 | LOOP J43 | ; DATASTAT |
| EOF9    | 3017 | DEC DX | ; /POINT AT STATUS PORT |
| EOFA    | 3018 | IN AL,DX | ; GET STATUS |
| EOFB    | 3019 | TEST AL,00H | ; TEST FOR NEC STILL BUSY |
| EOFC    | 3020 | JZ J44 | ; RESULTS_DONE |
| EODF    | 3021 | DEC DL | ; DECREMENT THE STATUS COUNTER |
| EODG    | 3022 | JNZ J50 | ; GO BACK FOR MORE |

System BIOS A-43
LaC OBJ

EFA8 EBn
EFAA
EFAA.
58
HAB SA
EFA.C: 59
EFAD
C3
A04500
EFBI
3AC5
EFBJ .6.04700
EFB6
740.6.
EFB8
8B0800
EFBE
ESAEFE
EFBE
8AC4
EfCD
FEeD
EFe7
CF
EFea
02
EFC9
25
EFeA
02
EFca
06
EFeE FF
EFeE
50
"f6
EFDO
19
Flo1 04
4-44 System BIOS
1100    ; DATA AREA PRINTER_BASE CONTAINS THE BASE ADDRESS OF THE PRINTER :
1101    ; CARD(S) AVAILABLE (LOCATED AT BEGINNING OF DATA SEGMENT) :
1102    ; 40BH ABSOLUTE, 3 WORDS) :
1103    ; :
1104    ; DATA AREA PRINT_TIM_OUT (BYTE) MAY BE CHANGED TO CAUSE DIFFERENT :
1105    ; TIME-OUT WAITS. DEFAULT=20 :
1106    ; :
1107    ; REGISTERS AH IS MODIFIED :
1108    ; ALL OTHERS UNCHANGED :
1109    ;--------------------------------------------------------------------
1110    ; ASSUME CS:CODE,DS:DATA
1111    ; ORG 0E700H
1112    ; PRINTER_ID PROC FAR
1113    ; STI ; INTERRUPTS BACK ON
1114    ; PUSH DS ; SAVE SEGMENT
1115    ; MOV SI,DX ; GET PRINTER PARM
1116    ; MOV SI,PRINT_TIM_OUT[SI] ; LOAD TIME-OUT PARM
1117    ; SHL SI,1 ; WORD OFFSET INTO TABLE
1118    ; MOVS DX,PRINTER_BASE[SI] ; GET BASE ADDRESS FOR PRINTER CARD
1119    ; OR DX,DX ; TEST DX FOR ZERO,
1120    ; JZ $+25 ; INDICATING NO PRINTER
1121    ; MOV B,AL ; RETURN
1122    ; OR AL,AH ; TEST FOR (AH)=0
1123    ; MOV AL,DX ; TEST FOR (AH)=1
1124    ; JNZ $+32 ; INIT_PRINTER
1125    ; DEC AL ; TEST FOR (AH)=2
1126    ; DEC AH ; TEST FOR (AH)=3
1127    ; JNZ B1 ; RETURN
1128    ; POP BX ; RECOVER REGISTERS
1129    ; POP DX ; RECOVER REGISTERS
1130    ; IRET
1131    ;------ PRINT THE CHARACTER IN (AL)
1132    ; B2:
1133    ; PUSH AX ; SAVE VALUE TO PRINT
1134    ; OUT DX,AL ; OUTPUT CHAR TO PORT
1135    ; INC DX ; POINT TO STATUS PORT
1136    ; B3:
1137    ; SUB CX,CX ; WAIT_BUSY
1138    ; B3:1:
1139    ; IN AL,DX ; GET STATUS
1140    ; MOVS AH,AL ; STATUS TO AH ALSO
1141    ; TEST AL,60H ; IS THE PRINTER CURRENTLY BUSY
1142    ; JNZ B4 ; OUT_STROBE
1143    ; LOOP B3_1 ; TRY AGAIN
1144    ; DEC BL ; DROP LOOP COUNT
1145    ; JNZ B3 ; GO TILL TIMEOUT ENDS
1146    ; OR AH,1 ; SET ERROR FLAG
1147    ; AND AH,0FH ; TURN OFF THE OTHER BITS
1148    ; JMP SHORT B7 ; RETURN WITH ERROR FLAG SET
1149    ; B4:
1150    ; MOVS AL,DX ; SET THE STROBE HIGH
1151    ; OUT DX,AL ; STROBE IS BIT 0 OF PORT C OF 8255
1152    ; OUT DX,AL ; SET THE STROBE LOW
1153    ; POP AX ; RECOVER THE OUTPUT CHAR
1154    ; B5:
1155    ;------ PRINTER STATUS
1156    ; B6:
1157    ; push ax ; save al reg
1158    ; mov dx,printer_base[si] ; get printer status
FO25 8004F8 3177  AND AL,0FH ; TURN OFF UNDEFINED BITS
FO27B 3179  POP DX ; STATUS_SET
FO29 5A 3180  MOV AL,DL ; RECOVER AL REG
FO29 8AC2 3181  XOR AH,AL ; GET CHARACTER INTO AL
FO2B 80F4F8 3182  JMP BI ; FLIP A COUPLE OF BITS
FO2E 5BC5 3183  RETURN FROM ROUTINE
3185
3186
3187
3189
3190
3191
3192
3193
3194
3195
3196
3197
3198
3199
3200
3201
3202
3203
3204
3205
3206
3207
3208
3209
3210
3211
3212
3213
3214
3215
3216
3217
3218
3219
3220
3221
3222
3223
3224
3225
3226
3227
3228
3229
3230
3231
3232
3233
3234
3235
3236
3237
3238
3239
3240
3241
3242
3243
3244
3245
3246
3247
3248
3249
3250
3251
3252
3253
3184 I------- INITIALIZE THE PRINTER PORT
3185
3186
3187
3189
3190
3191
3192
3193
3194
3195
3196
3197
3198
3199
3200
3201
3202
3203 I--- INT 10 ----------------------------------------
3204 1 VIDEO_ID:
3205 1 THESE ROUTINES PROVIDE THE CRT INTERFACE
3206 1 THE FOLLOWING FUNCTIONS ARE PROVIDED:
3207 1 (AH)=0 SET MODE (AL) CONTAINS MODE VALUE
3208 1 (AL)=0 40X25 BW (POWER ON DEFAULT)
3209 1 (AL)=1 40X25 COLOR
3210 1 (AL)=2 80X25 BW
3211 1 (AL)=3 80X25 COLOR
3212 1 GRAPHICS MODES
3213 1 (AL)=4 320X200 COLOR
3214 1 (AL)=5 320X200 BW
3215 1 (AL)=6 640X200 BW
3216 1 CRT MODE=7 80X25 B&W CARD (USED INTERNAL TO VIDEO ONLY)
3217 1 *** NOTE BW MODES OPERATE SAME AS COLOR MODES, BUT
3218 1 COLOR BURST IS NOT ENABLED
3219 1 (AH)=1 SET CURSOR TYPE
3220 1 (CH) = BITS 4-0 = START LINE FOR CURSOR
3221 1 ** HARDWARE WILL ALWAYS CAUSE BLINK
3222 1 ** SETTING BIT 5 OR 6 WILL CAUSE ERRATIC
3223 1 BLINKING ON NO CURSOR AT ALL
3224 1 (CL) = BITS 4-0 = END LINE FOR CURSOR
3225 1 (AH)=2 SET CURSOR POSITION
3226 1 (DH,DL) = ROW,COLUMN (0,0) IS UPPER LEFT
3227 1 (BH) = PAGE NUMBER (MUST BE 0 FOR GRAPHICS MODES)
3228 1 (AH)=3 READ CURSOR POSITION
3229 1 (BH) = PAGE NUMBER (MUST BE 0 FOR GRAPHICS MODES)
3230 1 ON EXIT (DH,DL) = ROW,COLUMN OF CURRENT CURSOR
3231 1 (CH,CL) = CURSOR MODE CURRENTLY SET
3232 1 (AH)=4 READ LIGHT PEN POSITION
3233 1 ON EXIT:
3234 1 (AH) = 0 -- LIGHT PEN SWITCH NOT DOWN/NOT TRIGGERED
3235 1 (AH) = 1 -- VALID LIGHT PEN VALUE IN REGISTERS
3236 1 (DH,DL) = ROW,COLUMN OF CHARACTER LP POSN
3237 1 (CH) = MASTER LINE (0-199)
3238 1 (DX) = PIXEL COLUMN (0-319,639)
3239 1 (AH)=5 SELECT ACTIVE DISPLAY PAGE (VALID ONLY FOR ALPHA MODES)
3240 1 (AL)=NEW PAGE VAL (0-7 FOR MODES 081, 0-3 FOR MODES 2A3)
3241 1 (AH)=6 SCROLL ACTIVE PAGE UP
3242 1 (AL) = NUMBER OF LINES, INPUT LINES BLANKED AT BOTTOM
3243 1 OF WINDOW
3244 1 AL = 0 MEANS BLANK ENTIRE WINDOW
3245 1 (CH,CL) = ROW,COLUMN OF UPPER LEFT CORNER OF SCROLL
3246 1 (DH,DL) = ROW,COLUMN OF LOWER RIGHT CORNER OF SCROLL
3247 1 (BH) = ATTRIBUTE TO BE USED ON BLANK LINE
3248 1 (AH)=7 SCROLL ACTIVE PAGE DOWN
3249 1 (AL) = NUMBER OF LINES, INPUT LINES BLANKED AT TOP
3250 1 OF WINDOW
3251 1 AL = 0 MEANS BLANK ENTIRE WINDOW
3252 1 (CH,CL) = ROW,COLUMN OF UPPER LEFT CORNER OF SCROLL
3253 1 (DH,DL) = ROW,COLUMN OF LOWER RIGHT CORNER OF SCROLL

A-46 System BIOS
(BH) = ATTRIBUTE TO BE USED ON BLANK LINE

(BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY)

(AL) = CHAR READ

(AL) = CHAR TO WRITE

(AL) = COUNT OF CHARACTERS TO WRITE

(BL) = ATTRIBUTE OF CHARACTER (ALPHA)/COLOR OF CHAR

FOR READ/WRITE CHARACTER INTERFACE WHILE IN GRAPHICS MODE, THE
CHARACTERS ARE FORMED FROM A CHARACTER GENERATOR IMAGE
MAINTAINED IN THE SYSTEM ROM. ONLY THE 1ST 128 CHAR.
ARE CONTAINED THERE. TO READ/WRITE THE SECOND 128
CHARS, THE USER MUST INITIALIZE THE POINTER AT
INTERRUPT 1FH LOCATION 0007CH TO POINT TO THE 1K Byte
TABLE CONTAINING THE CODE POINTS FOR THE SECOND
128 CHAR (128-255).

FOR WRITE CHARACTER INTERFACE IN GRAPHICS MODE, THE REPLICA:
FACTOR CONTAINED IN (CX) ON ENTRY Will PRODUCE VALID
RESULTS ONLY FOR CHARACTERS CONTAINED ON THE SAME ROW.
CONTINUATION TO SUCCEEDING LINES WILL NOT PRODUCE
CORRECTLY.

GRAPHICS INTERFACE

(AH) = 11 SET COLOR PALETTE

(BH) = PALETTE COLOR ID BEING SET (0-127)

(BL) = COLOR VALUE TO BE USED WITH THAT COLOR ID

COLOR ID = 0 SELECTS THE BACKGROUND COLOR (0-15):
COLOR ID = 1 SELECTS THE PALETTE TO BE USED:
0 = GREEN/1/RED(2)/YELLOW(3)
1 = CYNTHIA/MAGENTA(2)/WHITE(3)

IN 40X25 OR 80X25 ALPHA MODES, THE VALUE SET
FOR PALETTE COLOR 0 INDICATES THE
BORDER COLOR TO BE USED (VALUES 0-31, WHERE 16-31 SELECT THE HIGH INTENSITY
BACKGROUND SET.

(AH) = 12 WRITE DOT

(DX) = ROW NUMBER

(CX) = COLUMN NUMBER

(AL) = COLOR VALUE

IF BIT 7 OF AL = 1, THEN THE COLOR VALUE IS
EXCLUSIVE OR'D WITH THE CURRENT CONTENTS OF
THE DOT.

(AH) = 13 READ DOT

(DX) = ROW NUMBER

(CX) = COLUMN NUMBER

(AL) RETURNS THE DOT READ

ASCII TELETYPE ROUTINE FOR OUTPUT

(AH) = 14 WRITE TELETYPE TO ACTIVE PAGE

(AL) = CHAR TO WRITE

(BL) = FOREGROUND COLOR IN GRAPHICS MODE

NOTE -- SCREEN WIDTH IS CONTROLLED BY PREVIOUS MODE SET

(AH) = 15 CURRENT VIDEO STATE

RETURNS THE CURRENT VIDEO STATE

(AL) = MODE CURRENTLY SET (SEE AH=0 FOR EXPLANATION)

(AH) = NUMBER OF CHARACTER COLUMNS ON SCREEN

(BH) = CURRENT ACTIVE DISPLAY PAGE

CS,SS,DS,ES,CX,DX PRESERVED DURING CALL

ALL OTHERS DESTROYED

----------------------------------------
A-48  System BIOS
FOAM  IC  3402  DB  ICH, 2, 7, 6, 7
FOAD  07
FOAE  06
FOAF  07
FOBO  00  3403  DB  0, 0, 0, 0
FOBI  00
FOB2  00
FOB5  00
FO10  3404  M4  EQU  $-VIDEO_PARM
FOB4  71  3405  DB  71H, 50H, 5AH, 0AH, 1FH, 6, 19H ; SET UP FOR 80X25
FOB5  50
FOB6  9A
FOB7  0A
FOB8  1F
FOB9  06
FOBB  1C  3407  DB  ICH, 2, 7, 6, 7
FOBC  02
FOBE  06
FOBF  07
FOC0  00
FOCI  00
FOC2  00
FOC3  00
FOC4  38  3408  DB  38H, 20H, ZDH, 0AH, 7FH, 6, 6AH ; SET UP FOR GRAPHICS
FOC5  28
FOC6  2D
FOC7  0A
FOC8  7F
FOC9  06
FOCA  64
FOCB  70  3409  DB  70H, 2, 1, 6, 7
FOCC  02
FOCD  01
FOCE  06
FOCF  07
FODI  00
FOD2  00
FOD5  00
FOD4  61  3410  DB  61H, 50H, 52H, 0FH, 19H, 6, 19H ; SET UP FOR 80X25 B&W CARD
FOD5  50
FOD6  92
FOD7  0F
FOD8  19
FOD9  06
FODB  19  3411  DB  19H, 2, 0DH, 0BH, 0CH
FODC  02
FODD  00
FODE  08
FODF  0C
FOEO  00
FOEI  00
FOE2  00
FOE3  00
FOE4  3412  DB  0, 0, 0, 0
FOE5  50
FOE6  52
FOE7  0F
FOE8  19
FOE9  62
FOEB  9A
FOEC  40
FOED  00
FOEE  00
FOEF  50
FOFO  28
FOFC  38
FOFB  19
FOFA ]}</ref>
LOC OBJ | LINE | SOURCE

F0F2 50 | 3420 | i----- C_REG_TAB
F0F3 50 | 3429 |
F0F4 | 3430 | TABLE OF MODE SETS
F0F4 IC | 3431 | M7 LABEL BYTE
F0F4 2A | 3432 | DB 2CH,2DH,2EH,28H,2AH,2EH,28H,2AH
F0F5 2A | 3433 |
F0F6 2A | 3434 |
F0F7 29 | 3435 |
F0F8 2A | 3436 |
F0F9 2A | 3437 |
F0FA 2A | 3438 |
F0FB 2A | 3439 |
F0FC | 3440 | SET_MODE PROC HN
F0FD | 3441 | MOV DI,0304H ; ADDRESS OF COLOR CARD
F0FF B500 | 3442 | MOV BL,0 ; MODE SET FOR COLOR CARD
F101 83F830 | 3443 | CMP DI,130H ; IS BM CARD INSTALLED
F104 7506 | 3444 | JNE M0 ; OK WITH COLOR
F106 B007 | 3445 | MOV AL,7 ; INDICATE BM CARD MODE
F108 B204 | 3446 | MOV DL,0DH ; ADDRESS OF BM CARD (304)
F10A FECC | 3447 | INC DL ; MODE SET FOR BM CARD
F10C | 3448 | MOV AH,AL ; SAVE MODE IN AH
F10E A2500 | 3449 | MOV CRH,MODE,AL ; SAVE IN GLOBAL VARIABLE
F110 B02300 | 3450 | MOV ADDR_ADDR,DX ; SAVE ADDRESS OF BASE
F115 1E | 3451 | PUSH DS ; SAVE POINTER TO DATA SEGMENT
F116 50 | 3452 | PUSH AX ; SAVE MODE
F117 52 | 3453 | PUSH DX ; SAVE OUTPUT PORT VALUE
F118 83C204 | 3454 | ADD DX,4 ; POINT TO CONTROL REGISTER
F11B 4AC3 | 3455 | MOV AL,AL ; GET MODE SET FOR CARD
F11D EE | 3456 | OUT DX,AL ; RESET VIDEO
F11E 5A | 3457 | POP DX ; BACK TO BASE REGISTER
F11F 28C0 | 3458 | SUB AX,AX ; SET UP FOR ABS2 SEGMENT
F121 000F | 3459 | MOV DS,AX ; ESTABLISH VECTOR TABLE ADDRESSING
F123 C51E7400 | 3460 | ASSUME DS:ABS20
F125 5B | 3461 | LDS BX,PARM_PTR ; GET POINTER TO VIDEO PARMs
F127 5B | 3462 | POP AX ; RECOVER PARMs
F12B 91000 | 3463 | MOV CX,6A ; LENGTH OF EACH ROW OF TABLE
F12F 50C002 | 3464 | CMP AH,2 ; DETERMINE WHICH ONE TO USE
F13E 7210 | 3465 | JC M9 ; MODE IS 0 OR 1
F13F 00D9 | 3466 | ADD BX,CX ; MOVE TO NEXT ROW OF INIT TABLE
F142 00F04 | 3467 | CMP AH,4 ; MODE IS 2 OR 3
F143 7209 | 3468 | JC M9 ; MODE IS 2 OR 3
F145 00D9 | 3469 | ADD BX,CX ; MOVE TO GRAPhICS ROW OF INIT_TABLE
F149 00F07 | 3470 | CMP AH,7
F14C 7202 | 3471 | JC M9 ; MODE IS 4,5, OR 6
F14E 00D9 | 3472 | ADD BX,CX ; MOVE TO BM CARD ROW OF INIT_TABLE
F151 3549 | 3473 | i----- BST POINTS TO CORRECT ROW OF INITIALIZATION TABLE
F154 2A | 3474 | OUT_INIT
F155 50 | 3475 | PUSH AX ; SAVE MODE IN AH
F157 3264 | 3476 | XOR AH,AH ; AH WILL SERVE AS REGISTER
F157 77 | 3477 | i----- LOOP THROUGH TABLE, OUTPUTTING REG ADDRESS, THEN VALUE FROM TABLE
F159 8A | 3478 | i----- LOOP THROUGH TABLE, OUTPUTTING REG ADDRESS, THEN VALUE FROM TABLE
F15C | 3479 | i----- LOOP THROUGH TABLE, OUTPUTTING REG ADDRESS, THEN VALUE FROM TABLE
F15F | 3480 | M10: ; INIT LOOP
F163 BAC4 | 3481 | MOV AL,AH ; GET 6455 REGISTER NUMBER
F165 EE | 3482 | OUT DX,AL
F166 42 | 3483 | INC DX ; POINT TO DATA PORT
F167 FECC | 3484 | INC AH ; NEXT REGISTER VALUE
F16A 4007 | 3485 | MOV AL,BX ; NEXT REGISTER VALUE
F16C EE | 3486 | OUT DX,AL ; OUT TO CHIP
F16C 43 | 3487 | INC BX ; NEXT IN TABLE
F170 4A | 3488 | DEC DX ; BACK TO POINTER REGISTER
F174 2F3 | 3489 | LOOP M10 ; DO THE WHOLE TABLE
F178 50 | 3490 | POP AX ; GET MODE BACK
F181 1F | 3491 | POP DS ; RECOVER SEGMENT VALUE
F186 35FF | 3492 | ASSUME DS:DATA
F18F | 3493 | i----- FILL REGEN AREA WITH BLANK
F194 | 3494 | i----- FILL REGEN AREA WITH BLANK
F152 33FF | 3495 | XOR DI,DI ; SET UP POINTER FOR REGEN

A-50 System BIOS
AND NOT SET UP OVERSCAN REGISTER

F154 B93E4600 3494 MOV CRT_START.DI ; START ADDRESS SAVED IN GLOBAL
F156 B06620000 3497 MOV ACTIVE_PAGE.O ; SET PAGE VALUE
F15D B90020 3498 MOV CX,6192 ; NUMBER OF WORDS IN COLOR CARD
F160 69FC04 3499 CMP AH,4 ; TEST FOR GRAPHICS
F163 7000 3500 JC MS2 ; HD GRAPHICS_INIT
F165 69FC07 3501 CMP AH,7 ; TEST FOR BW CARD
F168 7004 3502 JE MI1 ; BW_CARD_INIT
F16A 33C0 3503 XOR AX,AX ; FILL FOR GRAPHICS MODE
F16C 8050 3504 JMP SHORT M13 ; CLEAR BUFFER
F16E M11: 3505 MOV AX, 'A' ; BH_CARD_INIT
F16E B508 3506 MOV CH,08H ; BUFFER SIZE ON BW CARD
F170 33E8 3507 MOV CH,0AH ; HD GRAPHICS_INIT
F170 B80007 3508 MOV AX,'A'+7*256 ; FILL CHAR FOR ALPHA
F175 M13: 3509 MOV CX,8192 ; NUMBER OF WORDS IN COLOR CARD
F175 B20007 3510 REP STOSW ; FILL THE REGEN BUFFER WITH BLANKS
F174 AB 3511

;----- ENABLE VIDEO AND CORRECT PORT SETTING
F175 C7000000706 3512 MOV CURSOR_MODE,607H ; SET CURRENT CURSOR MODE
F170 A0400 3514 MOV AL,CRT_MODE ; GET THE MODE
F176 32E4 3515 XOR AH,AH ; INTO AX REGISTER
F180 B8F0 3516 MOV SI,AX ; TABLE POINTER, INDEXED BY MODE
F182 80166300 3517 MOV DX,ADDR_6445 ; PREPARE TO OUTPUT TO
F184 80FC04 3518 MOV AL,CH ; VIDEO ENABLE PORT
F186 3520 3519 XOR AX,4 ; FILL THE REGEN BUFFER
F186 353A 3520 ADD DX,4 ; CLEAR BUFFER
F189 3521 MOV AL,C.S:[SI+OFFSET H6] ; FILL THE REGEN BUFFER
F18E EE 3522 OUT DX,AL ; SET VIDEO ENABLE PORT
F190 A56500 3523 MOV CRT_MODE_SET,AL ; SAVE THAT VALUE
F191 M24: 3524

;----- DETERMINE NUMBER OF COLUMNS, BOTH FOR ENTIRE DISPLAY
F192 3525 MOV AL,C.S:[SI+OFFSET M6] ;----- AND THE NUMBER TO BE USED FOR TTY INTERFACE
F197 3526 MOV AX,AH,AX
F199 3527 MOV CRT_COLS,AX ; NUMBER OF COLUMNS IN THIS SCREEN
F19C B100000 3528 MOV AL,C.S:[SI+OFFSET M6] ;----- SET CURSOR POSITIONS
F1A0 3529 MOV CX,CS:[SI+OFFSET M5] ;----- AND THE NUMBER TO BE USED FOR TTY INTERFACE
F1A5 3530 MOV CX,CS:[SI+OFFSET M5] ;----- LENGTH TO CLEAR
F1A9 B09000 3531 MOV CX,CH ; SAVE LENGTH OF CRT -- NOT USED FOR BM
F1AC 3532 MOV CX,8 ; CLEAR ALL CURSOR POSITIONS
F1AF E5000 3533 MOV AL,OFFSET_CURSOR_POSN
F1AF 1E 3534 PUSH DS ; ESTABLISH SEGMENT
F1BD 3535 POP ES ; ADDRESSING
F1BF 3536 POP AX,AX
F1B3 F3 3537 REP STOSW ; FILL WITH ZEROES
F1BD AB 3538

;----- SET UP OVERSCAN REGISTER
F1B5 42 3539 INC DX ; SET OVERSCAN PORT TO A DEFAULT
F1BD 0030 3540 MOV AL,30H ; VALUE OF 30H FOR ALL MODES
F1BD 6030 3541 MOV AL,30H ; VALUE OF 30H FOR ALL MODES
F1BD 083F900006 3542 MOV CX,CS:[SI+OFFSET M5] ; SEE IF THE MODE IS 640X200 BM
F1BD 7502 3543 CMP MS2,0 ; IF IT ISN'T 640X200, THEN GOTO REGULAR
F1BF 083F 3544 MOV AL,3FH ; IF IT IS 640X200, THEN PUT IN 3FH
F1C1 3545 M14: 3546 MOV AX,AH,AX
F1C1 1E 3547 OUT DX,AL ; OUTPUT THE CORRECT VALUE TO 3D9 PORT
F1CE A06400 3548 MOV CRT_PALETTE,AL ; SAVE THE VALUE FOR FUTURE USE
F1CE 3549

;----- NORMAL RETURN FROM ALL VIDEO RETURNS
F1C5 3550 VIDEO_RETURN:
F1C5 SF 3551 POP DI
F1C6 SE 3552 POP SI
F1C7 SB 3553 POP BX
F1C8 3554 M15: 3555 VIDEO_RETURN_C
F1C8 59 3556 POP CX
F1C9 SA 3557 POP DX
F1CA 1F 3558 POP DS
F1CB 07 3559 POP ES ; RECOVER SEGMENTS
F1CC CF 3560 JRET ; ALL DONE
F1CC 3561
F1CE 3562 SET_MODE.ENDP
F1C9 3563 I RET ; ALL DONE
F1C9 3564 SET_TYPE

Appendix A

System BIOS A-51
LOC OBJ | LINE | SOURCE
--- | --- | ---
3571 | ; THIS ROUTINE SETS THE CURSOR VALUE
3572 | ;
3573 | ; INPUT
3574 | ; OUTPUT
3575 | ; NONE
3576 | ;
3577 | ;
3578 | ;
3579 | ;
3580 | ;
3581 | ;
3582 | ;
3583 | ;
3584 | ;
3585 | ;
3586 | ;
3587 | ;
3588 | ;
3589 | ;
3590 | ;
3591 | ;
3592 | ;
3593 | ;
3594 | ;
3595 | ;
3596 | ;
3597 | ;
3598 | ;
3599 | ;
3600 | ;
3601 | ;
3602 | ;
3603 | ;
3604 | ;
3605 | ;
3606 | ;
3607 | ;
3608 | ;
3609 | ;
3610 | ;
3611 | ;
3612 | ;
3613 | ;
3614 | ;
3615 | ;
3616 | ;
3617 | ;
3618 | ;
3619 | ;
3620 | ;
3621 | ;
3622 | ;
3623 | ;
3624 | ;
3625 | ;
3626 | ;
3627 | ;
3628 | ;
3629 | ;
3630 | ;
3631 | ;
3632 | ;
3633 | ;
3634 | ;
3635 | ;
3636 | ;
3637 | ;
3638 | ;
3639 | ;
3640 | ;
3641 | ;
3642 | ;
3643 | ;
3644 | ;
3645 | ;
3646 | ;
3647 | ;

A-52 System BIOS
THE ROUTINE READS THE CURRENT CURSOR VALUE FROM THE

6845, FORMATS IT, AND SENDS IT BACK TO THE CALLER.

INPUT:

BH = PAGE OF CURSOR

OUTPUT:

DX = ROW, COLUMN OF THE CURRENT CURSOR POSITION

CX = CURRENT CURSOR MODE

READ_CURSOR PROC NEAR

F264 0B166300 MOV DX,ADDR_6845 ; I/O PORT FOR PALETTE
F265 03C205 ADD DX,5 ; OVERSCAN PORT
F266 0A0600 MOV AL,CRT_PALETTE ; GET THE CURRENT PALETTE VALUE
F267 0A0FF OR BH,BH ; IS THIS COLOR 0?
F268 756E JNZ M20 ; OUTPUT COLOR I
F270 3607 ; THE COLOR SELECTION IS UPDATED
F270 SET_COLOR PROC NEAR

F270 3607 MOV DX,ADDR_6845 ; I/O PORT FOR PALETTE
F270 3608 ADD DX,5 ; OVERSCAN PORT
F270 3609 MOV AL,CRT_PALETTE ; GET THE CURRENT PALETTE VALUE
F270 360A OR BH,BH ; IS THIS COLOR 0?
F270 360B JNZ M20 ; OUTPUT COLOR I
F270 360C ; THE COLOR SELECTION IS UPDATED

F271 360D ;----------------------- HANDLE COLOR 0 BY SETTING THE BACKGROUND COLOR
F271 360E ;----------------------- HANDLE COLOR 1 BY SELECTING THE PALETTE TO BE USED

Appendix A

System BIOS  A-53
A-54 System BIOS

LOC OBJ

LINE

SOURCE

F26A 3725 M2D:
F26A 24DF 3726 AND AL,0DFH ; TURN OFF PALETTE SELECT BIT
F26C D0EB 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 EB0F 3730 JMP M19 ; GO DO IT
F271 571 3731 SET_COLOR END

F272 : VIDEO STATE
F274 :-----------------------------
F277 5A264A00 3740 MOV AH,BYTE PTR CRT_COLS ; GET NUMBER OF COLUMNS
F27B A04000 3741 MOV AL,CRT_MODE ; CURRENT MODE
F27B 8A366200 3742 MOV BH,ACTIVE_PAGE ; GET CURRENT ACTIVE PAGE
F27F 5F 3743 POP DI ; RECOVER REGISTERS
F280 5E 3744 POP SI
F281 59 3745 POP CX ; DISCARD SAVED
F282 E943FF 3746 JMP M15 ; RETURN TO CALLER

F28E :-----------------------------
F285 : POSITION
F285 55 3758 PUSH BX ; SAVE REGISTER
F286 80B8 3759 MOV BX,AX
F288 8AC4 3760 MOV AL,AX ; ROWS TO AL
F29A F264A00 3761 MUL BYTE PTR CRT_COLS ; DETERMINE BYTES TO ROW
F29E 32FF 3762 XOR BH,BH
F29F 03C3 3763 ADD AX,BX ; ADD IN COLUMN VALUE
F2A0 D1E0 3764 SAL AX,1 ; * 2 FOR ATTRIBUTE BYTES
F2A4 5B 3765 POP BX
F2A9 C3 3766 BRET
F2B0 : POSITION
F2B7 :-----------------------------
F2B8 : SCROLL_UP
F2B8 53 3778 PUSH BX ; SAVE REGISTER
F2B9 80B8 3779 MOV BX,AX
F2BA F264A00 3780 MUL BYTE PTR CRT_COLS ; DETERMINE BYTES TO ROW
F2BD 7200 3781 JC N1 ; HANDLE SEPARATELY
F2BF 06FC07 3782 JMP AH,7 ; TEST FOR BM CARD
F2C4 7F03 3783 JE N1
F2CA 6F001 3784 JMP GRAPHICS_UP
F2C5 3791 N1: ; UP_CONTINUE
F2C5 53 3792 PUSH BX ; SAVE ADDRESS IN BL
F2C6 8BC1 3793 MOV AX,CX ; UPPER LEFT POSITION
F2CA 03700 3794 CALL SCROLL_POSITION ; GO SETUP FOR SCROLL
F2C4 7F31 3795 JZ NZ ; BLANK_FIELD
F2C4 03F0 3796 ADD SI,AX ; FROM ADDRESS
F2CF BA06 3797 MOV AH,DH ; 2 ROWS IN BLOCK
F2B1 2AE3 3798 SUB AH,BL ; 2 ROWS TO BE MOVED
F2D3 3799 NZ: ; ROW_LOOP
F2D3 E7700 3800 CALL NZ0 ; MOVE ONE ROW
F2E6 03F5 3801 ADD SI,BP

A-54 System BIOS
HANDLE

tOVE_ROW

f2BA FEee

Fll8

Fll4

FllC F6264AOO 3861

F3U 06

F320

F32B 57 3874

F328

F32:6

F327 C3

F323

F32A SE

F32E 5F

F32F SE

LOC OBJ

LINE

SOURCE

F2BA 03FD 3002 ADD DI, BP ; POINT TO NEXT LINE IN BLOCK

F2BA FECC 3003 DEC AH ; COUNT OF LINES TO MOVE

F2BC 7F5F 3004 JNZ H2 ; ROM_LOOP

F2BE N3: 3005 MVI N3: CLEAR_ENTRY

F2BE SA 3006 POP AX ; RECOVER ATTRIBUTE IN AH

F2BF 8020 3007 MOV AL, * ; FILL WITH BLANKS

F2C1 H4: 3008 MOV AL, H4 : CLEAR_LOOP

F2C1 E06000 3009 CALL H11 ; CLEAR THE ROW

F2C4 03FD 3010 ADD DI, BP ; POINT TO NEXT LINE

F2CA 7F5F 3011 DEC DL ; COUNTER OF LINES TO SCROLL

F2CA N5: 3012 JNZ H4 ; CLEAR_LOOP

F2CA E08C07 3013 CALL D05

F2CD 603E490007 3014 CMP CRT_MODE, 7 ; IS THIS THE BLACK AND WHITE CARD

F2DE 7947 3015 JE N6 ; IF SO, SKIP THE MODE RESET

F2DE 40500 3016 MOV AL, CRT_MODE_SET ; SET THE VALUE OF THE MODE SET

F2D7 B40003 3017 MOV DX, 03DAH ; ALWAYS SET COLOR CARD PORT

F2DA E1 3019 OUT DX, AL

F2DB N6: 3080 MOV DI, AX ; VIDEO_RET_HERE

F2DE 8EFE 3081 JMP VIDEO_RETURN

F2DF N7: 3082 MOV DL, BL ; GET ROW COUNT

F2E0 EBDC 3083 JMP N3 ; GO CLEAR THAT AREA

F2E5 3085 SCROLL_UP ENDP

F2E6 3086 ; ------ HANDLE COMMON SCROLL SET UP HERE

F2E7 3087 SCROLL_POSITION PROC NEAR

F2E8 603E490002 3089 CMP CRT_MODE, 2 ; TEST FOR SPECIAL CASE HERE

F2E9 7210 308A JB N9 ; HAVE TO HANDLE 80X25 SEPARATELY

F2EE 7711 308B CMP CRT_MODE, 3 ; CLEAR LOOP

F2E9 308C JA N9

F2EA 308D ; CLEAR ENTRY

F2EB 308E ROW_LOOP

F2EC 308F ADD DI, 30AH ; GUARANTEED TO BE COLOR CARD HERE

F2F0 52 3090 PUSH DX

F2F1 B4A03 3091 MOV DX, 30AH

F2F4 50 3092 PUSH AX

F2F5 N8: 3093 MOV AX, 0 ; WAIT_DISP_ENABLE

F2F6 EC 3094 IN AL, DX ; GET PORT

F2F7 A800 3095 TEST AL, 0 ; WAIT FOR VERTICAL RETRACE

F2F8 74FB 3096 JZ N8 ; WAIT_DISP_ENABLE

F2F9 D025 3097 MOV AL, ESH

F2FA D20B 3098 MOV DL, DS: DH

F2FB EE 3099 OUT DX, AL ; TURN OFF VIDEO

F2FF 5A 309A POP AX ; DURING VERTICAL RETRACE

F300 S5 309B POP DX

F301 N9: 309C CALL POSITION ; CONVERT TO REGEN POINTER

F302 0364E00 309D ADD AX, CRT_START ; OFFSET OF ACTIVE PAGE

F303 08F0 309E MOV SI, AX ; TO ADDRESS FOR SCROLL

F304 08F0 309F MOV SI, AX ; FROM ADDRESS FOR SCROLL

F305 2801 30A0 SUB DX, CX ; AX = # ROMS, WOLS IN BLOCK

F306 FE6C 30A1 INC DH

F310 32ED 30A2 INC DL ; INCREMENT FOR 0 ORIGIN

F312 EE0D 30A3 XOR CL, CH ; SET HIGH BYTE OF COUNT TO ZERO

F314 0B2E4A00 30A4 MOV BP, CRT_COLS ; GET NUMBER OF COLUMNS IN DISPLAY

F316 03D0 30A5 ADD BP, BP ; TIMES 2 FOR ATTRIBUTE BYTE

F31A 0AC3 30A6 MOV AL, BL ; GET LINE COUNT

F31C F8664A00 30A7 MUL BYTE PTR CRT_COLS ; DETERMINE OFFSET FROM ADDRESS

F322 0DC 30A8 ADD AX, AX ; #2 FOR ATTRIBUTE BYTE

F323 06 30A9 PUSH ES ; ESTABLISH ADDRESSING TO REGEN BUFFER

F325 1F 30AA POP DS ; FOR BOTH POINTERS

F326 08F000 30AB CMP BL, 0 ; 0 SCROLL MEANS BLANK FIELD

F327 C3 30AC RET ; RETURN WITH FLAGS SET

F32E 30AC SCROLL_POSITION ENDP

F330 30AD ; ------ MOVE_ROW

F331 30AE ; PROC NEAR

F332 04CA 30AF MOV CL, BL ; GET # OF COLS TO MOVE

F332 S6 30B0 PUSH SI

F332 S7 30B1 PUSH DI ; SAVE START ADDRESS

F332 F3 30B2 REP MOVSW ; MOVE THAT LINE ON SCREEN

F332 A5 30B3\n
F332 5F 30B4 POP DI ; RECOVER ADDRESSES

System BIOS A-55
LOC OBJ | LINE | SOURCE
--- | --- | ---
F330 C3 | 3878 | RET
F330 | N10 | ENDP
F330 | | 3880
F330 | | 3881
F330 | | 3882
F331 | 3883 | PROC NEAR
F331 | 3884 | MOV CL,DL ; GET % COLUMNS TO CLEAR
F335 | 3885 | PUSH DI
F335 | 3886 | REP STOSW ; STORE THE FILL CHARACTER
F335 | 3887 | AD
F336 | 3888 | SF
F337 | 3889 | CF
F337 | 3890 | CF
F338 | 3891 | CF
F339 | 3892 | CF
F340 | 3893 | CF
F341 | 3894 | CF
F342 | 3895 | CF
F343 | 3896 | CF
F344 | 3897 | CF
F345 | 3898 | CF
F346 | 3899 | CF
F347 | 3900 | CF
F348 | 3901 | CF
F349 | 3902 | CF
F350 | 3903 | CF
F351 | 3904 | CF
F352 | 3905 | CF
F353 | 3906 | CF
F354 | 3907 | STD ; DIRECTION FOR SCROLL DOWN
F355 | 3908 | MOV BL,AL ; LINE COUNT TO BL
F356 | 3909 | CMP AH,4 ; TEST FOR GRAPHICS
F357 | 3910 | JC N12
F358 | 3911 | JMP SF07 ; TEST FOR BM CARD
F359 | 3912 | JE N12
F359 | 3913 | JMP SF03
F359 | 3914 | JMP ENP
F360 | 3915 | SCROLL_DOWN PROC NEAR
F360 | 3916 | MOV BX,SI ; CONTINUE_DOWN
F361 | 3917 | CALL EXTENDED ; GET REGEN POSITION
F362 | 3918 | JC N16
F363 | 3919 | SUB SI,AX ; SI IS FROM ADDRESS
F364 | 3920 | MOV AH,DL ; GET TOTAL # ROWS
F364 | 3921 | MOV AH,BX ; COUNT TO MOVE IN SCROLL
F364 | 3922 | MOV AX,DX
F365 | 3923 | CALL N10 ; MOVE ONE ROW
F366 | 3924 | MOV DL,DH
F366 | 3925 | CALL HI
F367 | 3926 | MOV AH,DX ; CLEAR ONE ROW
F367 | 3927 | MOV DL,DX ; GO TO NEXT ROW
F368 | 3928 | MOV DH,DI
F368 | 3929 | MOV DL,SI
F369 | 3930 | MOV BI,DI
F370 | 3931 | MOV AX,AL
F370 | 3932 | MOV AX,AL
F370 | 3933 | MOV AX,AL
F370 | 3934 | MOV AX,AL
F370 | 3935 | MOV AX,AL
F370 | 3936 | MOV AX,AL
F370 | 3937 | MOV AX,AL
F370 | 3938 | MOV AX,AL
F370 | 3939 | MOV AX,AL
F370 | 3940 | MOV AX,AL
F370 | 3941 | MOV AX,AL
F371 | 3942 | MOV AX,AL
F371 | 3943 | MOV AX,AL
F371 | 3944 | MOV AX,AL
F371 | 3945 | MOV AX,AL
F371 | 3946 | MOV AX,AL
F371 | 3947 | MOV AX,AL
F371 | 3948 | MOV AX,AL
F371 | 3949 | MOV AX,AL
F371 | 3950 | MOV AX,AL
F371 | 3951 | MOV AX,AL
F371 | 3952 | MOV AX,AL
F371 | 3953 | MOV AX,AL
F371 | 3954 | MOV AX,AL
F371 | 3955 | MOV AX,AL
F371 | 3956 | MOV AX,AL
F371 | 3957 | MOV AX,AL
F371 | 3958 | MOV AX,AL
F371 | 3959 | MOV AX,AL
F371 | 3960 | MOV AX,AL
F371 | 3961 | MOV AX,AL
F371 | 3962 | MOV AX,AL
F371 | 3963 | MOV AX,AL
F371 | 3964 | MOV AX,AL
F371 | 3965 | MOV AX,AL
F371 | 3966 | MOV AX,AL
F371 | 3967 | MOV AX,AL
F371 | 3968 | MOV AX,AL
F371 | 3969 | MOV AX,AL
F371 | 3970 | MOV AX,AL
F371 | 3971 | MOV AX,AL
F371 | 3972 | MOV AX,AL
F371 | 3973 | MOV AX,AL
F371 | 3974 | MOV AX,AL
F371 | 3975 | MOV AX,AL
F371 | 3976 | MOV AX,AL
F371 | 3977 | MOV AX,AL
F371 | 3978 | MOV AX,AL
F371 | 3979 | MOV AX,AL
F371 | 3980 | MOV AX,AL
F371 | 3981 | MOV AX,AL
F371 | 3982 | MOV AX,AL
F371 | 3983 | MOV AX,AL
F371 | 3984 | MOV AX,AL
F371 | 3985 | MOV AX,AL
F371 | 3986 | MOV AX,AL
F371 | 3987 | MOV AX,AL
F371 | 3988 | MOV AX,AL
F371 | 3989 | MOV AX,AL
F371 | 3990 | MOV AX,AL
F371 | 3991 | MOV AX,AL
F371 | 3992 | MOV AX,AL
F371 | 3993 | MOV AX,AL
F371 | 3994 | MOV AX,AL
F371 | 3995 | MOV AX,AL
F371 | 3996 | MOV AX,AL
F371 | 3997 | MOV AX,AL
F371 | 3998 | MOV AX,AL
F371 | 3999 | MOV AX,AL
F371 | 4000 | MOV AX,AL
F371 | 4001 | MOV AX,AL
F371 | 4002 | MOV AX,AL
F371 | 4003 | MOV AX,AL
F371 | 4004 | MOV AX,AL
F371 | 4005 | MOV AX,AL
F371 | 4006 | MOV AX,AL
F371 | 4007 | MOV AX,AL
F371 | 4008 | MOV AX,AL
F371 | 4009 | MOV AX,AL
F371 | 4010 | MOV AX,AL
F371 | 4011 | MOV AX,AL
F371 | 4012 | MOV AX,AL
F371 | 4013 | MOV AX,AL
F371 | 4014 | MOV AX,AL
F371 | 4015 | MOV AX,AL
F371 | 4016 | MOV AX,AL
F371 | 4017 | MOV AX,AL
F371 | 4018 | MOV AX,AL
F371 | 4019 | MOV AX,AL
F371 | 4020 | MOV AX,AL
F371 | 4021 | MOV AX,AL
F371 | 4022 | MOV AX,AL
F371 | 4023 | MOV AX,AL
F371 | 4024 | MOV AX,AL
F371 | 4025 | MOV AX,AL
F371 | 4026 | MOV AX,AL
F371 | 4027 | MOV AX,AL
F371 | 4028 | MOV AX,AL
F371 | 4029 | MOV AX,AL
F371 | 4030 | MOV AX,AL
F371 | 4031 | MOV AX,AL
F371 | 4032 | MOV AX,AL
F371 | 4033 | MOV AX,AL
F371 | 4034 | MOV AX,AL
F371 | 4035 | MOV AX,AL
F371 | 4036 | MOV AX,AL
F371 | 4037 | MOV AX,AL
F371 | 4038 | MOV AX,AL
F371 | 4039 | MOV AX,AL
F371 | 4040 | MOV AX,AL
F371 | 4041 | MOV AX,AL
F371 | 4042 | MOV AX,AL
F371 | 4043 | MOV AX,AL
A-56 System BIOS
F374 60FC04 3957 CHX AH,4  ; IS THIS GRAPHICS
F374 7268 3958 JC P1  ; IS THIS BM CARD
F377 60FC07 3959 CHX AH,7  ; IS THIS BM CARD
F37C 7403 3960 JE P1
F37E 694002 3961 JMP GRAPHICS_READ
F391 6E1A00 3962 P1: CALL FIND_POSITION
F394 08F3 3963 MOV SI,8X  ; ESTABLISH ADDRESSING IN SI
F396 3964 ----- WAIT FOR HORIZONTAL RETRACE
F397 3965
F398 08163300 3966 MOV DX,ADDR_6045  ; GET BASE ADDRESS
F39A 03C206 3967 ADD DX,6  ; POINT AT STATUS PORT
F39D 06 3968 PUSH ES
F39E 1F 3969 POP DS  ; GET SEGMENT FOR QUICK ACCESS
F39F EC 3970 P2:  ; WAIT FOR RETRACE LOW
F3A0 5901 3971 IN AL,DX  ; GET STATUS
F3A2 75FB 3972 TEST AL,1  ; IS HORIZ RETRACE LOW
F3A4 FA 3973 JNZ P2  ; WAIT UNTIL IT IS
F3A6 3974 CLI  ; NO MORE INTERRUPTS
F3A7 EC 3975 IN AL,DX  ; GET STATUS
F3A9 5901 3976 TEST AL,1  ; IS IT HIGH
F3AB 74FB 3977 JZ P3  ; WAIT UNTIL IT IS
F3AC AD 3978 LODSW  ; GET THE CHAR/ATTR
F3AD EF27 FE 3979 JMP VIDEO_RETURN
F3AE 3980 READ_AC_CURRENT  ; READ AC CURRENT
F3E0 3981 FIND_POSITION  ; FIND POSITION
F3E4 0ACF 3982 MOV CL,BH  ; DISPLAY PAGE TO CX
F3E5 32ED 3983 XOR CH,CH
F3E6 82F1 3984 MOV SI,CX  ; MOVE TO SI FOR INDEX
F3E7 D166 3985 SAL SI,1  ; = 2 FOR WORD OFFSET
F3E8 81F50 3986 MOV AX,[SI+OFFSET POSITION]  ; GET ROW/COLUMN OF THAT PAGE
F3E9 33DB 3987 XOR BX,BX  ; SET START ADDRESS TO ZERO
F3EB 3306 3988 JCXZ P5  ; NO_PAGE
F3F1 01E4 3989 ADD BX,CRT_LEN  ; LENGTH OF BUFFER
F3F2 9400 3990 LOOP P4
F3F3 3991 PS:  ; NO_PAGE
F3F4 E0CFEE 3992 CALL POSITION  ; DETERMINE LOCATION IN REGEN
F3F5 0305 3993 ADD DX,AX  ; ADD TO START OF REGEN
F3F6 C3 3994 RET
F3F9 4000 FIND_POSITION  ; FIND POSITION
F3FA 4001  ; WRITE AC CURRENT
F3FB 4002  ; WRITE AC CURRENT
F3FC 4003  ; THIS ROUTINE WRITES THE ATTRIBUTE
F3FD 4004  ; AND CHARACTER AT THE CURRENT CURSOR
F3FE 4005  ; POSITION
F3FF 4006  ; INPUT
F400 4007  ; (AH) = CURRENT CRT MODE
F401 4008  ; (BH) = DISPLAY PAGE
F402 4009  ; (CX) = COUNT OF CHARACTERS TO WRITE
F403 400A  ; (AL) = CHAR TO WRITE
F404 400B  ; (BL) = ATTRIBUTE OF CHAR TO WRITE
F405 400C  ; (DS) = DATA SEGMENT
F406 400D  ; (ES) = REGEN SEGMENT
F407 400E  ; OUTPUT
F408 400F  ; NONE
F409 4010  ; WRITE AC CURRENT
F40A 4011  ; WRITE AC_CURRENT
F40B 4012  ; GET ATTRIBUTE TO AN
F40D 50 4025 CALL AX  ; SAVE ON STACK
F40E 51 4026 PUSH CX  ; SAVE WRITE COUNT
F40F 58 4027 CALL FIND_POSITION
F411 08FB 4028 MOV DI,BX  ; ADDRESS TO DI REGISTER
F412 59 4029 POP CX  ; WRITE COUNT
F413 5B 4030 POP BX  ; CHARACTER IN BX REG

Appendix A

System BIOS   A-57
LOC OBJ  LINEx SOURCE

F301  4031  P7:  WRITE_LOOP
F302  4032
F303  4033 ;----- WAIT FOR HORIZONTAL RETRACE
F304
F305  4035  MOV DX,ADDR_6045  ; GET BASE ADDRESS
F306  4036  ADD DX,6  ; POINT AT STATUS PORT
F307
F308  4037  P8:  ; THIS ROUTINE WILL WRITE A DOT, OR READ THE DOT AT
F309  4038  IN AL,DX  ; GET STATUS
F310  4039  TEST AL.1  ; IS IT LOW
F311  4040  JNZ P8  ; WAIT UNTIL IT IS
F312  4041  CLI  ; NO MORE INTERRUPTS
F313
F314  4042  P9:  ; INTERRUPTS BACK ON
F315  4043  IN AL,DX  ; GET STATUS
F316  4044  TEST AL.1  ; IS IT HIGH
F317  4045  JZ P9  ; WAIT UNTIL IT IS
F318  4046  MOV AX,BX  ; RECOVER THE CHAR/ATTR
F319  4047  STOSW  ; PUT THE CHAR/ATTR
F320  4048  STI  ; INTERRUPTS BACK ON
F321  4049  LOOP P7  ; AS MANY TIMES AS REQUESTED
F322  4050  JMP VIDEO_RETURN
F323  4051  WRITE_AC_CURRENT ENDP
F324  4052 ;---------------------------------------------------------------------
F325  4053 ; WRITE_C_CURRENT
F326  4054 ; THIS ROUTINE writes the character at
F327  4055 ; the current cursor position, attribute
F328  4056 ; unchanged
F329  4057 ; INPUT
F330  4058 ; (AH) = current CRT mode
F331  4059 ; (BH) = display page
F332  4060 ; (CX) = count of characters to write
F333  4061 ; (AL) = char to write
F334  4062 ; (DS) = data segment
F335  4063 ; (ES) = regen segment
F336  4064 ; output
F337  4065 ; none
F338  4066 ;---------------------------------------------------------------------
F339
F340  4067 WRITE_C_CURRENT PROC NEAR
F341  4068 CMP AH,4  ; IS THIS GRAPHICS
F342  4069 JC P10  ; YES
F343  4070 CMP AH,7  ; IS THIS DM CARD
F344  4071 JE P10  ; YES
F345  4072 JMP GRAPHICS_WRITE
F346
F347  4073 P10:
F348  4074 PUSH AX  ; SAVE ON STACK
F349  4075 PUSH CX  ; SAVE WRITE COUNT
F350  4076 CALL FIND_POSITION
F351  4077 MOV DL,BX  ; ADDRESS TO DL
F352  4078 MOV DX,BX  ; WRITE COUNT
F353  4079 MOV BX  ; BL HAS CHAR TO WRITE
F354  4080 P11:
F355  4081 ; WRITE_LOOP
F356
F357  4082 ;----- WAIT FOR HORIZONTAL RETRACE
F358
F359  4083 MOV DX,ADDR_6045  ; GET BASE ADDRESS
F360  4084 ADD DX,6  ; POINT AT STATUS PORT
F361  4085 ADD DX,6  ; POINT AT STATUS PORT
F362
F363  4086 P12:
F364  4087 IN AL,DX  ; GET STATUS
F365  4088 TEST AL.1  ; IS IT LOW
F366  4089 JNZ P12  ; WAIT UNTIL IT IS
F367  4090 CLI  ; NO MORE INTERRUPTS
F368  4091 P13:
F369  4092 IN AL,DX  ; GET STATUS
F370  4093 TEST AL.1  ; IS IT HIGH
F371  4094 JZ P13  ; WAIT UNTIL IT IS
F372  4095 MOV AL,DL  ; RECOVER CHAR
F373  4096 STOSB  ; PUT THE CHAR/ATTR
F374  4097 STI  ; INTERRUPTS BACK ON
F375  4098 INC DI  ; BUMP POINTER PAST ATTRIBUTE
F376  4099 LOOP P11  ; AS MANY TIMES AS REQUESTED
F377  4100 JMP VIDEO_RETURN
F378  4101 WRITE_C_CURRENT ENDP
F379
F380  4102 ;---------------------------------------------------------------------
F381  4103 ; READ DOT -- WRITE DOT
F382  4104 ; THESE ROUTINES WILL WRITE A DOT, OR READ THE DOT AT
F383  4105 ; THE INDICATED LOCATION
F384  4106 ; ENTRY --
F385  4107 ; DX = ROM (0-199) (THE ACTUAL VALUE DEPENDS ON THE MODE):

A-58 System BIOS
DETERMINE DOT VALUE TO WRITE (1-2 OR 4 BITS DEPENDING ON MODE, RIGHT JUSTIFIED) : 
BIT 7 OF AL+1 INDICATES XOR THE VALUE INTO THE LOCATION :

0 = DATA SEGMENT
1 = ES:REGEN SEGMENT

EXIT :
AL = DOT VALUE READ, RIGHT JUSTIFIED, READ ONLY : 

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

F41E
F41E EB3100
F421 8680A4
F422 22C4
F426 D2E0
F428 0ACE
F42A D2C0
F42C 996FD0
F42E 49F5
F43F 50
F43A 00F5
F434 22C4
F438 268AC
F43B 5B
F43C F6C100
F43F 7500
F441 0A4C
F444 32CC
F447 0AC1
F448 7D00
F44A 50
F44B E77FD0
F44C 3OC1
F44D EB5F
F452 S1
F453 56
F456 B028
F45A 52
F457 00E3FE
F45A F6E2
F45C 5A
F460 F6C201
F464 7000
F467 050020
F468 89F0
F46B 8801
F470 5D
F474 5A
F477 7403
F47B 020020
F47E 8800
F481 07
F488 8800

F41F READ_DOT PROC NEAR
F420 CALL R3
F421 MOV AL,ES:[SI]
F422 AND AL,AH
F423 SHR AL,CL
F424 MOV CL,DM
F425 ROL AL,CL
F426 JMP VIDEO_RETURN
F427 REDOT ENDP
F428
F42F WRITE_DOT PROC NEAR
F430 PVUSH AX
F431 PUSH AX
F432 CALL R3
F433 SHL AL,CL
F434 SRAINT OFF THE OTHER BITS
F435 MOV CL:[ES:SI]
F436 POP CX
F437 TEST BL,60H
F438 JNZ R2
F439 MOV ES:[SI],AL
F442 XOR AH
F443 AND CL,AH
F43R OR AL,CL
F441 FINISH_DOT
F444 MOV ES:[SI],AL
F445 RESTORE THE CURRENT BYTE
F446 XOR AX
F447 XOR CL,AL
F448 XOR AL,CL
F449 XOR AX
F44A XOR CL
F44B XOR AL
F44C XOR CL
F44D XOR AL
F44E XOR CL
F44F XOR AL
F450 PUSH R1
F451 FINISH UP THE WRITING
F452 R3 PROC NEAR
F453 S1
F455 S1
F458 S1
F45B S1
F45D S1
F45F S1
F462 R3
F463 S1
F464 S1
F467 S1
F46A S1
F46C S1
F46D S1
F46F S1
F472 S1
F475 S1
F477 S1
F47A S1
F47C S1
F47E S1
F480 S1
F482 S1

F426 D2E0
F428 0ACE
F42A D2C0
F42C 996FD0
F42E 49F5
F43F 7500
F441 0A4C
F444 32CC
F447 0AC1
F448 7D00
F44A 50
F44B E77FD0
F44C 3OC1
F44D EB5F
F451 S1
F453 S1
F456 B028
F45A 52
F457 00E3FE
F45A F6E2
F45C 5A
F460 F6C201
F464 7000
F467 050020
F468 8801

System BIOS A-59
A-60 System BIOS
DETERMINE THE SOURCE ADDRESS IN THE BUFFER

<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>LINE</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>F40D 7364</td>
<td>4262</td>
<td>JNC R7</td>
<td>; FIND_SOURCE</td>
</tr>
<tr>
<td>4263</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F426</td>
<td></td>
<td></td>
<td>; ---- MEDIUM REG UP</td>
</tr>
<tr>
<td>4265</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F46F D0E2</td>
<td>4266</td>
<td>SAL DL,1</td>
<td>; # COLUMNS = 2, SINCE 2 BYTES/CHAR</td>
</tr>
<tr>
<td>F4B1 D1E7</td>
<td>4267</td>
<td>SAL DI,1</td>
<td>; OFFSET #3 SINCE 2 BYTES/CHAR</td>
</tr>
<tr>
<td>4268</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F469</td>
<td></td>
<td></td>
<td>; ---- DETERMINE THE SOURCE ADDRESS IN THE BUFFER</td>
</tr>
<tr>
<td>4270</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F4B3</td>
<td>4271</td>
<td>R7:</td>
<td>; FIND_SOURCE</td>
</tr>
<tr>
<td>F4B4 D6</td>
<td>4272</td>
<td>PUSH ES</td>
<td>; GET SEGMENTS BOTH POINTING TO REGEN</td>
</tr>
<tr>
<td>F4B4 IF</td>
<td>4273</td>
<td>POP DS</td>
<td></td>
</tr>
<tr>
<td>F4B5 2AED</td>
<td>4274</td>
<td>SUB CH,CH</td>
<td>; ZERO TO HIGH OF COUNT REG</td>
</tr>
<tr>
<td>F4B7 D0E3</td>
<td>4275</td>
<td>SAL BL,1</td>
<td>; MULTIPLY NUMBER OF LINES BY 4</td>
</tr>
<tr>
<td>F4B9 D0E3</td>
<td>4276</td>
<td>SAL BL,1</td>
<td></td>
</tr>
<tr>
<td>F4BB 7AED</td>
<td>4277</td>
<td>JZ R11</td>
<td>; IF ZERO, THEN BLANK ENTIRE FIELD</td>
</tr>
<tr>
<td>F4BD 8AC3</td>
<td>4278</td>
<td>MOV AL,BL</td>
<td>; GET NUMBER OF LINES IN AL</td>
</tr>
<tr>
<td>F4BF D450</td>
<td>4279</td>
<td>MOV AH,00</td>
<td>; 80 BYTES/ROW</td>
</tr>
<tr>
<td>F4C1 FEE4</td>
<td>4280</td>
<td>MUL AH</td>
<td>; DETERMINE OFFSET TO SOURCE</td>
</tr>
<tr>
<td>F4CE 0BF7</td>
<td>4281</td>
<td>MOV SI,DI</td>
<td>; SET UP SOURCE</td>
</tr>
<tr>
<td>F4C3 03F0</td>
<td>4282</td>
<td>ADD SI,AX</td>
<td>; ADD IN OFFSET TO IT</td>
</tr>
<tr>
<td>F4C7 0AE6</td>
<td>4283</td>
<td>MOV AH,0H</td>
<td>; NUMBER OF ROWS IN FIELD</td>
</tr>
<tr>
<td>F4C9 2AED</td>
<td>4284</td>
<td>SUB AH,BL</td>
<td>; DETERMINE NUMBER TO MOVE</td>
</tr>
<tr>
<td>4285</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F426</td>
<td></td>
<td></td>
<td>; ------ LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD FIELDS</td>
</tr>
<tr>
<td>4220</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F4CB</td>
<td>4286</td>
<td>RB:</td>
<td>; ROM_LOOP</td>
</tr>
<tr>
<td>F4CB E0000</td>
<td>4287</td>
<td>CALL R17</td>
<td>; MOVE ONE ROW</td>
</tr>
<tr>
<td>F4CE 01ED01F</td>
<td>4288</td>
<td>SUB SI,2000H-80</td>
<td>; MOVE TO NEXT ROW</td>
</tr>
<tr>
<td>F4D2 01BF01F</td>
<td>4289</td>
<td>SUB DI,2000H-80</td>
<td></td>
</tr>
<tr>
<td>F4DE FEEC</td>
<td>4290</td>
<td>DEC AH</td>
<td>; NUMBER OF ROWS TO MOVE</td>
</tr>
<tr>
<td>F4DF 75F1</td>
<td>4291</td>
<td>JNZ R8</td>
<td>; CONTINUE TILL ALL MOVED</td>
</tr>
<tr>
<td>4292</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F49A</td>
<td>4293</td>
<td>CALL R17</td>
<td>; CLEAR ENTRY</td>
</tr>
<tr>
<td>F49A 8AC7</td>
<td>4294</td>
<td>MOV AL,0H</td>
<td>; ATTRIBUTE TO FILL WITH</td>
</tr>
<tr>
<td>F4C0</td>
<td>4295</td>
<td>R10:</td>
<td></td>
</tr>
<tr>
<td>F4DC E0000</td>
<td>4296</td>
<td>CALL R18</td>
<td>; CLEAR THAT ROW</td>
</tr>
<tr>
<td>F4DE 01ED01F</td>
<td>4297</td>
<td>SUB SI,2000H-80</td>
<td></td>
</tr>
<tr>
<td>F4E3 FEEC</td>
<td>4298</td>
<td>DEC BL</td>
<td>; NUMBER OF LINES TO FILL</td>
</tr>
<tr>
<td>F4E5 75F1</td>
<td>4299</td>
<td>JNZ R10</td>
<td>; CLEAR_LOOP</td>
</tr>
<tr>
<td>F4E7 E000FC</td>
<td>4300</td>
<td>JMP VIDEO_RETURN</td>
<td>; EVERYTHING DONE</td>
</tr>
<tr>
<td>F4EA</td>
<td>4301</td>
<td>R11:</td>
<td>; BLANK_FIELD</td>
</tr>
<tr>
<td>F4EA 0ADE</td>
<td>4302</td>
<td>MOV BL,0H</td>
<td>; SET BLANK COUNT TO</td>
</tr>
<tr>
<td>4303</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F4EC EBEC</td>
<td>4304</td>
<td>JMP R9</td>
<td>; CLEAR THE FIELD</td>
</tr>
<tr>
<td>4305</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F4EC EBEC</td>
<td>4306</td>
<td>GRAPHICS_UP</td>
<td>; ENDP</td>
</tr>
<tr>
<td>4309</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F4E5</td>
<td></td>
<td></td>
<td>; SCROLL DOWN</td>
</tr>
<tr>
<td>4310</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F4E5</td>
<td></td>
<td></td>
<td>; THIS ROUTINE SCROLLS DOWN THE INFORMATION ON THE CRT</td>
</tr>
<tr>
<td>4312</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F4E5</td>
<td></td>
<td></td>
<td>; ENTRY</td>
</tr>
<tr>
<td>4313</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F4E5</td>
<td></td>
<td></td>
<td>; CH,CL = UPPER LEFT CORNER OF REGION TO SCROLL</td>
</tr>
<tr>
<td>4314</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F4E5</td>
<td></td>
<td></td>
<td>; DH,DL = LOWER RIGHT CORNER OF REGION TO SCROLL</td>
</tr>
<tr>
<td>4315</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F4E5</td>
<td></td>
<td></td>
<td>; BOTH OF THE ABOVE ARE IN CHARACTER POSITIONS</td>
</tr>
<tr>
<td>4316</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F4E5</td>
<td></td>
<td></td>
<td>; BL = FILL VALUE FOR BLANKED LINES</td>
</tr>
<tr>
<td>4317</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F4E5</td>
<td></td>
<td></td>
<td>; AL = # LINES TO SCROLL (AL=0 MEANS BLANK THE ENTIRE</td>
</tr>
<tr>
<td>4318</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F4E5</td>
<td></td>
<td></td>
<td>; FIELD)</td>
</tr>
<tr>
<td>4319</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F4E5</td>
<td></td>
<td></td>
<td>; DS = DATA SEGMENT</td>
</tr>
<tr>
<td>4320</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F4E5</td>
<td></td>
<td></td>
<td>; ES = REGEN SEGMENT</td>
</tr>
<tr>
<td>4321</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F4E5</td>
<td></td>
<td></td>
<td>; EXIT</td>
</tr>
<tr>
<td>4322</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F4E5</td>
<td></td>
<td></td>
<td>; NOTHING, THE SCREEN IS SCROLLED</td>
</tr>
<tr>
<td>4323</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F4E5</td>
<td></td>
<td></td>
<td>; -----------------------------------------------</td>
</tr>
<tr>
<td>4324</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F4E6</td>
<td></td>
<td></td>
<td>; GRAPHICS_DOWN PROC NEAR</td>
</tr>
<tr>
<td>4325</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F4E6 FD</td>
<td>4326</td>
<td>STD</td>
<td>; SET DIRECTION</td>
</tr>
<tr>
<td>F4E6 DAEB</td>
<td>4327</td>
<td>MOV BL,AL</td>
<td>; SAVE LINE COUNT IN BL</td>
</tr>
<tr>
<td>F4F1 0BC2</td>
<td>4328</td>
<td>MOV AX,DX</td>
<td>; GET LOWER RIGHT POSITION INTO AX REG</td>
</tr>
<tr>
<td>4329</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F432</td>
<td></td>
<td></td>
<td>; ---- USE CHARACTER SUBROUTINE FOR POSITIONING</td>
</tr>
<tr>
<td>4330</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F433</td>
<td></td>
<td></td>
<td>; ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE</td>
</tr>
<tr>
<td>4332</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F433 E0F02</td>
<td>4333</td>
<td>CALL GRAPH_PUSH</td>
<td></td>
</tr>
<tr>
<td>F434 08F8</td>
<td>4334</td>
<td>MOV DI,AX</td>
<td>; SAVE RESULT AS DESTINATION ADDRESS</td>
</tr>
<tr>
<td>F435</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F436</td>
<td></td>
<td></td>
<td>; ---- DETERMINE SIZE OF WINDOW</td>
</tr>
<tr>
<td>F437</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F437 2BD1</td>
<td>4338</td>
<td>SUB DX,CK</td>
<td></td>
</tr>
</tbody>
</table>

System BIOS  A-61
LOC OBJ  
LINE  
SOURCE

F4FA 81C20101 4359  ADD  DX,101H ; ADJUST VALUES
F4FE D0E6 4360  SAL  DH,1 ; MULTIPLY # ROWS BY 4
F500 D0E6 4361  SAL  DH,1 ; SINCE A VERY DOTS/CHAR
F500 4362  SAL  DH,1 ; AND EVEN/ODD ROWS
F500 4363  i----- DETERMINE CRT MODE
F502 803E990006 4364  CMP  CRT_MODE,6 ; TEST FOR MEDIUM RES
F507 7305 4365  JNC  R12 ; FIND_SOURCE_DOWN
F507 4366  ;
F509 D0E2 4367  i----- MEDIUM RES DOWN
F50B D1E7 4368  SAL  DL,1 ; # COLUMNS = 2, SINCE
F50D 4369  SAL  DL,1 ; 2 BYTES/CHAR (OFFSET OK)
F50D 436A  INC  DI ; OFFSET = # SINCE 2 BYTES/CHAR
F50D 436B  ;
F50E 436C  i----- DETERMINE THE SOURCE ADDRESS IN THE BUFFER
F50E 436D  R12: ; FIND_SOURCE_DOWN
F50E 06 436E 4359  PUSH  ES ; BOTH SEGMENTS TO REGEN
F50F 1F 436F 4360  POP  DS ;
F510 2AE0 4370  SUB  CH,CH ; ZERO TO HIGH OF COUNT REG
F512 81C70000 4371  ADD  DL,0 ; POINT TO LAST ROW OF PIXELS
F514 D0E3 4372  SAL  BL,1 ; MULTIPLY NUMBER OF LINES BY 4
F514 4373  SAL  BL,1 ;
F515 74D6 4374  JZ  R16 ; IF ZERO, THEN BLANK ENTIRE FIELD
F515 742E 4375  MOV  AL,DL ; GET NUMBER OF LINES IN AL
F515 7AC3 4376  MOV  AH,80 ; 80 BYTES/ROW
F516 4B50 4377  MOV  AH,80 ;
F518 4A68 4378  MUL  AL ; DETERMINE OFFSET TO SOURCE
F518 4A68 4379  MOV  SI,DI ; SET UP SOURCE
F518 4A68 437A  MOV  SI,AX ; SUBTRACT THE OFFSET
F518 4A68 437B  MOV  AH,DX ; NUMBER OF ROWS IN FIELD
F518 4A68 437C  SUB  AH,DL ; DETERMINE NUMBER TO MOVE
F518 4A68 437D  ;
F519 75F1 437E  i----- LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD FIELDS
F519 37F1 437F 4377  CALL  R17 ; MOV ONE ROW
F51B 80E0 4380  SUB  SI,2000H+80 ; MOVE TO NEXT ROW
F51F 31F50200 4381  MOV  SI,2000H+80 ;
F525 FECC 4382  DEC  AH ; NUMBER OF ROWS TO MOVE
F527 75F1 4383 4381  JNZ  R13 ; CONTINUE TILL ALL MOVED
F527 75F1 4384  ;
F529 4A 4385  i----- FILL IN THE VACATED LINE(S)
F529 35A9 4386 4380  MOV  AL,AH ; ATTRIBUTE TO FILL WITH
F52B 3BC7 4387 4381  CALL  R12 ; CLEAR_LOOP_DOWN
F52B 35A9 4388 4382  CALL  R12 ; CLEAR A ROW
F52B 81F50200 4389 4383  CALL  R12 ; POINT TO NEXT LINE
F52F 4FEB 438A 4384  DEC  BL ; NUMBER OF LINES TO FILL
F533 75F5 438B 4385  JNZ  R15 ; CLEAR_LOOP_DOWN
F535 FC 438C 4386  CLO  ; RESET THE DIRECTION FLAG
F537 E976FC 438D 4387  JMP  VIDEO_RETURN ; EVERYTHING DONE
F539 4A 438E 4388  ; BLANK_FIELD_DOWN
F540 4EAD 438F 4389  MOV  BL,0 ; SET BLANK COUNT TO EVERYTHING
F540 4EAD 4390 438A  IN  ; IN FIELD
F540 4EAD 4391 438B  JMP  R14 ; CLEAR THE FIELD
F540 4EAD 4392 438C  GRAPHICS_DOWN ENDP
F540 4EAD 4393  ;
F544 BEEB 4394 4400  i----- ROUTINE TO MOVE ONE ROW OF INFORMATION
F544 4401 4401 4402  PROC  NEAR
F545 EACA 4402 4403  MOV  CL,DL ; NUMBER OF BYTES IN THE ROW
F545 56 4403 4404  PUSH  SI ;
F545 57 4404 4405  PUSH  DI ; SAVE POINTERS
F545 F3 4405 4406  REP  MOVSB ; MOVE THE EVEN FIELD
F545 4406 4407  ;
F545 5F 4407 4408  POP  DI ;
F545 SF 4408 4409  POP  SI ;
F545 5E 4409 440A  ADD  SI,2000H ; POINT TO THE ODD FIELD
F556 61C60020 440A 440B  ADD  DI,2000H ;
F556 61C70020 440B 440C  ADD  SI,2000H ;
F556 61C70020 440C 440D  ADD  DI,2000H ;
F557 5E 440D 440E  PUSH  SI ; SAVE THE POINTERS
F557 57 440E 440F  PUSH  DI ; COUNT BACK
F557 57 440F 4410  MOV  CL,DL ; MOVE THE ODD FIELD
F562 F3 4410 4411  REP  MOVSB
A-62 System BIOS
LOC OBJ

SOURCE

F563 A4
F564 5F
F565 5E
F566 C3
F57 2.1
F57 2.1
F57 2.1
F57 2.1
F57 2.1
F57 2.1
F57 2.1
F57 2.1
GRAPHICS_WRITE PROC NEAR

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

GRAPHICS_WRITE PROC NEAR

;--------------------------------
; GRAPHICS_WRITE
;--------------------------------

4415 POP DI

; POINTERS BACK

4416 POP SI

4417 RET

; RETURN TO CALLER

4418 R17 ENDP

4419

;------- CLEAR A SINGLE ROW

4420

4421

F567 BACA

4422 MOV CL,DL

; NUMBER OF BYTES IN FIELD

4423 PUSH DI

; SAVE POINTER

4424 REP STOSB

; STORE THE NEW VALUE

4425

4426 MOV DI

; POINTER BACK

4427 ADD DI,2000H

; POINT TO ODD FIELD

4428 PUSH DI

4429 MOV CL,DL

4430 REP STOSB

; FILL THE ODD FIELD

4431 MOV DI

4432 RET

; RETURN TO CALLER

4433 R18 ENDP

4434

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

4435 ; GRAPHICS_WRITE

4436 ; THIS ROUTINE WRITES THE ASCII CHARACTER TO THE

4437 ; CURRENT POSITION ON THE SCREEN.

4438 ; ENTRY

4439 ; AL = CHARACTER TO WRITE

4440 ; DL = 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 ;

4450 ; GRAPHICS_READ

4451 ; THIS ROUTINE READS THE ASCII CHARACTER AT THE CURRENT

4452 ; CURSOR POSITION ON THE SCREEN BY MATCHING THE DOTS ON

4453 ; THE SCREEN TO THE CHARACTER GENERATOR CODE POINTS

4454 ; ENTRY

4455 ; AL = CHARACTER TO READ AT THAT POSITION (0 RETURNED IF

4456 ; AL IS ASSUMED AS THE BACKGROUND COLOR)

4457 ; EXIT

4458 ;

4459 ; FOR BOTH ROUTINES, THE IMAGES USED TO FORM CHAR

4460 ; ARE CONTAINED IN ROM FOR THE 1ST 128 CHAR. TO ACCESS CHAR

4461 ; IN THE SECOND HALF, THE USER MUST INITIALIZIZE THE VECTOR AT

4462 ; INTERRUPT 1AH (LOCATION 0007CH) TO POINT TO THE USER

4463 ; TABLE OF GRAPHIC IMAGES (8X8 BOXES).

4464 ; FAILURE TO DO SO WILL CAUSE IN STRANGE RESULTS

4465 ;--------------------------------

4466 ; ASSEMBLING

4467 ;--------------------------------

4468 ; GRAPHICS_WRITE PROC NEAR

4469 GRAPHICS_WRITE PROC NEAR

4470 ;

4471 ; DETERMINE POSITION IN REGEN BUFFER TO PUT CODE POINTS

4472 ;

4473 ; DETERMINE REGION TO GET CODE POINTS FROM

4474 ;

4475 ; IMAGE IS IN FIRST HALF, CONTAINED IN ROM

4476 ;

4477 ; IMAGE IS IN SECOND HALF, IN USER RAM

4478 ;

4479 ;

4480 ;

4481 ;

4482 ;

4483 ;

4484 ;

4485 ;

4486 ;

4487 ;

4488 ;

System BIOS  A-63
A-64  System BIOS

```
HIGH RESOLUTION HODE

F5A0 2C00  SUB AL,00H | EXTEND_CHAR
F5A1 1E    PUSH DS | ZERO ORIGIN FOR SECOND HALF
F5A2 80F6  SUB SI,DI | SAVE DATA POINTER
F5A3 6EDE  MOV DS,SI | ESTABLISH VECTOR ADDRESSING
F5A5 C367C00 LDS SI,EXT_PTR | GET THE OFFSET OF THE TABLE
F5A6 6CDA  MOV DX,DS | GET THE SEGMENT OF THE TABLE
F5A7 3F    ADD DS:DI | RECOVER DATA SEGMENT
F5A8 52    PUSH DX | SAVE TABLE SEGMENT ON STACK
F5A9 00    MOV AX,0
F5AA 01    MOV AL,0
F5AB 02    MOV BL,0
F5AC 03    MOV CH,0
F5AD 04    MOV DL,0
F5AE 05    MOV SI,0
F5AF 06    MOV DI,0
F5B0 E8    CALL S21 | DOUBLE UP ALL THE BITS
F5B1 80    MOV AL,0 | DOUBLE UP ALL THE BITS
F5B2 81    MOV AX,0 | CONVERT THEM TO FOREGROUND
F5B3 80    MOV AH,0 | COLOR (0 BACK)
F5B4 FFC00  TEST DL,00H | IS THIS XOR FUNCTION
F5B5 7A07  JNZ SI0 | NO, STORE IT IN AS IT IS
F5B6 263215  XOR AL,ES:[DI] | DO FUNCTION WITH HALF
F5B7 263295  XOR AL,ES:[DI]+1 | AND WITH OTHER HALF
F5B8 8B82  MOV BL,ES:[DI]+1 | STORE FIRST BYTE
F5B9 2600501 MOV SI,ES:[DI]+1,AL | STORE SECOND BYTE
F5BA AC    LODS SI | GET CODE POINT
F5BB E8C500 CALL S21
```
AND
F5FE 23C3 4566 AND AX,AX  ; CONVERT TO COLOR
F600 44C200 4567 TEST DL,00H  ; AGAIN, IS THIS XOR FUNCTION
F603 740A 4568 JZ SI1  ; NO, JUST STORE THE VALUES
F605 262A50020 4569 XOR AH,ES:[DI+2000H]  ; FUNCTION WITH FIRST HALF
F60A 262B50110 4570 XOR AL,ES:[DI+2001H]  ; AND WITH SECOND HALF
F60F 4571 SI1:
F60F 26080020 4572 MOV ES:[DI+2000H],AL  ; STORE IN SECOND PORTION OF BUFFER
F614 260805010 4573 MOV ES:[DI+2005H],AL  ; STORE IN SECOND PORTION OF BUFFER
F619 03C750 4574 ADD DI,00  ; POINT TO NEXT LOCATION
F61C FECE 4575 DEC DH  ; KEEP GOING
F61E 75C1 4576 JNZ SP  ; RECOVER CODE POINTER
F620 5E 4577 POP DI  ; RECOVER REGEN POINTER
F621 5F 4578 POP DI  ; RECOVERY TO NEXT CHAR POSITION
F622 47 4579 INC DI  ; POINT TO NEXT CHAR POSITION
F623 47 4580 INC DI  ; POINT TO NEXT CHAR POSITION
F624 E807 4581 LOOP 5B  ; MORE TO WRITE
F626 E99CFB 4582 JMP VIDEO_RETURN

GRAPHICS_READ PROC NEAR
F629 E80600 4583 CALL S26  ; CONVERTED TO OFFSET IN REGN
F62C 08F0 4584 MOV SI,AX  ; SAVE IN SI
F62E 03EC08 4585 SUB SP,8  ; ALLOCATE SPACE TO SAVE THE
F62F 4586 80EC 4587 MOV BP,SP  ; READ CODE POINT
F631 72A 4588 JMP SI  ; POINTER TO SAVE AREA

GRAPHICS_WRITE ENDP

; GRAPHICS_READ PROC NEAR
F632 003E400006 4589 CMP CR1_MODE,6
F633 06 4590 PUSH ES
F639 1F 4591 POP DS  ; POINT TO REGEN SEGMENT
F63A 72A 4592 JC SI3  ; MEDIUM RESOLUTION
F63C B604 4593 MOV DH,4  ; NUMBER OF PASSES
F63E 512 4594 MOV SI:[DI+2000H]  ; GET FIRST BYTE
F640 806000 4595 MOV [BP],AL  ; SAVE IN STORAGE AREA
F643 45 4596 INC BP  ; NEXT LOCATION
F644 8A60020 4597 MOV AL,[SI+2000H]  ; GET LOWER REGION BYTE
F645 8A6600 4598 MOV [BP],AL  ; ADJUST AND STORE
F646 45 4599 INC BP  ; NEXT LOCATION
F647 80C50 4600 ADD SI,60  ; POINTER INTO REGEN
F64F FECE 4601 DEC DH  ; LOOP CONTROL
F651 75EB 4602 JMP SI2  ; DO IT SOME MORE
F653 E81790 4603 JMP SI5  ; GO MATCH THE SAVED CODE POINTS
F655 B604 4604 MOV DH,4  ; MEDIUM RESOLUTION READ
F65F 513 4605 MOV SI:[DI+2000H]  ; GET FIRST BYTE
F665 806000 4606 MOV [BP],AL  ; SAVE IN STORAGE AREA
F668 45 4607 INC BP  ; NEXT LOCATION
F669 8A60020 4608 MOV AL,[SI+2000H]  ; GET LOWER REGION BYTE
F66A 8A6600 4609 MOV [BP],AL  ; ADJUST AND STORE
F66B 45 4610 INC BP  ; NEXT LOCATION
F66C 83C50 4611 ADD SI,60  ; POINTER INTO REGEN
F66F FECE 4612 DEC DH  ; LOOP CONTROL
F671 75EB 4613 JMP SI2  ; DO IT SOME MORE
F673 EB1790 4614 JMP SI5  ; GO MATCH THE SAVED CODE POINTS
F675 B604 4615 MOV DH,4  ; MEDIUM RESOLUTION READ
F679 513 4616 MOV SI:[DI+2000H]  ; GET FIRST BYTE
F67F 806000 4617 MOV [BP],AL  ; SAVE IN STORAGE AREA
F682 45 4618 INC BP  ; NEXT LOCATION
F683 8A60020 4619 MOV AL,[SI+2000H]  ; GET LOWER REGION BYTE
F684 8A6600 461A MOV [BP],AL  ; ADJUST AND STORE
F685 45 461B INC BP  ; NEXT LOCATION
F686 80C50 461C ADD SI,60  ; POINTER INTO REGEN
F68F FECE 461D DEC DH  ; LOOP CONTROL
F691 75EB 461E JMP SI2  ; DO IT SOME MORE
F693 EB1790 461F JMP SI5  ; GO MATCH THE SAVED CODE POINTS
F695 B604 4620 MOV DH,4  ; MEDIUM RESOLUTION READ
F699 513 4621 MOV SI:[DI+2000H]  ; GET FIRST BYTE
F69F 806000 4622 MOV [BP],AL  ; SAVE IN STORAGE AREA
F6A2 45 4623 INC BP  ; NEXT LOCATION
F6A3 8A60020 4624 MOV AL,[SI+2000H]  ; GET LOWER REGION BYTE
F6A4 8A6600 4625 MOV [BP],AL  ; ADJUST AND STORE
F6A5 45 4626 INC BP  ; NEXT LOCATION
F6A6 80C50 4627 ADD SI,60  ; POINTER INTO REGEN
F6A7 81EE00 4628 SUB SI,[BP]+80  ; ADJUST POINTER BACK INTO UPPER
F6AA FECE 4629 DEC DH  ; LOOP CONTROL
F6AB 75EB 4630 JMP SI2  ; DO IT SOME MORE
F6AC EB1790 4631 JMP SI5  ; GO MATCH THE SAVED CODE POINTS

; SAVE AREA HAS CHARACTER IN IT, MATCH IT
F6A6 75EB 4632 MOV SI:[DI+OFFSET CRT_CHAR_GEN]  ; ESTABLISH ADDRESSING
F6A6 BFAEAFA90 4633 MOV DI,OFFSET CRT_CHAR_GEN  ; ESTABLISH ADDRESSING
F6B0 0E 4634 MOV DS,SI
F6B1 FF 4635 PUSH CS
F6B4 57 4636 MOV DS,SI
F6B7 FF 4637 MOV DL,DX
F6B8 3ED06 4638 MOV DL,AX
F6B9 8F05 4639 MOV AL,0
F6BA 73F 4640 MOV SI,8
F6BB E807 4641 JMP SI
F6BC E80600 4642 MOV DS,SI
F6BB B604 4643 MOV DH,4
F6C3 513 4644 MOV SI:[DI+2000H]  ; GET FIRST BYTE
F6C3 806000 4645 MOV [BP],AL  ; SAVE IN STORAGE AREA
F6C6 45 4646 INC BP  ; NEXT LOCATION
F6C7 8A60020 4647 MOV AL,[SI+2000H]  ; GET LOWER REGION BYTE
F6C8 8A6600 4648 MOV [BP],AL  ; ADJUST AND STORE
F6C9 45 4649 INC BP  ; NEXT LOCATION
F6CB 80C50 464A ADD SI,60  ; POINTER INTO REGEN
F6CC 81EE00 464B SUB SI,[BP]+80  ; ADJUST POINTER BACK INTO UPPER
F6CF FECE 464C DEC DH  ; LOOP CONTROL
F6D0 75EB 464D JMP SI2  ; DO IT SOME MORE
F6D3 EB1790 464E JMP SI5  ; GO MATCH THE SAVED CODE POINTS

; FIND CHAR
F6D6 BFAEAFA90 464F MOV DI,OFFSET CRT_CHAR_GEN  ; ESTABLISH ADDRESSING
F6D7 0E 4650 MOV DS,SI
F6D8 FF 4651 PUSH CS
F6DA 57 4652 MOV DS,SI
F6DB FF 4653 MOV DL,DX
F6DC 3ED06 4654 MOV DL,AX
F6DD 8F05 4655 MOV AL,0
F6DE 73F 4656 MOV SI,8
F6DF E807 4657 JMP SI
F6E0 E80600 4658 MOV DS,SI
F6EB B604 4659 MOV DH,4
F6EC 513 465A MOV SI:[DI+2000H]  ; GET FIRST BYTE
F6EE 806000 465B MOV [BP],AL  ; SAVE IN STORAGE AREA
F6F1 45 465C INC BP  ; NEXT LOCATION
F6F2 8A60020 465D MOV AL,[SI+2000H]  ; GET LOWER REGION BYTE
F6F3 8A6600 465E MOV [BP],AL  ; ADJUST AND STORE
F6F4 45 465F INC BP  ; NEXT LOCATION
F6F5 80C50 4660 ADD SI,60  ; POINTER INTO REGEN
F6F6 81EE00 4661 SUB SI,[BP]+80  ; ADJUST POINTER BACK INTO UPPER
F6F9 FECE 4662 DEC DH  ; LOOP CONTROL
F6FA 75EB 4663 JMP SI2  ; DO IT SOME MORE
F6FB EB1790 4664 JMP SI5  ; GO MATCH THE SAVED CODE POINTS

; FIND CHAR

Appendix A

System BIOS A-65
A-66  System BIOS
F6CD 2309 4719 AND BX,CX ; USE MASK TO EXTRACT A BIT
F6CF 08D3 4720 OR DX,BX ; PUT INTO RESULT REGISTER
F6D1 D1E0 4721 SHL AX.1 ; SHIFT BASE AND MASK BY 1
F6D3 D1E1 4722 SHL CX.1 ; BASE TO TEMP
F6D5 80D0 4723 MOV BX,AX ; BASE TO TEMP
F6D7 2309 4724 AND BX,CX ; EXTRACT THE SAME BIT
F6D9 08D3 4725 OR DX,BX ; PUT INTO RESULT
F6DB D1E1 4726 SHL CX.1 ; SHIFT ONLY MASK NOW.
F6D7 730C 4727 MOV BX,AX ; MOVING TO NEXT BASE
F6D9 08C2 4728 POP AX ; USE MASK BIT COMING OUT TO TERMINATE
F6EA 5B 4729 MOV AX,BX ; RESULT TO FARM REGISTER
F6D3 59 4730 POP CX ; RECOVER REGISTERS
F6D3 5A 4731 POP DX
F6D3 C3 4732 RET ; ALL DONE

F6D9 S21 ENDP 4733

F6C0 B4A4 4734 ;---------------------------------------------
F6C5 BA4401 4735 | MED_READ_BYTE :
F6D6 B9080C 4736 | THIS ROUTINE WILL TAKE 2 BYTES FROM THE REGEN :
F6D5 B200 4737 | BUFFER, COMPARE AGAINST THE CURRENT FOREGROUND :
F6D6 85C1 4738 | COLOR, AND PLACE THE CORRESPONDING ON/OFF BIT :
F6D7 F0 4739 | PATTERN INTO THE CURRENT POSITION IN THE SAVE :
F6D9 7401 4740 | AREA :
F6D7 F9 4741 | ENTRY :
F6D8 8D02 4742 ; SI.OS = POINTER TO AREA OF INTEREST :
F6D6 08D2 4743 ; BX = EXPANDED FOREGROUND COLOR :
F6D8 8D00 4744 ; BP = POINTER TO SAVE AREA :
F6D6 8D0C 4745 ; BP IS INCREMENT AFTER SAVE :
F6D7 0200 4746 ;---------------------------------------------
F6E5 S23 PROC NEAR 4747
F6E6 B4A4 4748 MOV AX,[SI] ; GET FIRST BYTE
F6E7 BA4401 4749 MOV AL,[SI]+1 ; GET SECOND BYTE
F6E8 B9080C 4750 MOV CX,[CO08H] ; 2 BIT MASK TO TEST THE ENTRIES
F6E9 B200 4751 MOV DI.0 ; RESULT REGISTER
F6E7 85C1 4752 S24: TEST AX,CX ; IS THIS SECTION BACKGROUND?
F6F1 F0 4753 CLR ; CLEAR CARRY IN HOPES THAT IT IS
F6F2 7401 4754 JZ S25 ; IF ZERO, IT IS BACKGROUND
F6F4 F9 4755 STC ; WASN'T, SO SET CARRY
F6F5 08D2 4756 S25: RCL DI.1 ; MOVE THAT BIT INTO THE RESULT
F6F7 D1E9 4757 SHR CX.1 ; MOVE THE MASK TO THE RIGHT BY 2 BITS
F6F9 D1E9 4758 SHR CX.1 ; IF AGAIN IF MASK DIDN'T FALL OUT
F6FD 8D00 4759 MOV [BP].DL ; STORE RESULT IN SAVE AREA
F6FD 8D04 4760 MOV [BP].DH ; ADJUST POINTER
F700 C5 4761 JS ; ALL DONE
F701 C3 4762 RET

F702 S23 ENDP 4763

F702 A15000 4764 ;---------------------------------------------
F705 53 4765 | MED_READ_BYTE :
F706 80D8 4766 | THIS ROUTINE TAKES THE CURSOR POSITION :
F707 80D0 4767 | CONTENTS IN THE MEMORY LOCATION, AND :
F708 B4A4 4768 | CONVERTS IT INTO AN OFFSET INTO THE :
F709 F624400 4769 | REGEN BUFFER, ASSUMING ONE BYTE/CHAR. :
F70E D1E0 4770 | FOR MEDIUM RESOLUTION GRAPHICS, :
F710 D1E0 4771 ; THE NUMBER MUST BE DOUBLED. :
F712 2AFF 4772 ; ENTRY :
F714 03C3 4773 ; NO REGISTERS. MEMORY LOCATION :
F716 5B 4774 ; CURSOR_POSH IS USED :
F717 C3 4775 ; AX CONTAINS OFFSET INTO REGEN BUFFER :
F718 S26 ENDP 4776 ;---------------------------------------------

System BIOS A-67
LOCATION: DETERMINE VALUE TO SCROLL REQUIRED

A-68 System BIOS
Appendix A

System BIOS

A-69
NOLO TEST FOR TRIGGER HAS DETERMINE LOC OBJ
F7AA A802
F7AC 7503
F7AE 90100

F7B1
F7B1 B410
F7B3 B166300
F7B7 8A4
F7B9 EE
F7BA 42
F7BD EC
F7BE 4A
F7BF EC4
F7C1 8AC4
F7C3 EE
F7C6 8AE5

F7C0 8A1E4900
F7C2 2AFF
F7CE 2BAF94F7
F7D3 2Bc3
F7D9 B1E400
F7DD 7902
F7DF 2Bc0

F7E1
F7E1 B103
F7E3 03E490004
F7E5 722A
F7E6 03E490007
F7E7 7423

F7F1 B2A
F7F3 F6F2

F7F5 8AE8
F7F7 02ED
F7F9 8ADC
F7FA 2AFF
F7FD 03E490006
F7F0 7504
F7F4 B104
F7F6 D0E4
F7F8 38E3

F7FA AAD4
F7FC AAF0
F7FE D0E1
F810 D0EE

A-70 System BIOS
; ALPHON - 46K BASE

F014 5826 jmp short vs ; LIGHT_PEN_RETURN_SET

F013 5827 ;-------- ALPHA MODE ON LIGHT PEN

F014 5828 vs: div byte ptr crt_cols ; DETERMINE ROW,COLUMN VALUE

F017 5829 mov dl,ah ; ROWS TO DL

F01A 582A mov dl,al ; COLS TO DL

F01C 582B sal al,cl ; MULTIPLY ROWS X COLS

F01E 582C mov ch,al ; GET RASTER VALUE TO RETURN REG

F020 582D mov bl,ah ; COLUMN VALUE

F022 5831 xor bh,dl ; TO BX

F024 5832 sal bx,cl ;

F026 5833 vs: mov ah,1 ; LIGHT_PEN_RETURN_SET

F028 5834 vs: mov ah,1 ; INDIQUE EVERYTHING SET

F02A 5836 vs: push dx ; SAVE RETURN VALUE (IN CASE)

F02C 5837 mov dx,addr_6045 ; GET BASE ADDRESS

F02E 5838 add dx,7 ; POINT TO RESET PARM

F030 5839 out dx,al ; ADDRESS, NOT DATA, IS IMPORTANT

F031 5840 pop dx ; RECOVER VALUE

F033 5841 vs: push di

F035 5842 vs: push si

F037 5844 vs: push ds ; DISCARD SAVED BX,CX,DX

F03F 5845 vs: pop ds

F041 5846 vs: pop ds

F043 5847 vs: pop ds

F046 5848 vs: pop es

F049 5849 iret

F050 584A read_len: ; READ Len

F051 5851 --- INT 12 ---

F053 5852 ; MEMORY_SIZE_DET

F054 5853 ; THIS ROUTINE DETERMINES THE AMOUNT OF MEMORY IN THE SYSTEM

F055 5854 ; AS REPRESENTED BY THE SWITCHES ON THE PLANAR. NOTE THAT THE

F056 5855 ; SYSTEM MAY NOT BE ABLE TO USE 3/0 MEMORY UNLESS THERE IS A FULL

F057 5856 ; COMPLEMENT OF 64K BYTES ON THE PLANAR.

F058 5857 ;

F059 5858 ;

F05A 5859 ;

F05B 585A ;

F05C 585B ;

F05D 585C ;

F05E 585D ;

F05F 585E ;

F060 585F ;

F061 5860 ;

F062 5861 ;

F063 5862 ;

F064 5863 ;

F065 5864 ;

F066 5865 ;

F067 5866 ;

F068 5867 ;

F069 5868 ;

F06A 5869 ;

F06B 586A ;

F06C 586B ;

F06D 586C ;

F06E 586D ;

F06F 586E ;

F070 586F ;

F071 5870 ;

F072 5871 --- INT 11 ---

F073 5872 ; EQUIPMENT DETECTION

F074 5873 ; THIS ROUTINE ATTEMPTS TO DETERMINE WHAT OPTIONAL

F075 5874 ; DEVICES ARE ATTACHED TO THE SYSTEM.

F076 5875 ;

F077 5876 ;

F078 5877 ;

F079 5878 ;

F07A 5879 ;

F07B 587A ;

F07C 587B ;

F07D 587C ;

F07E 587D ;

F07F 587E ;

F080 587F ;

F081 5880 --- INT 11 ---

F082 5881 --- INT 11 ---

F083 5882 --- INT 11 ---

F084 5883 --- INT 11 ---

F085 5884 --- INT 11 ---

F086 5885 --- INT 11 ---
(AX) IS SET, BIT SIGNIFICANT, TO INDICATE ATTACHED I/O :

BIT IS NOT USED

BIT I2 IS GAME I/O ATTACHED

BIT 11,10,9 = NUMBER OF RS232 CARDS ATTACHED

BIT 0 UNUSED

BIT 7,6 = NUMBER OF DISKETTE DRIVES

00=1, 01=2, 10=3, 11=4 ONLY IF BIT 0 = 1

BIT 5,4 = INITIAL VIDEO MODE

00 - UNUSED

01 - 40X25 BM USING COLOR CARD

10 - 80X25 BM USING COLOR CARD

11 - 40X25 BM USING BM CARD

BOX IS NOT \n
BOX IS USING A \n
40X25 BW USING COLOR CARD

SEE IF

BOX IS

INT 15

THIS BIT INDICATES THAT

A-72 System BIOS
LOC OBJ  LINE  SOURCE

F866 E640  5174  OUT OAH,AL
F868 E641  5175  IN AL,PORT_B
F86A 0C30  5176  OR AL,00110000B ; TOGGLE PARITY CHECK ENABLES
F86C E641  5177  OUT PORT_B,AL
F86E 24CF  5178  AND AL,11001111B
F869 E641  5179  OUT PORT_B,AL
F892 081E300  5180  MOV BX,MEMORY_SIZE ; GET MEMORY SIZE WORD
F896 FC  5181  CLD ; SET DIR FLAG TO INCREMENT
F897 2B02  5182  SUB DX,DX ; POINT DX AT START OF MEM
F899 5183  NMI_LOOP:
F899 0EDA  5184  MOV DS,DX
F89B 0EC2  5185  MOV ES,DX
F89D 090040  5186  MOV CX,4000H ; SET FOR 16KB SCAN
F8A0 2B06  5187  SUB SI,SI ; SET SI TO BE RELATIVE TO
F8A2 F3  5188  XOR AX,AX ; START OF SS
F8A3 AC  5189  IN AL,PORT_C ; IF PARITY CHECK HAPPENED
F8A6 0C60  5190  AND AL,10000000B ; IF PARITY CHECK HAPPENED
F8A8 7612  5191  JNZ PTR_MMI ; GO PRINT ADDRESS IF IT DID
F8AA 01C2004  5192  ADD DX,4000H ; POINT TO NEXT 16KB BLOCK
F8AE 65B0  5193  SUB BX,160
F8B1 7566  5194  JNZ NMI_LOOP
F8B3 085F990  5195  MOV SI,(OFFSET D3A) ; PRINT ROW OF ?? ?? IF PARITY
F8B7 E801  5196  CALL PTR_MSG ; CHECK COULD NOT BE RE-CREATED
F8BA FA  5197  CLI
F8BB F4  5198  HLT ; HALT SYSTEM
F8BC 5200  PR!_MII:
F8BC 0CDA  5201  MOV DX,DS
F8BE E81097  5202  CALL PTR_SEG ; PRINT SEGMENT VALUE
F8C1 BA1302  5203  MOV DX,0213H
F8C4 E000  5204  MOV AL,00 ; DISABLE EXPANSION BOX
F8C6 EE  5205  OUT DX,AL ; (CAN'T WRITE TO MEM)
F8C7 D028  5206  MOV AL,'\'
F8C9 E00000  5207  CALL PTR_HEX
F8CB B55A5  5208  MOV AX,045AH
F8CF 08C0  5209  MOV CX,AX
F8D1 2900  5210  SUB BX,AX
F8D3 8907  5211  MOV EBX,A;
F8D5 90  5212  NOP
F8D6 90  5213  NOP
F8D7 8B07  5214  MOV AX,[BX] ; HAD THE ERROR
F8D9 38C1  5215  CMP AX,CX ; IS IT THERE?
F8DB 7607  5216  JE SYS_BOX_ERR ; YES- MUST BE SYS UNIT
F8D7 045  5217  MOV AL,'E' ; NO-MUST BE IN EXP. BOX
F8DE E88A00  5218  CALL PTR_HEX
F8E2 EB05  5219  JMP SHORT HLT_MII
F8E4 5220  SYS_BOX_ERR:
F8E4 D053  5221  MOV AL,','
F8E6 E8300  5222  CALL PTR_HEX
F8E9 5223  HLT_MII:
F8EA D029  5224  MOV AL,')'
F8EB E8A00  5225  CALL PTR_HEX
F8EE FA  5226  CLI ; HALT SYSTEM
F8F0 5228  D16:
F8F0 5A  5229  POP AX ; RESTORE ORIG CONTENTS OF AX
F8F1 CF  5230  IRET
F8F3 5231  HINT_INT ENDP
F8F4 5232
F8F5 5233  ; ---------------------
F8F6 5234  ; ROS CHECKSUM SUBROUTINE
F8F7 5235  ; ---------------------
F8F8 E90020  5236  ROS_CHECKSUM PROC NEAR ; NEXT ROS_MODULE
F8F9 5237  MOV CX,0192 ; NUMBER OF BYTES TO ADD
F8FA 3600  5238  ROS_CHECKSUM_CNT: ; ENTRY FOR OPTIONAL ROS TEST
F8FB 32C0  5239  XOR AL,AL
F8FC 240  5240  C26:
F8FF 0207  5241  ADD AL,DOS:IX1
F8F0 43  5242  INC BX ; POINT TO NEXT BYTE
F8F2 2F8B  5243  LOOP C26 ; ADD ALL BYTES IN ROS MODULE
F8F3 0AC0  5244  OR AL,AL ; SUM = 0?
F8F4 C3  5245  RET
F8F5 5246  ROS_CHECKSUM ENDP
F8F6 5247
F8F7 5248  ; MESSAGE AREA FOR POST
F8F8 5249  ; ---------------------
LOC OBJ     LINE       SOURCE
FF00 313031  5250  E0    DB '101',13,10 ; SYSTEM BOARD ERROR
FF02 0D     5251  E1    DB '201',13,10 ; MEMORY ERROR
FF03 0A     5252  F3A   DB 'ROM',13,10 ; ROM CHECKSUM ERROR
FF04 29823031 5253  F3C   DB '1001',13,10 ; EXPANSION IO BOX ERROR
FF05 0D     5254  D1    DB 'PARITY CHECK 2',13,10
FF06 0A     5255  D2    DB 'PARITY CHECK 1',13,10
FF07 0A     5256  DZA   DB '?????',13,10
FF08 0A     5257
5258
5259
5260
5261
5262
5263
5264
5265
5266
5267
5268
5269
5270
5271
5272
5273
5274
5275
5276
5277
5278
5279
5280
5281
5282
5283
5284
5285
5286
5287
5288
5289
5290
5291
5292
5293
5294
5295
5296
5297
5298
5299
5300
5301
5302
5303
5304
5305
5306
5307
5308
5309

A-74  System BIOS
F908  XPC_BYTE     PROC NEAR
F908 50  ; SAVE FOR LOW NIBBLE DISPLAY
F90C  D104  ; SHF COUNT
F90E  D1E8  ; NIBBLE SNAP
F999  E80300  ; DO THE HIGH NIBBLE DISPLAY
F999  58  ; RECOVER THE NIBBLE
F99A  240F  ; FALL INTO LOW NIBBLE CONVERSION
F99C  XLOT_PR  ; CONVERT 00-FF TO ASCII CHARACTER
F99C  D040  ; ADD AI.040H
F99C  27  ; ADJUST FOR NUMERIC AND ALPHA RANGE
F996  D040  ; ADD AI.040H
F996  27  ; ADJUST HIGH NIBBLE TO ASCII RANGE
F99C  PRT_HEX  ; DISPLAY CHARACTER IN AL
F99D  D070  ; MDV 89H
F9A0  C010  ; MDV 0H
F9A2  C3  ; CALL VIDEO_ID
F9A3  F4  ; LABEL WORD
F9A3  DC03  ; PRINT SOURCE TABLE
F9A5  7003  ; DW 36CH
F9A7  7002  ; DW 278H
F9A9  F4E  ; ENTRY REQUIREMENTS:
F999  E80300  ; THIS SUBROUTINE WILL PRINT A MESSAGE ON THE DISPLAY :
F9AC  E01C00  ; SI = OFFSET(ADDRESS) OF MESSAGE BUFFER :
F9AE  1E  ; CX = MESSAGE BYTE COUNT :
F9AF  E8A700  ; MAXIMUM MESSAGE LENGTH IS 36 CHARACTERS :
F9B2  A01000  ; F90B FA  ; SET BP non-zero to flag ERR
F9B4  88EE  ; MFG_HAL SYSTEM:
F9B4  E81C00  ; YES - HALT SYSTEM
F9BC  E663  ; CMD_PORT_AL
F9BE  E005  ; AI.01000010B
F9B9  E661  ; OUT PORT_B_AL
F9C2  A01500  ; MDV AI.MFG_ERR_FLAG
F9C5  E660  ; OUT PORT_A_AL
F9C7  F4  ; SET INTO 8255 REG
F9C8  GIZ2:  ; HALT SYS
F9CA  1F  ; POP DS
F9C9  C3  ; WRITE_MSG:
F9C9  E9  ; RET
F9CA  P_MSG  ; INITIAL RELIABILITY TEST -- SUBROUTINES :
F9CA  2E0A04  ; ASSUME CS:CODE,DS:DATA
F9CD  46  ; PUT CHAR IN AL
F9CE  50  ; SAVE PRINT CHAR
F9CF  8BCAFF  ; CALL PRT_HEX
F9D0  50  ; RECOVER PRINT CHAR
F9D3  3C0A  ; WAS IT LINE FEED?
F9D5  7F3F  ; NO, KEEP PRINTING STRING
F9D7  C3  ; RET
SUBROUTINES FOR POWER ON DIAGNOSTICS

CONTINUE BEEPING SPEAKER ROUTINE TO SOUND BEEPER

A-76 System BIOS
<table>
<thead>
<tr>
<th>LOC OBJ</th>
<th>LINE</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>FE26</td>
<td>0000C606</td>
<td>FE6C00</td>
</tr>
<tr>
<td>FE2E</td>
<td>0000C6C6</td>
<td>C6C600</td>
</tr>
<tr>
<td>FE36</td>
<td>0000CCDC</td>
<td>7CCD00</td>
</tr>
<tr>
<td>FE3E</td>
<td>0000CFB0</td>
<td>3064FC0</td>
</tr>
<tr>
<td>FE46</td>
<td>1C030E30</td>
<td>0301C00</td>
</tr>
<tr>
<td>FE4E</td>
<td>181B0D01</td>
<td>80100000</td>
</tr>
<tr>
<td>FE56</td>
<td>03030C30</td>
<td>03000000</td>
</tr>
<tr>
<td>FE6E</td>
<td>06FC0000</td>
<td>C6000000</td>
</tr>
<tr>
<td>FF66</td>
<td>0010360C</td>
<td>060EFE00</td>
</tr>
</tbody>
</table>

| FEE6 | FE6E | FB | FE6F | JE | FE70 | 006A5B | FE73 | 0A6E | FE75 | 7467 | FE77 | 7ECC | FE79 | 7466 | FE7B | FE7B | FB | FE7C | 1F | FE7D | CF | FE7E | FE7F | FE7F | FA | FE7F | A0700D | FE82 | C66700000 | FE85 | 80EE60D | FE88 | B066C00 | FE8D | E8E6A | FE91 | FE91 | FA | FE92 | 8911660D | FE96 | 90E06E00 | FE9A | C66700000 | FE9F | E8E6B | FE9 | FE9 | 16F0D0D | FE9 | 16F0D0D | FE9 | 16F0D0D | FE9 | 16F0D0D | FE9 | 16F0D0D |

### Appendix A

<table>
<thead>
<tr>
<th>ORG</th>
<th>$0FE6E</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORG</td>
<td>$0FE6F</td>
</tr>
<tr>
<td>ORG</td>
<td>$0FE70</td>
</tr>
<tr>
<td>ORG</td>
<td>$0FE73</td>
</tr>
<tr>
<td>ORG</td>
<td>$0FE75</td>
</tr>
<tr>
<td>ORG</td>
<td>$0FE77</td>
</tr>
<tr>
<td>ORG</td>
<td>$0FE79</td>
</tr>
<tr>
<td>ORG</td>
<td>$0FE7B</td>
</tr>
<tr>
<td>ORG</td>
<td>$0FE7C</td>
</tr>
<tr>
<td>ORG</td>
<td>$0FE7D</td>
</tr>
<tr>
<td>ORG</td>
<td>$0FE7E</td>
</tr>
<tr>
<td>ORG</td>
<td>$0FE7F</td>
</tr>
<tr>
<td>ORG</td>
<td>$0FE91</td>
</tr>
<tr>
<td>ORG</td>
<td>$0FE92</td>
</tr>
<tr>
<td>ORG</td>
<td>$0FE96</td>
</tr>
<tr>
<td>ORG</td>
<td>$0FE9A</td>
</tr>
</tbody>
</table>

### TIME_OF_DAY PROC FAR

- STI: INTERRUPTS BACK ON
- PUSH DS: SAVE SEGMENT
- CALL DOS: DOS CALL
- OR AH, AH: AH = 0
- JZ T2: READ_TIME
- DEC AH: AH - 1
- JZ T3: SET_TIME
- JMP T1: TO_RETURN
- POP DS: RECOVER SEGMENT
- IRET: RETURN TO CALLER
- CLI: NO TIMER INTERRUPTS WHILE READING
- MOV AL, TIMER_OPL: GET OVERFLOW, AND RESET THE FLAG
- MOV CX, TIMER_HIGH: SET THE TIME
- MOV DX, TIMER_LOW: OVERFLOW
- JMP T1: TO_RETURN
- CLI: NO INTERRUPTS WHILE WRITING
- MOV TIMER_LOW, DX: OVERFLOW
- MOV TIMER_HIGH, CX: SET THE TIME
- JMP T1: TO_RETURN

### TIME_OF_DAY ENDP
FEAS FB 5697 STI 1 INTERRUPTS BACK ON
FEAL IE 5690 PUSH DS
FEAQ 50 5699 PUSH AX
FEAQ 52 5700 PUSH DX
FEAQ 66A0FB 5701 CALL DOS
FEAC FF06CC00 5702 INC TIMER_LOW 1 INCREMENT TIME
FEAD 7504 5703 JNZ T4 1 TEST DAY
FEAE FF06CC00 5704 INC TIMER_HIGH 1 INCREMENT HIGH WORD OF TIME
FEAE 5705 T4: CMP TIMER_HIGH,61BH 1 TEST FOR COUNT EQUALING 24 HOURS
FEAF 7515 5707 JNZ T5 1 DISKETTE_CTL
FEAG 0136CC0800 5708 CMP TIMER_LOW,080H
FEAC 750D 5709 JNZ T5 1 DISKETTE_CTL
SE7 5710
FEAS 2BC0 5711 1----- TIMER HAS GONE 24 HOURS
FEAQ 96C0 5712 SUB AX,AX
FEAQ A6ED00 5713 MOV TIMER_HIGH,AX
FEAQ A6CC00 5714 MOV TIMER_LOW,AX
FEAQ C606700000 5715 MOV TIMER_OFL,1
FEAQ 5716
FEAQ 5717
FEAQ 5718
FEB2 5719
FEB2 FE04000 5720 T5: DISKETTE_CTL
FEB2 750B 5721 DEC MOTOR_COUNT
FEB2 750C 5722 JNZ T6 1 RETURN IF COUNT NOT OUT
FEB2 B263FD00 5723 AND MOTOR_STATUS,0FH 1 TURN OFF MOTOR RUNNING BITS
FEB2 600C 5724 MOV AL,0CH
FEBF BAF203 5725 MOV DX,03F2H 1 FOC CTL PORT
FEED EE 5726 OUT DX,AL 1 TURN OFF THE MOTOR
FEED EE 5727 T6: TIMER_RET:
FEED EE 5728 DCD 1 CH 1 TRANSFER CONTROL TO A USER ROUTINE
FEED EE 5729 MOV AX,0EI
FEED EE 5730 OUT 020H,AL 1 END OF INTERRUPT TO 8259
FEED EE 5731 POP DX
FEED EE 5732 POP AX
FEED EE 5733 POP DS 1 RESET MACHINE STATE
FEEC CF 5734 IRET 1 RETURN FROM INTERRUPT
FEEF 5735 TIMER_INT ENDP
SE7 5736
SE7 5737 1----------------------------------------------
SE7 5738 | THESE ARE THE VECTORS WHICH ARE MOVED INTO : |
SE7 5739 | THE 6066 INTERRUPT AREA DURING POWER ON, |
SE7 5740 | ONLY THE OFFSETS ARE DISPLAYED HERE, CODE |
SE7 5741 | SEGMENT WILL BE ADDED FOR ALL OF THEM, EXCEPT |
SE7 5742 | WHERE NOTED. |
SE7 5743 |-----------------------------------------------|
FEEF 5744 ASSUME CS:CODE
FEEF 5745 ORG 00FF3H
FEEF 5746 VECTOR_TABLE LABEL WORD 1 VECTOR TABLE FOR MOVE TO INTERRUPTS
FEEF 5747 DW OFFSET TIMER_INT 1 INTERRUPT B
FEEF 5748 DW OFFSET KB_INT 1 INTERRUPT 9
FEEF 5749 DW OFFSET KB_11 1 INTERRUPT 9
FEEF 574A DW OFFSET DISK_INT 1 INTERRUPT E
FEEF 574B DW OFFSET DISK_INT 1 INTERRUPT E
FEEF 574C DW OFFSET DISK_INT 1 INTERRUPT E
FEEF 574D DW OFFSET DISK_INT 1 INTERRUPT E
FEEF 574E DW OFFSET DISK_INT 1 INTERRUPT E
FEEF 574F DW OFFSET DISK_INT 1 INTERRUPT E
FEEF 5750 DW OFFSET VIDEO_ID 1 INTERRUPT 10H
FEEF 5751 DW OFFSET EQUIPMENT 1 INTERRUPT 11H
FEEF 5752 DW OFFSET MEMORY_SIZE_DET 1 INTERRUPT 12H
FEEF 5753 DW OFFSET DISKETTE_ID 1 INTERRUPT 13H
FEEF 5754 DW OFFSET RS232_ID 1 INTERRUPT 14H
FEEF 5755 DW CASSETTE_ID 1 INTERRUPT 15H FORMER CASSETTE_ID)
FEEF 5756 DW OFFSET PRINTER_ID 1 INTERRUPT 16H
FEEF 5757 DW OFFSET PRINTER_ID 1 INTERRUPT 17H
FEEF 5758
FEEF 5759 DW 0000H 1 INTERRUPT 10H
FEEF 575A DW 0060H 1 MUST BE INSERTED INTO TABLE LATER
FEEF 575B
FEEF 575C DW OFFSET BOOTSTRAP 1 INTERRUPT 19H
FEEF 575D DW OFFSET KB_1 1 INTERRUPT 1AH -- TIME OF DAY
FEEF 575E DW OFFSET KB_2 1 INTERRUPT 1BH -- KEYBOARD BREAK ADDR
FEEF 575F DW TIME_OF_DAY 1 INTERRUPT 1CH -- TIMER BREAK ADDR
FEEF 5760 DW VIDEO_PARM 1 INTERRUPT 1D -- VIDEO PARAMETERS
FEEF 5761 DW OFFSET_DISK 1 INTERRUPT 1E -- DISK PARMs
FEEF 5762 DW 0 1 INTERRUPT 1F -- POINTER TO VIDEO EXT

A-80 System BIOS
IHT 5 ---------------------------------------------------------------

1. THIS ROUTINE IS ALSO LEFT IN PLACE AFTER THE:

2. 'FF' FOR NON-HARDWARE INTERRUPTS THAT WAS:

EXECUTED ACCIDENTALLY.

LOC OBJ

IHT 5 ---------------------------------------------------------------

1. THIS ROUTINE IS ALSO LEFT IN PLACE AFTER THE:

2. 'FF' FOR NON-HARDWARE INTERRUPTS THAT WAS:

EXECUTED ACCIDENTALLY.

LOC OBJ

IHT 5 ---------------------------------------------------------------

1. THIS ROUTINE IS ALSO LEFT IN PLACE AFTER THE:

2. 'FF' FOR NON-HARDWARE INTERRUPTS THAT WAS:

EXECUTED ACCIDENTALLY.

LOC OBJ

IHT 5 ---------------------------------------------------------------

1. THIS ROUTINE IS ALSO LEFT IN PLACE AFTER THE:

2. 'FF' FOR NON-HARDWARE INTERRUPTS THAT WAS:

EXECUTED ACCIDENTALLY.

LOC OBJ

IHT 5 ---------------------------------------------------------------

1. THIS ROUTINE IS ALSO LEFT IN PLACE AFTER THE:

2. 'FF' FOR NON-HARDWARE INTERRUPTS THAT WAS:

EXECUTED ACCIDENTALLY.

LOC OBJ

IHT 5 ---------------------------------------------------------------

1. THIS ROUTINE IS ALSO LEFT IN PLACE AFTER THE:

2. 'FF' FOR NON-HARDWARE INTERRUPTS THAT WAS:

EXECUTED ACCIDENTALLY.

LOC OBJ

IHT 5 ---------------------------------------------------------------

1. THIS ROUTINE IS ALSO LEFT IN PLACE AFTER THE:

2. 'FF' FOR NON-HARDWARE INTERRUPTS THAT WAS:

EXECUTED ACCIDENTALLY.

LOC OBJ

IHT 5 ---------------------------------------------------------------

1. THIS ROUTINE IS ALSO LEFT IN PLACE AFTER THE:

2. 'FF' FOR NON-HARDWARE INTERRUPTS THAT WAS:

EXECUTED ACCIDENTALLY.

LOC OBJ

IHT 5 ---------------------------------------------------------------

1. THIS ROUTINE IS ALSO LEFT IN PLACE AFTER THE:

2. 'FF' FOR NON-HARDWARE INTERRUPTS THAT WAS:

EXECUTED ACCIDENTALLY.

LOC OBJ

IHT 5 ---------------------------------------------------------------

1. THIS ROUTINE IS ALSO LEFT IN PLACE AFTER THE:

2. 'FF' FOR NON-HARDWARE INTERRUPTS THAT WAS:

EXECUTED ACCIDENTALLY.

LOC OBJ

IHT 5 ---------------------------------------------------------------

1. THIS ROUTINE IS ALSO LEFT IN PLACE AFTER THE:

2. 'FF' FOR NON-HARDWARE INTERRUPTS THAT WAS:

EXECUTED ACCIDENTALLY.

LOC OBJ

IHT 5 ---------------------------------------------------------------

1. THIS ROUTINE IS ALSO LEFT IN PLACE AFTER THE:

2. 'FF' FOR NON-HARDWARE INTERRUPTS THAT WAS:

EXECUTED ACCIDENTALLY.

LOC OBJ

IHT 5 ---------------------------------------------------------------

1. THIS ROUTINE IS ALSO LEFT IN PLACE AFTER THE:

2. 'FF' FOR NON-HARDWARE INTERRUPTS THAT WAS:

EXECUTED ACCIDENTALLY.

LOC OBJ

IHT 5 ---------------------------------------------------------------

1. THIS ROUTINE IS ALSO LEFT IN PLACE AFTER THE:

2. 'FF' FOR NON-HARDWARE INTERRUPTS THAT WAS:

EXECUTED ACCIDENTALLY.

LOC OBJ

IHT 5 ---------------------------------------------------------------

1. THIS ROUTINE IS ALSO LEFT IN PLACE AFTER THE:

2. 'FF' FOR NON-HARDWARE INTERRUPTS THAT WAS:

EXECUTED ACCIDENTALLY.
A-82 System BIOS
5926 INT 17H ; SEND THE LINE FEED
5929 XOR AH, AH ; XOR FOR THE CR
5930 MOV AL, 150 ; CR
5931 INT 17H ; SEND THE CARRIAGE RETURN
5932 RET

5933 \--- END \---

5934 \-------------- \--------------

5935 PRINT A SEGMENT VALUE TO LOOK LIKE A 20 BIT ADDRESS :
5936 \-------------- \--------------

5937 DX MUST CONTAIN SEGMENT VALUE TO BE PRINTED :
5938 \-------------- \--------------

5939 \-------------- \--------------

5939 PRT_SEG PROC NEAR
5940 MOV AL, 0DH ; GET MSB
5941 CALL XPC_BYTE
5942 MOV AL, 01H ; LSB
5943 CALL XPC_BYTE
5944 MOV AL, '0' ; PRINT A '0'
5945 CALL PRT_HEX
5946 MOV AL, ' ' ; SPACE
5947 CALL PRT_HEX
5948 RET
5949 PRT_SEG ENDP
5950 \-------------- \--------------

5951 CODE ENDS
5952 \-------------- \--------------

5953 \-------------- \--------------

5954 \-------------- \--------------

5955 \-------------- \--------------

5956 \-------------- \--------------

5957 \-------------- \--------------

5958 \-------------- \--------------

5959 \-------------- \--------------

5960 JMP \-------------- \--------------

5961 \-------------- \--------------

5962 DB '11/08/82' ; RELEASE MARKER
5963 \-------------- \--------------

5964 \-------------- \--------------

5965 System BIOS A-83
**Fixed Disk BIOS**

```
; TITLE (FIXED DISK BIOS FOR IBM DISK CONTROLLER)

; INT 13 ---------------------------------------------

; FIXED DISK I/O INTERFACE

; THIS INTERFACE PROVIDES ACCESS TO 5 1/4" FIXED DISKS
; THROUGH THE IBM FIXED DISK CONTROLLER.

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

; 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.

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

<table>
<thead>
<tr>
<th>INPUT</th>
<th>(AH = HEX VALUE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(AH)=00</td>
<td>RESET DISK (DL = 80H, AH) / DISKETE</td>
</tr>
<tr>
<td>(AH)=01</td>
<td>READ THE STATUS OF THE LAST DISK OPERATION INTO (AL)</td>
</tr>
<tr>
<td>NOTE: DL &lt; 80H - DISKETE</td>
<td></td>
</tr>
<tr>
<td>DL &gt; 80H - DISK</td>
<td></td>
</tr>
<tr>
<td>(AH)=02</td>
<td>READ THE DESIRED SECTORS INTO MEMORY</td>
</tr>
<tr>
<td>(AH)=03</td>
<td>WRITE THE DESIRED SECTORS FROM MEMORY</td>
</tr>
<tr>
<td>(AH)=04</td>
<td>VERIFY THE DESIRED SECTORS</td>
</tr>
<tr>
<td>(AH)=05</td>
<td>FORMAT THE DESIRED TRACK</td>
</tr>
<tr>
<td>(AH)=06</td>
<td>FORMAT THE DESIRED TRACK AND SET BAD SECTOR FLAGS</td>
</tr>
<tr>
<td>(AH)=07</td>
<td>FORMAT THE DRIVE STARTING AT THE DESIRED TRACK</td>
</tr>
<tr>
<td>(AH)=08</td>
<td>RETURN THE CURRENT DRIVE PARAMETERS</td>
</tr>
<tr>
<td>(AH)=09</td>
<td>INITIALIZE DRIVE PAIR CHARACTERISTICS</td>
</tr>
<tr>
<td>(AH)=0A</td>
<td>READ LONG</td>
</tr>
<tr>
<td>(AH)=0B</td>
<td>WRITE LONG</td>
</tr>
<tr>
<td>NOTE: READ AND WRITE LONG ENCOMPASS 512 + 4 BYTES ECC</td>
<td></td>
</tr>
<tr>
<td>(AH)=OC</td>
<td>SEEK</td>
</tr>
<tr>
<td>(AH)=OD</td>
<td>ALTERNATE DISK RESET (SEE DL)</td>
</tr>
<tr>
<td>(AH)=0E</td>
<td>READ SECTOR BUFFER</td>
</tr>
<tr>
<td>(AH)=0F</td>
<td>WRITE SECTOR BUFFER</td>
</tr>
<tr>
<td>(AH)=10</td>
<td>RECOMMENDED PRACTICE BEFORE FORMATTING</td>
</tr>
<tr>
<td>(AH)=11</td>
<td>TEST DRIVE READY</td>
</tr>
<tr>
<td>(AH)=12</td>
<td>CALIBRATE</td>
</tr>
<tr>
<td>(AH)=13</td>
<td>CONTROLLER RAM DIAGNOSTIC</td>
</tr>
<tr>
<td>(AH)=14</td>
<td>DRIVE DIAGNOSTIC</td>
</tr>
<tr>
<td>(AH)=15</td>
<td>CONTROLLER INTERNAL DIAGNOSTIC</td>
</tr>
<tr>
<td>(AH)=16</td>
<td>REGISTERS USED FOR FIXED DISK OPERATIONS</td>
</tr>
<tr>
<td>(DL)</td>
<td>DRIVE NUMBER (60H-87H FOR DISK, VALUE CHECKED)</td>
</tr>
<tr>
<td>(OH)</td>
<td>HEAD NUMBER (0-7 ALLOWED, NOT VALUE CHECKED)</td>
</tr>
<tr>
<td>(CH)</td>
<td>CYLINDER NUMBER (10-1025), NOT VALUE CHECKED! (SEE CL)</td>
</tr>
<tr>
<td>(CL)</td>
<td>SECTOR NUMBER (1-17), NOT VALUE CHECKED</td>
</tr>
<tr>
<td>(AL)</td>
<td>NUMBER OF SECTORS (MAXIMUM POSSIBLE RANGE 1-80H,</td>
</tr>
<tr>
<td>FOR READ/ WRITE LONG 1-79H)</td>
<td></td>
</tr>
<tr>
<td>(ES:BX)</td>
<td>INTERLEAVE VALUE FOR FORMAT 1-160</td>
</tr>
<tr>
<td>(ES:BX)</td>
<td>ADDRESS OF BUFFER FOR READS AND WRITES</td>
</tr>
<tr>
<td>(AH)=0F</td>
<td>NOT REQUIRED FOR VERIFY</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OUTPUT</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AH = STATUS OF CURRENT OPERATION</td>
<td></td>
</tr>
<tr>
<td>CY = 0</td>
<td>SUCCESSFUL OPERATION (AH=0 ON RETURN)</td>
</tr>
<tr>
<td>CY = 1</td>
<td>FAILED OPERATION (AH HAS ERROR REASON)</td>
</tr>
<tr>
<td>NOTE: ERROR 1H INDICATES THAT THE DATA READ HAS A RECOVERABLE</td>
<td></td>
</tr>
<tr>
<td>ERROR WHICH WAS CORRECTED BY THE ECC ALGORITHM. THE DATA</td>
<td></td>
</tr>
<tr>
<td>IS PROBABLY GOOD, HOWEVER THE BIOS ROUTINE INDICATES AN</td>
<td></td>
</tr>
<tr>
<td>ERROR TO ALLOW THE CONTROLLING PROGRAM A CHANCE TO DECIDE</td>
<td></td>
</tr>
<tr>
<td>FOR ITSELF. THE ERROR MAY NOT RECUR IF THE DATA IS</td>
<td></td>
</tr>
</tbody>
</table>

---

A-84 Fixed Disk BIOS
REWRITTEN. (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 000H ; UNDEFINED ERROR OCCURRED

0000

TIME_OUT EQU 080H ; ATTACHMENT FAILED TO RESPOND

0070

BAD_SEEK EQU 40H ; SEEK OPERATION FAILED

0020

BAD_CTRL EQU 20H ; CONTROLLER HAS FAILED

0011

DATA_CORRECTED EQU 11H ; ECC CORRECTED DATA ERROR

0010

BAD_ECC EQU 10H ; BAD ECC ON DISK READ

0010

BAD_TRACK EQU 08H ; BAD TRACK FLAG DETECTED

0039

DMA_BOUNDARY EQU 09H ; ATTEMPT TO DMA ACROSS 64K BOUNDARY

0007

INIT_FAIL EQU 07H ; DRIVE PARAMETER ACTIVITY FAILED

0005

BAD_RESET EQU 05H ; RESET FAILED

0004

RECORD_NOT_FMT EQU 04H ; REQUESTED SECTOR NOT FOUND

0002

BAD_ADDR MARK EQU 02H ; ADDRESS MARK

0001

BAD_CMD EQU 01H ; BAD COMMAND PASSED TO DISK I/O

-------------------------------------------------------------------------

INTERRUPT AND STATUS AREAS:

-------------------------------------------------------------------------

DUMMY SEGMENT AT 0

ORG 000H*4 ; FIXED DISK INTERRUPT VECTOR

ORG 001H*4 ; DISK INTERRUPT VECTOR

ORG 002H*4 ; BOOTSTRAP INTERRUPT VECTOR

ORG 003H*4 ; DISKETTE INTERRUPT VECTOR

ORG 004H*4 ; NEW DISKETTE INTERRUPT VECTOR

ORG 005H*4 ; FIXED DISK PARAMETER VECTOR

ORG 006H*4 ; DISKETTE LOAD VECTOR

ORG 007H*4 ; DISKETTE LOAD VECTOR

ORG 008H*4 ; DISKETTE LOAD VECTOR

ORG 009H*4 ; DISKETTE LOAD VECTOR

ORG 00AH*4 ; DISKETTE LOAD VECTOR

DATA SEGMENT AT 40H

ORG 42H

ORG 43H

ORG 44H

ORG 45H

ORG 46H

ORG 47H

ORG 48H

ORG 49H

ORG 4AH

ORG 4BH

ORG 4CH

ORG 4DH

ORG 4EH

ORG 4FH

ORG 50H

ORG 51H

ORG 52H

ORG 53H

ORG 54H

ORG 55H

ORG 56H

ORG 57H

ORG 58H

ORG 59H

ORG 5AH

ORG 5BH

ORG 5CH

ORG 5DH

ORG 5EH

ORG 5FH

ORG 60H

ORG 61H

ORG 62H

ORG 63H

ORG 64H

ORG 65H

ORG 66H

ORG 67H

ORG 68H

ORG 69H

ORG 6AH

ORG 6BH

ORG 6CH

ORG 6DH

ORG 6EH

ORG 6FH

 Hawks Disk BIOS A-85

Hardware Specific Values:

Controller I/O Port

> When Read From:
A-86  Fixed Disk BIOS
Fixed Disk BIOS

A-87
```assembly
0065 6ECD  MOV ES,AX  ; SET SEGMENT
0067 20DB  SUB BX,BX
0069 B000F  MOV AX,0F00H  ; WRITE SECTOR BUFFER
006C CD13  INT 13H
006F 7252  JC ERROR_EX
0070 FE067500  INC HF_NRM  ; DRIVE ZERO RESPONDED
0072 B13D2  MOV DX,DX,1M  ; EXPANSION BOX
0074 B000  MOV AL,0  ; EXPANSION BOX OFF
0075 31F  OUT DX,AL  ; TURN BOX OFF
0076 00E5 2508  MOV AX,OFDH  ; WRITE SECTOR BUFFER
007E 7252  JC ERROR_EX
0080 FO67500  INC HF_NRM  ; DRIVE ZERO RESPONDED
0082 DFOF  MOV AX,AL,0FH
0083 1C8F  CMP AX,0FH  ; ... IN THE SYSTEM UNIT
0085 74067500  INC HF_HUN  ; TALLY ANOTHER DRIVE
0087 074067500  MOV AX,AL,0FH
0089 74067500  INC HF_HUN
008B 074067500  MOV AX,AL,0FH
008D 74067500  INC HF_HUN
008F 74067500  INC HF_HUN
0091 E7066C004A01  MOV DX,TIMER_LOW,424D  ; CONTROLLER IS IN SYSTEM UNIT
0093 3C072  JP BOX_ON  ; EXPANSION BOX OFF
0095 BDF0  MOV BP,OFH  ; POD ERROR FLAG
0097 2BCD  SUB AX,AX  ; RESET
0099 8B06009  MOV CX,F17L  ; MESSAGE CHARACTER COUNT
009B 5100  LODS  ; MESSAGE CHARACTER COUNT
009D 89060090  MOV CX,F17L
009F 5100  LODS  ; MESSAGE CHARACTER COUNT
00A1 8700  MOV BH,0  ; PAGE ZERO
00A3 E0A464E801  MOV AL,C5:F17511  ; GET BYTE
00A5 5100  LODS  ; MESSAGE CHARACTER COUNT
00A7 B40E  MOV AH,14D  ; VIDEO OUT
00A9 5100  LODS  ; MESSAGE CHARACTER COUNT
00AB C010  INT 10H  ; DISPLAY CHARACTER
00AD 5100  LODS  ; MESSAGE CHARACTER COUNT
00AF 5100  LODS  ; MESSAGE CHARACTER COUNT
00B1 F9  OUT CH:  ; DO MORE
00B3 5100  LODS  ; MESSAGE CHARACTER COUNT
00B5 FA  CLI  ; BE SURE TIMER IS DISABLED
00B7 ED21  IN AL,021H
00B9 EC31  OR AL,01H
00BB ED21  OUT 021H,AL
00BD F8  STI
00BF 6A5000  CALL DSBL
00C3 CB  RET
00C5 31373031  F17 DB '1701',0DH,0AH
00C7 0D  MOV CX  ; SAVE REGISTER
00C9 0A  MOV DX  
00CA 06  F17L EQU $-F17
00CE 31F  MOV AX,CX  ; SAVE REGISTER
00D0 51  PUSH CX
00D2 52  PUSH DX
```
INT 19 ---------------------------------------------------

THE FIXED DISK ROM REPLACES THE INTERRUPT 19 BOOT STRAP VECTOR WITH A POINTER TO THIS BOOT ROUTINE

RESET THE DEFAULT DISK AND DISKETTE PARAMETER VECTORS

THE BOOT BLOCK to BE READ IN WILL BE ATTEMPTED FROM CYLINDER 0 SECTOR 1 OF THE DEVICE.

IF THE ABOVE FAILS CONTROL IS PASSED TO RESIDENT BASIC

----------------------------------------------------------------

0186 FA
0186 26CO
0188 8ED8
0188 C7060401E703
0191 acOE0601
0195 C70676000102
019B 6CCE7A00
019F FB
01A8 740A
01B3 890100
01B6 890300
01B8 890300
01B9 2B02
01BB 59
01BC 755A
01BE 60F200
01C1 745A
01C3 7230
01C5 EB0490
01CB 6400H
01CE 69H
01CF 63H
01D0 62H
01D2 61H
01D4 60H
01D6 5FH
01D8 5EH
01DB 5DH
01DC 5CH
01DD 5BH
01DE 5AH
01E0 59H
01E2 58H
01E4 57H
01E6 56H
01E8 55H
01EA 54H
01EC 53H
01EE 52H
01F0 51H
01F2 50H
01F4 4FH
01F6 4EH
01F8 4DH
01FA 4CH
01FB 4BH
01FD 4AH
01FE 49H
01FF 48H
0200 47H
0201 46H
0202 45H
0203 44H
0204 43H
0205 42H
0206 41H
0207 40H
0208 3FH
0209 3EH
020A 3DH
020B 3CH
020C 3BH
020D 3AH
020E 39H
020F 38H
0210 37H
0211 36H
0212 35H
0213 34H
0214 33H
0215 32H
0216 31H
0217 30H
0218 2FH
0219 2EH
021A 2DH
021B 2CH
021C 2BH
021D 2AH
021E 29H
021F 28H
0220 27H
0221 26H
0222 25H
0223 24H
0224 23H
0225 22H
0226 21H
0227 20H
0228 1FH
0229 1EH
022A 1DH
022B 1CH
022C 1BH
022D 1AH
022E 19H
022F 18H
0230 17H
0231 16H
0232 15H
0233 14H
0234 13H
0235 12H
0236 11H
0237 10H
0238 9H
0239 8H
023A 7H
023B 6H
023C 5H
023D 4H
023E 3H
023F 2H
0240 1H
0241 0H

Appendix A

Fixed Disk BIOS  A-89
ATTEMPT BOOTSTRAP FROM FIXED DISK

LOC 01C8 EAA07C0000
  1 ------  ATTEMPT BOOTSTRAP FROM FIXED DISK
  2
  3
  4
  5
  6
  7
  8
  9
 10
 11
 12
 13
 14
 15
 16
 17
 18
 19
 20
 21
 22
 23
 24
 25
 26
 27
 28
 29
 30
 31
 32
 33
 34
 35
 36
 37
 38
 39
 40
 41
 42
 43
 44
 45
 46
 47
 48
 49
 50
 51
 52
 53
 54
 55
 56
 57
 58
 59
 60
 61
 62
 63
 64
 65
 66
 67
 68
 69
 70
 71
 72
 73
 74
 75
 76
 77
 78
 79
 80
 81
 82
 83
 84
 85
 86
 87
 88
 89
 90
 91
 92
 93
 94
 95
 96
 97
 98
 99
 100
 101
 102
 103
 104
 105
 106
 107
 108
 109
 110
 111
 112
 113
 114
 115
 116
 117
 118
 119
 120
 121
 122
 123
 124
 125
 126
 127
 128
 129
 130
 131
 132
 133
 134
 135
 136
 137
 138
 139
 140
 141
 142
 143
 144
 145
 146
 147
 148
 149
 150
 151
 152
 153
 154
 155
 156
 157
 158
 159
 160
 161
 162
 163
 164
 165
 166
 167
 168
 169
 170
 171
 172
 173
 174
 175
 176
 177
 178
 179
 180
 181
 182
 183
 184
 185
 186
 187
 188
 189
 190
 191
 192
 193
 194
 195
 196
 197
 198
 199
 200
 201
 202
 203
 204
 205
 206
 207
 208
 209
 210
 211
 212
 213
 214
 215
 216
 217
 218
 219
 220
 221
 222
 223
 224
 225
 226
 227
 228
 229
 230
 231
 232
 233
 234
 235
 236
 237
 238
 239
 240
 241
 242
 243
 244
 245
 246
 247
 248
 249
 250
 251
 252
 253
 254
 255
 256
 257
 258
 259
 260
 261
 262
 263
 264
 265
 266
 267
 268
 269
 270
 271
 272
 273
 274
 275
 276
 277
 278
 279
 280
 281
 282
 283
 284
 285
 286
 287
 288
 289
 290
 291
 292
 293
 294
 295
 296
 297
 298
 299
 300
 301
 302
 303
 304
 305
 306
 307
 308
 309
 310
 311
 312
 313
 314
 315
 316
 317
 318
 319
 320
 321
 322
 323
 324
 325
 326
 327
 328
 329
 330
 331
 332
 333
 334
 335
 336
 337
 338
 339
 340
 341
 342
 343
 344
 345
 346
 347
 348
 349
 350
 351
 352
 353
 354
 355
 356
 357
 358
 359
 360
 361
 362
 363
 364
 365
 366
 367
 368
 369
 370
 371
 372
 373
 374
 375
 376
 377
 378
 379
 380
 381
 382
 383
 384
 385
 386
 387
 388
 389
 390
 391
 392
 393
 394
 395
 396
 397
 398
 399
 400
 401
 402
 403
 404
 405
 406
 407
 408
 409
 410
 411
 412
 413
 414
 415
 416
 417
 418
 419
 420
 421
 422
 423
 424
 425
 426
 427
 428
 429
 430
 431
 432
 433
 434
 435
 436
 437
 438
 439
 440
 441
 442
 443
 444
 445
 446
 447
 448
 449
 450
 451
 452
 453
 454
 455
 456
 457
 458
 459
 460
 461
 462
 463
 464
 465
 466
 467
 468
 469
 470
 471
 472
 473
 474
 475
 476
 477
 478
 479
 480
 481
 482
 483
 484
 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
 513
 514
 515
 516
 517
 518
 519
 520
 521
 522
 523
 524
 525
 526
 527
 528
 529
 530
 531
 532
 533

A-90  Fixed Disk BIOS
LOC OBJ  |  LINE  |  SOURCE
0247 FA  |  534  |  CLI  AL.021H  ;  DISABLE INTERRUPTS
0248 E241 |  535  |  IN  AL.021H  ;  DISABLE INTERRUPTS
024A 0C20 |  536  |  OR  AL.021H  ;  DISABLE INTERRUPTS
024C 6621 |  537  |  OUT  021H,AL  ;  DISABLE INTERRUPTS
024E FB  |  538  |  STI  ;  ENABLE INTERRUPTS
024F 56  |  539  |  POP  AX  ;  RESTORE OFFSET
0250 00267700  |  540  |  MOV  PORT.OFF, AH  ;  RESTORE SEGMENT
0254 1F  |  541  |  POP  DS  ;  RESTORE SEGMENT
0255 C3  |  542  |  RET  ;  RESTORE SEGMENT
0256  |  543  |  DB  ENDP  ;  FIXED DISK BIOS ENTRPY POINT
0256  |  544  |  DISK_ID_PROC  FAR  ;  FIXED DISK BIOS ENTRPY POINT
0256  |  545  |  -------------------------------
0256  |  546  |  -------------------------------
0256  |  547  |  DISK_ID_PROC  FAR  ;  FIXED DISK BIOS ENTRPY POINT
0256  |  548  |  ASSUME  DS:NOTHING.ES:NOTHING  ;  FIXED DISK BIOS ENTRPY POINT
0256  |  549  |  00FA80  |  551  |  CMP  DL:00H  ;  TEST FOR FIXED DISK DRIVE
0259 7355 |  552  |  JAE  HARD.DISK  ;  YES, HANDLE HERE
025D CD40  |  553  |  INT  40H  ;  DISKETTE HANDLER
025D  |  554  |  RET  ;  FIXED DISK BIOS ENTRPY POINT
025D  |  555  |  RET  2  ;  BACK TO CALLER
0260  |  556  |  HARD.DISK:  ;  FIXED DISK BIOS ENTRPY POINT
0260  |  557  |  ASSUME  DS:DATA  ;  FIXED DISK BIOS ENTRPY POINT
0260  |  558  |  6B  ;  ENABLE INTERRUPTS
0261 8A44 |  559  |  OR  AH,AH  ;  FIXED DISK BIOS ENTRPY POINT
0263 7599 |  560  |  JNZ  A3  ;  FIXED DISK BIOS ENTRPY POINT
0265 CD40 |  561  |  INT  40H  ;  TEST FOR FIXED DISK DRIVE
0267 2A44 |  562  |  SUB  AH,AH  ;  FIXED DISK BIOS ENTRPY POINT
0269 00FA81 |  563  |  CMP  DL:(OH + S_MAX_FILE - 1)  ;  FIXED DISK BIOS ENTRPY POINT
026C 77EF |  564  |  JA  RET_2  ;  FIXED DISK BIOS ENTRPY POINT
026E  |  565  |  A5:  ;  FIXED DISK BIOS ENTRPY POINT
026E  |  566  |  CMP  AH,AH  ;  FIXED DISK BIOS ENTRPY POINT
0271 7503 |  567  |  JNZ  A2  ;  FIXED DISK BIOS ENTRPY POINT
0273 9E1A01 |  568  |  JMP  GET.PARM.N  ;  FIXED DISK BIOS ENTRPY POINT
0276  |  569  |  A2:  ;  FIXED DISK BIOS ENTRPY POINT
0276  |  570  |  PUSH  BX  ;  FIXED DISK BIOS ENTRPY POINT
0277 51  |  571  |  PUSH  CX  ;  FIXED DISK BIOS ENTRPY POINT
0278 52  |  572  |  PUSH  DX  ;  FIXED DISK BIOS ENTRPY POINT
0279 56  |  573  |  PUSH  DS  ;  FIXED DISK BIOS ENTRPY POINT
027A 06  |  574  |  PUSH  ES  ;  FIXED DISK BIOS ENTRPY POINT
027B 56  |  575  |  PUSH  SI  ;  FIXED DISK BIOS ENTRPY POINT
027C 57  |  576  |  PUSH  DI  ;  FIXED DISK BIOS ENTRPY POINT
027D 8B4A00 |  577  |  CALL  DISK_ID_CONT  ;  FIXED DISK BIOS ENTRPY POINT
027D  |  578  |  CALL  DISK_ID_CONT  ;  FIXED DISK BIOS ENTRPY POINT
027E  |  579  |  ---  ;  FIXED DISK BIOS ENTRPY POINT
0280 50  |  580  |  PUSH  AX  ;  FIXED DISK BIOS ENTRPY POINT
0281 850FF |  581  |  CALL  DSBL  ;  FIXED DISK BIOS ENTRPY POINT
0284 800400 |  582  |  MOV  AX,DATA  ;  FIXED DISK BIOS ENTRPY POINT
0287 8006 |  583  |  MOV  DS,AH  ;  FIXED DISK BIOS ENTRPY POINT
0289 56  |  584  |  POP  AX  ;  FIXED DISK BIOS ENTRPY POINT
028A 8A267400 |  585  |  MOV  AH,DISK_STATUS  ;  FIXED DISK BIOS ENTRPY POINT
028E 80FC01 |  586  |  CMP  AH,1  ;  FIXED DISK BIOS ENTRPY POINT
0291 F5  |  587  |  CMP  ;  FIXED DISK BIOS ENTRPY POINT
0292 5F  |  588  |  POP  DI  ;  FIXED DISK BIOS ENTRPY POINT
0293 5E  |  589  |  POP  SI  ;  FIXED DISK BIOS ENTRPY POINT
0294 67  |  590  |  POP  ES  ;  FIXED DISK BIOS ENTRPY POINT
0295 1F  |  591  |  POP  DS  ;  FIXED DISK BIOS ENTRPY POINT
0296 54  |  592  |  POP  DX  ;  FIXED DISK BIOS ENTRPY POINT
0297 59  |  593  |  POP  CX  ;  FIXED DISK BIOS ENTRPY POINT
0298 5B  |  594  |  POP  BX  ;  FIXED DISK BIOS ENTRPY POINT
0299 CA0200 |  595  |  RET  2  ;  FIXED DISK BIOS ENTRPY POINT
0299  |  596  |  DISK_ID_ENDP  ;  FIXED DISK BIOS ENTRPY POINT
029C 599  |  597  |  M1  ;  FIXED DISK BIOS ENTRPY POINT
029C 3603 |  598  |  DW  DISK_RESET  ;  FIXED DISK BIOS ENTRPY POINT
029E 4003 |  599  |  DW  RETURN_STATUS  ;  FIXED DISK BIOS ENTRPY POINT
02A0 5603 |  600  |  DW  DISK_READ  ;  FIXED DISK BIOS ENTRPY POINT
02A2 6003 |  601  |  DW  DISK_WRITE  ;  FIXED DISK BIOS ENTRPY POINT
02A4 6403 |  602  |  DW  DISK_VERF  ;  FIXED DISK BIOS ENTRPY POINT
02A6 7203 |  603  |  DW  FMT.TRK  ;  FIXED DISK BIOS ENTRPY POINT
02A8 7903 |  604  |  DW  FMT.BAD  ;  FIXED DISK BIOS ENTRPY POINT
02A9 6003 |  605  |  DW  FMT.DRY  ;  FIXED DISK BIOS ENTRPY POINT
02AC 3C00 |  606  |  DW  BAD.COMMAND  ;  FIXED DISK BIOS ENTRPY POINT
02AE 2704 |  607  |  DW  INIT.DRV  ;  FIXED DISK BIOS ENTRPY POINT
02B0 CF04 |  608  |  DW  RD_LONG  ;  FIXED DISK BIOS ENTRPY POINT
02B2 DD04 |  609  |  DW  WR_LONG  ;  FIXED DISK BIOS ENTRPY POINT
LOC OBJ

0264 F204 611 DW DISK_SEEK 00CH
0266 3003 612 DW DISK_RESET 00DH
0268 F004 613 DW RD_BUFF 0012H
026A 0705 614 DW WR_BUFF 00FH
026C 1505 615 DW TST_RDY 010H
026E 1C05 616 DW HDDISK_RECAL 011H
0270 2305 617 DW RAM_DIAG 012H
0272 2A05 618 DW CHK_DRV 013H
0274 3105 619 DW CNTLR_DIAG 014H

002A 620 MIL EQU 9-MI

02C6 621 SETUP_PROC NEAR

02CC 6C07740000 622 MOV DISK_STATUS,0 ; RESET THE STATUS INDICATOR
02CC 51 625 PUSH CX ; SAVE CX
02CC 26 626 CALL SETUP_PROC

02CE 8AEE 627 ----- CALCULATE THE PORT OFFSET

02CE 8ACA01 628 MOV CH,DL ; SAVE DL
02CD FECA 630 OR DL,1
02CE B00000 631 DEC DL
02CF 0000 632 SML DL,1 ; GENERATE OFFSET
02D0 6817700 633 MOV ,PORT_OFF,DL ; STORE OFFSET
02D2 5A05 634 MOV DL,CH ; RESTORE DL
02D4 0E01 635 AND DL,1
02D6 5105 636 MOV CL,5 ; SHIFT COUNT
02D6 E0E2 637 SML DL,CL
02D7 5A06 638 OR DL,DL
02D8 60162000 639 MOV CMD_BLOCK+1,DL
02DE 59 640 PDP CX
02E0 C3 641 RET

02E2 642 SETUP_PROC ENDP

02E4 8AEA 643 DISK_IO_CONT PROC NEAR
02E6 E0 644 PUSH AX
02E8 040000 645 MOV AX,DATA
02ED 0000 646 MOV DS,AX ; ESTABLISH SEGMENT
02F0 50 647 POP AX
02F1 00FC01 648 CMP AH,01H ; RETURN STATUS
02F4 7503 649 JNZ AX
02F6 EB5900 650 JMP RETURN_STATUS
02F9 653 651 SUB DL,OH
02FA 0EAE00 652 CMP DL,MAX_FILE ; LEGAL DRIVE TEST
02FB 732F 653 JAE BAD_COMMAND
0301 655 654 CALL SETUP_A

0303 E8C2FF 656 ----- SET UP COMMAND BLOCK

0306 FEC9 657 DISK_STATUS.BAD_CMDS = 0
0306 C049+00000 658 MOV CMP_BLOCK+0,0
0306 6B00+4400 659 MOV CMP_BLOCK+2,CL
030F 602E+5000 660 MOV CMP_BLOCK+3,CH
0315 A46000 661 MOV CMP_BLOCK+4,AL
0316 67000 662 MOV AL,CMD BYTE
0319 A24700 663 MOV CMP_BLOCK+5,AL
031C 50 664 PUSH AX ; SAVE AX
031D 8AC4 665 MOV AL,AM ; GET INTO LOW BYTE
031F 32E4 666 XOR AH,AM ; ZERO HIGH BYTE
0321 D1E0 667 SAL AX,1 ; #2 FOR TABLE LOOKUP
0323 88F0 668 CMP SX,AX
0325 3DA400 669 CMP AX,MIL ; TEST WITHIN RANGE
0328 50 670 POP AX ; RESTORE AX
0329 7305 671 JNB BAD_COMMAND
032B 5FCC00 672 JMP WORD PTR CS:[SI + OFFSET MI]
0330 67 673 BAD_COMMAND
0330 C06740001 674 MOV DISK_STATUS,BAD_CMD ; COMMAND ERROR
0335 3000 675 MOV AL,0
0337 C3 676 RET

0339 677 DISK_IO_CONT ENDP

A-92 Fixed Disk BIOS
0338 E04306 669 CALL PORT_1 ; RES} PORT
0338 EE 690 OUT DX,AL ; ISSUE RESET
033C E03F04 691 CALL PORT_1 ; CONTROLLER HARDWARE STATUS
033F EC 692 IN AL,DX ; GET STATUS
0340 2042 693 AND AL,E ; ERROR BIT
0342 7406 694 JJZ DRI
0344 C0074000E 695 MOV DISK_STATUS,BAD_RESET
0349 C3 696 RET
034A E90A00 697 DR1: JMP INIT_DRV ; SET THE DRIVE PARAMETERS
034A DISK_RESET ENDP 699 ; DISK RESET ROUTINE (AH = 00H)
0350 700 ; DISK_PROTOCOL ROUTINE (AH = 00H)
035A 701 ; DISK_READ_ROUTINE (AH = 00H)
0360 702 ; DISK_WRITE ROUTINE (AH = 00H)
036A 703 ; DISK_VERIFY ROUTINE (AH = 00H)
0370 704 ; DISK_VERIFY ROUTINE (AH = 00H)
037E 705 ; FORMAT Track ROUTINE (AH = 00H)
0382 706 ; FORMAT Drive ROUTINE (AH = 00H)
038E 707 ; FORMAT CONT ROUTINE (AH = 00H)
0394 708 ; FORMAT BAG TRACK ROUTINE (AH = 00H)
039A 709 ; FORMAT DRIVE ROUTINE (AH = 00H)
039E 710 ; ZERO OUT SECTOR

Appendix A

Fixed Disk BIOS  A-93
Line 764: [-----------------------------]
Line 765: | GET PARAMETERS (AH = 0) |
Line 766: [-----------------------------]
Line 767: |------------------------------|
Line 768: | GET_PARM_N | LABEL NEAR |
Line 769: | GET_PARM | PROC FAR |
Line 770: | SAVE Registers |
Line 771: |------------------|
Line 772: |------------------|
Line 773: |------------------|
Line 774: |------------------|
Line 775: |------------------|
Line 776: |------------------|
Line 777: |------------------|
Line 778: |------------------|
Line 779: |------------------|
Line 780: |------------------|
Line 781: |------------------|
Line 782: |------------------|
Line 783: |------------------|
Line 784: |------------------|
Line 785: |------------------|
Line 786: |------------------|
Line 787: |------------------|
Line 788: |------------------|
Line 789: |------------------|
Line 790: |------------------|
Line 791: |------------------|
Line 792: |------------------|
Line 793: |------------------|
Line 794: |------------------|
Line 795: |------------------|
Line 796: |------------------|
Line 797: |------------------|
Line 798: |------------------|
Line 799: |------------------|
Line 800: |------------------|
Line 801: |------------------|
Line 802: |------------------|
Line 803: |------------------|
Line 804: |------------------|
Line 805: |------------------|
Line 806: |------------------|
Line 807: |------------------|
Line 808: |------------------|
Line 809: |------------------|
Line 810: |------------------|
Line 811: |------------------|
Line 812: |------------------|
Line 813: |------------------|
Line 814: |------------------|
Line 815: |------------------|
Line 816: |------------------|
Line 817: |------------------|
Line 818: |------------------|
Line 819: |------------------|
Line 820: |------------------|
Line 821: |------------------|
Line 822: |------------------|
Line 823: |------------------|
Line 824: |------------------|
Line 825: |------------------|
Line 826: |------------------|
Line 827: |------------------|
Line 828: |------------------|
Line 829: |------------------|
Line 830: |------------------|
Line 831: |------------------|
Line 832: |------------------|
Line 833: |------------------|
Line 834: |------------------|
Line 835: |------------------|
Line 836: |------------------|
Line 837: |------------------|
Line 838: |------------------|
Line 839: |------------------|
Line 840: |------------------|
Line 841: |------------------|
Line 842: |------------------|

A-94 Fixed Disk BIOS
8.2 RESERVED FOR FUTURE USE

8.3 TO DYNAMICALLY DEFINE A SET OF PARAMETERS

8.4 BUILD A TABLE OF VALUES AND PLACE THE CORRESPONDING VECTOR INTO INTERRUPT 41.

8.5 NOTE: THE DEFAULT TABLE IS VECTORED IN FOR AN INTERRUPT 19H (BOOTSTRAP)

8.6 ON THE CARD SWITCH SETTINGS

8.7 DRIVE 0 DRIVE 1

8.8 ----------------------------

8.9 THE DEFAULT TABLE IS VECTORED IN FOR

8.10 A TABLE OF VALUES AND PLACE THE CORRESPONDING VECTOR INTO INTERRUPT 41.

8.11 ON THE CARD SETTINGS

8.12 ----------------------------

8.13 TRANSLATION TABLE

8.14 DRIVE 0 DRIVE 1

8.15 ----------------------------

8.16 STANDARD

8.17 CHECK DRIVE

8.18 DRIVE TYPE 00

8.19 DRIVE TYPE 01

8.20 DRIVE TYPE 02

8.21 DRIVE TYPE 03

--- FIXED DISK BIOS A-95 ---

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0367</td>
<td>3201</td>
</tr>
<tr>
<td>0369</td>
<td>02</td>
</tr>
<tr>
<td>036A</td>
<td>3201</td>
</tr>
<tr>
<td>036C</td>
<td>0000</td>
</tr>
<tr>
<td>036E</td>
<td>00</td>
</tr>
<tr>
<td>0370</td>
<td>0C</td>
</tr>
<tr>
<td>0372</td>
<td>04</td>
</tr>
<tr>
<td>0374</td>
<td>08</td>
</tr>
<tr>
<td>0376</td>
<td>00</td>
</tr>
<tr>
<td>0378</td>
<td>00</td>
</tr>
<tr>
<td>037A</td>
<td>02</td>
</tr>
<tr>
<td>037C</td>
<td>04</td>
</tr>
<tr>
<td>037E</td>
<td>06</td>
</tr>
<tr>
<td>0380</td>
<td>00</td>
</tr>
<tr>
<td>0382</td>
<td>00</td>
</tr>
<tr>
<td>0384</td>
<td>0C</td>
</tr>
<tr>
<td>0386</td>
<td>04</td>
</tr>
<tr>
<td>0388</td>
<td>08</td>
</tr>
<tr>
<td>038A</td>
<td>00</td>
</tr>
<tr>
<td>038C</td>
<td>00</td>
</tr>
<tr>
<td>038E</td>
<td>00</td>
</tr>
<tr>
<td>0390</td>
<td>0C</td>
</tr>
<tr>
<td>0392</td>
<td>04</td>
</tr>
<tr>
<td>0394</td>
<td>08</td>
</tr>
<tr>
<td>0396</td>
<td>00</td>
</tr>
<tr>
<td>0398</td>
<td>0C</td>
</tr>
<tr>
<td>039A</td>
<td>04</td>
</tr>
<tr>
<td>039C</td>
<td>08</td>
</tr>
<tr>
<td>039E</td>
<td>00</td>
</tr>
</tbody>
</table>

--- FIXED DISK BIOS A-95 ---

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0407</td>
<td>3201</td>
</tr>
<tr>
<td>0409</td>
<td>06</td>
</tr>
<tr>
<td>040B</td>
<td>00</td>
</tr>
<tr>
<td>040D</td>
<td>00</td>
</tr>
<tr>
<td>040F</td>
<td>00</td>
</tr>
<tr>
<td>0411</td>
<td>04</td>
</tr>
<tr>
<td>0413</td>
<td>00</td>
</tr>
<tr>
<td>0415</td>
<td>04</td>
</tr>
<tr>
<td>0417</td>
<td>3201</td>
</tr>
<tr>
<td>0419</td>
<td>04</td>
</tr>
</tbody>
</table>

--- FIXED DISK BIOS A-95 ---

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0401</td>
<td>04</td>
</tr>
<tr>
<td>0403</td>
<td>00</td>
</tr>
<tr>
<td>0405</td>
<td>00</td>
</tr>
<tr>
<td>0407</td>
<td>00</td>
</tr>
<tr>
<td>0409</td>
<td>00</td>
</tr>
<tr>
<td>040B</td>
<td>00</td>
</tr>
<tr>
<td>040D</td>
<td>00</td>
</tr>
<tr>
<td>040F</td>
<td>00</td>
</tr>
<tr>
<td>0411</td>
<td>00</td>
</tr>
<tr>
<td>0413</td>
<td>00</td>
</tr>
<tr>
<td>0415</td>
<td>00</td>
</tr>
<tr>
<td>0417</td>
<td>00</td>
</tr>
<tr>
<td>0419</td>
<td>00</td>
</tr>
</tbody>
</table>

--- FIXED DISK BIOS A-95 ---
LOC    OBJ    LINE    SOURCE
041A  3201    919    DH    0306D
041C  0000    920    DW    0000D
041E  00    921    DB    00H
041F  05    922    DB    05H
0420  0C    923    DB    0CH ; STANDARD
0421  B4    924    DB    084H ; FORMAT DRIVE
0422  20    925    DB    020H ; CHECK DRIVE
0423  00000000    926    DB    0.0.0.0
0427    928    INIT_DRV    PROC    NEAR
0434    934    CALL    INIT_DRV_R
0437    937    ;----- DO DRIVE ZERO
0436    936    INIT_DRV_R    PROC
0438    938    JS    03
0440    940    MOV    CMD_BLOCK+0,INIT_DRV_CMD
0442    942    MOV    CMD_BLOCK+1,0
0443    943    CALL    INIT_DRV_R
0447    947    ;----- DO DRIVE ONE
0446    946    MOV    CMD_BLOCK+0,INIT_DRV_CMD
0448    948    MOV    CMD_BLOCK+1,00100000B
0449    949    CALL    INIT_DRV_R
0455    955    IF
0456    956    MOV    DS.AX
0458    958    LES    BX,HF_TBL_VEC
045A    959    POP    DS
045C    960    CALL    INIT_DRV_S
0462    962    MOV    01,0
0466    964    MOV    01,1
0467    965    MOV    01,2
0468    967    MOV    01,3
0469    969    MOV    01,4
046A    970    MOV    01,5
046B    971    CALL    INIT_DRV_S
046C    973    JC    B3
046E    976    JC    B3
046F    977    JC    B3
0470    978    JC    B3
0471    979    JC    B3
0472    980    JC    B3
0473    981    JC    B3
0477    984    JC    B3
047A    986    MOV    DI.4
047C    987    MOV    DI.5
047E    988    MOV    DI.6
047F    989    MOV    DI.7
0480    990    MOV    DI.8
0481    991    CALL    INIT_DRV_S
0482    992    JC    B3
0483    993    JC    B3
0484    994    JC    B3
0485    995    JC    B3

A-96    Fixed Disk BIOS
0498 7215 996 JC BS
0499 BF00 997 MOV DI,0 ; DRIVE STEP OPTION
049A 268A01 999 MOV ALES:(BX + DI)
049B A7600 1000 MOV CONTROL_BYTE,AL
049C 01 1001
049D 2BC9 1002 SUB CX,CX
049E 80302 1003 CALL PORT_1
049F 1005 IN AL,DX
04A0 A00 1006 TEST ARI управление ; STATUS INPUT MODE
04A1 7599 1007 JNZ B6
04A2 E276 1008 LOOP BS
04A3 268A0007 1009 MOV DISK_STATUS,INIT_FAIL ; OPERATION FAILED
04A4 F9 1011 STC
04A5 C3 1012 RET
04A6 EC 1013
04A7 8201 1014 B6: CALL PORT_0
04A8 1016 IN AL,DX
04A9 2402 1017 AND AL,2 ; MASK ERROR BIT
04AA 75F1 1018 JNZ B5
04AB C3 1019 RET
04AC 75F1 1020 ASSUME ES:NOTHING
04AD 1021 INIT_DRV_R ENDP
04AE 75F1 1022
04AF 8033 1023 ; ----- SEND THE BYTE OUT TO THE CONTROLLER
04B0 7207 1024
04B1 1025 INIT_DRV_S PROC NEAR
04B2 E81900 1026 CALL CHK_LONG
04B3 7207 1027 JC DI
04B4 8047 1028 CALL PORT_0
04B5 1029 MOV ALES:(BX + DI)
04B6 EE 1030 OUT DX,AL
04B7 C3 1031 DI: RET
04B8 C3 1032 INIT_DRV_S ENDP
04B9 1033
04BA 75F1 1034 ; READ LONG (AH = OAH)
04BB F8 1035 CMC
04BC 1036 ; WRITE LONG (AH = 08H)
04BD 1037
04BE 3E8D 1038 CMP AL,0A0H
04BF F8 1039 CMC
04C0 C3 1040 RET
04C1 1041
04C2 1042 INIT_DRV_R PROC NEAR
04C3 1043 CALL HD_WAIT_REQ
04C4 7207 1044 JC DI
04C5 E8A702 1045 CALL PORT_0
04C6 268A01 1046 MO. AL,ES:[BX + 1]
04C7 EE 1047 OUT DX,AL
04C8 C3 1048 RET
04C9 1049 INIT_DRV_S ENDP
04CA 1050
04CB 1051 ; SEEK IAH = OCH)
04Cd 1052 DISK..SEEK PROC NEAR
04Ce 1053 CALL CHK_LONG
04Cf 7207 1054 JC G8
04D0 1055 MO. CMD_BLOCK+6,RD_LONG_CMD
04D1 8047 1056 MO. AL,DMA_READ
04D2 E860 1057 JMP SHORT DMA_OPN
04D3 1058 RD_LONG ENDP
04D4 1059
04D5 1060 ; WRITE LONG (AH = 0DH)
04D6 1061
04D7 1062 CMP AL,0A0H
04D8 F8 1063 CMC
04D9 C3 1064 RET
04DA 1065 CHK_LONG ENDP
04DB 1066
04DC 1067 ; SEEK (AH = 08H)
04DD 1068 ; ------ SEND THE BYTE OUT TO THE CONTROLLER
04DE 1069
04E0 1070 DISK_SEEK PROC NEAR
04E1 1071 MO. CMD_BLOCK,SEEK_CMD
04E2 1072 JMP SHORT DMA_OPN
04E3
04E4
04E5
04E6 1073
04E7 1074
04E8 1075
04E9 1076
04EA
04EB
04EC
04ED
04EE 1077
04EF
04F0 1078
04F1
04F2
04F3
04F4
04F5
04F6
04F7
04F8
04F9
04FA
04FB
04FC
04FD
04FE
04FF

Fixed Disk BIOS  A-97
LOC OBJ  LINE  SOURCE
1073  DISK_SEEK  ENDP
1074  
1075  |-----------------------------------------------|
1076  | READ SECTOR BUFFER (AH = 0EH) |
1077  |-----------------------------------------------|
1078  
1079  04F9  1079  RD_BUFF PROC NEAR
1080  04F9 C00642000E  1080  MOV CMD_BLOCK+0,RD_BUFF_CMD
1081  04FE C006460001  1081  MOV CMD_BLOCK+4,1 ; ONLY ONE BLOCK
1082  0503 EB47  1082  MOV AL,DMA_READ
1083  0505 EB3E  1083  JMP SHORT DMA_OPEN
1084  0504 RD_BUFF ENDP
1085  
1086  |-----------------------------------------------|
1087  | WRITE SECTOR BUFFER (AH = 0FH) |
1088  |-----------------------------------------------|
1089  
1090  0507  1090  WR_BUFF PROC NEAR
1091  0507 C00642000F  1091  MOV CMD_BLOCK+0,WR_BUFF_CMD
1092  050C C006460001  1092  MOV CMD_BLOCK+4,1 ; ONLY ONE BLOCK
1093  0511 B048  1093  MOV AL,DMA_WRITE
1094  0513 EB30  1094  JMP SHORT DMA_OPEN
1095  0509 WR_BUFF ENDP
1096  
1097  |-----------------------------------------------|
1098  | TEST DISK READY (AH = 010H) |
1099  |-----------------------------------------------|
1100  
1101  0515  1101  TST_DRV PROC NEAR
1102  0515 C006420000  1102  MOV CMD_BLOCK+0,TST_DRV_CMD
1103  051A EB1A  1103  JMP SHORT DMA_OPEN
1104  0510 TST_DRV ENDP
1105  
1106  |-----------------------------------------------|
1107  | RECALIBRATE (AH = 011H) |
1108  |-----------------------------------------------|
1109  
1110  0510  1110  HDISK_RECAL PROC NEAR
1111  0510 C006420001  1111  MOV CMD_BLOCK+0,RECAL_CMD
1112  0521 EB13  1112  JMP SHORT DMA_OPEN
1113  0518 EB1A  1113  HDISK_RECAL ENDP
1114  
1115  |-----------------------------------------------|
1116  | CONTROLLER RAM DIAGNOSTICS (AH = 012H) |
1117  |-----------------------------------------------|
1118  
1119  0523  1119  RAM_DIAG PROC NEAR
1120  0523 C006420000  1120  MOV CMD_BLOCK+0,RAM_DIAG_CMD
1121  0528 EB0C  1121  JMP SHORT DMA_OPEN
1122  0520 RAM_DIAG ENDP
1123  
1124  |-----------------------------------------------|
1125  | DRIVE DIAGNOSTICS (AH = 013H) |
1126  |-----------------------------------------------|
1127  
1128  052A  1128  CHK_DRV PROC NEAR
1129  052A C006420003  1129  MOV CMD_BLOCK+0,CHK_DRV_CMD
1130  052F EB05  1130  JMP SHORT DMA_OPEN
1131  052A CHK_DRV ENDP
1132  
1133  |-----------------------------------------------|
1134  | CONTROLLER INTERNAL DIAGNOSTICS (AH = 014H) |
1135  |-----------------------------------------------|
1136  
1137  0531  1137  CHNL_DRV PROC NEAR
1138  0531 C006420004  1138  MOV CMD_BLOCK+0,CHNL_DRV_CMD
1139  0533 EB0C  1139  CHNL_DRV ENDP
1140  
1141  |-----------------------------------------------|
1142  | SUPPORT ROUTINES |
1143  |-----------------------------------------------|
1144  
1145  0536  1145  DMA_OPEN:
1146  0536 B002  1146  MOV AL,02H
1147  053B EB1700  1147  CALL COMMAND ; ISSUE THE COMMAND
1148  053B 7221  1148  JC G11
1149  0530 EB16  1149  JMP SHORT G3
1150  

A-98  Fixed Disk BIOS
Fixed Disk BIOS  A-99
LaC OBJ
LINE SOURCE
1227 ;
1228 ; BYTE 2:
1229 ; BITS 7-5 CYLINDER HIGH:
1230 ; BITS 4-0 SECTOR NUMBER:
1231 ;
1232 ; BYTE 3:
1233 ; BITS 7-0 CYLINDER LOW:
1234 ;
1235 ;--------------------------------------------------
1236
05C6 1237 ERROR_CHK PROC NEAR
1238 ASSUME ES:DATA
05C6 0A7H00 1239 MOV AL,DISK_STATUS ; CHECK IF THERE WAS AN ERROR
05F9 0A60 1240 OR AL,AL
05A1 7501 1241 JNZ G21
05A3 C3 1242 RET
05A4 1243 ;----- PERFORM SENSE STATUS
05A4 B94000 1244
05A4 BB4000 1245 MOV AX,DATA
05A7 0EC0 1246 MOV ES,AX ; ESTABLISH SEGMENT
05A9 28C0 1247 SUB AX,AX
05AD BB00 1248 MOV DI,AX
05AD C606420003 1249 MOV CMD_BLOCK+SENSE_CMD
05B2 2AC0 1250 SUB AL,AL
05B4 E8FFF 1251 CALL COMMAND ; ISSUE SENSE STATUS COMMAND
05B7 7223 1252 JC SENSE_ABORT ; CANNOT RECOVER
05B9 B94000 1253 MOV CX,4
05B5 1254
05B5 0BCE00 1255 CALL HD_WAIT_REQ
05B5 7220 1256 JC G24
05C1 E0ADD1 1257 CALL PORT_0
05C4 EC 1258 IN AL,AL
05C9 0C66420003 1259 MOV CMD_BLOCK+SENSE_CMD
05C9 47 1260 INC DI
05CA E8B101 1261 CALL PORT_1
05CD E2ED 1262 LOOP G22
05CF E80000 1263 CALL HD_WAIT_REQ
05D2 7220 1264 JC G24
05D4 E9A01 1265 CALL PORT_0
05D7 EC 1266 IN AL,AL
05D8 A002 1267 TEST AL,2
05DA 740F 1268 JC STAT_ERR
05DC E8ADD1 1269 CALL HD_WAIT_REQ
05DA 7220 1270 JC G24
05EA 006
05EA F4 1271 STC
05E2 C3 1272 RET
05E2 1273 ERROR_CHK ENDP
05E3 1A06 1274 T_0 DW TYPE_0
05E5 2706 1275 T_1 DW TYPE_1
05E7 6A06 1276 T_2 DW TYPE_2
05E9 7706 1277 T_3 DW TYPE_3
05EB 1278
05EB 268A1E4200 1279 MOV BL,ES:HD_ERROR ; GET ERROR BYTE
05F0 0AC3 1280 MOV AL,AL
05F2 260F 1281 AND AL,0FH
05F4 0E330 1282 AND BL,00110000B ; ISOLATE TYPE
05F7 2AFF 1283 SUB DH,DL
05F9 B13 1284 MOV CL,3
05F9 DD8B 1285 SHR DX,CL ; ADJUST
05FD 2E3FA7E305 1286 JMP WORD PTR CS:IBX + OFFSET T_0)
05FD 1287 ASSUME ES:NOTHING
05FA 268A1E4200 1288
0602 1289 TYPE0_TABLE LABEL BYTE
0608 00020402000002 1290 DB 0,BAD_CHKLR,BAD_SEEK,BAD_CHKLR,TIME_OUT,0,BAD_CHKLR
0609 0040 1291 DB 0,BAD_SEEK
060A 0009 1292 DB 0,BAD_CHKLR,BAD_SEEK,BAD_CHKLR,TIME_OUT,0,BAD_CHKLR
0609 0040 1293 DB 0,BAD_SEEK
060B 1294 TYPE0_LEN EQU $-TYPE0_TABLE
060B 0100200004 1295 DB BAD_ECC,BAD_ECC,BAD_ADDR_MARK,0,RECORD_NOT_FND
0610 400499110B 1296 DB BAD_SEEK,0,0,DATACORRECTED,BAD_TRACK
061A 1300 1297 DB BAD_CHKLR,BAD_CHKLR,TIME_OUT,0
0615 1302 TYPE2_TABLE LABEL BYTE
0615 C302 1298 DB BAD_CHKLR,BAD_CHKLR,TIME_OUT,0
0615 0102 1303 DB BAD_CHKLR,BAD_CHKLR,TIME_OUT,0

A-100  Fixed Disk BIOS
0002 1304  TYPE2_LEN EQU $-TYPE2_TABLE
0017 1309  TYPE3_TABLE LABEL BYTE
0017 202010 1306  DB BAD_CNTL,BAD_CNTL,BAD_ECC
0003 1307  TYPE2_LEN EQU $-TYPE2_TABLE
1308 1309  I----- TYPE 0 ERROR
1310 1311  TYPE 0:
061A 1312  MOV BX,OFFSET TYPE2_TABLE
061D 3C09 1313  CMP AL,TYPE2_LEN  ; CHECK IF ERROR IS DEFINED
061F 7E55 1314  JAE UNDEF_ERR_L
0621 2E07 1315  XLAT CS:TYPE2_TABLE  ; TABLE LOOKUP
0623 AE7400 1316  MOV DISK_STATUS,AL  ; SET ERROR CODE
0626 C3 1317  RET
1318 1319  I----- TYPE 1 ERROR
1320 1321  TYPE 1:
0627 BB0006 1322  MOV BX,OFFSET TYPE2_TABLE
062A 3C08 1323  MOV CX,AX
062C 3C0A 1324  CMP AL,TYPE2_LEN  ; CHECK IF ERROR IS DEFINED
062E 7E55 1325  JAE UNDEF_ERR_L
0630 2E07 1326  XLAT CS:TYPE2_TABLE  ; TABLE LOOKUP
0632 AE7400 1327  MOV DISK_STATUS,AL  ; SET ERROR CODE
0635 801008 1328  AND CL,00H  ; CORRECTED ECC
0638 60F900 1329  CMP CL,00H
063B 750A 1330  JNZ G30
1331 1332  I----- OBTAIN ECC ERROR BURST LENGTH
1333 063D C66420000D 1334  MOV CHD_BLOCK+0,RO_ECC_CMD
0642 2AC0 1335  SUB AL,AL
0644 E808FF 1336  CALL COMMAND
0647 7E1E 1337  JC G30
0649 E83000 1338  CALL HD_WAIT_REQ
064C 7E19 1339  JC G30
064E 602001 1340  CALL PORT 0
0651 7E1C 1341  JM AL,DX
0652 6AC8 1342  MOV CL,AL
0654 E83000 1343  CALL HD_WAIT_REQ
0657 7E0E 1344  JC G30
0659 E81001 1345  CALL PORT 0
065C 7E1C 1346  JM AL,DX
065D A001 1347  TEST AL,01H
065F 7E46 1348  JZ G30
0661 C66740020 1349  MOV DISK_STATUS,BAD_CNTL
0666 F9 1350  STC
0667 750A 1351  G30:
0667 6AC1 1352  MOV AL,CL
0669 C3 1353  RET
1354 1355  I----- TYPE 2 ERROR
1356 066A 1357  TYPE 2:
066A BB1506 1358  MOV BX,OFFSET TYPE2_TABLE
066D 3C08 1359  CMP AL,TYPE2_LEN  ; CHECK IF ERROR IS DEFINED
066F 7E13 1360  JC UNDEF_ERR_L
0671 2E07 1361  XLAT CS:TYPE2_TABLE  ; TABLE LOOKUP
0673 AE7400 1362  MOV DISK_STATUS,AL  ; SET ERROR CODE
0676 C3 1363  RET
1364 1365  I----- TYPE 3 ERROR
1366 0677 1367  TYPE 3:
0677 BB1706 1368  MOV BX,OFFSET TYPE3_TABLE
067A 3C08 1369  CMP AL,TYPE3_LEN  ; CHECK IF ERROR IS DEFINED
067C 7E13 1370  JC UNDEF_ERR_L
067E 2E07 1371  XLAT CS:TYPE3_TABLE
0680 AE7400 1372  MOV DISK_STATUS,AL
0683 C3 1373  RET
1374 1375  UNDEF_ERR_L:
0684 C66740000D 1376  MOV DISK_STATUS,UNDEF_ERR
0689 C3 1377  RET
1378 1379  HD_WAIT_REQ PROC NEAR
068A 81 1380  PUSH CX
Appendix A: Fixed Disk BIOS
HD_SETUP PROC NEAR

PUSH AX
PUSH AL
MOV AL,CMD_BLOCK+4
CMP AL,81H ; BLOCK COUNT OUT OF RANGE
JNC J1
INC CH ; CARRY MEANS HIGH 4 BITS MUST BE INC
J33:
PUSH AX ; SAVE START ADDRESS
MOV AL,OFFH
PUSH AX
MOV AX,516D ; ONE BLOCK (512) PLUS 4 BYTES ECC
PUSH BX
MOV AX,S16D ; ONE BLOCK (512) PLUS 4 BYTES ECC

J20:
MOV AX,ES
SHL AX,1 ; MULTIPLY BY 512 BYTES PER SECTOR
DEC AX
AND AL,OFH
MOV AX,OFFH
OUT DMA_HIGH,AL ; OUTPUT THE HIGH 4 BITS TO PAGE REG
J----- DETERMINE COUNT

J33:
J----- HANDLE READ AND WRITE LONG (5160 BYTE BLOCKS)

A-102 Fixed Disk BIOS
WAIT_INT PROC NEAR

WAIT_INT
STI ; TURN ON INTERRUPTS

PUSH BX ; PRESERVE REGISTERS

PUSH CX

PUSH DS

ASSUME DS:NULL

SUB AX,AX ; Establish segment

LES SI,HF_TBL_VEC

ASSUME DS:DATA

716 1F
PUSH DS

716 1F
RET ; RETURN TO CALLER. CFLAGS SET BY ABOVE IF ERROR

Next function

; Wait for interrupt

1800 1873

; Standard time out

; Format drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive

; Check drive
LOC OBJ | LINE | SOURCE
--- | --- | ---
075C  07 | 1535 | POP ES
075D  59 | 1536 | POP CX
075E  5B | 1537 | POP BX
075F  C3 | 1538 | RET

| 1539 | WAIT_INT | ENDP |

| 1540 | |

| 0760 | 1541 | HD_INT | PROC | NEAR |

| 0760  58 | 1542 | PUSH AX |

| 0761  B020 | 1543 | MOV AL,E81I |; END OF INTERRUPT |

| 0763  E620 | 1544 | OUT INT_CTL_PORT,AL |

| 0765  B007 | 1545 | MOV AL,07H |; SET DMA MODE TO DISABLE |

| 0767  E60A | 1546 | OUT DMA=10,AL |

| 0769  E621 | 1547 | IN AL,021H |

| 076B  0C20 | 1548 | OR AL,020H |

| 076D  E621 | 1549 | OUT 021H,AL |

| 076F  58 | 1550 | POP AX |

| 0770  CF | 1551 | INRET |

| 1552 | HD_INT | ENDP |

| 1553 | |

| 0771 | 1554 | PORT_0 | PROC | NEAR |

| 0771  UA2003 | 1555 | MOV DX,HF_PORT |; BASE VALUE |

| 0774  58 | 1556 | PUSH AX |

| 0776  2A4C | 1557 | SUB AH,AX |

| 0777  A07700 | 1558 | MOV AL,PORT_OFF |; ADD IN THE OFFSET |

| 077A  0300 | 1559 | ADD DX,AX |

| 077C  58 | 1560 | POP AX |

| 077D  C3 | 1561 | RET |

| 1562 | PORT_0 | ENDP |

| 1563 | |

| 077E | 1564 | PORT_1 | PROC | NEAR |

| 077E  E800FF | 1565 | CALL PORT_0 |

| 0781  42 | 1566 | INC DX |; INCREMENT TO PORT ONE |

| 0782  C3 | 1567 | RET |

| 1568 | PORT_1 | ENDP |

| 1569 | |

| 0783 | 1570 | PORT_2 | PROC | NEAR |

| 0783  E800FF | 1571 | CALL PORT_1 |

| 0786  42 | 1572 | INC DX |; INCREMENT TO PORT TWO |

| 0787  C3 | 1573 | RET |

| 1574 | PORT_2 | ENDP |

| 1575 | |

| 0788 | 1576 | PORT_3 | PROC | NEAR |

| 0788  E800FF | 1577 | CALL PORT_2 |

| 0796  42 | 1578 | INC DX |; INCREMENT TO PORT THREE |

| 0797  C3 | 1579 | RET |

| 1580 | PORT_3 | ENDP |

| 1581 | |

| 0798 | 1582 | | SUPPORTED |

| 079D  08 | 1583 | CALL PORT_3 |

| 07A0  8A901000 | 1584 | MOV AH,CMND_BLOCK+1 |

| 07A0  80E420 | 1585 | AND AH,0D100000B |; DRIVE 0 OR 1 |

| 07A2  7504 | 1586 | JSZ SW2_AND |

| 07A6  D0E0 | 1587 | SHR AL,1 |; ADJUST |

| 07A7  D0F8 | 1588 | SHR AL,1 |

| 07A8  1589 | SW2_AND: |

| 07A9  2403 | 1590 | AND AL,011B |; ISOLATE |

| 07AA  B104 | 1591 | MOV CL,4 |

A-104  Fixed Disk BIOS
<table>
<thead>
<tr>
<th>LOC OBJ</th>
<th>LINE</th>
<th>OBJ</th>
<th>CODE</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>07AC D2E0</td>
<td>1612</td>
<td>SHL</td>
<td>AL, CL</td>
<td>1 ADJUST</td>
</tr>
<tr>
<td>07AE DAE4</td>
<td>1613</td>
<td>SUB</td>
<td>AH, AH</td>
<td></td>
</tr>
<tr>
<td>07B0 C3</td>
<td>1614</td>
<td>RET</td>
<td></td>
<td></td>
</tr>
<tr>
<td>07B1</td>
<td>1615</td>
<td>SUB</td>
<td>AH, AH</td>
<td></td>
</tr>
<tr>
<td>07B1 F9</td>
<td>1616</td>
<td>STC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>07B2 C3</td>
<td>1617</td>
<td>RET</td>
<td></td>
<td></td>
</tr>
<tr>
<td>07B3 30302030</td>
<td>1618</td>
<td>SWZ_OFFS</td>
<td>END</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>07B5 30302030</td>
<td>1619</td>
<td>RET</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>07B6</td>
<td>1620</td>
<td>DB</td>
<td>'08/16/82'</td>
<td>1 RELEASE MARKER</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>07B6</td>
<td>1621</td>
<td>END_ADDRESS</td>
<td>LABEL</td>
<td>BYTE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>07B6</td>
<td>1622</td>
<td>CODE</td>
<td>ENDS</td>
<td></td>
</tr>
<tr>
<td>07B6</td>
<td>1623</td>
<td>ENDS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>07B6</td>
<td>1624</td>
<td>END</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Appendix A**
Notes:
### 8088 Register Model

<table>
<thead>
<tr>
<th>AX:</th>
<th>AH</th>
<th>AL</th>
</tr>
</thead>
<tbody>
<tr>
<td>BX:</td>
<td>BH</td>
<td>BL</td>
</tr>
<tr>
<td>CX:</td>
<td>CH</td>
<td>CL</td>
</tr>
<tr>
<td>DX:</td>
<td>DH</td>
<td>DL</td>
</tr>
</tbody>
</table>

- **AX**: Accumulator
- **BX**: Base
- **CX**: Count
- **DX**: Data

<table>
<thead>
<tr>
<th>SP</th>
<th>Stack Pointer</th>
</tr>
</thead>
<tbody>
<tr>
<td>BP</td>
<td>Base Pointer</td>
</tr>
<tr>
<td>SI</td>
<td>Source Index</td>
</tr>
<tr>
<td>DI</td>
<td>Destination Index</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IP</th>
<th>Instruction Pointer</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>FLAGS</th>
<th>FLAGSL</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS</td>
<td>Code Segment</td>
</tr>
<tr>
<td>DS</td>
<td>Data Segment</td>
</tr>
<tr>
<td>SS</td>
<td>Stack Segment</td>
</tr>
<tr>
<td>ES</td>
<td>Extra Segment</td>
</tr>
</tbody>
</table>

### Instructions which reference the flag register file as a 16-bit object

Instructions that reference the flag register file 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

### Flags

- **AF**: Auxiliary Carry - BCD
- **CF**: Carry Flag
- **PF**: Parity Flag
- **SF**: Sign Flag
- **ZF**: Zero Flag

```
8080 Flags
```

- **DF**: Direction Flag (Strings)
- **IF**: Interrupt Enable Flag
- **OF**: Overflow Flag (CF ⊕ SF)
- **TF**: Trap - Single Step Flag

```
8088 Flags
```
### 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

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

#### r/m | Operand Address
---|-------------------
000 | (BX) + (SI) + DISP  
001 | (BX) + (DI) + DISP  
010 | (BP) + (SI) + DISP  
011 | (BP) + (DI) + DISP  
100 | (SI) + DISP  
101 | (DI) + DISP  
110 | (BP) + DISP*  
111 | (BX) + DISP  

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

Logical Memory Space

Displacement

MSB
LSB
BYTE

Word

Extra Data Segment

Data Segment

Stack Segment

Code Segment

7
0
FFFFFH

64KB

Selected Segment Register

CS
SS
DS
ES

0000
0000
0000
0000

Adder

Physical Address Latch

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>

B-4 8088 Instruction Reference
Data Transfer

**MOV** = Move
Register/memory to/from register

\[
\begin{array}{cccc}
1 & 0 & 0 & 0 1 0 d w \\
\end{array}
\]
mod reg r/m

Immediate to register/memory

\[
\begin{array}{cccc}
1 & 1 & 0 & 0 0 1 1 w \\
\end{array}
\]
mod 0 0 0 r/m

\[
\begin{array}{c}
data \\
data if w=1
\end{array}
\]

Immediate to register

\[
\begin{array}{c}
1 & 0 & 1 & 1 w \ reg \\
\end{array}
\]
data

\[
\begin{array}{c}
data if w=1
\end{array}
\]

Memory to accumulator

\[
\begin{array}{cc}
1 & 0 & 1 & 0 0 0 0 w \\
\end{array}
\]
addr-low

\[
\begin{array}{c}
addr-high
\end{array}
\]

Accumulator to memory

\[
\begin{array}{cc}
1 & 0 & 1 & 0 0 0 1 w \\
\end{array}
\]
addr-low

\[
\begin{array}{c}
addr-high
\end{array}
\]

Register/memory to segment register

\[
\begin{array}{ccc}
1 & 0 & 0 0 1 1 1 0 \\
\end{array}
\]
mod 0 reg r/m

Segment register to register/memory

\[
\begin{array}{ccc}
1 & 0 & 0 0 1 1 0 0 \\
\end{array}
\]
mod 0 reg r/m

**PUSH** = Push
Register/memory

\[
\begin{array}{cccc}
1 & 1 & 1 & 1 1 1 1 1 \\
\end{array}
\]
mod 1 1 0 r/m

Register

\[
\begin{array}{c}
0 & 1 & 0 & 1 0 \ reg \\
\end{array}
\]

Segment register

\[
\begin{array}{c}
0 & 0 & 0 \ reg 1 1 0 \\
\end{array}
\]

**POP** = Pop
Register/memory

\[
\begin{array}{cccc}
1 & 0 & 0 0 1 1 1 1 \\
\end{array}
\]
mod 0 0 0 r/m

Register

\[
\begin{array}{c}
0 & 1 & 0 & 1 1 \ reg \\
\end{array}
\]

Segment register

\[
\begin{array}{c}
0 & 0 & 0 \ reg 1 1 1 \\
\end{array}
\]
### B-6  8088 Instruction Reference

**XCHG** = Exchange Register/memory with register

```
<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1 1 w</td>
</tr>
</tbody>
</table>
```

Register with accumulator

```
<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>reg</td>
</tr>
</tbody>
</table>
```

**IN** = Input to AL/AX from Fixed port

```
<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1 0 w</td>
</tr>
</tbody>
</table>
```

Variable port (DX)

```
<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1 0 w</td>
</tr>
</tbody>
</table>
```

**OUT** = Output from AL/AX to Fixed port

```
<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1 1 w</td>
</tr>
</tbody>
</table>
```

Variable port (DX)

```
<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1 0 w</td>
</tr>
</tbody>
</table>
```

**XLAT** = Translate byte to AL

```
<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0 1 1 1</td>
</tr>
</tbody>
</table>
```

**LEA** = Load EA to register

```
<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1 1 0 1</td>
</tr>
</tbody>
</table>
```

**LDS** = Load pointer to DS

```
<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1 0 1 1</td>
</tr>
</tbody>
</table>
```

**LES** = Load pointer to ES

```
<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1 0 0 0</td>
</tr>
</tbody>
</table>
```

**LAHF** = Load AH with flags

```
<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1 1 1 1</td>
</tr>
</tbody>
</table>
```

**SAHF** = Store AH into flags

```
<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1 1 1 0</td>
</tr>
</tbody>
</table>
```

**PUSHF** = Push flags

```
<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1 1 0 0</td>
</tr>
</tbody>
</table>
```

**POPF** = Pop flags

```
<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1 1 0 1</td>
</tr>
</tbody>
</table>
```
### Arithmetic

**ADD** = Add  
Register/memory with register to either

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

Immediate to register/memory

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

Immediate to accumulator

| 0 0 0 0 0 1 0 w | data | data if w=1 |

**ADC** = Add with carry  
Register/memory and register to either

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

Immediate to register/memory

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

Immediate to accumulator

| 0 0 0 1 0 1 0 w | data | data if w=1 |

**INC** = Increment  
Register/memory

| 1 1 1 1 1 1 w | mod 0 0 0 r/m |

Register

| 0 1 0 0 0 reg |

**AAA** = ASCII adjust for add

| 0 0 1 1 0 1 1 1 |

**DAA** = Decimal adjust for add

| 0 0 1 0 0 1 1 1 |

**SUB** = Subtract  
Register/memory and register to either

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

Immediate from register/memory

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

Immediate from accumulator

| 0 0 1 0 1 1 0 w | data | data if w=1 |
**SBB** = Subtract with borrow
Register/memory and register to either

<table>
<thead>
<tr>
<th>0 0 0 1 1 0 d w</th>
<th>mod</th>
<th>reg</th>
<th>r/m</th>
</tr>
</thead>
</table>

Immediate from register/memory

<table>
<thead>
<tr>
<th>1 0 0 0 0 0 s w</th>
<th>mod</th>
<th>0 1 1 r/m</th>
<th>data</th>
<th>data if s:w=01</th>
</tr>
</thead>
</table>

Immediate from accumulator

<table>
<thead>
<tr>
<th>0 0 0 1 1 1 0 w</th>
<th>data</th>
<th>data if w=1</th>
</tr>
</thead>
</table>

**DEC** = Decrement
Register/memory

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

Register

<table>
<thead>
<tr>
<th>0 1 0 0 1 reg</th>
</tr>
</thead>
</table>

**NEG** = Change sign

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

**CMP** = Compare
Register/memory and register

<table>
<thead>
<tr>
<th>0 0 1 1 1 0 d w</th>
<th>mod</th>
<th>reg</th>
<th>r/m</th>
</tr>
</thead>
</table>

Immediate with register/memory

<table>
<thead>
<tr>
<th>1 0 0 0 0 0 s w</th>
<th>mod</th>
<th>1 1 1 r/m</th>
<th>data</th>
<th>data if s:w=01</th>
</tr>
</thead>
</table>

Immediate with accumulator

<table>
<thead>
<tr>
<th>0 0 1 1 1 1 0 w</th>
<th>data</th>
<th>data if w=1</th>
</tr>
</thead>
</table>

**AAS** = ASCII adjust for subtract

<table>
<thead>
<tr>
<th>0 0 1 1 1 1 1 1</th>
</tr>
</thead>
</table>

**DAS** = Decimal adjust for subtract

<table>
<thead>
<tr>
<th>0 0 1 0 1 1 1 1</th>
</tr>
</thead>
</table>

**MUL** = Multiply (unsigned)

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

**IMUL** = Integer multiply (signed)

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

**AAM** = ASCII adjust for multiply

<table>
<thead>
<tr>
<th>1 1 0 1 0 1 0 0</th>
<th>0 0 0 0 1 0 1 0</th>
</tr>
</thead>
</table>

**DIV** = Divide (unsigned)

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

B-8 8088 Instruction Reference
**DIV** = Integer divide (signed)

\[
\begin{array}{c|c}
1 & 1 & 1 & 1 & 0 & 1 & 1 & w & \text{mod} & 1 & 1 & 1 & r/m
\end{array}
\]

**AAD** = ASCII adjust for divide

\[
\begin{array}{c|c}
1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0
\end{array}
\]

**CBW** = Convert byte to word

\[
\begin{array}{c|c}
1 & 0 & 0 & 1 & 1 & 0 & 0 & 0
\end{array}
\]

**CWD** = Convert word to double word

\[
\begin{array}{c|c}
1 & 0 & 0 & 1 & 1 & 0 & 0 & 1
\end{array}
\]

**Logic**

**NOT** = Invert

\[
\begin{array}{c|c}
1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & w & \text{mod} & 0 & 1 & 0 & r/m
\end{array}
\]

**SHL/SAL** = Shift logical/arithmetic left

\[
\begin{array}{c|c}
1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & v & w & \text{mod} & 1 & 0 & 0 & r/m
\end{array}
\]

**SHR** = Shift logical right

\[
\begin{array}{c|c}
1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & v & w & \text{mod} & 1 & 0 & 1 & r/m
\end{array}
\]

**SAR** = Shift arithmetic right

\[
\begin{array}{c|c}
1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & v & w & \text{mod} & 1 & 1 & 1 & r/m
\end{array}
\]

**ROL** = Rotate left

\[
\begin{array}{c|c}
1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & v & w & \text{mod} & 0 & 0 & 0 & r/m
\end{array}
\]

**ROR** = Rotate right

\[
\begin{array}{c|c}
1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & v & w & \text{mod} & 0 & 0 & 1 & r/m
\end{array}
\]

**RCL** = Rotate through carry left

\[
\begin{array}{c|c}
1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & v & w & \text{mod} & 0 & 1 & 0 & r/m
\end{array}
\]

**RCR** = Rotate through carry right

\[
\begin{array}{c|c}
1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & v & w & \text{mod} & 0 & 1 & 1 & r/m
\end{array}
\]

**AND** = And

Register/memory and register to either

\[
\begin{array}{c|c|c|c}
0 & 0 & 1 & 0 & 0 & 0 & d & w & \text{mod} & \text{reg} & r/m
\end{array}
\]

Immediate to register/memory

\[
\begin{array}{c|c|c|c|c|c}
1 & 0 & 0 & 0 & 0 & 0 & 0 & w & \text{mod} & 1 & 0 & 0 & r/m & \text{data} & \text{data if } w=1
\end{array}
\]

Immediate to accumulator

\[
\begin{array}{c|c|c|c|c|c}
0 & 0 & 1 & 0 & 0 & 1 & 0 & w & \text{data} & \text{data if } w=1
\end{array}
\]
**TEST** = And function to flags, no result
Register/memory and register

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

Immediate data and register/memory

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

Immediate data and accumulator

| 1 0 1 0 1 0 0 w | data | data if w=1 |

**OR** = OR
Register/memory and register to either

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

Immediate to register/memory

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

Immediate to accumulator

| 0 0 0 0 1 1 0 w | data | data if w=1 |

**XOR** = Exclusive or
Register/memory and register to either

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

Immediate to register/memory

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

Immediate to accumulator

| 0 0 1 1 0 1 0 w | data | data if w=1 |

---

**String Manipulation**

**REP** = Repeat

| 1 1 1 1 0 0 1 z |

**MOVS** = Move String

| 1 0 1 0 0 1 0 w |

**CMPS** = Compare String

| 1 0 1 0 0 1 1 w |

**SCAS** = Scan String

| 1 0 1 0 1 1 1 w |

---

**B-10 Instruction Reference**
**LODS** = Load String

```
1 0 1 0 1 1 0 w
```

**STOS** = Store String

```
1 0 1 0 1 0 1 w
```

**Control Transfer**

**CALL** = Call
- Direct within segment
  ```
  1 1 1 0 1 0 0 0
  ```
  disp-low   disp-high
- Indirect within segment
  ```
  1 1 1 1 1 1 1 1
  ```
  mod 0 1 0 r/m
- Direct intersegment
  ```
  1 0 0 1 1 0 1 0
  ```
  offset-low  offset-high
  seg-low     seg-high

**JMP** = Unconditional Jump
- Direct within segment
  ```
  1 1 1 0 1 0 0 1
  ```
  disp-low   disp-high
- Direct within segment-short
  ```
  1 1 1 0 1 0 1 1
  ```
  disp
- Indirect within segment
  ```
  1 1 1 1 1 1 1 1
  ```
  mod 1 0 0 r/m
- Direct intersegment
  ```
  1 1 1 0 1 0 1 0
  ```
  offset-low  offset-high
  seg-low     seg-high
- Indirect intersegment
  ```
  1 1 1 1 1 1 1 1
  ```
  mod 1 0 1 r/m
RET = Return from CALL
Within segment

```
1 1 0 0 0 0 1 1
```

Within segment adding immediate to SP

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

Intersegment

```
1 1 0 0 1 0 1 1
```

Intersegment, adding immediate to SP

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

JE/JZ = Jump on equal/zero

```
0 1 1 1 1 0 1 0 0  disp
```

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

```
0 1 1 1 1 1 0 0  disp
```

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

```
0 1 1 1 1 1 1 0  disp
```

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

```
0 1 1 1 1 0 0 1 0  disp
```

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

```
0 1 1 1 1 0 1 1 0  disp
```

JP/JPE = Jump on parity/parity even

```
0 1 1 1 1 0 1 0 1 0  disp
```

JO = Jump on overflow

```
0 1 1 1 1 0 0 0 0  disp
```

JS = Jump on sign

```
0 1 1 1 1 1 0 0 0  disp
```

JNE/JNZ = Jump on not equal/not zero

```
0 1 1 1 0 1 0 1 0  disp
```

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

```
0 1 1 1 1 1 0 1  disp
```

JNE/JNZ = Jump on not equal/not zero

```
0 1 1 1 0 1 0 1  disp
```

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

```
0 1 1 1 1 1 0 1  disp
```
<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
<th>Opcode</th>
</tr>
</thead>
<tbody>
<tr>
<td>JNLE/JG</td>
<td>Jump on not less or equal/greater</td>
<td>11111111 disp</td>
</tr>
<tr>
<td>JNB/JAE</td>
<td>Jump on not below/above or equal</td>
<td>1110011 disp</td>
</tr>
<tr>
<td>JNBE/JA</td>
<td>Jump on not below or equal/above</td>
<td>1110111 disp</td>
</tr>
<tr>
<td>JNP/JPO</td>
<td>Jump on not parity/parity odd</td>
<td>1111011 disp</td>
</tr>
<tr>
<td>JNO</td>
<td>Jump on not overflow</td>
<td>1110001 disp</td>
</tr>
<tr>
<td>JNS</td>
<td>Jump on not sign</td>
<td>1111001 disp</td>
</tr>
<tr>
<td>LOOP</td>
<td>Loop CX times</td>
<td>11100010 disp</td>
</tr>
<tr>
<td>LOOPZ/LOOPE</td>
<td>Loop while zero/equal</td>
<td>1110001 disp</td>
</tr>
<tr>
<td>LOOPNZ/LOOPNE</td>
<td>Loop while not zero/not equal</td>
<td>1110000 disp</td>
</tr>
<tr>
<td>JCXZ</td>
<td>Jump on CX zero</td>
<td>1110001 disp</td>
</tr>
</tbody>
</table>
## 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 0F) = 1</td>
<td>“less” or “not greater or equal”</td>
</tr>
<tr>
<td>JLE or JNG</td>
<td>((SF xor 0F) 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 0F) = 0</td>
<td>“not less” or “greater or equal”</td>
</tr>
<tr>
<td>JNLE or JG</td>
<td>((SF xor 0F) 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 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
```

---

8088 Instruction Reference
CLC = Clear carry
1 1 1 1 1 0 0 0

CMC = Complement carry
1 1 1 1 0 1 0 1

CLD = Clear direction
1 1 1 1 1 1 0 0

CLI = Clear interrupt
1 1 1 1 1 0 1 0

HLT = Halt
1 1 1 1 0 1 0 0

LOCK = Bus lock prefix
1 1 1 1 0 0 0 0

Processor Control

STC = Set carry
1 1 1 1 1 0 0 1

NOP = No operation
1 0 0 1 0 0 0 0

STD = Set direction
1 1 1 1 1 1 0 1

STI = Set interrupt
1 1 1 1 1 0 1 1

WAIT = Wait
1 0 0 1 1 0 1 1

ESC = Escape (to external device)
1 1 0 1 1 x x x mod x x x r/m

Footnotes:
if \( d = 1 \) then "to"; if \( d = 0 \) then "from"
if \( w = 1 \) then word instruction; if \( w = 0 \) then byte instruction
if \( s:w = 01 \) then 16 bits of immediate data from the operand
if \( s:w = 11 \) then an immediate data byte is 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
### 8088 Instruction Set Matrix

<table>
<thead>
<tr>
<th>LO</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</td>
<td>ADD w,ia</td>
<td>PUSH</td>
<td>POP</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</td>
<td>ADC w,i</td>
<td>PUSH</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</td>
<td>AND w,i</td>
<td>SEG</td>
<td>=ES</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</td>
<td>XOR w,i</td>
<td>SEG</td>
<td>=SS</td>
</tr>
<tr>
<td>4</td>
<td>INC AX</td>
<td>INC CX</td>
<td>INC w,t,r/m</td>
<td>INC w,t,r/m</td>
<td>INC SP</td>
<td>INC BP</td>
<td>INC</td>
<td>SI</td>
</tr>
<tr>
<td>5</td>
<td>PUSH AX</td>
<td>PUSH CX</td>
<td>PUSH w,t,r/m</td>
<td>PUSH w,t,r/m</td>
<td>PUSH SP</td>
<td>PUSH BP</td>
<td>PUSH</td>
<td>DI</td>
</tr>
<tr>
<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>JO JNO</td>
<td>JB/ JNAE</td>
<td>JNB/ JE/ JZ</td>
<td>JNB/ JNE/ JNZ</td>
<td>JBE/ JNE/ JBE/</td>
<td>JNB/ JNE/ JBE/ JA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</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>9</td>
<td>NOP 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>
<td>DI</td>
</tr>
<tr>
<td>A</td>
<td>MOV AL m</td>
<td>MOV AX m</td>
<td>MOV b,f,r/m</td>
<td>MOV b,t,r/m</td>
<td>MOVs b</td>
<td>MOVs w</td>
<td>CMPS b</td>
<td>CMPS w</td>
</tr>
<tr>
<td>B</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>
<td>MOV</td>
</tr>
<tr>
<td>C</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>MOV</td>
</tr>
<tr>
<td>D</td>
<td>Shift b v</td>
<td>Shift b,v</td>
<td>Shift w,v</td>
<td>AAM</td>
<td>AAD</td>
<td>XLAT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>LOOPNZ/ LOOPOE</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>F</td>
<td>LOCK</td>
<td>REP z</td>
<td>HLT</td>
<td>CMC</td>
<td>Grp 1 b,r/m</td>
<td>Grp 1 w,r/m</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **b** = byte operation
- **d** = direct
- **f** = from CPU reg
- **i** = immediate
- **ia** = immed. to accum.
- **id** = indirect
- **is** = immed. byte, sign ext.
- **l** = long ie. intersegment
- **m** = memory
- **r/m** = EA is second byte
- **si** = short intrasegment
- **sr** = segment register
- **t** = to CPU reg
- **v** = variable
- **w** = word operation
- **z** = zero

---

**B-16 8088 Instruction Reference**
### 8088 Instruction Set Matrix

<table>
<thead>
<tr>
<th>LO</th>
<th>HI</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>OR</td>
<td>b,t,r/m</td>
<td>w,f,r/m</td>
<td>OR</td>
<td>b,t,r/m</td>
<td>OR</td>
<td>w,t,r/m</td>
<td>OR</td>
<td>w,i</td>
</tr>
<tr>
<td>1</td>
<td>SBB</td>
<td>b,t,r/m</td>
<td>w,f,r/m</td>
<td>SBB</td>
<td>b,t,r/m</td>
<td>SBB</td>
<td>w,t,r/m</td>
<td>SBB</td>
<td>w,i</td>
</tr>
<tr>
<td>2</td>
<td>SUB</td>
<td>b,t,r/m</td>
<td>w,f,r/m</td>
<td>SUB</td>
<td>b,t,r/m</td>
<td>SUB</td>
<td>w,t,r/m</td>
<td>SUB</td>
<td>w,i</td>
</tr>
<tr>
<td>3</td>
<td>CMP</td>
<td>b,t,r/m</td>
<td>w,f,r/m</td>
<td>CMP</td>
<td>b,t,r/m</td>
<td>CMP</td>
<td>w,t,r/m</td>
<td>CMP</td>
<td>w,i</td>
</tr>
<tr>
<td>4</td>
<td>DEC</td>
<td>AX</td>
<td>DEC</td>
<td>CX</td>
<td>DEC</td>
<td>BX</td>
<td>DEC</td>
<td>SP</td>
<td>DEC</td>
</tr>
<tr>
<td>5</td>
<td>POP</td>
<td>AX</td>
<td>POP</td>
<td>CX</td>
<td>POP</td>
<td>DX</td>
<td>POP</td>
<td>SP</td>
<td>POP</td>
</tr>
<tr>
<td>6</td>
<td>JNS</td>
<td>JS</td>
<td>JPE</td>
<td>JNS</td>
<td>JP /JPE</td>
<td>JNP/  JPO</td>
<td>JL/  JNGE</td>
<td>JNL/ JG</td>
<td>JLE/ JNG</td>
</tr>
<tr>
<td>7</td>
<td>MOV</td>
<td>b,t,r/m</td>
<td>w,f,r/m</td>
<td>MOV</td>
<td>b,t,r/m</td>
<td>MOV</td>
<td>w,t,r/m</td>
<td>MOV</td>
<td>sr,t,r/m</td>
</tr>
<tr>
<td>8</td>
<td>CBW</td>
<td>CALL</td>
<td>l,d</td>
<td>CALL</td>
<td>WAIT</td>
<td>PUSHF</td>
<td>POPF</td>
<td>SAHF</td>
<td>LAHF</td>
</tr>
<tr>
<td>9</td>
<td>TEST</td>
<td>b,i</td>
<td>TEST</td>
<td>w,i</td>
<td>STOS</td>
<td>b</td>
<td>STOS</td>
<td>w</td>
<td>Lods</td>
</tr>
<tr>
<td>A</td>
<td>MOV</td>
<td>AX</td>
<td>MOV</td>
<td>CX</td>
<td>MOV</td>
<td>AX</td>
<td>MOV</td>
<td>BX</td>
<td>MOV</td>
</tr>
<tr>
<td>B</td>
<td>RET</td>
<td>l</td>
<td>RET</td>
<td>l,(i-SP)</td>
<td>INT</td>
<td>l</td>
<td>INT</td>
<td>Type3</td>
<td>INT</td>
</tr>
<tr>
<td>C</td>
<td>ESC</td>
<td>0</td>
<td>ESC</td>
<td>1</td>
<td>ESC</td>
<td>2</td>
<td>ESC</td>
<td>3</td>
<td>ESC</td>
</tr>
<tr>
<td>D</td>
<td>CALL</td>
<td>d</td>
<td>JMP</td>
<td>d</td>
<td>JMP</td>
<td>l,d</td>
<td>JMP</td>
<td>si,d</td>
<td>IN</td>
</tr>
<tr>
<td>E</td>
<td>CLC</td>
<td>STC</td>
<td>CLI</td>
<td>STI</td>
<td>CLD</td>
<td>STD</td>
<td>Grp 2</td>
<td>b,r/m</td>
<td>Grp 2</td>
</tr>
<tr>
<td>F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Additional Information

- **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 —
<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>MOVS</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>As Text Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Color/Graphics Monitor Adapter</td>
</tr>
<tr>
<td>Hex</td>
<td>Dec</td>
<td>Symbol</td>
</tr>
<tr>
<td>00</td>
<td>0</td>
<td>Blank (Null)</td>
</tr>
<tr>
<td>01</td>
<td>1</td>
<td>😊</td>
</tr>
<tr>
<td>02</td>
<td>2</td>
<td>😞</td>
</tr>
<tr>
<td>03</td>
<td>3</td>
<td>😍</td>
</tr>
<tr>
<td>04</td>
<td>4</td>
<td>🌟</td>
</tr>
<tr>
<td>05</td>
<td>5</td>
<td>💐</td>
</tr>
<tr>
<td>06</td>
<td>6</td>
<td>🌹</td>
</tr>
<tr>
<td>07</td>
<td>7</td>
<td>⭐️</td>
</tr>
<tr>
<td>08</td>
<td>8</td>
<td>Ctrl H, Backspace, Shift Backspace</td>
</tr>
<tr>
<td>09</td>
<td>9</td>
<td>⚫️</td>
</tr>
<tr>
<td>0A</td>
<td>10</td>
<td>⚫️</td>
</tr>
<tr>
<td>0B</td>
<td>11</td>
<td>⚫️</td>
</tr>
<tr>
<td>0C</td>
<td>12</td>
<td>⚫️</td>
</tr>
<tr>
<td>0D</td>
<td>13</td>
<td>⚫️</td>
</tr>
<tr>
<td>0E</td>
<td>14</td>
<td>⚫️</td>
</tr>
<tr>
<td>0F</td>
<td>15</td>
<td>⚫️</td>
</tr>
<tr>
<td>10</td>
<td>16</td>
<td>⚫️</td>
</tr>
<tr>
<td>11</td>
<td>17</td>
<td>⚫️</td>
</tr>
<tr>
<td>12</td>
<td>18</td>
<td>⚫️</td>
</tr>
<tr>
<td>13</td>
<td>19</td>
<td>⚫️</td>
</tr>
<tr>
<td>14</td>
<td>20</td>
<td>⚫️</td>
</tr>
<tr>
<td>15</td>
<td>21</td>
<td>⚫️</td>
</tr>
<tr>
<td>16</td>
<td>22</td>
<td>⚫️</td>
</tr>
<tr>
<td>17</td>
<td>23</td>
<td>⚫️</td>
</tr>
<tr>
<td>Value</td>
<td>As Characters</td>
<td>Color/Graphics Monitor Adapter</td>
</tr>
<tr>
<td>-------</td>
<td>--------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Background</td>
</tr>
<tr>
<td>18  24</td>
<td>Ctrl X</td>
<td>Blue</td>
</tr>
<tr>
<td>19  25</td>
<td>Ctrl Y</td>
<td>Blue</td>
</tr>
<tr>
<td>1A  26</td>
<td>Ctrl Z</td>
<td>Blue</td>
</tr>
<tr>
<td>1B  27</td>
<td>Ctrl [, Esc, Shift Esc, Ctrl Esc</td>
<td>Blue</td>
</tr>
<tr>
<td>1C  28</td>
<td>Ctrl \</td>
<td>Blue</td>
</tr>
<tr>
<td>1D  29</td>
<td>Ctrl ]</td>
<td>Blue</td>
</tr>
<tr>
<td>1E  30</td>
<td>Ctrl 6</td>
<td>Blue</td>
</tr>
<tr>
<td>1F  31</td>
<td>Ctrl —</td>
<td>Blue</td>
</tr>
<tr>
<td>20  32</td>
<td>Blank Space</td>
<td>Green</td>
</tr>
<tr>
<td></td>
<td>Space Bar, Shift, Space, Ctrl Space, Alt Space</td>
<td></td>
</tr>
<tr>
<td>21  33</td>
<td>Shift !</td>
<td>Green</td>
</tr>
<tr>
<td>22  34</td>
<td>Shift &quot;</td>
<td>Green</td>
</tr>
<tr>
<td>23  35</td>
<td>Shift #</td>
<td>Green</td>
</tr>
<tr>
<td>24  36</td>
<td>Shift $</td>
<td>Green</td>
</tr>
<tr>
<td>25  37</td>
<td>Shift %</td>
<td>Green</td>
</tr>
<tr>
<td>26  38</td>
<td>Shift &amp;</td>
<td>Green</td>
</tr>
<tr>
<td>27  39</td>
<td>Shift '</td>
<td>Green</td>
</tr>
<tr>
<td>28  40</td>
<td>Shift (</td>
<td>Green</td>
</tr>
<tr>
<td>29  41</td>
<td>Shift )</td>
<td>Green</td>
</tr>
<tr>
<td>2A  42</td>
<td>Shift Note 1</td>
<td>Green</td>
</tr>
<tr>
<td>28  43</td>
<td>Shift +</td>
<td>Green</td>
</tr>
<tr>
<td>2C  44</td>
<td>Shift '</td>
<td>Green</td>
</tr>
<tr>
<td>2D  45</td>
<td>Shift —</td>
<td>Green</td>
</tr>
<tr>
<td>2E  46</td>
<td>Shift .</td>
<td>Green</td>
</tr>
</tbody>
</table>

C-2 Of Characters, Keystrokes, and Colors
<table>
<thead>
<tr>
<th>Hex</th>
<th>Dec</th>
<th>Symbol</th>
<th>As Characters</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></td>
<td></td>
<td>Green</td>
<td>High Intensity</td>
</tr>
<tr>
<td>30</td>
<td>48</td>
<td>0</td>
<td>Cyan</td>
<td>0</td>
<td>Note 3</td>
<td>Black</td>
<td>Normal</td>
</tr>
<tr>
<td>31</td>
<td>49</td>
<td>1</td>
<td>Cyan</td>
<td>1</td>
<td>Note 3</td>
<td>Blue</td>
<td>Underline</td>
</tr>
<tr>
<td>32</td>
<td>50</td>
<td>2</td>
<td>Cyan</td>
<td>2</td>
<td>Note 3</td>
<td>Green</td>
<td>Normal</td>
</tr>
<tr>
<td>33</td>
<td>51</td>
<td>3</td>
<td>Cyan</td>
<td>3</td>
<td>Note 3</td>
<td>Cyan</td>
<td>Normal</td>
</tr>
<tr>
<td>34</td>
<td>52</td>
<td>4</td>
<td>Cyan</td>
<td>4</td>
<td>Note 3</td>
<td>Red</td>
<td>Normal</td>
</tr>
<tr>
<td>35</td>
<td>53</td>
<td>5</td>
<td>Cyan</td>
<td>5</td>
<td>Note 3</td>
<td>Magenta</td>
<td>Normal</td>
</tr>
<tr>
<td>36</td>
<td>54</td>
<td>6</td>
<td>Cyan</td>
<td>6</td>
<td>Note 3</td>
<td>Brown</td>
<td>Normal</td>
</tr>
<tr>
<td>37</td>
<td>55</td>
<td>7</td>
<td>Cyan</td>
<td>7</td>
<td>Note 3</td>
<td>Light Grey</td>
<td>Normal</td>
</tr>
<tr>
<td>38</td>
<td>56</td>
<td>8</td>
<td>Cyan</td>
<td>8</td>
<td>Note 3</td>
<td>Dark Grey</td>
<td>High Intensity</td>
</tr>
<tr>
<td>39</td>
<td>57</td>
<td>9</td>
<td>Cyan</td>
<td>9</td>
<td>Note 3</td>
<td>Light Blue</td>
<td>High Intensity Underline</td>
</tr>
<tr>
<td>3A</td>
<td>58</td>
<td>:</td>
<td>Shift</td>
<td></td>
<td></td>
<td>Cyan</td>
<td>Light Green High Intensity</td>
</tr>
<tr>
<td>3B</td>
<td>59</td>
<td>;</td>
<td>Cyan</td>
<td></td>
<td></td>
<td>Light Cyan</td>
<td>High Intensity</td>
</tr>
<tr>
<td>3C</td>
<td>60</td>
<td>&lt;</td>
<td>Shift</td>
<td></td>
<td></td>
<td>Cyan</td>
<td>Light Red High Intensity</td>
</tr>
<tr>
<td>3D</td>
<td>61</td>
<td>=</td>
<td>Cyan</td>
<td></td>
<td></td>
<td>Light Magenta</td>
<td>High Intensity</td>
</tr>
<tr>
<td>3E</td>
<td>62</td>
<td>&gt;</td>
<td>Shift</td>
<td></td>
<td></td>
<td>Cyan</td>
<td>Yellow High Intensity</td>
</tr>
<tr>
<td>3F</td>
<td>63</td>
<td>?</td>
<td>Shift</td>
<td></td>
<td></td>
<td>Cyan</td>
<td>White High Intensity</td>
</tr>
<tr>
<td>40</td>
<td>64</td>
<td>@</td>
<td>Shift</td>
<td></td>
<td></td>
<td>Red</td>
<td>Black Normal</td>
</tr>
<tr>
<td>41</td>
<td>65</td>
<td>A</td>
<td>Note 4</td>
<td>A</td>
<td>Note 4</td>
<td>Red</td>
<td>Blue Underline</td>
</tr>
<tr>
<td>42</td>
<td>66</td>
<td>B</td>
<td>Note 4</td>
<td>B</td>
<td>Note 4</td>
<td>Red</td>
<td>Green Normal</td>
</tr>
<tr>
<td>43</td>
<td>67</td>
<td>C</td>
<td>Note 4</td>
<td>C</td>
<td>Note 4</td>
<td>Red</td>
<td>Cyan Normal</td>
</tr>
<tr>
<td>44</td>
<td>68</td>
<td>D</td>
<td>Note 4</td>
<td>D</td>
<td>Note 4</td>
<td>Red</td>
<td>Normal</td>
</tr>
<tr>
<td>45</td>
<td>69</td>
<td>E</td>
<td>Note 4</td>
<td>E</td>
<td>Note 4</td>
<td>Red</td>
<td>Magenta Normal</td>
</tr>
<tr>
<td>46</td>
<td>70</td>
<td>F</td>
<td>Note 4</td>
<td>F</td>
<td>Note 4</td>
<td>Red</td>
<td>Brown Normal</td>
</tr>
<tr>
<td>47</td>
<td>71</td>
<td>G</td>
<td>Note 4</td>
<td>G</td>
<td>Note 4</td>
<td>Red</td>
<td>Light Grey Normal</td>
</tr>
<tr>
<td>48</td>
<td>72</td>
<td>H</td>
<td>Note 4</td>
<td>H</td>
<td>Note 4</td>
<td>Red</td>
<td>Dark Grey High Intensity</td>
</tr>
<tr>
<td>49</td>
<td>73</td>
<td>I</td>
<td>Note 4</td>
<td>I</td>
<td>Note 4</td>
<td>Red</td>
<td>Light Blue High Intensity</td>
</tr>
<tr>
<td>4A</td>
<td>74</td>
<td>J</td>
<td>Note 4</td>
<td>J</td>
<td>Note 4</td>
<td>Red</td>
<td>Light Green High Intensity</td>
</tr>
<tr>
<td>Value</td>
<td>As Characters</td>
<td>As Text Attributes</td>
<td>Color/Graphics Monitor Adapter</td>
<td>IBM Monochrome Display Adapter</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>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Background</td>
<td>Foreground</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>
</tr>
<tr>
<td>4B</td>
<td>K</td>
<td>K</td>
<td>Red</td>
<td>Light Cyan</td>
<td>High Intensity</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4C</td>
<td>L</td>
<td>L</td>
<td>Red</td>
<td>Light Red</td>
<td>High Intensity</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4D</td>
<td>M</td>
<td>M</td>
<td>Red</td>
<td>Light Magenta</td>
<td>High Intensity</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4E</td>
<td>N</td>
<td>N</td>
<td>Red</td>
<td>Yellow</td>
<td>High Intensity</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4F</td>
<td>O</td>
<td>O</td>
<td>Red</td>
<td>White</td>
<td>High Intensity</td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>P</td>
<td>P</td>
<td>Magenta</td>
<td>Black</td>
<td>Normal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>51</td>
<td>Q</td>
<td>Q</td>
<td>Magenta</td>
<td>Blue</td>
<td>Underline</td>
<td></td>
<td></td>
</tr>
<tr>
<td>52</td>
<td>R</td>
<td>R</td>
<td>Magenta</td>
<td>Green</td>
<td>Normal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>53</td>
<td>S</td>
<td>S</td>
<td>Magenta</td>
<td>Cyan</td>
<td>Normal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>54</td>
<td>T</td>
<td>T</td>
<td>Magenta</td>
<td>Red</td>
<td>Normal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>55</td>
<td>U</td>
<td>U</td>
<td>Magenta</td>
<td>Magenta</td>
<td>Normal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>56</td>
<td>V</td>
<td>V</td>
<td>Magenta</td>
<td>Brown</td>
<td>Normal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>57</td>
<td>W</td>
<td>W</td>
<td>Magenta</td>
<td>Light Grey</td>
<td>Normal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>58</td>
<td>X</td>
<td>X</td>
<td>Magenta</td>
<td>Dark Grey</td>
<td>High Intensity</td>
<td></td>
<td></td>
</tr>
<tr>
<td>59</td>
<td>Y</td>
<td>Y</td>
<td>Magenta</td>
<td>Light Blue</td>
<td>High Intensity Underline</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5A</td>
<td>Z</td>
<td>Z</td>
<td>Magenta</td>
<td>Light Green</td>
<td>High Intensity</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5B</td>
<td>[</td>
<td>[</td>
<td>Magenta</td>
<td>Light Cyan</td>
<td>High Intensity</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5C</td>
<td>\</td>
<td>\</td>
<td>Magenta</td>
<td>Light Red</td>
<td>High Intensity</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5D</td>
<td>]</td>
<td>]</td>
<td>Magenta</td>
<td>Light Magenta</td>
<td>High Intensity</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5E</td>
<td>^</td>
<td>^</td>
<td>Shift</td>
<td>Yellow</td>
<td>High Intensity</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5F</td>
<td>-</td>
<td>-</td>
<td>Shift</td>
<td>White</td>
<td>High Intensity</td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>\</td>
<td>\</td>
<td>Yellow</td>
<td>Black</td>
<td>Normal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>61</td>
<td>a</td>
<td>a</td>
<td>Note 5</td>
<td>Yellow</td>
<td>Blue Underline</td>
<td></td>
<td></td>
</tr>
<tr>
<td>62</td>
<td>b</td>
<td>b</td>
<td>Note 5</td>
<td>Yellow</td>
<td>Green Normal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>63</td>
<td>c</td>
<td>c</td>
<td>Note 5</td>
<td>Yellow</td>
<td>Cyan Normal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>d</td>
<td>d</td>
<td>Note 5</td>
<td>Yellow</td>
<td>Red Normal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>65</td>
<td>e</td>
<td>e</td>
<td>Note 5</td>
<td>Yellow</td>
<td>Magenta Normal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>66</td>
<td>f</td>
<td>f</td>
<td>Note 5</td>
<td>Yellow</td>
<td>Brown Normal</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

C-4  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 Dec Symbol Keystrokes Modes</td>
<td>Background</td>
<td>Foreground</td>
<td>Adapter</td>
</tr>
<tr>
<td>67 103 g g Note 5</td>
<td>Yellow</td>
<td>Light Grey</td>
<td>Normal</td>
</tr>
<tr>
<td>68 104 h h Note 5</td>
<td>Yellow</td>
<td>Dark Grey</td>
<td>High Intensity</td>
</tr>
<tr>
<td>69 105 i i Note 5</td>
<td>Yellow</td>
<td>Light Blue</td>
<td>High Intensity Underline</td>
</tr>
<tr>
<td>6A 106 j j Note 5</td>
<td>Yellow</td>
<td>Light Green</td>
<td>High Intensity</td>
</tr>
<tr>
<td>6B 107 k k Note 5</td>
<td>Yellow</td>
<td>Light Cyan</td>
<td>High Intensity</td>
</tr>
<tr>
<td>6C 108 l l Note 5</td>
<td>Yellow</td>
<td>Light Red</td>
<td>High Intensity</td>
</tr>
<tr>
<td>6D 109 m m Note 5</td>
<td>Yellow</td>
<td>Light Magenta</td>
<td>High Intensity</td>
</tr>
<tr>
<td>6E 110 n n Note 5</td>
<td>Yellow</td>
<td>Yellow</td>
<td>High Intensity</td>
</tr>
<tr>
<td>6F 111 o o Note 5</td>
<td>Yellow</td>
<td>White</td>
<td>High Intensity</td>
</tr>
<tr>
<td>70 112 p p Note 5</td>
<td>White</td>
<td>Black</td>
<td>Reverse Video</td>
</tr>
<tr>
<td>71 113 q q Note 5</td>
<td>White</td>
<td>Blue</td>
<td>Underline</td>
</tr>
<tr>
<td>72 114 r r Note 5</td>
<td>White</td>
<td>Green</td>
<td>Normal</td>
</tr>
<tr>
<td>73 115 s s Note 5</td>
<td>White</td>
<td>Cyan</td>
<td>Normal</td>
</tr>
<tr>
<td>74 116 f f Note 5</td>
<td>White</td>
<td>Red</td>
<td>Normal</td>
</tr>
<tr>
<td>75 117 u u Note 5</td>
<td>White</td>
<td>Magenta</td>
<td>Normal</td>
</tr>
<tr>
<td>76 118 v v Note 5</td>
<td>White</td>
<td>Brown</td>
<td>Normal</td>
</tr>
<tr>
<td>77 119 w w Note 5</td>
<td>White</td>
<td>Light Grey</td>
<td>Normal</td>
</tr>
<tr>
<td>78 120 x x Note 5</td>
<td>White</td>
<td>Dark Grey</td>
<td>Reverse Video</td>
</tr>
<tr>
<td>79 121 y y Note 5</td>
<td>White</td>
<td>Light Blue</td>
<td>High Intensity Underline</td>
</tr>
<tr>
<td>7A 122 z z Note 5</td>
<td>White</td>
<td>Light Green</td>
<td>High Intensity</td>
</tr>
<tr>
<td>7B 123</td>
<td>Shift</td>
<td>White</td>
<td>Light Cyan</td>
</tr>
<tr>
<td>7C 124</td>
<td>Shift</td>
<td>White</td>
<td>Light Red</td>
</tr>
<tr>
<td>7D 125</td>
<td>Shift</td>
<td>White</td>
<td>Light Magenta</td>
</tr>
<tr>
<td>7E 126</td>
<td>Shift</td>
<td>White</td>
<td>Yellow</td>
</tr>
<tr>
<td>7F 127 Δ Ctrl --</td>
<td>White</td>
<td>White</td>
<td>High Intensity</td>
</tr>
</tbody>
</table>

Of Characters, Keystrokes, and Colors  C-5
<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>80</td>
<td>Ç</td>
<td>Black</td>
<td>Non-Display</td>
</tr>
<tr>
<td>81</td>
<td>ü</td>
<td>Black</td>
<td>Underline</td>
</tr>
<tr>
<td>82</td>
<td>é</td>
<td>Black</td>
<td>Green</td>
</tr>
<tr>
<td>83</td>
<td>ã</td>
<td>Black</td>
<td>Cyan</td>
</tr>
<tr>
<td>84</td>
<td>à</td>
<td>Black</td>
<td>Red</td>
</tr>
<tr>
<td>85</td>
<td>à</td>
<td>Black</td>
<td>Magenta</td>
</tr>
<tr>
<td>86</td>
<td>á</td>
<td>Black</td>
<td>Brown</td>
</tr>
<tr>
<td>87</td>
<td>ç</td>
<td>Black</td>
<td>Light Grey</td>
</tr>
<tr>
<td>88</td>
<td>è</td>
<td>Black</td>
<td>Dark Grey</td>
</tr>
<tr>
<td>89</td>
<td>é</td>
<td>Black</td>
<td>Light Blue</td>
</tr>
<tr>
<td>8A</td>
<td>è</td>
<td>Black</td>
<td>Light Green</td>
</tr>
<tr>
<td>8B</td>
<td>i</td>
<td>Black</td>
<td>Light Cyan</td>
</tr>
<tr>
<td>8C</td>
<td>ì</td>
<td>Black</td>
<td>Light Red</td>
</tr>
<tr>
<td>8D</td>
<td>i</td>
<td>Black</td>
<td>Light Magenta</td>
</tr>
<tr>
<td>8E</td>
<td>A</td>
<td>Black</td>
<td>Yellow</td>
</tr>
<tr>
<td>8F</td>
<td>À</td>
<td>Black</td>
<td>White</td>
</tr>
<tr>
<td>90</td>
<td>É</td>
<td>Blue</td>
<td>Black</td>
</tr>
<tr>
<td>91</td>
<td>æ</td>
<td>Blue</td>
<td>Underline</td>
</tr>
<tr>
<td>92</td>
<td>ÀE</td>
<td>Blue</td>
<td>Green</td>
</tr>
<tr>
<td>93</td>
<td>ô</td>
<td>Blue</td>
<td>Cyan</td>
</tr>
<tr>
<td>94</td>
<td>ö</td>
<td>Blue</td>
<td>Red</td>
</tr>
<tr>
<td>95</td>
<td>ò</td>
<td>Blue</td>
<td>Magenta</td>
</tr>
<tr>
<td>96</td>
<td>û</td>
<td>Blue</td>
<td>Brown</td>
</tr>
<tr>
<td>97</td>
<td>ù</td>
<td>Blue</td>
<td>Light Grey</td>
</tr>
<tr>
<td>98</td>
<td>ý</td>
<td>Blue</td>
<td>Dark Grey</td>
</tr>
<tr>
<td>99</td>
<td>ò</td>
<td>Blue</td>
<td>Light Blue</td>
</tr>
<tr>
<td>9A</td>
<td>ü</td>
<td>Blue</td>
<td>Light Green</td>
</tr>
</tbody>
</table>

80 to FF Hex are Flashing in both Color & IBM Monochrome

C-6 Of Characters, Keystrokes, and Colors
<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>9B</td>
<td>€</td>
<td>Blue Light Cyan</td>
<td>High Intensity</td>
<td></td>
</tr>
<tr>
<td>9C</td>
<td>£</td>
<td>Blue Light Red</td>
<td>High Intensity</td>
<td></td>
</tr>
<tr>
<td>9D</td>
<td>¥</td>
<td>Blue Light Magenta</td>
<td>High Intensity</td>
<td></td>
</tr>
<tr>
<td>9E</td>
<td>Pt</td>
<td>Blue Yellow</td>
<td>High Intensity</td>
<td></td>
</tr>
<tr>
<td>9F</td>
<td>¥</td>
<td>Blue White</td>
<td>High Intensity</td>
<td></td>
</tr>
<tr>
<td>A0</td>
<td>á</td>
<td>Green Black</td>
<td>Normal</td>
<td></td>
</tr>
<tr>
<td>A1</td>
<td>í</td>
<td>Green Blue</td>
<td>Underline</td>
<td></td>
</tr>
<tr>
<td>A2</td>
<td>ó</td>
<td>Green Green</td>
<td>Normal</td>
<td></td>
</tr>
<tr>
<td>A3</td>
<td>ü</td>
<td>Green Cyan</td>
<td>Normal</td>
<td></td>
</tr>
<tr>
<td>A4</td>
<td>Ñ</td>
<td>Green Red</td>
<td>Normal</td>
<td></td>
</tr>
<tr>
<td>A5</td>
<td>Ñ</td>
<td>Green Magenta</td>
<td>Normal</td>
<td></td>
</tr>
<tr>
<td>A6</td>
<td>a</td>
<td>Green Brown</td>
<td>Normal</td>
<td></td>
</tr>
<tr>
<td>A7</td>
<td>o</td>
<td>Green Light Grey</td>
<td>Normal</td>
<td></td>
</tr>
<tr>
<td>A8</td>
<td>ç</td>
<td>Green Dark Grey</td>
<td>High Intensity</td>
<td></td>
</tr>
<tr>
<td>A9</td>
<td>r</td>
<td>Green Light Blue</td>
<td>High Intensity Underline</td>
<td></td>
</tr>
<tr>
<td>AA</td>
<td>ņ</td>
<td>Green Light Green</td>
<td>High Intensity</td>
<td></td>
</tr>
<tr>
<td>AB</td>
<td>≡</td>
<td>Green Light Cyan</td>
<td>High Intensity</td>
<td></td>
</tr>
<tr>
<td>AC</td>
<td>¼</td>
<td>Green Light Red</td>
<td>High Intensity</td>
<td></td>
</tr>
<tr>
<td>AD</td>
<td>i</td>
<td>Green Light Magenta</td>
<td>High Intensity</td>
<td></td>
</tr>
<tr>
<td>AE</td>
<td>&lt;&lt;</td>
<td>Green Yellow</td>
<td>High Intensity</td>
<td></td>
</tr>
<tr>
<td>AF</td>
<td>&gt;&gt;</td>
<td>Green White</td>
<td>High Intensity</td>
<td></td>
</tr>
<tr>
<td>B0</td>
<td>···</td>
<td>Cyan Black</td>
<td>Normal</td>
<td></td>
</tr>
<tr>
<td>B1</td>
<td>☄</td>
<td>Cyan Blue</td>
<td>Underline</td>
<td></td>
</tr>
<tr>
<td>B2</td>
<td>☄</td>
<td>Cyan Green</td>
<td>Normal</td>
<td></td>
</tr>
<tr>
<td>B3</td>
<td>☄</td>
<td>Cyan Cyan</td>
<td>Normal</td>
<td></td>
</tr>
<tr>
<td>B4</td>
<td>☄</td>
<td>Cyan Red</td>
<td>Normal</td>
<td></td>
</tr>
<tr>
<td>B5</td>
<td>☄</td>
<td>Cyan Magenta</td>
<td>Normal</td>
<td></td>
</tr>
<tr>
<td>B6</td>
<td>☄</td>
<td>Cyan Brown</td>
<td>Normal</td>
<td></td>
</tr>
<tr>
<td>Hex</td>
<td>Dec</td>
<td>Symbol</td>
<td>Keystrokes</td>
<td>Modes</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>--------</td>
<td>------------</td>
<td>-------</td>
</tr>
<tr>
<td>B7</td>
<td>183</td>
<td></td>
<td>Alt 183</td>
<td>Note 6</td>
</tr>
<tr>
<td>B8</td>
<td>184</td>
<td></td>
<td>Alt 184</td>
<td>Note 6</td>
</tr>
<tr>
<td>B9</td>
<td>185</td>
<td></td>
<td>Alt 185</td>
<td>Note 6</td>
</tr>
<tr>
<td>BA</td>
<td>186</td>
<td></td>
<td>Alt 186</td>
<td>Note 6</td>
</tr>
<tr>
<td>BB</td>
<td>187</td>
<td></td>
<td>Alt 187</td>
<td>Note 6</td>
</tr>
<tr>
<td>BC</td>
<td>188</td>
<td></td>
<td>Alt 188</td>
<td>Note 6</td>
</tr>
<tr>
<td>BD</td>
<td>189</td>
<td></td>
<td>Alt 189</td>
<td>Note 6</td>
</tr>
<tr>
<td>BE</td>
<td>190</td>
<td></td>
<td>Alt 190</td>
<td>Note 6</td>
</tr>
<tr>
<td>BF</td>
<td>191</td>
<td></td>
<td>Alt 191</td>
<td>Note 6</td>
</tr>
<tr>
<td>C0</td>
<td>192</td>
<td></td>
<td>Alt 192</td>
<td>Note 6</td>
</tr>
<tr>
<td>C1</td>
<td>193</td>
<td></td>
<td>Alt 193</td>
<td>Note 6</td>
</tr>
<tr>
<td>C2</td>
<td>194</td>
<td></td>
<td>Alt 194</td>
<td>Note 6</td>
</tr>
<tr>
<td>C3</td>
<td>195</td>
<td></td>
<td>Alt 195</td>
<td>Note 6</td>
</tr>
<tr>
<td>C4</td>
<td>196</td>
<td></td>
<td>Alt 196</td>
<td>Note 6</td>
</tr>
<tr>
<td>C5</td>
<td>197</td>
<td></td>
<td>Alt 197</td>
<td>Note 6</td>
</tr>
<tr>
<td>C6</td>
<td>198</td>
<td></td>
<td>Alt 198</td>
<td>Note 6</td>
</tr>
<tr>
<td>C7</td>
<td>199</td>
<td></td>
<td>Alt 199</td>
<td>Note 6</td>
</tr>
<tr>
<td>C8</td>
<td>200</td>
<td></td>
<td>Alt 200</td>
<td>Note 6</td>
</tr>
<tr>
<td>C9</td>
<td>201</td>
<td></td>
<td>Alt 201</td>
<td>Note 6</td>
</tr>
<tr>
<td>CA</td>
<td>202</td>
<td></td>
<td>Alt 202</td>
<td>Note 6</td>
</tr>
<tr>
<td>CB</td>
<td>203</td>
<td></td>
<td>Alt 203</td>
<td>Note 6</td>
</tr>
<tr>
<td>CC</td>
<td>204</td>
<td></td>
<td>Alt 204</td>
<td>Note 6</td>
</tr>
<tr>
<td>CD</td>
<td>205</td>
<td></td>
<td>Alt 205</td>
<td>Note 6</td>
</tr>
<tr>
<td>CE</td>
<td>206</td>
<td></td>
<td>Alt 206</td>
<td>Note 6</td>
</tr>
<tr>
<td>CF</td>
<td>207</td>
<td></td>
<td>Alt 207</td>
<td>Note 6</td>
</tr>
<tr>
<td>DO</td>
<td>208</td>
<td></td>
<td>Alt 208</td>
<td>Note 6</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>Dec</td>
<td>Symbol</td>
<td>Keystrokes</td>
</tr>
<tr>
<td>D1</td>
<td>209</td>
<td>Alt 209</td>
<td>Note 6</td>
</tr>
<tr>
<td>D2</td>
<td>210</td>
<td>Alt 210</td>
<td>Note 6</td>
</tr>
<tr>
<td>D3</td>
<td>211</td>
<td>Alt 211</td>
<td>Note 6</td>
</tr>
<tr>
<td>D4</td>
<td>212</td>
<td>Alt 212</td>
<td>Note 6</td>
</tr>
<tr>
<td>D5</td>
<td>213</td>
<td>Alt 213</td>
<td>Note 6</td>
</tr>
<tr>
<td>D6</td>
<td>214</td>
<td>Alt 214</td>
<td>Note 6</td>
</tr>
<tr>
<td>D7</td>
<td>215</td>
<td>Alt 215</td>
<td>Note 6</td>
</tr>
<tr>
<td>D8</td>
<td>216</td>
<td>Alt 216</td>
<td>Note 6</td>
</tr>
<tr>
<td>D9</td>
<td>217</td>
<td>Alt 217</td>
<td>Note 6</td>
</tr>
<tr>
<td>DA</td>
<td>218</td>
<td>Alt 218</td>
<td>Note 6</td>
</tr>
<tr>
<td>DB</td>
<td>219</td>
<td>Alt 219</td>
<td>Note 6</td>
</tr>
<tr>
<td>DC</td>
<td>220</td>
<td>Alt 220</td>
<td>Note 6</td>
</tr>
<tr>
<td>DD</td>
<td>221</td>
<td>Alt 221</td>
<td>Note 6</td>
</tr>
<tr>
<td>DE</td>
<td>222</td>
<td>Alt 222</td>
<td>Note 6</td>
</tr>
<tr>
<td>DF</td>
<td>223</td>
<td>Alt 223</td>
<td>Note 6</td>
</tr>
<tr>
<td>E0</td>
<td>224</td>
<td>α</td>
<td>Alt 224</td>
</tr>
<tr>
<td>E1</td>
<td>225</td>
<td>β</td>
<td>Alt 225</td>
</tr>
<tr>
<td>E2</td>
<td>226</td>
<td>Γ</td>
<td>Alt 226</td>
</tr>
<tr>
<td>E3</td>
<td>227</td>
<td>π</td>
<td>Alt 227</td>
</tr>
<tr>
<td>E4</td>
<td>228</td>
<td>Σ</td>
<td>Alt 228</td>
</tr>
<tr>
<td>E5</td>
<td>229</td>
<td>σ</td>
<td>Alt 229</td>
</tr>
<tr>
<td>E6</td>
<td>230</td>
<td>μ</td>
<td>Alt 230</td>
</tr>
<tr>
<td>E7</td>
<td>231</td>
<td>τ</td>
<td>Alt 231</td>
</tr>
<tr>
<td>E8</td>
<td>232</td>
<td>Φ</td>
<td>Alt 232</td>
</tr>
<tr>
<td>E9</td>
<td>233</td>
<td>θ</td>
<td>Alt 233</td>
</tr>
<tr>
<td>EA</td>
<td>234</td>
<td>Ω</td>
<td>Alt 234</td>
</tr>
<tr>
<td>EB</td>
<td>235</td>
<td>δ</td>
<td>Alt 235</td>
</tr>
</tbody>
</table>

Of Characters, Keystrokes, and Colors  C-9
<table>
<thead>
<tr>
<th>Value</th>
<th>As Characters</th>
<th>As Text Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Color/Graphics Monitor Adapter</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Background</td>
</tr>
<tr>
<td>EC 236</td>
<td>$\infty$</td>
<td>Alt 236</td>
</tr>
<tr>
<td>ED 237</td>
<td>$\phi$</td>
<td>Alt 237</td>
</tr>
<tr>
<td>EE 238</td>
<td>$\epsilon$</td>
<td>Alt 238</td>
</tr>
<tr>
<td>EF 239</td>
<td>$\ominus$</td>
<td>Alt 239</td>
</tr>
<tr>
<td>F0 240</td>
<td>$\equiv$</td>
<td>Alt 240</td>
</tr>
<tr>
<td>F1 241</td>
<td>±</td>
<td>Alt 241</td>
</tr>
<tr>
<td>F2 242</td>
<td>$\geq$</td>
<td>Alt 242</td>
</tr>
<tr>
<td>F3 243</td>
<td>$\leq$</td>
<td>Alt 243</td>
</tr>
<tr>
<td>F4 244</td>
<td>$\downarrow$</td>
<td>Alt 244</td>
</tr>
<tr>
<td>F5 245</td>
<td>$\nearrow$</td>
<td>Alt 245</td>
</tr>
<tr>
<td>F6 246</td>
<td>÷</td>
<td>Alt 246</td>
</tr>
<tr>
<td>F7 247</td>
<td>$\approx$</td>
<td>Alt 247</td>
</tr>
<tr>
<td>F8 248</td>
<td>$\circ$</td>
<td>Alt 248</td>
</tr>
<tr>
<td>F9 249</td>
<td>$\bullet$</td>
<td>Alt 249</td>
</tr>
<tr>
<td>FA 250</td>
<td>$\cdot$</td>
<td>Alt 250</td>
</tr>
<tr>
<td>FB 251</td>
<td>$\sqrt{}$</td>
<td>Alt 251</td>
</tr>
<tr>
<td>FC 252</td>
<td>$\eta$</td>
<td>Alt 252</td>
</tr>
<tr>
<td>FD 253</td>
<td>2</td>
<td>Alt 253</td>
</tr>
<tr>
<td>FE 254</td>
<td>$\blacksquare$</td>
<td>Alt 254</td>
</tr>
<tr>
<td>FF 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 [Prt Sc] key or 2) in shift mode hit the 
   key.

NOTE 2  Period (.) can easily be keyed using two methods:  
1) hit the [>] key or 2) in shift or Num Lock 
   mode hit the [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 Decimal Value</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>16</td>
<td>32</td>
<td>48</td>
<td>64</td>
<td>80</td>
<td>96</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>&quot;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>🎃</td>
<td>🗯️</td>
<td>!!</td>
<td>#</td>
<td>3</td>
<td>C</td>
<td>S</td>
<td>c</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>♦️</td>
<td>🤛</td>
<td>$</td>
<td>4</td>
<td>D</td>
<td>T</td>
<td>d</td>
<td>t</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>♣️</td>
<td>⌚️</td>
<td>%</td>
<td>5</td>
<td>E</td>
<td>U</td>
<td>e</td>
<td>u</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>♠️</td>
<td>🕳️</td>
<td>&amp;</td>
<td>6</td>
<td>F</td>
<td>V</td>
<td>f</td>
<td>v</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>•</td>
<td>🎃</td>
<td>'</td>
<td>7</td>
<td>G</td>
<td>W</td>
<td>g</td>
<td>w</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>•</td>
<td>🎃</td>
<td>(</td>
<td>8</td>
<td>H</td>
<td>X</td>
<td>h</td>
<td>x</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>🎃</td>
<td>🎃</td>
<td>)</td>
<td>9</td>
<td>I</td>
<td>Y</td>
<td>i</td>
<td>y</td>
</tr>
<tr>
<td>10</td>
<td>A</td>
<td>🎃</td>
<td>🎃</td>
<td>—</td>
<td>:</td>
<td>J</td>
<td>Z</td>
<td>j</td>
<td>z</td>
</tr>
<tr>
<td>11</td>
<td>B</td>
<td>🎃</td>
<td>🎃</td>
<td>+</td>
<td>;</td>
<td>K</td>
<td>[</td>
<td>k</td>
<td>{</td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>🎃</td>
<td>🎃</td>
<td>,</td>
<td>&lt;</td>
<td>L</td>
<td>\</td>
<td>l</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>D</td>
<td>🎃</td>
<td>🎃</td>
<td>—</td>
<td>=</td>
<td>M</td>
<td>]</td>
<td>m</td>
<td>}</td>
</tr>
<tr>
<td>14</td>
<td>E</td>
<td>🎃</td>
<td>🎃</td>
<td>.</td>
<td>&gt;</td>
<td>N</td>
<td>^</td>
<td>n</td>
<td>~</td>
</tr>
<tr>
<td>15</td>
<td>F</td>
<td>🎃</td>
<td>🎃</td>
<td>/</td>
<td>?</td>
<td>O</td>
<td>_</td>
<td>o</td>
<td>△</td>
</tr>
</tbody>
</table>

C-12  Of Characters, Keystrokes, and Colors
<table>
<thead>
<tr>
<th>DECIMAL VALUE</th>
<th>HEXA VALUE</th>
<th>128</th>
<th>144</th>
<th>160</th>
<th>176</th>
<th>192</th>
<th>208</th>
<th>224</th>
<th>240</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Ç</td>
<td>É</td>
<td>á</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>ü</td>
<td>æ</td>
<td>í</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>é</td>
<td>Æ</td>
<td>ó</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>ã</td>
<td>õ</td>
<td>û</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>ä</td>
<td>ö</td>
<td>ñ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>à</td>
<td>ò</td>
<td>Ñ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>á</td>
<td>û</td>
<td>Ñ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>ç</td>
<td>ù</td>
<td>ò</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>ê</td>
<td>ë</td>
<td>õ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>è</td>
<td>Ö</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>A</td>
<td>é</td>
<td>Ü</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>B</td>
<td>ï</td>
<td>ì</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>I</td>
<td>£</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>D</td>
<td>¥</td>
<td>i</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>E</td>
<td>Ä</td>
<td>ã</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>F</td>
<td>Å</td>
<td>s</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Of Characters, Keystrokes, and Colors  C-13
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
D-2 Logic Diagrams
Keyboard - Type 1 (Sheet 2 of 2)
D-16 Logic Diagrams
D-22 Logic Diagrams  Printer (Sheet 1 of 2)
Monochrome Display Adapter (Sheet 1 of 10)
Monochrome Display Adapter (Sheet 2 of 10)
Monochrome Display Adapter (Sheet 3 of 10)
Monochrome Display Adapter (Sheet 7 of 10)
Color/Graphics Monitor Adapter (Sheet 2 of 6)
Color/Graphics Monitor Adapter (Sheet 5 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

Logic Diagrams D-43
Monochrome Display (Sheet 1 of 1)

DANGER
HAZARDOUS VOLTAGES
UP TO 450 VOLTS EXIST
ON THE PRINTED
CIRCUIT BOARDS

NOTES:
1. RESISTOR VALUES ARE IN OHMS (1.0K = 1,000, 10K = 10,000, 100K = 100,000)
2. ALL RESISTORS ARE 1/4W EXCEPT WHERE OTHERWISE INDICATED
3. ALL CAPACITORS ARE 1/4W EXCEPT WHERE OTHERWISE INDICATED
4. CAPACITOR VALUES ARE pF UNLESS OTHERWISE INDICATED
5. AC WIRE 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:

<table>
<thead>
<tr>
<th>Drive 1</th>
<th>Drive 2</th>
<th>Drive 3</th>
<th>Drive 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signals</td>
<td>Signals</td>
<td>Signals</td>
<td>Signals</td>
</tr>
<tr>
<td>10</td>
<td>16</td>
<td>16</td>
<td>10</td>
</tr>
<tr>
<td>11</td>
<td>15</td>
<td>15</td>
<td>11</td>
</tr>
<tr>
<td>12</td>
<td>14</td>
<td>14</td>
<td>12</td>
</tr>
<tr>
<td>13</td>
<td>13</td>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td>14</td>
<td>11</td>
<td>11</td>
<td>14</td>
</tr>
<tr>
<td>15</td>
<td>12</td>
<td>12</td>
<td>15</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>10</td>
<td>16</td>
</tr>
</tbody>
</table>

2. ALL DRIVES ARE JUMPERED FOR MULTIPLEX OPERATION. HEAD LOAD WITH DRIVE SELECT AND DRIVE SELECT VIA INPUT PIN 12. TERMINATING R-PINS ALL LEFT OVER DRIVES 1 & 3 ONLY.

3. MC4044 should be adjacent to module MS287, 7439, 14S74, 14S72, 14LS31. A 14LS111 8, 2.5 OHM RESISTORS SHOULD BE SHOWN ASSOCIATED P1 PINS.

4. ALL SIGNAL LINES HIGHER THAN OR EQUAL TO IN1X SHOULD BE KEPT TO THE SHORTEST POSSIBLE LENGTH. THIS IS A PRIMARY DESIGN GOAL.

5. MAKE NO CONNECTION TO UNUSED PINS OR 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)
D-46 Logic Diagrams
5-1/4 Inch Diskette Drive Adapter (Sheet 3 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. 5%
2. ALL CAPS ARE IN μF.
3. ALL DIODES ARE IN 444&.
4. ALL TRANSISTORS NPN 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, ±2%, 1/4W.
2. ±1% RESISTORS ARE 1/2W.
3. CAPACITORS ARE IN µF, ±20%, 35V.
Fixed Disk Drive Adapter (Sheet 5 of 6)
Fixed Disk Drive Adapter (Sheet 6 of 6)

NOTES:
1. UNLESS OTHERWISE SPECIFIED:
   a. ALL RESISTORS 1/4 W, 5% CARBON RESISTOR.
   b. ALL CAPS +12V OR GREATER >10uF.
   c. NO MORE THAN 15 LOADS PER PULLUP NET.
D-60  Logic Diagrams
Fixed Disk Drive - Type 1 (Sheet 2 of 3)
Fixed Disk Drive - Type 1 (Sheet 3 of 3)
Logic Diagrams

Appendix D

Fixed Disk Drive - Type 2 (Sheet 1 of 3)

NOTES:
1. SHEET TO SHEET CONNECTION IS AS FOLLOWS:
2. UNLESS OTHERWISE SPECIFIED ALL RESISTORS ARE 1/4W 5% VALUE IN OHMS
   CAPACITORS = NO. 5N, VALUE IN uF
3. J7-1 IS A PROGRAMMABLE JUMPER SOCKET
Fixed Disk Drive - Type 2 (Sheet 2 of 3)

D-64 Logic Diagrams
Logic Diagrams

Appendix D

D-67

32K Memory Expansion Option (Sheet 2 of 3)
64K Memory Expansion Option (Sheet 2 of 3)
64/256K Memory Expansion Option (Sheet 3 of 4)
Game Control Adapter (Sheet 1 of 1)
Logic Diagrams D-79

Binary Synchronous Communications Adapter (Sheet 1 of 2)
SDLC 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 kg)

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
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
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:

**Diagram Notes:**
- *Not used when business machine clocking is used.
- **Not standardized by EIA (Electronics Industry Association).
- ***Not standardized by CCITT.

---

**Table:**

<table>
<thead>
<tr>
<th>EIA/CCITT Line Number</th>
<th>Lead Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>AA/101</td>
</tr>
<tr>
<td>2</td>
<td>BB/104</td>
</tr>
<tr>
<td>3</td>
<td>BA/103</td>
</tr>
<tr>
<td>4</td>
<td>CA/105</td>
</tr>
<tr>
<td>5</td>
<td>CB/106</td>
</tr>
<tr>
<td>6</td>
<td>CC/107</td>
</tr>
<tr>
<td>7</td>
<td>AB/102</td>
</tr>
<tr>
<td>8</td>
<td>CF/109</td>
</tr>
<tr>
<td>9</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>CH/111</td>
</tr>
<tr>
<td>11</td>
<td>DD/115</td>
</tr>
<tr>
<td>12</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>DB/114</td>
</tr>
<tr>
<td>14</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>**/108.1</td>
</tr>
<tr>
<td>21</td>
<td>**/108.2</td>
</tr>
<tr>
<td>22</td>
<td>DE/125</td>
</tr>
<tr>
<td>23</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
</tr>
</tbody>
</table>

---

**Figure:**

External Modem Cable Connector

---

**F-2 Communications**
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.

<table>
<thead>
<tr>
<th>Switched Timing Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Terminal Ready</td>
</tr>
<tr>
<td>Data Set Ready</td>
</tr>
<tr>
<td>Request to Send</td>
</tr>
<tr>
<td>Clear to Send</td>
</tr>
<tr>
<td>Transmitted Data</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Nonswitched Timing Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Terminal Ready</td>
</tr>
<tr>
<td>Data Terminal Ready</td>
</tr>
<tr>
<td>Request to Send</td>
</tr>
<tr>
<td>Clear to Send</td>
</tr>
<tr>
<td>Transmitted Data</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, which causes the modem at terminal A to generate a carrier signal.

2. Normally the 'data set ready' lines from the modems are active whenever the modems are powered on.

3. Terminal A activates the 'request to send' line, which causes modem B to detect 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.

4. 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.

5. 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.

6. Terminal 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.

9. Terminal B now becomes the transmitter to respond to the request from terminal A. To transmit data, terminal B activates the 'request to send' line, which causes modem B to transmit a carrier to modem A.

10. Modem B begins a delay that is longer than the echo delay at modem A before turning on the 'clear to send' line. The longer delay (called request-to-send to clear-to-send delay) ensures that modem A is ready to receive when terminal B begins transmitting data. After the delay, modem B activates the 'clear to send' line to indicate that terminal B can begin transmitting its response.

11. After the echo delay at modem A, modem A senses the carrier from modem B (the carrier was activated in step 13 when terminal B activated the 'request to send' line) and activates the 'received line signal detector' line to terminal A.

12. Modem A and terminal A are now ready to receive the response from terminal B. Remember, the response was not transmitted until after the request-to-send to clear-to-send delay at modem B (step 14).
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

System Board Switch Settings ...................... G-3
  System Board Switch ................................ G-3
  Math Coprocessor Switch Setting .................. G-3
  System Board Memory Switch Settings .......... G-4
  Monitor Type Switch Settings ................... G-4
  5-1/4” Diskette Drive Switch Settings .......... G-5

Extender Card Switch Settings ..................... G-6

Memory Option Switch Settings ..................... G-7
  288K Total Memory .................................. G-7
  320K Total Memory .................................. G-8
  352K Total Memory .................................. G-9
  384K Total Memory .................................. G-10
  416K Total Memory .................................. G-11
  448K Total Memory .................................. G-12
  480K Total Memory .................................. G-13
  512K Total Memory .................................. G-14
  544K Total Memory .................................. G-15
  576K Total Memory .................................. G-16
  608K Total Memory .................................. G-17
  640K Total Memory .................................. G-18
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>
### Memory Option Switch Settings

288K Total Memory  
32K + (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 - 32K option</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Switch Settings G-7
### 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></td>
<td></td>
</tr>
<tr>
<td>1 - 64K option</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 - 32K options</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
352K Total Memory
96K + (256K on System Board)

<table>
<thead>
<tr>
<th>Scenario</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 64K installed 1 - 32K option</td>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="image3.png" alt="Image" /></td>
</tr>
<tr>
<td>1 - 64K option 1 - 32K option</td>
<td><img src="image4.png" alt="Image" /></td>
<td><img src="image5.png" alt="Image" /></td>
<td><img src="image6.png" alt="Image" /></td>
</tr>
<tr>
<td>3 - 32K options</td>
<td><img src="image7.png" alt="Image" /></td>
<td><img src="image8.png" alt="Image" /></td>
<td><img src="image9.png" alt="Image" /></td>
</tr>
</tbody>
</table>
### 384K Total Memory
128K + (256K on System Board)

<table>
<thead>
<tr>
<th>Configuration</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 64K option installed</td>
<td>![Card Switches Image]</td>
<td>![Card Switches Image]</td>
<td>![Card Switches Image]</td>
</tr>
<tr>
<td>1 - 64K option</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 - 64K options</td>
<td>![Card Switches Image]</td>
<td>![Card Switches Image]</td>
<td>![Card Switches Image]</td>
</tr>
<tr>
<td>1 - 64/256K option with 64K installed</td>
<td>![Card Switches Image]</td>
<td>![Card Switches Image]</td>
<td>![Card Switches Image]</td>
</tr>
<tr>
<td>2 - 32K options</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 - 64K option</td>
<td>![Card Switches Image]</td>
<td>![Card Switches Image]</td>
<td>![Card Switches Image]</td>
</tr>
<tr>
<td>2 - 32K options</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 416K Total Memory

160K + (256K on System Board)

<table>
<thead>
<tr>
<th>Configuration</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 64K installed</td>
<td><img src="image1.png" alt="Switches" /></td>
<td><img src="image2.png" alt="Switches" /></td>
<td><img src="image3.png" alt="Switches" /></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>2 - 64K options</td>
<td></td>
<td><img src="image4.png" alt="Switches" /></td>
<td><img src="image5.png" alt="Switches" /></td>
</tr>
<tr>
<td>1 - 32K option</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 - 64/256K option with 128K installed</td>
<td><img src="image6.png" alt="Switches" /></td>
<td></td>
<td><img src="image7.png" alt="Switches" /></td>
</tr>
<tr>
<td>1 - 32K option</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Memory Option</td>
<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>
<td>--------------------------</td>
</tr>
<tr>
<td>1 - 64/256K option with 192K installed</td>
<td>![Diagram]</td>
<td>![Diagram]</td>
<td>![Diagram]</td>
</tr>
<tr>
<td>1 - 64/256K option with 128K installed</td>
<td>![Diagram]</td>
<td>![Diagram]</td>
<td>![Diagram]</td>
</tr>
<tr>
<td>1 - 64K option</td>
<td>![Diagram]</td>
<td>![Diagram]</td>
<td>![Diagram]</td>
</tr>
<tr>
<td>1 - 64/256K option with 64K installed</td>
<td>![Diagram]</td>
<td>![Diagram]</td>
<td>![Diagram]</td>
</tr>
<tr>
<td>2 - 64K options</td>
<td>![Diagram]</td>
<td>![Diagram]</td>
<td>![Diagram]</td>
</tr>
<tr>
<td>3 - 64K options</td>
<td>![Diagram]</td>
<td>![Diagram]</td>
<td>![Diagram]</td>
</tr>
<tr>
<td>1 - 64/256K option with 128 installed</td>
<td>![Diagram]</td>
<td>![Diagram]</td>
<td>![Diagram]</td>
</tr>
<tr>
<td>2 - 32K options</td>
<td>![Diagram]</td>
<td>![Diagram]</td>
<td>![Diagram]</td>
</tr>
</tbody>
</table>
### 480K Total Memory

224K + (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 192K installed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 - 32K option</td>
<td></td>
<td></td>
</tr>
<tr>
<td><img src="image1" alt="Switch Settings Diagram" /></td>
<td></td>
<td></td>
</tr>
<tr>
<td><img src="image2" alt="Switch Settings Diagram" /></td>
<td></td>
<td></td>
</tr>
<tr>
<td><img src="image3" alt="Switch Settings Diagram" /></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| 1 - 64/256K option with 128K installed  |
| 1 - 64K option |
| 1 - 32K option |
| <img src="image4" alt="Switch Settings Diagram" />  |
| <img src="image5" alt="Switch Settings Diagram" />  |
| <img src="image6" alt="Switch Settings Diagram" />  |</p>
<table>
<thead>
<tr>
<th>Memory Configuration</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="switch1.png" alt="Switch Settings" /></td>
<td><img src="switch2.png" alt="Switch Settings" /></td>
<td><img src="switch3.png" alt="Switch Settings" /></td>
</tr>
<tr>
<td>2 - 64K options</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 - 64/256K option with 192K installed</td>
<td><img src="switch1.png" alt="Switch Settings" /></td>
<td><img src="switch2.png" alt="Switch Settings" /></td>
<td><img src="switch3.png" alt="Switch Settings" /></td>
</tr>
<tr>
<td>1 - 64K option</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 - 64/256K option with 192K installed</td>
<td><img src="switch1.png" alt="Switch Settings" /></td>
<td><img src="switch2.png" alt="Switch Settings" /></td>
<td><img src="switch3.png" alt="Switch Settings" /></td>
</tr>
<tr>
<td>2 - 32K options</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 - 64/256K option with 256K installed</td>
<td><img src="switch1.png" alt="Switch Settings" /></td>
<td><img src="switch2.png" alt="Switch Settings" /></td>
<td><img src="switch3.png" alt="Switch Settings" /></td>
</tr>
</tbody>
</table>
### 544K Total Memory
288K + (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 192K installed</td>
<td>1 - 64K option</td>
<td>1 - 32K option</td>
</tr>
<tr>
<td>1 - 64K option</td>
<td>1 - 32K option</td>
<td></td>
</tr>
<tr>
<td>1 - 64/256K option with 256K installed</td>
<td>1 - 32K option</td>
<td></td>
</tr>
</tbody>
</table>

*Switch Settings*

*Appendix C*
<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 192K installed</td>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="image3.png" alt="Image" /></td>
</tr>
<tr>
<td>2 - 64K options</td>
<td><img src="image4.png" alt="Image" /></td>
<td><img src="image5.png" alt="Image" /></td>
<td><img src="image6.png" alt="Image" /></td>
</tr>
<tr>
<td>1 - 64/256K option with 256K installed</td>
<td><img src="image7.png" alt="Image" /></td>
<td><img src="image8.png" alt="Image" /></td>
<td><img src="image9.png" alt="Image" /></td>
</tr>
<tr>
<td>1 - 64/256K option with 64K installed</td>
<td><img src="image10.png" alt="Image" /></td>
<td><img src="image11.png" alt="Image" /></td>
<td><img src="image12.png" alt="Image" /></td>
</tr>
<tr>
<td>1 - 64/256K option with 256K installed</td>
<td><img src="image13.png" alt="Image" /></td>
<td><img src="image14.png" alt="Image" /></td>
<td><img src="image15.png" alt="Image" /></td>
</tr>
<tr>
<td>1 - 64/256K option with 64K option</td>
<td><img src="image16.png" alt="Image" /></td>
<td><img src="image17.png" alt="Image" /></td>
<td><img src="image18.png" alt="Image" /></td>
</tr>
<tr>
<td>1 - 64/256K option with 256K installed</td>
<td><img src="image19.png" alt="Image" /></td>
<td><img src="image20.png" alt="Image" /></td>
<td><img src="image21.png" alt="Image" /></td>
</tr>
<tr>
<td>2 - 32K options</td>
<td><img src="image22.png" alt="Image" /></td>
<td><img src="image23.png" alt="Image" /></td>
<td><img src="image24.png" alt="Image" /></td>
</tr>
</tbody>
</table>

576K Total Memory
320K + (256K on System Board)
### 608K Total Memory

352K + (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 256K installed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 - 64/256K option with 64K installed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 - 32K option</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 256K installed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 - 64K option</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 - 32K option</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 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>&lt;br&gt;<strong>1 - 64/256K option with 64K installed</strong>&lt;br&gt;<strong>1 - 64K option</strong>&lt;br&gt;<strong>1 - 64/256K option with 256K installed</strong>&lt;br&gt;<strong>2 - 64K options</strong>&lt;br&gt;<strong>1 - 64/256K option with 256K installed</strong>&lt;br&gt;<strong>1 - 64/256K option with 128K installed</strong></td>
<td><img src="image1.png" alt="Switch Settings" /></td>
<td><img src="image2.png" alt="Switch Settings" /></td>
</tr>
<tr>
<td><img src="image3.png" alt="Switch Settings" /></td>
<td><img src="image4.png" alt="Switch Settings" /></td>
<td><img src="image5.png" alt="Switch Settings" /></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.
**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.
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 alphanumeric mode. (2) The most frequent value in the statistical sense.
modem: (Modulator-Demodulator) A device that converts serial (bit by bit) digital signals from a business machine (or data terminal equipment) to analog signals which are suitable for transmission in a telephone network. The inverse function is also performed by the modem on reception of analog signals.

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

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

monitor: (1) A device that observes and verifies the operation of a data processing system and indicates any specific departure from the norm. (2) A television type display, such as the IBM Monochrome Display. (3) Software or hardware that observes, supervises, controls, or verifies the operations of a system.

ms: Millisecond; one thousandth of a second.

multiplexer: A device capable of 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.
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/asynchronous 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
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
connector specifications 1-146
direct-drive connector specifications 1-146
display buffer basic operation 1-127
graphics mode 1-132
graphics mode (high resolution) 1-135
graphics mode (low resolution) 1-132
graphic mode (medium resolution) 1-133
light pen connector specifications 1-147
major components 1-126
memory requirements 1-145
mode control and status register 1-143
mode register summary 1-143
mode select register 1-141
programming considerations 1-137
RF modulator connector specifications 1-147
sequence of events 1-144
status register 1-143
summary of available color 1-135
colors, summary of available 1-137
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,
  diskette drive adapter 1-151
  fixed disk drive adapter 1-179
communications adapter cable 1-285
  connector specifications 1-286
communications F-1
  establishing a link F-3
component diagram,
  system board 1-13
compressed (printer) 1-93
compressed off (printer) 1-93
connector specifications,
  asynchronous communications adapter 1-215
  binary synchronous communications 1-245
  color/graphics monitor adapter 1-147
  communications adapter cable 1-286
  diskette drive adapter (external) 1-174
  diskette drive adapter (internal) 1-173
  game control adapter 1-203
  keyboard interface 1-67
  monochrome display adapter 1-115
  printer adapter 1-81
  synchronous data link control adapter 1-293
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
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
FCHS (coprocessor) 1-38
FCLEX/FNCLEX (coprocessor) 1-38
FCOM (coprocessor) 1-38
FCOMP (coprocessor) 1-39
FCOMPP (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
FPATAN 1-52
FPREM 1-52
FPTAN 1-53
FRNDINT 1-53
FRSTOR 1-53
FSAVE/FNSAVE (coprocessor) 1-54
FScale (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
FSTS/FSNSTS (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
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
hardware interrupt listing 1-9
home head (printer) 1-95
horizontal tab (printer) 1-92
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
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
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

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

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
FRNDINT 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

Index J-15
Motorola 6845 CRT Controller,
(see color/graphics monitor adapter)
(see monochrome display adapter)

N

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

O

one bit delay mode register 1-283
operating mode register 1-280
OSC (oscillator) 1-16
Oscillator (OSC), I/O channel 1-16

P

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
  connector 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
subscript/superscript (printer) 1-99
subscript/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
8250,
    (see asynchronous communications 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
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:
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:
BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 123 BOCA RATON, FLORIDA 33432

POSTAGE WILL BE PAID BY ADDRESSEE

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