LIMITED WARRANTY

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

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

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

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

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

THIS WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS, AND YOU MAY ALSO HAVE OTHER RIGHTS WHICH MAY VARY FROM STATE TO STATE.
Federal Communications Commission
Radio Frequency Interference Statement

WARNING: This equipment has been certified to comply with the limits for a Class B computing device, pursuant to Subpart J of Part 15 of FCC rules. Only peripherals (computer input/output devices, terminals, printers, etc.) certified to comply with the Class B limits may be attached to this computer. Operation with non-certified peripherals is likely to result in interference to radio and TV reception. If peripherals not offered by IBM are used with this equipment, it is suggested to use shielded grounded cables with in-line filters if necessary.

Notice: As sold by the manufacturer, the 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 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 Technical Reference manual describes the hardware design and provides interface information for the IBM Personal Computer. 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, 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
Part Number 6025000

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
Part Number 6025072

*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 System Unit ........................................ 1-3
- IBM Personal Computer Math Coprocessor .................................. 1-33
- IBM Keyboard ............................................................................. 1-73
- IBM Expansion Unit .................................................................... 1-79
- IBM 80 CPS Printers .................................................................... 1-91
- IBM Printer Adapter ...................................................................... 1-117
- IBM Monochrome Display and Printer Adapter .............................. 1-123
- IBM Monochrome Display ............................................................ 1-131
- IBM Color/Graphics Display Adapter ............................................ 1-133
- IBM Color Display ....................................................................... 1-157
- IBM 5-¼” Diskette Drive Adapter ................................................ 1-159
- IBM 5-¼” Diskette Drive ............................................................... 1-183
- Diskettes ....................................................................................... 1-185
- IBM Fixed Disk Drive Adapter ...................................................... 1-187
- IBM 10MB Fixed Disk Drive .......................................................... 1-203
- IBM Memory Expansion Options ................................................. 1-205
- IBM Game Control Adapter ........................................................... 1-211
- IBM Prototype Card ...................................................................... 1-217
- IBM Asynchronous Communications Adapter .............................. 1-223
- IBM Binary Synchronous Communications Adapter ..................... 1-251
- IBM Synchronous Data Link Control (SDLC) Communication Adapter .... 1-271
- IBM Communications Adapter Cable ............................................ 1-301

## Section 2: ROM BIOS and System Usage

- ROM BIOS .................................................................................. 2-2
- Keyboard Encoding and Usage ...................................................... 2-11
- BIOS Cassette Logic ..................................................................... 2-21

## 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 (16/64K) ........................................ D-2
System Board (64/256K) ..................................... D-12
Keyboard - Type 1 ....................................... D-22
Keyboard - Type 2 ..................................... D-24
Expansion Board ........................................ D-25
Extender Card ........................................ D-26
Receiver Card ........................................ D-29
Printer ................................................ D-32
Printer Adapter ......................................... D-35
Monochrome Display Adapter ....................... D-36
Color/Graphics Monitor Adapter ..................... D-46
Color Display ........................................ D-52
Monochrome Display ................................ D-54
5-¼ Inch Diskette Drive Adapter .................... D-55
5-¼ Inch Diskette Drive – Type 1 .................... D-59
5-¼ Inch Diskette Drive – Type 2 .................... D-62
Fixed Disk Drive Adapter ............................ D-64
Fixed Disk Drive – Type 1 ............................ D-70
Fixed Disk Drive – Type 2 ............................ D-73
32K Memory Expansion Option ..................... D-76
64K Memory Expansion Option ..................... D-79
64/256K Memory Expansion Option ................. D-82
Game Control Adapter ................................. D-86
Prototype Card ........................................ D-87
Asynchronous Communications Adapter .......... D-88
Binary Synchronous Communications Adapter .... D-89
SDLC Communications Adapter ................. D-91

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  ........................
SECTION 1: HARDWARE

IBM Personal Computer System Unit .................. 1-3
IBM Personal Computer Math Coprocesser ............ 1-33
IBM Keyboard .................................... 1-73
IBM Expansion Unit ................................ 1-79
IBM 80 CPS Printers ................................ 1-91
IBM Printer Adapter ............................... 1-117
IBM Monochrome Display and Printer Adapter ....... 1-123
IBM Monochrome Display ............................. 1-131
IBM Color/Graphics Display Adapter .................. 1-133
IBM Color Display .................................. 1-157
IBM 5-1/4” Diskette Drive Adapter ................... 1-159
IBM 5-1/4” Diskette Drive ........................... 1-183
Diskettes ............................................ 1-185
IBM Fixed Disk Drive Adapter ......................... 1-187
IBM 10MB Fixed Disk Drive ........................... 1-203
IBM Memory Expansion Options ........................ 1-205
IBM Game Control Adapter .............................. 1-211
IBM Prototype Card .................................. 1-217
IBM Asynchronous Communications Adapter .......... 1-223
IBM Binary Synchronous Communications Adapter .... 1-251
IBM Synchronous Data Link Control (SDLC)
  Communication Adapter .............................. 1-271
IBM Communications Adapter Cable .................... 1-301
System Block Diagram
The system unit is the standalone tabletop unit that contains the power supply, the speaker, and the system board.

The system unit contains one of two system boards. One system board supports 16K to 64K of read/write memory. The other system board supports 64K to 256K of read/write memory. Both system boards are functionally identical.

The power supply provides dc voltage to the system board and the internal drive(s).

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, audio cassette, and speaker. Five 62-pin card edge-sockets are also mounted on the board. The I/O channel is bussed across these five I/O slots.

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

The system board consists of five functional areas: the 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 a 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 48K x 8 of ROM or EPROM. Six module sockets are provided, each of which can accept an 8K by 8 byte device. Five of the sockets are populated with 40K bytes of ROM. This ROM contains the cassette BASIC interpreter, cassette operating system, 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 24-pin modules and has an access time of 250 ns and a cycle time of 375 ns.

The difference between the R/W memory on the two system boards is shown in the following chart.

<table>
<thead>
<tr>
<th>System Board</th>
<th>Minimum Storage</th>
<th>Maximum Storage</th>
<th>Memory Modules</th>
<th>Soldered (Bank 0)</th>
<th>Pluggable (Bank 1-3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>16/64K</td>
<td>16K</td>
<td>64K</td>
<td>16K by 1 Bit</td>
<td>1 Bank of 9</td>
<td>3 Banks of 9</td>
</tr>
<tr>
<td>64/256K</td>
<td>64K</td>
<td>256K</td>
<td>64K by 1 Bit</td>
<td>1 Bank of 9</td>
<td>3 Banks of 9</td>
</tr>
</tbody>
</table>

Memory greater than either system board's maximum is obtained by adding memory cards in the expansion slots. All memory is parity-checked and consists of dynamic 16K by 1 bit or (64K by 1 bit) chips with an access time of 250 ns and a cycle time of 410 ns.
The system board contains circuits for attaching an audio cassette, the keyboard, and the speaker. The cassette adapter allows the attachment of any good quality audio cassette through the earphone output and either the microphone or auxiliary inputs. The system board has a jumper for either input. This interface also provides a cassette motor control line for transport starting and stopping under program control. This interface reads and writes the audio cassette at a data rate of between 1,000 and 2,000 baud. The baud rate is variable and dependent on data content, because a different bit-cell time is used for 0’s and 1’s. For diagnostic purposes, the tape interface can loop read to write for testing the system board’s circuits. The ROM cassette software blocks cassette data and generates a cyclic redundancy check (CRC) to check this data.

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

Both the keyboard and cassette interfaces are 5-pin DIN connectors on the system board that extend 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.

1-6 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 0060</th>
<th>PA0 +Keyboard Scan Code</th>
<th>0</th>
<th>IPL 5-1/4 Diskette Drive (SW1-1)</th>
<th>Reserved (SW1-2)</th>
<th>System Board Read/Write *(SW1-3)</th>
<th>Memory Size</th>
<th>System Board Read/Write *(SW1-4)</th>
<th>Memory Size</th>
<th>+Display Type 1 **(SW1-5)</th>
<th>+Display Type 2 ***SW1-6)</th>
<th>No. of 5-1/4 Drives ****(SW1-7)</th>
<th>No. of 5-1/4 Drives ****(SW1-8)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hex Port Number 0061</td>
<td>PA0 1 +Timer 2 Gate Speaker</td>
<td>0</td>
<td>Speaker Data</td>
<td>-----------------</td>
<td>Enable Read/Write Memory</td>
<td>------------</td>
<td>Enable I/0 Channel Check</td>
<td>------------</td>
<td>Hold Keyboard Clock Low</td>
<td>-------------------</td>
<td>(Enable Keyboard) or + (Clear Keyboard and Enable Sense Switches)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PA1 2 +Speaker Data</td>
<td>1</td>
<td>Read/Write Memory Size or (Read Spare Key)</td>
<td>-----------------</td>
<td>Enable Read/Write Memory</td>
<td>------------</td>
<td>Enable I/0 Channel Check</td>
<td>------------</td>
<td>Hold Keyboard Clock Low</td>
<td>-------------------</td>
<td>(Enable Keyboard) or + (Clear Keyboard and Enable Sense Switches)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PA2 3 +Cassette Motor Off</td>
<td>2</td>
<td>Read/Write Memory Size or (Read Spare Key)</td>
<td>-----------------</td>
<td>Enable Read/Write Memory</td>
<td>------------</td>
<td>Enable I/0 Channel Check</td>
<td>------------</td>
<td>Hold Keyboard Clock Low</td>
<td>-------------------</td>
<td>(Enable Keyboard) or + (Clear Keyboard and Enable Sense Switches)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PA3 4 +Cassette Data In</td>
<td>3</td>
<td>Read/Write Memory Size or (Read Spare Key)</td>
<td>-----------------</td>
<td>Enable Read/Write Memory</td>
<td>------------</td>
<td>Enable I/0 Channel Check</td>
<td>------------</td>
<td>Hold Keyboard Clock Low</td>
<td>-------------------</td>
<td>(Enable Keyboard) or + (Clear Keyboard and Enable Sense Switches)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PA4 5 +Timer Channel 2 Out</td>
<td>4</td>
<td>Read/Write Memory Size or (Read Spare Key)</td>
<td>-----------------</td>
<td>Enable Read/Write Memory</td>
<td>------------</td>
<td>Enable I/0 Channel Check</td>
<td>------------</td>
<td>Hold Keyboard Clock Low</td>
<td>-------------------</td>
<td>(Enable Keyboard) or + (Clear Keyboard and Enable Sense Switches)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PA5 6 +I/O Channel Check</td>
<td>6</td>
<td>Read/Write Memory Size or (Read Spare Key)</td>
<td>-----------------</td>
<td>Enable Read/Write Memory</td>
<td>------------</td>
<td>Enable I/0 Channel Check</td>
<td>------------</td>
<td>Hold Keyboard Clock Low</td>
<td>-------------------</td>
<td>(Enable Keyboard) or + (Clear Keyboard and Enable Sense Switches)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PA6 7 +Read/Write Memory Parity Check</td>
<td>7</td>
<td>Read/Write Memory Size or (Read Spare Key)</td>
<td>-----------------</td>
<td>Enable Read/Write Memory</td>
<td>------------</td>
<td>Enable I/0 Channel Check</td>
<td>------------</td>
<td>Hold Keyboard Clock Low</td>
<td>-------------------</td>
<td>(Enable Keyboard) or + (Clear Keyboard and Enable Sense Switches)</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>
<td></td>
<td></td>
</tr>
<tr>
<td>Command/Mode Register</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hex 99</td>
<td>Mode Register Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7 6 5 4 3 2 1 0</td>
<td>1 0 0 1 1 0 0 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>* PA3 Sw1-4 PA2 Sw1-3</th>
<th>Amount of Memory Located on System Board</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0</td>
<td>16K</td>
</tr>
<tr>
<td>0 1</td>
<td>32K</td>
</tr>
<tr>
<td>1 0</td>
<td>48K</td>
</tr>
<tr>
<td>1 1</td>
<td>64 to 256K</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>** PA5 Sw1-6 PA4 Sw1-5</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>*** PA7 Sw1-8 PA6 Sw1-7</th>
<th>Number of 5-1/4” 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

1-12 System Unit
<table>
<thead>
<tr>
<th>Start Address</th>
<th>Hex</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decimal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>00000</td>
<td></td>
</tr>
<tr>
<td>16K</td>
<td>04000</td>
<td>16 to 64K Read/Write Memory on System Board</td>
</tr>
<tr>
<td>32K</td>
<td>08000</td>
<td></td>
</tr>
<tr>
<td>48K</td>
<td>0C000</td>
<td></td>
</tr>
<tr>
<td>64K</td>
<td>10000</td>
<td></td>
</tr>
<tr>
<td>80K</td>
<td>14000</td>
<td></td>
</tr>
<tr>
<td>96K</td>
<td>18000</td>
<td></td>
</tr>
<tr>
<td>112K</td>
<td>1C000</td>
<td></td>
</tr>
<tr>
<td>128K</td>
<td>20000</td>
<td></td>
</tr>
<tr>
<td>144K</td>
<td>24000</td>
<td></td>
</tr>
<tr>
<td>160K</td>
<td>28000</td>
<td></td>
</tr>
<tr>
<td>176K</td>
<td>2C000</td>
<td></td>
</tr>
<tr>
<td>192K</td>
<td>30000</td>
<td></td>
</tr>
<tr>
<td>208K</td>
<td>34000</td>
<td></td>
</tr>
<tr>
<td>224K</td>
<td>38000</td>
<td></td>
</tr>
<tr>
<td>240K</td>
<td>3C000</td>
<td></td>
</tr>
<tr>
<td>256K</td>
<td>40000</td>
<td>Up to 576K Read/Write Memory in I/O Channel</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></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 for 16/64K System Board (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</td>
</tr>
<tr>
<td>720K</td>
<td>B4000</td>
</tr>
<tr>
<td>736K</td>
<td>B8000</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</td>
</tr>
<tr>
<td>816K</td>
<td>CC000</td>
</tr>
<tr>
<td>832K</td>
<td>D0000</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</td>
</tr>
<tr>
<td>976K</td>
<td>F4000</td>
</tr>
<tr>
<td>992K</td>
<td>F8000</td>
</tr>
<tr>
<td>1008K</td>
<td>FC000</td>
</tr>
</tbody>
</table>

128K Reserved

Monochrome

Color/Graphics

Fixed Disk Control

192K Read Only Memory

Expansion and Control

Reserved

48K Base System ROM

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

System Memory Map for 64/256K System Board (Part 1 of 2)
<table>
<thead>
<tr>
<th>Start Address</th>
<th>Hex</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>640K</td>
<td>A0000</td>
<td>128K Reserved</td>
</tr>
<tr>
<td>656K</td>
<td>A4000</td>
<td></td>
</tr>
<tr>
<td>672K</td>
<td>A8000</td>
<td></td>
</tr>
<tr>
<td>688K</td>
<td>AC000</td>
<td></td>
</tr>
<tr>
<td>704K</td>
<td>B0000</td>
<td>Monochrome</td>
</tr>
<tr>
<td>720K</td>
<td>B4000</td>
<td></td>
</tr>
<tr>
<td>736K</td>
<td>B8000</td>
<td>Color/Graphics</td>
</tr>
<tr>
<td>752K</td>
<td>BC000</td>
<td></td>
</tr>
<tr>
<td>768K</td>
<td>C0000</td>
<td></td>
</tr>
<tr>
<td>784K</td>
<td>C4000</td>
<td></td>
</tr>
<tr>
<td>800K</td>
<td>C8000</td>
<td>Fixed Disk Control</td>
</tr>
<tr>
<td>816K</td>
<td>CC000</td>
<td></td>
</tr>
<tr>
<td>832K</td>
<td>D0000</td>
<td></td>
</tr>
<tr>
<td>848K</td>
<td>D4000</td>
<td>192K Read Only Memory</td>
</tr>
<tr>
<td>864K</td>
<td>D8000</td>
<td>Expansion and Control</td>
</tr>
<tr>
<td>880K</td>
<td>DC000</td>
<td></td>
</tr>
<tr>
<td>896K</td>
<td>E0000</td>
<td></td>
</tr>
<tr>
<td>912K</td>
<td>E4000</td>
<td></td>
</tr>
<tr>
<td>928K</td>
<td>E8000</td>
<td></td>
</tr>
<tr>
<td>944K</td>
<td>EC000</td>
<td></td>
</tr>
<tr>
<td>960K</td>
<td>F0000</td>
<td>Reserved</td>
</tr>
<tr>
<td>976K</td>
<td>F4000</td>
<td></td>
</tr>
<tr>
<td>992K</td>
<td>F8000</td>
<td>48K Base System ROM</td>
</tr>
<tr>
<td>1008K</td>
<td>FC000</td>
<td></td>
</tr>
</tbody>
</table>

System Memory Map for 64/256K System Board (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 adapter are located in "Appendix G: Switch Settings."

1-16 System Unit
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 512 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 five system unit expansion slots, assuming two low-power Schottky loads per slot. The IBM I/O adapters typically use only one load.

The following pages describe the system board’s I/O channel.
I/O Channel Diagram

1-18 System Unit
## I/O Channel Description

The following is a description of the IBM Personal Computer 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>Signal</td>
<td>I/O</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-----</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</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>I/O CHK</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>Signal</td>
<td>I/O</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-----</td>
<td>-------------</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>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>Signal</td>
<td>I/O Description</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>----------------</td>
<td></td>
</tr>
<tr>
<td>AEN</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>
<td></td>
</tr>
<tr>
<td>T/C</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>
<td></td>
</tr>
</tbody>
</table>

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 — Controller 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
Power Supply

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

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

It is a dc-switching power supply designed for continuous operation at 63.5 watts. It has a fused 120-Vac input and provides four regulated dc output voltages: 7 A of +5 Vdc, 2 A of +12 Vdc, 0.3 A of −5 Vdc, and 0.25 A of −12 Vdc. These outputs are over-voltage, over-current, open-circuit, and short-circuit protected. If a dc overload or over-voltage condition occurs, all dc outputs are shut down as long as the condition exists.

The +5 Vdc powers the logic on the system board and the diskette drives and allows approximately 4 A of +5 Vdc for the adapters in the system-unit expansion slots. The +12 Vdc power level is designed to power the system’s dynamic memory and the two internal 5-1/4 inch diskette drive motors. It is assumed that only one drive is active at a time. The −5 Vdc level is designed for dynamic memory bias voltage; it tracks the +5 Vdc and +12 Vdc very quickly at power-on and has a longer decay on power-off than the +5 Vdc and +12 Vdc outputs. The +12 Vdc and −12 Vdc are used for powering the EIA drivers on the communications adapters. All four power levels are bussed across the five system-unit expansion slots.
Operating Characteristics

Input Requirements

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

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

Vdc Output

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

<table>
<thead>
<tr>
<th>Voltage (Vdc)</th>
<th>Current (Amps)</th>
<th>Regulation (Tolerance)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nominal</td>
<td>Minimum</td>
<td>Maximum</td>
</tr>
<tr>
<td>+5.0</td>
<td>2.3</td>
<td>7.0</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>2.0</td>
</tr>
<tr>
<td>-12.0</td>
<td>0.0</td>
<td>0.25</td>
</tr>
</tbody>
</table>

Vac Output

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

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

Primary (Input)

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

<table>
<thead>
<tr>
<th>Voltage (Nominal Vac)</th>
<th>Type Protection</th>
<th>Rating (Amps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>120</td>
<td>Fuse</td>
<td>2</td>
</tr>
</tbody>
</table>

Secondary (Output)

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

Power-Good Signal

When the power supply is turned on after it has been off for a minimum of 5 seconds, it generates a power-good signal which indicates that there is adequate power for processing. When the four output voltages are above the minimum sense levels, as described below, the signal sequences to a TTL-compatible up level (2.4 Vdc to 5.5 Vdc), which is capable of sourcing 60 µA. When any of the four output voltages is below its minimum sense level or above its maximum sense level, the power good signal will be a TTL-compatible down level (0.0 Vdc to 0.4 Vdc) capable of sourcing 500 µA. The power good signal has a turn-on delay of 100 ms after the output voltages have reached their respective minimum sense levels.
<table>
<thead>
<tr>
<th>Output Voltage</th>
<th>Under-Voltage Nominal Sense Level</th>
<th>Over-Voltage Nominal Sense Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>+5 Vdc</td>
<td>+4.0 Vdc</td>
<td>+5.9 Vdc</td>
</tr>
<tr>
<td>-5 Vdc</td>
<td>-4.0 Vdc</td>
<td>-5.9 Vdc</td>
</tr>
<tr>
<td>+12 Vdc</td>
<td>+9.6 Vdc</td>
<td>+14.2 Vdc</td>
</tr>
<tr>
<td>-12 Vdc</td>
<td>-9.6 Vdc</td>
<td>-14.2 Vdc</td>
</tr>
</tbody>
</table>

**Cassette Interface**

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

A 2 by 2 Berg pin and a jumper are used on the cassette ‘data out’ line. The jumper allows use of the ‘data out’ line as a 0.075-Vdc microphone input when placed across the M and C pins of the Berg connector. A 0.68-Vdc auxiliary input to the cassette recorder is available when the jumper is placed across the A and C pins of the Berg connector. The “System Board Component Diagram” shows the location of the cassette Berg pins.
Cassette Circuit Block Diagrams

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

Cassette Interface Read Hardware Block Diagram

Cassette Interface Write Hardware Block Diagram
Cassette Motor Control Block Diagram
## 5-Pin DIN Connector

### Pin Signal Electrical Characteristics

<table>
<thead>
<tr>
<th>Pin</th>
<th>Signal</th>
<th>Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Motor Control Common from Relay</td>
<td>Common from Relay</td>
</tr>
<tr>
<td>2</td>
<td>Ground</td>
<td>Relay N.O. (6 Vdc at 1A)</td>
</tr>
<tr>
<td>3</td>
<td>Motor Control Relay N.O.</td>
<td>500nA at ±13V - at 1,000 - 2,000 Baud</td>
</tr>
<tr>
<td>4</td>
<td>Data In</td>
<td>Data Out</td>
</tr>
<tr>
<td>5</td>
<td>Data Out (Microphone or Auxiliary)</td>
<td>250 μA at 0.68 Vdc or 0.075 Vdc</td>
</tr>
</tbody>
</table>

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

**Data out can be chosen using a jumper located on the system board.

(Auxiliary — 0.68 Vdc or Microphone — 0.075 Vdc).

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

---

**Cassette Interface Connector Specifications**
Notes:
The IBM Personal Computer Math Coprocessor enables the IBM Personal Computer to perform high speed arithmetic, logarithmic functions, and trigonometric operations with extreme accuracy.

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

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

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

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

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

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

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Bits</th>
<th>Significant Digits (Decimal)</th>
<th>Approximate Range (decimal)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word Integer</td>
<td>16</td>
<td>4</td>
<td>-32,768 ≤ X ≤ +32,767</td>
</tr>
<tr>
<td>Short Integer</td>
<td>32</td>
<td>9</td>
<td>-2×10^9 ≤ X ≤ +2×10^9</td>
</tr>
<tr>
<td>Long Integer</td>
<td>64</td>
<td>18</td>
<td>-9×10^{18} ≤ X ≤ +9×10^{18}</td>
</tr>
<tr>
<td>Packed Decimal</td>
<td>80</td>
<td>18</td>
<td>-9.9...99 ≤ X ≤ +9.9...99 (18 digits)</td>
</tr>
<tr>
<td>Short Real*</td>
<td>32</td>
<td>6-7</td>
<td>8.43×10^{-37} ≤</td>
</tr>
<tr>
<td>Long Real*</td>
<td>64</td>
<td>15-16</td>
<td>4.19×10^{-307} ≤</td>
</tr>
<tr>
<td>Temporary Real</td>
<td>80</td>
<td>19</td>
<td>3.4×10^{-4932} ≤</td>
</tr>
</tbody>
</table>

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

Data Types

1-34 Coprocessor
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 at 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

79 64 63 0

<table>
<thead>
<tr>
<th>Exponent</th>
<th>Significand</th>
</tr>
</thead>
</table>

Sign

Register Structure

1-38 Coprocessor
**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 Diagram](image)

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

**Status Word Format**

Coprocessor 1-39
Control Word

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

15

  IC RC PC IEM PM UM OM ZM DM IM

Exception Masks (1 = Exception is Masked)
- Invalid Operation
- Denormalized Operand
- Zerodivide
- 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 \( \infty \))
- 10 = Round Up (toward \( \infty \))
- 11 = Chop (Truncate Toward Zero)

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

Control Word Format
Tag Word

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

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>
<th>INSTRUCTION OPCODE$^{(2)}$</th>
<th>INSTRUCTION ADDRESS$^{(1)}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>10</td>
</tr>
</tbody>
</table>

$^{(1)}$ 20-bit physical address
$^{(2)}$ 11 least significant bits of opcode: 5 most significant bits are always COPROCESSOR HOOK (11011B)

Exception Pointers Format
Number System

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

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

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

Coprocessor Number System

1-42 Coprocessor
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>Typical</td>
</tr>
<tr>
<td>(no operands)</td>
<td>14</td>
</tr>
</tbody>
</table>

FADD

Addition

FADD // source/destination,source

FADD P 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>Typical</td>
</tr>
<tr>
<td>//ST,ST(i)/ST(i),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>

1-44 Coprocessor
### FADDP

<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></td>
<td></td>
</tr>
<tr>
<td>ST(I),ST</td>
<td>90</td>
<td>75-105</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

### FIADD

<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></td>
<td></td>
</tr>
<tr>
<td>word-integer</td>
<td>120+EA</td>
<td>102-137+EA</td>
<td>2</td>
<td>2-4</td>
</tr>
<tr>
<td>short-integer</td>
<td>125+EA</td>
<td>108-143+EA</td>
<td>4</td>
<td>2-4</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>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></td>
<td></td>
</tr>
<tr>
<td>packed-decimal</td>
<td>300+EA</td>
<td>290-310+EA</td>
<td>10</td>
<td>2-4</td>
</tr>
</tbody>
</table>

Coprocessor 1-45
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>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: I</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands</td>
<td>Execution Clocks</td>
</tr>
<tr>
<td></td>
<td>Typical</td>
</tr>
<tr>
<td>(no operands)</td>
<td>15</td>
</tr>
</tbody>
</table>

FCLEX/FNCLEX

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

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

1-46 Coprocessor
FCOM

FCOM/ /source

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

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

C3 C0 Order

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></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 &gt; 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-reals</td>
<td>68+EA</td>
</tr>
<tr>
<td>long-reals</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></td>
<td>Typical</td>
</tr>
<tr>
<td>(no operands)</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></td>
<td>Typical</td>
</tr>
<tr>
<td>(no operands)</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>
</tbody>
</table>
| //ST(i),ST | 198     | 193-203 | 0   | 2   | FDIV
| short-real | 220+EA  | 215-225+EA | 4   | 2.4 | FDIV DISTANCE
| long-real  | 225+EA  | 220-230+EA | 8   | 2.4 | FDIV ARC[DI] |

<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>
</tbody>
</table>
| word-integer | 230+EA  | 224-238+EA | 2   | 2.4 | FIDIV SURVEY_OBSERVATIONS
| short-integer | 236+EA  | 230-243+EA | 4   | 2.4 | FIDIVدراسة 角[DII] |

Coprocessor 1-49
FDIVR

Reversed Division

FDIVR / /source/ destination,source

FDIVRP destination,source

FIDIVR source

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

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

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

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

1-50 Coprocessor
FENI/FNENI

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

<table>
<thead>
<tr>
<th>FENI/FNENI (no operands)</th>
<th>Exceptions: None</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operands</strong></td>
<td><strong>Execution Clocks</strong></td>
</tr>
<tr>
<td></td>
<td>Typical</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><strong>Operands</strong></td>
<td><strong>Execution Clocks</strong></td>
</tr>
<tr>
<td></td>
<td>Typical</td>
</tr>
<tr>
<td>ST(i)</td>
<td>186</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><strong>Operands</strong></td>
<td><strong>Execution Clocks</strong></td>
</tr>
<tr>
<td></td>
<td>Typical</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-inter</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-inter</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>

1-52 Coprocessor
FINIT/FNINIT

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

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

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

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

FIST destination

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

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

FISTP

FISTP destination

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

<table>
<thead>
<tr>
<th>FISTP</th>
<th>Exceptions: I, P</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operand</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 8088</th>
<th>Bytes</th>
<th>Coding Example</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Typical</td>
<td>Range</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ST(i)</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>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>FLD ST(0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>FLD READING[SI].PRESSURE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>FLD [BP].TEMPERATURE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>FLD SAVEREADING</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 8088</th>
<th>Bytes</th>
<th>Coding Example</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Typical</td>
<td>Range</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-bytes</td>
<td>10+EA</td>
<td>7-14+EA</td>
<td>2</td>
<td>2-4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>FLDCW CONTROL_WORD</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>Operation</th>
<th>Execution Clocks</th>
<th>Transfers</th>
<th>Bytes</th>
<th>Coding Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLDENV</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operands</td>
<td>Typical</td>
<td>Range</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14-bytes</td>
<td>40+EA</td>
<td>35-45+EA</td>
<td>14</td>
<td>FLDENV [BP+6]</td>
</tr>
</tbody>
</table>

FLDLG2

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

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

FLDLN2

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

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

1-56 Coprocessor
FLDL2E

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

<table>
<thead>
<tr>
<th>FLDL2E (no operands)</th>
<th>Exceptions: I</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>18</td>
</tr>
</tbody>
</table>

FLDL2T

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

<table>
<thead>
<tr>
<th>FLDL2T (no operands)</th>
<th>Exceptions: I</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>19</td>
</tr>
</tbody>
</table>

FLDPI

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

<table>
<thead>
<tr>
<th>FLDPI (no operands)</th>
<th>Exceptions: I</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>19</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: 1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operands</strong></td>
<td><strong>Execution Clocks</strong></td>
</tr>
<tr>
<td>(no operands)</td>
<td>Typical 14</td>
</tr>
</tbody>
</table>

**FLD1**

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

<table>
<thead>
<tr>
<th>FLD1 (no operands)</th>
<th>Exceptions: 1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operands</strong></td>
<td><strong>Execution Clocks</strong></td>
</tr>
<tr>
<td>(no operands)</td>
<td>Typical 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>Operands</td>
<td>Execution Clocks</td>
</tr>
<tr>
<td></td>
<td>Typical</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>Operands</td>
<td>Execution Clocks</td>
</tr>
<tr>
<td></td>
<td>Typical</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>Operands</td>
<td>Execution Clocks</td>
</tr>
<tr>
<td></td>
<td>Typical</td>
</tr>
<tr>
<td>word-integer</td>
<td>130+EA</td>
</tr>
<tr>
<td>short-integer</td>
<td>136+EA</td>
</tr>
</tbody>
</table>
FNOP

FNOP (no operation) stores the stack top 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 \( \theta = \text{ARCTAN} \left( \frac{Y}{X} \right) \). 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>
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>Operand</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>Operand</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>Operand</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>Operands</td>
<td>Execution Clocks</td>
</tr>
<tr>
<td></td>
<td>Typical</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:

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

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>Operands</td>
<td>Execution Clocks</td>
</tr>
<tr>
<td></td>
<td>Typical</td>
</tr>
<tr>
<td>(no operands)</td>
<td>35</td>
</tr>
</tbody>
</table>

FSQRT

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

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

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

FST destination

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

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

FSTCW/FNSTCW

FSTCW/FNSTCW destination

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

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

FSTENV/FNSTENV destination

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

<table>
<thead>
<tr>
<th>FSTENV/FNSTENV</th>
<th>Exceptions: None</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operands</strong></td>
<td><strong>Execution Clocks</strong></td>
</tr>
<tr>
<td></td>
<td>Typical</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><strong>Operands</strong></td>
<td><strong>Execution Clocks</strong></td>
</tr>
<tr>
<td></td>
<td>Typical</td>
</tr>
<tr>
<td>ST(i)</td>
<td>20</td>
</tr>
<tr>
<td>short-real</td>
<td>89+EA</td>
</tr>
<tr>
<td>long-real</td>
<td>102+EA</td>
</tr>
<tr>
<td>temp-real</td>
<td>55+EA</td>
</tr>
</tbody>
</table>

1-64 Coprocessor
**FSTSW/FNSTSW**

**FSTSW/FNSTSW destination**

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

<table>
<thead>
<tr>
<th>FSTSW/FNSTSW</th>
<th>Exceptions: None</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operands</strong></td>
<td><strong>Execution Clocks</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Typical</strong></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><strong>Operands</strong></td>
<td><strong>Execution Clocks</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Typical</strong></td>
</tr>
<tr>
<td>//ST,ST(i)/ST(i),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>
FSUBP

Exceptions: I, D, O, U, 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>Range</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ST(i),ST</td>
<td>90</td>
<td>75-105</td>
<td>0</td>
<td>FSUBP ST(2),ST</td>
</tr>
</tbody>
</table>

FISUB

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>Range</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Typical</td>
<td>Range</td>
<td></td>
<td></td>
</tr>
<tr>
<td>word-integer</td>
<td>120+EA</td>
<td>102-137+EA</td>
<td>2</td>
<td>FISUB BASE_FREQUENCY</td>
</tr>
<tr>
<td>short-integer</td>
<td>125+EA</td>
<td>108-143+EA</td>
<td>4</td>
<td>FISUB TRAIN_SIZE[DI]</td>
</tr>
</tbody>
</table>

FSUBR

Reversed Subtraction

FSUBR / /source/destination,source

FSUBRP destination,source

FISUBR source

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

FSUBR

Exceptions: I, D, O, U, 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>Range</td>
<td></td>
<td></td>
</tr>
<tr>
<td>//ST,ST(i)/ST(i),ST</td>
<td>87</td>
<td>70-100</td>
<td>0</td>
<td>FSUBR ST,ST(1)</td>
</tr>
<tr>
<td>short-real</td>
<td>105+EA</td>
<td>90-120+EA</td>
<td>4</td>
<td>FSUBR VECTOR[SI]</td>
</tr>
<tr>
<td>long-real</td>
<td>110+EA</td>
<td>95-125+EA</td>
<td>8</td>
<td>FSUBR [BX]:INDEX</td>
</tr>
</tbody>
</table>

1-66 Coprocessor
FSUBRP

<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>ST(i),ST</td>
<td>90 75-105</td>
<td>0</td>
<td>2</td>
<td>FSUBRP ST(1),ST</td>
</tr>
</tbody>
</table>

FISUBR

<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>word-integer</td>
<td>120+EA 103-139+EA</td>
<td>2</td>
<td>2-4</td>
<td>FISUBR FLOOR[BX] [SI]</td>
</tr>
<tr>
<td>short-integer</td>
<td>125+EA 109-144+EA</td>
<td>4</td>
<td>2-4</td>
<td>FISUBR BALANCE</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.

FTST (no operands)

<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>42 38-48</td>
<td>0</td>
<td>2</td>
<td>FTST</td>
</tr>
</tbody>
</table>

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

FWAIT (processor instruction)

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

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

FXAM

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

<table>
<thead>
<tr>
<th>FXAM</th>
<th>Exceptions: None</th>
</tr>
</thead>
<tbody>
<tr>
<td><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>17</td>
</tr>
<tr>
<td>Condition Code</td>
<td>Interpretation</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>C3  C2  C1  C0</td>
<td>+ Unnormal</td>
</tr>
<tr>
<td>0    0    0  0</td>
<td>+ NaN</td>
</tr>
<tr>
<td>0  0  0  1</td>
<td>- Unnormal</td>
</tr>
<tr>
<td>0  0  1  0</td>
<td>- NaN</td>
</tr>
<tr>
<td>0  0  1  1</td>
<td>+ Normal</td>
</tr>
<tr>
<td>0  1  0  0</td>
<td>+ ∞</td>
</tr>
<tr>
<td>0  1  0  1</td>
<td>- Normal</td>
</tr>
<tr>
<td>0  1  1  0</td>
<td>- ∞</td>
</tr>
<tr>
<td>0  1  1  1</td>
<td>Empty</td>
</tr>
<tr>
<td>1  0  0  0</td>
<td>+ 0</td>
</tr>
<tr>
<td>1  0  0  1</td>
<td>Empty</td>
</tr>
<tr>
<td>1  0  1  0</td>
<td>- 0</td>
</tr>
<tr>
<td>1  0  1  1</td>
<td>Empty</td>
</tr>
<tr>
<td>1  1  0  0</td>
<td>+ Denormal</td>
</tr>
<tr>
<td>1  1  0  1</td>
<td>Empty</td>
</tr>
<tr>
<td>1  1  1  0</td>
<td>- Denormal</td>
</tr>
<tr>
<td>1  1  1  1</td>
<td>Empty</td>
</tr>
</tbody>
</table>

**FXCH**

**FXCH/ /destination**

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

<table>
<thead>
<tr>
<th>FXCH</th>
<th>Exceptions: 1</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>12</td>
</tr>
</tbody>
</table>
FXTRACT

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

<table>
<thead>
<tr>
<th>FXTRACT</th>
<th>Exceptions: I</th>
</tr>
</thead>
<tbody>
<tr>
<td>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>

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.

\[ \log_2 2 \cdot \log_2 X \]

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

1-70 Coprocessor
**FYL2XP1**

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

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

**F2XM1**

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

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

<table>
<thead>
<tr>
<th>F2XM1</th>
<th>Exceptions: U, P (operands not checked)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operands</strong></td>
<td><strong>Execution Clocks</strong></td>
</tr>
<tr>
<td>(no operands)</td>
<td>Typical 8088</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.
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>4</td>
<td>Ground</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>+5 Volts</td>
<td>+5 Vdc</td>
</tr>
</tbody>
</table>

#### Power Supply Voltages

<table>
<thead>
<tr>
<th>Voltage</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>+5 Vdc</td>
</tr>
</tbody>
</table>

---

1-78  Keyboard
Expansion Unit

The expansion unit option upgrades the IBM Personal Computer 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.

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

1-80  Expansion Unit
Expansion Channel

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

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

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

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

(*) With respect to the system unit.

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

The expansion unit dc power supply is a 130-watt, 4 voltage level switching regulator. It is integrated into the expansion unit and supplies power for the expansion unit, and its options. The supply provides 15 A of +5 Vdc, plus or minus 5%, 4.2A 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 power supply is located at the right rear of the expansion unit. It supplies operating voltages to the expansion board, and provides two separate connections for power to the fixed disk drives. The nominal power requirements and output voltages are listed in the following tables:

<table>
<thead>
<tr>
<th>Voltage (Vac at 50/60 Hz)</th>
<th>Frequency (Hz)</th>
<th>Current (Amps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nominal</td>
<td>Minimum</td>
<td>Maximum</td>
</tr>
<tr>
<td>110</td>
<td>90</td>
<td>137</td>
</tr>
</tbody>
</table>

Input Requirements

<table>
<thead>
<tr>
<th>Voltage (Vdc)</th>
<th>Current (Amps)</th>
<th>Regulation (Tolerance)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nominal</td>
<td>Minimum</td>
<td>Maximum</td>
</tr>
<tr>
<td>+5.0</td>
<td>2.3</td>
<td>15.0</td>
</tr>
<tr>
<td>−5.0</td>
<td>0.0</td>
<td>0.3</td>
</tr>
<tr>
<td>+12.0</td>
<td>0.4</td>
<td>4.2</td>
</tr>
<tr>
<td>−12.0</td>
<td>0.0</td>
<td>0.25</td>
</tr>
</tbody>
</table>

Vdc Output

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

Vac Output

1-82 Expansion Unit
Power Supply Connectors and Pin Assignments

The power connector on the expansion 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>
Extender Card

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

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

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

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

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

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

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

(All values in hex)

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

1-86 Expansion Unit
Extender Card Block Diagram
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 216</td>
<td>Read low-order address bits (A0 - A7)</td>
</tr>
<tr>
<td>(*) Example:</td>
<td>Write to memory location F123:4 = 00</td>
</tr>
<tr>
<td></td>
<td>Read Port 215 = 12</td>
</tr>
<tr>
<td></td>
<td>Read Port 216 = 34</td>
</tr>
</tbody>
</table>

(All values in hex)

The expansion unit is automatically enabled upon power-up. The expansion unit and the system unit will be written to, if the expansion unit is not disabled when writing to FXXXX. However, the system unit and the expansion unit are read back separately.

1-88 Expansion Unit
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.

![Interface Diagram]

<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**

1-90  Expansion Unit
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.
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>(1) Print Method:</strong></td>
<td>Serial-impact dot matrix</td>
</tr>
<tr>
<td><strong>(2) Print Speed:</strong></td>
<td>80 cps</td>
</tr>
<tr>
<td><strong>(3) Print Direction:</strong></td>
<td>Bidirectional with logical seeking</td>
</tr>
<tr>
<td><strong>(4) Number of Pins in Head:</strong></td>
<td>9</td>
</tr>
<tr>
<td><strong>(5) Line Spacing:</strong></td>
<td>1/16 inch (4.23 mm) or programmable</td>
</tr>
<tr>
<td><strong>(6) Printing Characteristics</strong></td>
<td></td>
</tr>
<tr>
<td>Matrix:</td>
<td>9 x 9</td>
</tr>
<tr>
<td>Character Set:</td>
<td>Full 96-character ASCII with descenders plus 9 international characters/symbols.</td>
</tr>
<tr>
<td>Graphic Character:</td>
<td>See “Additional Printer Specifications”</td>
</tr>
<tr>
<td><strong>(7) Printing Sizes</strong></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><strong>(8) Media Handling</strong></td>
<td></td>
</tr>
<tr>
<td>Paper Feed:</td>
<td>Adjustable sprocket pin feed</td>
</tr>
<tr>
<td>Paper Width Range:</td>
<td>4 inch (101.6 mm) to 10 inch (254 mm)</td>
</tr>
<tr>
<td>Copies:</td>
<td>One original plus two carbon copies (total thickness not to exceed 0.012 inch (0.3 mm)). Minimum paper thickness is 0.0025 inch (0.064 mm).</td>
</tr>
<tr>
<td>Paper Path:</td>
<td>Rear</td>
</tr>
<tr>
<td><strong>(9) Interfaces</strong></td>
<td></td>
</tr>
<tr>
<td>Standard:</td>
<td>Parallel 8-bit Data and Control Lines</td>
</tr>
<tr>
<td><strong>(10) Inked Ribbon</strong></td>
<td></td>
</tr>
<tr>
<td>Color:</td>
<td>Black</td>
</tr>
<tr>
<td>Type:</td>
<td>Cartridge</td>
</tr>
<tr>
<td>Life Expectancy:</td>
<td>3 million characters</td>
</tr>
<tr>
<td><strong>(11) Environmental Conditions</strong></td>
<td></td>
</tr>
<tr>
<td>Operating Temperature Range:</td>
<td>41 to 95°F (5 to 35°C)</td>
</tr>
<tr>
<td>Operating Humidity:</td>
<td>10 to 80% non-condensing</td>
</tr>
<tr>
<td><strong>(12) Power Requirement</strong></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><strong>(13) Physical Characteristics</strong></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.

(6) Printing Characteristics
IBM 80 CPS Graphics Printer
Extra Character Set: Set 1
   Additional ASCII numbers 160 to 175 contain European characters. Numbers 176 to 223 contain graphic characters. Numbers 224 to 239 contain selected Greek characters. Numbers 240 to 255 contain math and extra symbols.
   Set 2
   The difference in set 2 are ASCII numbers 3, 4, 5, 6, and 21. ASCII numbers 128 to 175 contain European characters.

Graphics: There are 20 block characters and programmable graphics.

(7) Printing Sizes

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

Additional Printer Specifications
Setting the DIP Switches

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

Location of Printer DIP Switches

<table>
<thead>
<tr>
<th>Switch Number</th>
<th>Function</th>
<th>On</th>
<th>Off</th>
<th>Factory-Set Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1</td>
<td>Not Applicable</td>
<td>—</td>
<td>—</td>
<td>On</td>
</tr>
<tr>
<td>1-2</td>
<td>CR</td>
<td>Print Only</td>
<td>Print &amp; Line Feed</td>
<td>On</td>
</tr>
<tr>
<td>1-3</td>
<td>Buffer Full</td>
<td>Print Only</td>
<td>Print &amp; Line Feed</td>
<td>Off</td>
</tr>
<tr>
<td>1-4</td>
<td>Cancel Code</td>
<td>Invalid</td>
<td>Valid</td>
<td>Off</td>
</tr>
<tr>
<td>1-5</td>
<td>Delete Code</td>
<td>Invalid</td>
<td>Valid</td>
<td>On</td>
</tr>
<tr>
<td>1-6</td>
<td>Error Buzzer</td>
<td>Sounds</td>
<td>Does Not Sound</td>
<td>On</td>
</tr>
<tr>
<td>1-7</td>
<td>Character Generator</td>
<td>N.A.</td>
<td>Graphic Patterns Select</td>
<td>Off</td>
</tr>
<tr>
<td>1-8</td>
<td>SLCT IN Signal</td>
<td>Fixed</td>
<td>Not Fixed</td>
<td>On</td>
</tr>
</tbody>
</table>

Functions and Conditions of DIP Switch 1 (Matrix)
### Switches Factory-Set

<table>
<thead>
<tr>
<th>Switch Number</th>
<th>Function</th>
<th>On Condition</th>
<th>Off Condition</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 2 (Matrix)

<table>
<thead>
<tr>
<th>Switch Number</th>
<th>Function</th>
<th>On Condition</th>
<th>Off Condition</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>Not Applicable</td>
<td>--</td>
<td>--</td>
<td>On</td>
</tr>
<tr>
<td>1-6</td>
<td>Error Buzzer</td>
<td>Sound</td>
<td>Does Not Sound</td>
<td>On</td>
</tr>
<tr>
<td>1-7</td>
<td>Character Generator</td>
<td>Set 2</td>
<td>Set 1</td>
<td>Off</td>
</tr>
<tr>
<td>1-8</td>
<td>SLCT IN Signal</td>
<td>Fixed Internally</td>
<td>Not Fixed Internally</td>
<td>On</td>
</tr>
</tbody>
</table>

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

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

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

Specifications:

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

Connector: Plug: 57-30360 (Amphenol)

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

Data transfer sequence:

![Parallel Interface Timing Diagram](image-url)
<table>
<thead>
<tr>
<th>Signal Pin No.</th>
<th>Return Pin No.</th>
<th>Signal</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>19</td>
<td>STROBE</td>
<td>In</td>
<td>STROBE pulse to read data in. Pulse width must be more than 0.5 μs at receiving terminal. The signal level is normally “high”; read-in of data is performed at the “low” 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 “high” level when data is logical “1” and “low” when logical “0.”</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; “low” 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 “high” signal indicates that the printer cannot receive data. The signal becomes “high” in the following cases: 1. During data entry. 2. During printing operation. 3. In “offline” state. 4. During printer error status.</td>
</tr>
</tbody>
</table>

Connector Pin Assignment and Descriptions of Interface Signals (Part 1 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>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>OV</td>
<td>—</td>
<td>Logic GND level.</td>
</tr>
<tr>
<td>17</td>
<td>—</td>
<td>CHASSIS-GND</td>
<td>—</td>
<td>Printer chassis GND. In the printer, the chassis GND and the logic GND are isolated from each other.</td>
</tr>
<tr>
<td>18</td>
<td>—</td>
<td>NC</td>
<td>—</td>
<td>Not used.</td>
</tr>
<tr>
<td>19-30</td>
<td>—</td>
<td>GND</td>
<td>—</td>
<td>&quot;Twisted-Pair Return&quot; signal; GND level.</td>
</tr>
<tr>
<td>31</td>
<td>—</td>
<td>INIT</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 &quot;low&quot; when the printer is in &quot;Paper End&quot; state, &quot;Offline&quot; state and &quot;Error&quot; state.</td>
</tr>
<tr>
<td>33</td>
<td>—</td>
<td>GND</td>
<td>—</td>
<td>Same as with pin numbers 19 to 30.</td>
</tr>
<tr>
<td>34</td>
<td>—</td>
<td>NC</td>
<td>—</td>
<td>Not used.</td>
</tr>
<tr>
<td>35</td>
<td></td>
<td></td>
<td></td>
<td>Pulled up to +5 Vdc through 4.7 k-ohms resistance.</td>
</tr>
<tr>
<td>36</td>
<td>—</td>
<td>SLCT IN</td>
<td>In</td>
<td>Data entry to the printer is possible only when the level of this signal is &quot;low.&quot; (Internal fixing can be carried out with DIP SW 1-8. The condition at the time of shipment is set &quot;low&quot; for this signal.)</td>
</tr>
</tbody>
</table>

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

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

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

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

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

On the following pages 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 tab stop. Tab stops are set with ESC D. No tab stops are set when the printer is powered on. (Graphics Printer sets a tab stop every 8 columns when powered on.)</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>Spacing 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><strong>SO</strong></td>
<td><strong>Shift Out (Double Width)</strong>&lt;br&gt;Changes the printer to the Double Width print mode.&lt;br&gt;Note: A Carriage Return, Line Feed or DC4 cancels Double Width print mode.&lt;br&gt;Example: LPRINT CHR$(14);</td>
</tr>
<tr>
<td><strong>SI</strong></td>
<td><strong>Shift In (Compressed)</strong>&lt;br&gt;Changes the printer to the Compressed Character print mode.&lt;br&gt;Example: LPRINT CHR$(15);</td>
</tr>
<tr>
<td><strong>DC1</strong></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: LPRINT CHR$(17);</td>
</tr>
<tr>
<td><strong>DC2</strong></td>
<td><strong>Device Control 2 (Compressed Off)</strong>&lt;br&gt;Stops printing in the Compressed print mode.&lt;br&gt;Example: LPRINT CHR$(18);</td>
</tr>
<tr>
<td><strong>DC3</strong></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: LPRINT CHR$(19);</td>
</tr>
<tr>
<td><strong>DC4</strong></td>
<td><strong>Device Control 4 (Double Width Off)</strong>&lt;br&gt;Stops printing in the Double Width print mode.&lt;br&gt;Example: LPRINT CHR$(20);</td>
</tr>
<tr>
<td><strong>CAN</strong></td>
<td><strong>Cancel</strong>&lt;br&gt;Clears the printer buffer. Control codes, except SO, remain in effect.&lt;br&gt;Example: LPRINT CHR$(24);</td>
</tr>
<tr>
<td><strong>ESC</strong></td>
<td><strong>Escape</strong>&lt;br&gt; Lets the printer know that the next data sent is a printer command.&lt;br&gt;(See the following list of commands.)&lt;br&gt;Example: LPRINT CHR$(27);</td>
</tr>
<tr>
<td>Printer Code</td>
<td>Printer Function</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>ESC –</td>
<td>Escape Minus (Underline)</td>
</tr>
<tr>
<td></td>
<td>Format: ESC –;n;</td>
</tr>
<tr>
<td></td>
<td>(Graphics Printer only)</td>
</tr>
<tr>
<td></td>
<td>ESC - followed by a 1, prints all of the following data with an underline.</td>
</tr>
<tr>
<td></td>
<td>ESC - followed by a 0 (zero), cancels the Underline print mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>LPRINT CHR$(27);CHR$(45);CHR$(1);</td>
</tr>
<tr>
<td>ESC 0</td>
<td>Escape Zero (1/8-Inch Line Feeding)</td>
</tr>
<tr>
<td></td>
<td>Changes paper feeding to 1/8 inch.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>LPRINT CHR$(27);CHR$(48);</td>
</tr>
<tr>
<td>ESC 1</td>
<td>Escape 1 (7/72-Inch Line Feeding)</td>
</tr>
<tr>
<td></td>
<td>Changes paper feed to 7/72 inch.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>LPRINT CHR$(27);CHR$(49);</td>
</tr>
<tr>
<td>ESC 2</td>
<td>Escape Two (Starts Variable Line Feeding)</td>
</tr>
<tr>
<td></td>
<td>ESC 2 is an execution command for ESC A. If no ESC A command has been given, line feeding returns to 1/6-inch.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>LPRINT CHR$(27);CHR$(50);</td>
</tr>
<tr>
<td>ESC 3</td>
<td>Escape Three (Variable Line Feeding)</td>
</tr>
<tr>
<td></td>
<td>Format: ESC 3;n;</td>
</tr>
<tr>
<td></td>
<td>(Graphics Printer only)</td>
</tr>
<tr>
<td></td>
<td>Changes the-paper feeding to n/216-inch. The example below sets the paper feeding to 54/216 (1/4) inch. The value of n must be between 1 and 255.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>LPRINT CHR$(27);CHR$(51);CHR$(54);</td>
</tr>
<tr>
<td>ESC 6</td>
<td>Escape Six (Select Character Set 2)</td>
</tr>
<tr>
<td></td>
<td>(Graphics Printer only)</td>
</tr>
<tr>
<td></td>
<td>Selects character set 2. (See &quot;Printer Character Set 2.&quot;)</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>LPRINT CHR$(27);CHR$(54);</td>
</tr>
<tr>
<td>ESC 7</td>
<td>Escape Seven (Select Character Set 1.)</td>
</tr>
<tr>
<td></td>
<td>(Graphics Printer only)</td>
</tr>
<tr>
<td></td>
<td>Selects character set 1. (See &quot;Printer Character Set 1.&quot;)</td>
</tr>
<tr>
<td></td>
<td>Character set 1 is selected when the printer is powered on or reset.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>LPRINT CHR$(27);CHR$(55);</td>
</tr>
<tr>
<td>ESC 8</td>
<td>Escape Eight (Ignore Paper End)</td>
</tr>
<tr>
<td></td>
<td>Allows the printer to print to the end of the paper. The printer ignores the Paper End switch.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>LPRINT CHR$(27);CHR$(56);</td>
</tr>
<tr>
<td>Printer Code</td>
<td>Printer Function</td>
</tr>
<tr>
<td>-------------</td>
<td>---------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ESC 9</td>
<td><strong>Escape Nine (Cancel Ignore Paper End)</strong></td>
</tr>
<tr>
<td></td>
<td>Cancels the Ignore Paper End command. ESC 9 is selected when the printer is</td>
</tr>
<tr>
<td></td>
<td>powered on or reset.</td>
</tr>
<tr>
<td></td>
<td>Example</td>
</tr>
<tr>
<td></td>
<td>LPRINT CHR$(27);CHR$(57);</td>
</tr>
<tr>
<td>ESC &lt;</td>
<td><strong>Escape Less Than (Home Head)</strong></td>
</tr>
<tr>
<td></td>
<td>(Graphics Printer only)</td>
</tr>
<tr>
<td></td>
<td>The print head will return to the left margin to print the line following</td>
</tr>
<tr>
<td></td>
<td>ESC &lt;. This will occur for one line only.</td>
</tr>
<tr>
<td></td>
<td>Example</td>
</tr>
<tr>
<td></td>
<td>LPRINT CHR$(27);CHR$(60);</td>
</tr>
<tr>
<td>ESC A</td>
<td><strong>Escape A (Sets Variable Line Feeding)</strong></td>
</tr>
<tr>
<td></td>
<td>Format: ESC A;n;</td>
</tr>
<tr>
<td></td>
<td>Escape A sets the line-feed to n/72-inch. The example below tells the printer</td>
</tr>
<tr>
<td></td>
<td>to set line feeding to 24/72-inch. ESC 2 must be sent to the printer before the</td>
</tr>
<tr>
<td></td>
<td>line feeding will change. For example, ESC A;24 (text) ESC 2 (text). The text</td>
</tr>
<tr>
<td></td>
<td>following ESC A;24 will space at the previously set line-feed increments. The</td>
</tr>
<tr>
<td></td>
<td>text following ESC 2 will be printed with new line-feed increments of 24/72-inch.</td>
</tr>
<tr>
<td></td>
<td>Any increment between 1/72 and 85/72 may be used.</td>
</tr>
<tr>
<td></td>
<td>Example</td>
</tr>
<tr>
<td></td>
<td>LPRINT CHR$(27);CHR$(65);CHR$(24);CHR$(27);CHR$(50);</td>
</tr>
<tr>
<td>ESC B</td>
<td><strong>Escape B (Set Vertical Tabs)</strong></td>
</tr>
<tr>
<td></td>
<td>Format: ESC B;n₁;n₂;...;nₖ;NUL;</td>
</tr>
<tr>
<td></td>
<td>(Graphics Printer ignores ESC B)</td>
</tr>
<tr>
<td></td>
<td>Sets vertical tab stop positions. Up to 64 vertical tab stop positions are</td>
</tr>
<tr>
<td></td>
<td>recognized by the printer. The n’s, in the format above, are used to indicate</td>
</tr>
<tr>
<td></td>
<td>tab stop positions. Tab stop numbers must be received in ascending numeric order.</td>
</tr>
<tr>
<td></td>
<td>The tab stop numbers will not become valid until the NUL code is entered. Once</td>
</tr>
<tr>
<td></td>
<td>vertical tab stops are established, they will be valid until new tab stops are</td>
</tr>
<tr>
<td></td>
<td>specified. (If the printer is reset or powered Off, set tab stops are cleared.)</td>
</tr>
<tr>
<td></td>
<td>If no tab stop is set, the Vertical Tab command behaves as a Line Feed command.</td>
</tr>
<tr>
<td></td>
<td>ESC B followed only by NUL will cancel tab stops. The form length must be set by</td>
</tr>
<tr>
<td></td>
<td>the ESC C command prior to setting tabs.</td>
</tr>
<tr>
<td></td>
<td>Example</td>
</tr>
<tr>
<td></td>
<td>LPRINT CHR$(27);CHR$(66);CHR$(10);CHR$(20);CHR$(40);CHR$(0);</td>
</tr>
<tr>
<td>Printer Code</td>
<td>Printer Function</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>ESC C</td>
<td>Escape C (Set Lines per Page)</td>
</tr>
<tr>
<td></td>
<td>Format: ESC C;n;</td>
</tr>
<tr>
<td></td>
<td>Sets the page length. The ESC C command must have a value following it to specify the length of page desired. (Maximum form length for the printer is 127 lines.)</td>
</tr>
<tr>
<td></td>
<td>The example below sets the page length to 55 lines. The printer defaults to 66 lines per page when powered on or reset.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>LPRINT CHR$(27);CHR$(67);CHR$(55);</td>
</tr>
<tr>
<td>Escape C</td>
<td>Escape C (Set Inches per Page)</td>
</tr>
<tr>
<td></td>
<td>Format: ESC C;n;m;</td>
</tr>
<tr>
<td></td>
<td>(Graphics Printer only)</td>
</tr>
<tr>
<td></td>
<td>Escape C sets the length of the page in inches. This command requires a value of 0 (zero) for n, and a value between 1 and 22 for m.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>LPRINT CHR$(27);CHR$(67);CHR$(0);CHR$(12);</td>
</tr>
<tr>
<td>ESC D</td>
<td>Escape D (Set Horizontal Tab Stops)</td>
</tr>
<tr>
<td></td>
<td>Format: ESC D;n_1;n_2;...;n_k;NUL;</td>
</tr>
<tr>
<td></td>
<td>Sets the horizontal tab stop positions. The example below shows the horizontal tab stop positions set at printer column positions of 10, 20, and 40. They are followed by CHR$(O), the NUL code. They must also be in ascending numeric order as shown. Tab stops can be set between 1 and 80. When in the Compressed print mode, tab stops can be set up to 132.</td>
</tr>
<tr>
<td></td>
<td>The maximum number of tabs that can be set is 112. The Graphics Printer can have a maximum of 28 tab stops. The HT (CHR$(9)) is used to execute a tab operation.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>LPRINT CHR$(27);CHR$(68);CHR$(10);CHR$(20);CHR$(40);CHR$(0);</td>
</tr>
<tr>
<td>ESC E</td>
<td>Escape E (Emphasized)</td>
</tr>
<tr>
<td></td>
<td>Changes the printer to the Emphasized print mode. The speed of the printer is reduced to half speed during the Emphasized print mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>LPRINT CHR$(27);CHR$(69);</td>
</tr>
<tr>
<td>ESC F</td>
<td>Escape F (Emphasized Off)</td>
</tr>
<tr>
<td></td>
<td>Stops printing in the Emphasized print mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>LPRINT CHR$(27);CHR$(70);</td>
</tr>
<tr>
<td>ESC G</td>
<td>Escape G (Double Strike)</td>
</tr>
<tr>
<td></td>
<td>Changes the printer to the Double Strike print mode. The paper is spaced 1/216 of an inch before the second pass of the print head.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>LPRINT CHR$(27);CHR$(71);</td>
</tr>
<tr>
<td>Printer Code</td>
<td>Printer Function</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------</td>
</tr>
</tbody>
</table>
| ESC H        | **Escape H (Double Strike Off)**  
|              | Stops printing in the Double Strike mode.  
|              | Example:  
|              | LPRINT CHR$(27);CHR$(72);  
| ESC J        | **Escape J (Set Variable Line Feeding)**  
|              | Format: ESC J;n;  
|              | (Graphics Printer only)  
|              | When ESC J is sent to the printer, the paper will feed in increments of n/216 of an inch. The value of n must be between 1 and 255.  
|              | The example below gives a line feed of 50/216-inch. ESC J is canceled after the line feed takes place.  
|              | Example:  
|              | LPRINT CHR$(27);CHR$(74);CHR$(50);  
| ESC K        | **Escape K (480 Bit-Image Graphics Mode)**  
|              | Format ESC K;n_1;n_2;v_1;v_2;...v_k;  
|              | (Graphics Printer only)  
|              | Changes from the Text mode to the Bit-Image Graphics mode. n_1 and n_2 are one byte, which specify the number of bit-image data bytes to be transferred. v_1 through v_k are the bytes of the bit-image data. The number of bit-image data bytes (k) is equal to n_1 + 256n_2 and cannot exceed 480 bytes. At every horizontal position, each byte can print up to 8 vertical dots. Bit-image data may be mixed with text data on the same line.  
|              | **Note:** Assign values to n_1 and n_2 as follows:  
|              | n_1 represents values from 0 - 255.  
|              | n_2 represents values from 0 - 1 x 256.  
|              | MSB is most significant bit and LSB is least significant bit.  

<table>
<thead>
<tr>
<th>n_2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>MSB</td>
<td>LSB</td>
</tr>
<tr>
<td>15</td>
<td>14 13 12 11 10 9 8</td>
</tr>
<tr>
<td>7</td>
<td>6 5 4 3 2 2 2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>n_1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>MSB</td>
<td>LSB</td>
</tr>
<tr>
<td>2</td>
<td>2 2 2 2 2 2 1 0</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 \((20 \times 6 = 120)\). The printable portion left in Bit-Image mode is 360 dot positions \((480 - 120 = 360)\).

Data sent to the printer.

**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$(2)+CHR$(4)+CHR$(8)
6  SLASH$=SLASH$+CHR$(16)+CHR$(32)+CHR$(64)+CHR$(128)+CHR$(0)
7  GAP$=CHR$(0)+CHR$(0)+CHR$(0)
8  NDOTS=480
9  'ESCKN1
10 PRINT #1,CHR$(27);"K";CHR$(NDOTS MOD 256);CHR$(FIX (NDOTS/256));
11  'SEND NDOTS NUMBER OF BIT IMAGE BYTES
12 FOR I=1 TO NDOTS/12 'NUMBER OF SLASHES TO PRINT USING
13 GRAPHICS
14 PRINT #1,SLASH$;GAP$;
15 NEXT I
16 CLOSE
17 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>ESC L</td>
<td><strong>Escape L (960 Bit-Image Graphics Mode)</strong>&lt;br&gt;Format: ESC L;n₁;n₂;v₁;v₂;...vₖ;&lt;br&gt;(Graphics Printer only)&lt;br&gt;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>ESC N</td>
<td><strong>Escape N (Set Skip Perforation)</strong>&lt;br&gt;Format: ESC N;n;&lt;br&gt;(Graphics Printer only)&lt;br&gt;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.&lt;br&gt;Example:&lt;br&gt;CHR$(27);CHR$(78);CHR$(12);</td>
</tr>
<tr>
<td>ESC O</td>
<td><strong>Escape O (Cancel Skip Perforation)</strong>&lt;br&gt;(Graphics Printer only)&lt;br&gt;Cancels the Skip Perforation function.&lt;br&gt;Example:&lt;br&gt;LPRINT CHR$(27);CHR$(79);</td>
</tr>
<tr>
<td>ESC S</td>
<td><strong>Escape S (Subscript/Superscript)</strong>&lt;br&gt;Format: ESC S;n;&lt;br&gt;(Graphics Printer only)&lt;br&gt;Changes the printer to the Subscript print mode when ESC S is followed by a 1, as in the example below. When ESC S is followed by a 0 (zero), the printer will print in the Superscript print mode.&lt;br&gt;Example:&lt;br&gt;LPRINT CHR$(27);CHR$(83);CHR$(1);</td>
</tr>
<tr>
<td>ESC T</td>
<td><strong>Escape T (Subscript/Superscript Off)</strong>&lt;br&gt;(Graphics Printer only)&lt;br&gt;The printer stops printing in the Subscript or Superscript print mode.&lt;br&gt;Example:&lt;br&gt;LPRINT CHR$(27);CHR$(84);</td>
</tr>
<tr>
<td>ESC U</td>
<td><strong>Escape U (Unidirectional Printing)</strong>&lt;br&gt;Format: ESC U;n;&lt;br&gt;(Graphics Printer only)&lt;br&gt;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.&lt;br&gt;Example:&lt;br&gt;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;...vₘ;</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;...vₘ;</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)
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>BEL</td>
<td>HT</td>
<td>LF</td>
<td>VT</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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>150</td>
<td>151</td>
<td>152</td>
<td>153</td>
<td>154</td>
<td>155</td>
<td>156</td>
<td>157</td>
<td>158</td>
<td>159</td>
</tr>
<tr>
<td>CAN</td>
<td>ESC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>160</td>
<td>161</td>
<td>162</td>
<td>163</td>
<td>164</td>
<td>165</td>
<td>166</td>
<td>167</td>
<td>168</td>
<td>169</td>
</tr>
<tr>
<td>á</td>
<td>í</td>
<td>ó</td>
<td>ú</td>
<td>ñ</td>
<td>Ñ</td>
<td>a</td>
<td>o</td>
<td>ñ</td>
<td></td>
</tr>
<tr>
<td>170</td>
<td>171</td>
<td>172</td>
<td>173</td>
<td>174</td>
<td>175</td>
<td>176</td>
<td>177</td>
<td>178</td>
<td>179</td>
</tr>
<tr>
<td>1/2</td>
<td>1/4</td>
<td>i</td>
<td>&lt;&lt;</td>
<td>&gt;&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>180</td>
<td>181</td>
<td>182</td>
<td>183</td>
<td>184</td>
<td>185</td>
<td>186</td>
<td>187</td>
<td>188</td>
<td>189</td>
</tr>
<tr>
<td></td>
<td></td>
<td></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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>230</td>
<td>231</td>
<td>232</td>
<td>233</td>
<td>234</td>
<td>235</td>
<td>236</td>
<td>237</td>
<td>238</td>
<td>239</td>
</tr>
<tr>
<td>μ</td>
<td>τ</td>
<td>φ</td>
<td>Θ</td>
<td>Ω</td>
<td>δ</td>
<td>∞</td>
<td>Ø</td>
<td>ε</td>
<td></td>
</tr>
<tr>
<td>240</td>
<td>241</td>
<td>242</td>
<td>243</td>
<td>244</td>
<td>245</td>
<td>246</td>
<td>247</td>
<td>248</td>
<td>249</td>
</tr>
<tr>
<td>≡</td>
<td>±</td>
<td>≥</td>
<td>≤</td>
<td>∫</td>
<td>∴</td>
<td>÷</td>
<td>≈</td>
<td>°</td>
<td></td>
</tr>
<tr>
<td>250</td>
<td>251</td>
<td>252</td>
<td>253</td>
<td>254</td>
<td>255</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>√</td>
<td>n</td>
<td>2</td>
<td></td>
<td></td>
<td>SP</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Graphics Printer Character Set 1 (Part 2 of 2)
Graphics Printer Character Set 2 (Part 1 of 2)
Graphics Printer Character Set 2 (Part 2 of 2)
IBM Printer Adapter

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

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

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

When this adapter is used to attach a printer, data or printer commands are loaded into an 8-bit, latched, output port, and the strobe line is activated, writing data to the printer. The program then may read the input ports for printer status indicating when the next character can be written, or it may use the interrupt line to indicate “not busy” to the software.

The output ports may also be read at the card’s interface for diagnostic loop functions. This allows faults to be isolated between the adapter and the attaching device.

This same function is also part of the combination IBM Monochrome Display and Printer Adapter. A block diagram of the printer adapter is on the next page.
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</td>
<td>Bit 3 Bit 2 Bit 1 Bit 0</td>
</tr>
<tr>
<td>Pin 9 Pin 8 Pin 7 Pin 6</td>
<td>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.

<table>
<thead>
<tr>
<th>IBM Monochrome Display &amp; Printer Adapter</th>
<th>Printer Adapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output to address hex 3BE</td>
<td>Output to address hex 37A</td>
</tr>
<tr>
<td>Bit 4 IRQ Enable</td>
<td>Bit 3 Bit 2 Bit 1 Bit 0</td>
</tr>
<tr>
<td>Pin 17 Pin 16</td>
<td>Pin 14 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 $+5 \text{ Vdc}$ through 4.7 k-ohm resistors. They can each sink approximately 7 mA and maintain 0.8 volts down-level.

<table>
<thead>
<tr>
<th>IBM Monochrome Display &amp; Printer Adapter</th>
<th>Printer Adapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input from address hex 3BC</td>
<td>Input from address hex 378</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 ground rules) at the time of an input, this data will be ORed with the latch contents.

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

This command presents realtime status to the processor from the pins as follows.

<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>

1-120 Printer Adapter
This instruction causes the data present on pins 1, 14, 16, 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>IRQ Enable</td>
<td>Pin 17</td>
<td>Pin 16</td>
<td>Pin 14</td>
<td>Pin 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Por=0</td>
<td>Por=1</td>
<td>Por=0</td>
<td>Por=1</td>
<td>Por=1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

These pins assume the states shown after a reset from the processor.
Rear Panel

Note: All outputs are software-generated, and all inputs are real-time signals (not latched).

At Standard TTL Levels

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

Connector Specifications

1-122 Printer Adapter
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 80 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 Display 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.
<table>
<thead>
<tr>
<th>Bit Number</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>+High Resolution Mode</td>
</tr>
<tr>
<td>1</td>
<td>Not Used</td>
</tr>
<tr>
<td>2</td>
<td>Not Used</td>
</tr>
<tr>
<td>3</td>
<td>+Video Enable</td>
</tr>
<tr>
<td>4</td>
<td>Not Used</td>
</tr>
<tr>
<td>5</td>
<td>+Enable Blink</td>
</tr>
<tr>
<td>6,7</td>
<td>Not Used</td>
</tr>
</tbody>
</table>

### 6845 CRT Control Port 1 (Hex 3B8)

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

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

<table>
<thead>
<tr>
<th>IBM Monochrome Display</th>
<th>IBM Monochrome Display and Printer Adapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>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>

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
IBM Monochrome Display

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

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

Operating Characteristics

Screen

- High-persistence green phosphor (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 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.
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 to this buffer during the horizontal-retrace intervals. While the processor may write to the required buffer at any time, a small amount of display interference will result if this does not occur during the horizontal-retrace intervals.

Character Generator

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

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

Composite Color Generator

This generator produces base band video color information.

Alphanumeric Mode

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

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

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

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

<table>
<thead>
<tr>
<th>FG</th>
<th>Background</th>
<th>Foreground</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>0 0 0</td>
<td>1 1 1</td>
</tr>
<tr>
<td>B</td>
<td>1 1 1</td>
<td>1 0 0</td>
</tr>
<tr>
<td>B</td>
<td>0 0 0</td>
<td>1 0 0</td>
</tr>
<tr>
<td>B</td>
<td>1 1 1</td>
<td>1 1 1</td>
</tr>
</tbody>
</table>

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

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

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>7 6 5 4 3 2 1 0</td>
<td>Background Color</td>
<td>Character Color</td>
</tr>
<tr>
<td>FG</td>
<td>R</td>
<td>G</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

The monochrome display adapter will produce white characters on a white background with any other code. The color/graphics adapter will change foreground and background colors according to the color value selected. The color values for the various red, green, blue, and intensity bit settings are given in the following 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>Mode</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 table:

<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>C1</td>
<td>C0</td>
<td>C1</td>
<td>C0</td>
<td>C1</td>
<td>C0</td>
<td>C1</td>
<td>C0</td>
</tr>
<tr>
<td>First Display PEL</td>
<td>Second Display PEL</td>
<td>Third Display PEL</td>
<td>Fourth Display PEL</td>
<td></td>
<td></td>
<td></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.

Two two colors sets are:

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

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

High-Resolution Black-and-White Graphics Mode

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

- Contains a maximum of 200 rows of 640 PELs, with each PEL being 1-high by 1-wide.

- Supports black-and-white mode only.

- Requires 16,000 bytes of read/write memory (on the adapter).
• Addressing and mapping procedures are the same as medium-resolution color graphics, but the data format is different. In this mode, each bit in memory is mapped to a PEL on the screen.

• Formats 8 PELs per byte in the following manner:

![Diagram showing memory addressing and mapping procedures]

**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)

**Color/Graphics Adapter 1-145**
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."

<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.

Summary of Available Colors

1-146 Color/Graphics Adapter
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>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 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

6845 Register Description

1-148 Color/Graphics Adapter
Programming the Mode Control and Status Register

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

<table>
<thead>
<tr>
<th>Hex Address</th>
<th>A9 A8 A7 A6 A5 A4 A3 A2 A1 A0</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.
Bits 0, 1, 2, 3  These bits select the screen's border color in the 40 by 25 alphanumeric mode. They select the screen's background color (C0-C1) in the medium-resolution (320 by 200) color-graphics mode.

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

Bit 5  This bit is only used in the medium-resolution (320 by 200) color-graphics mode. It is used to select the active set of screen colors for the display.

When bit 5 is set to 1, colors are determined as follows:

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

1-150  Color/Graphics Adapter
When bit 5 is set to 0, colors are determined as follows:

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

**Mode-Select Register**

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

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

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

Bit 0  A 1 selects 80 by 25 alphanumeric mode  
A 0 selects 40 by 25 alphanumeric mode

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

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

Bit 3  A 1 enables the video signal at certain times when modes are being changed. The video signal should be disabled when changing modes.
Bit 4 A 1 selects the high-resolution (640 by 200) black-and-white graphics mode. One color of 8 can be selected on direct-drive sets in this mode by using register hex 3D9.

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

**Mode Register Summary**

<table>
<thead>
<tr>
<th>Bits</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 1 1 0 1</td>
<td>40 x 25 Alphanumeric Black-and-White</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0 0 1 0 1</td>
<td>40 x 25 Alphanumeric Color</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 0 1 1 0 1</td>
<td>80 x 25 Alphanumeric Black-and-White</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 0 0 1 0 1</td>
<td>80 x 25 Alphanumeric Color</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 1 1 1 0 z</td>
<td>320 x 200 Black-and-White Graphics</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 1 0 1 0 z</td>
<td>320 x 200 Color Graphics</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 1 1 1 1 z</td>
<td>640 x 200 Black-and-White Graphics</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- 0 1 1 1 1 z = don’t care condition
- Enable Blink Attribute
- 640 x 200 Black-and-White
- Enable Video Signal
- Select Black-and-White Mode
- Select 320 x 200 Graphics
- 80 x 25 Alphanumeric Select

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

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

<table>
<thead>
<tr>
<th>Bit 0</th>
<th>Display Enable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 1</td>
<td>Light-Pen Trigger Set</td>
</tr>
<tr>
<td>Bit 2</td>
<td>Light-Pen Switch Made</td>
</tr>
<tr>
<td>Bit 3</td>
<td>Vertical Sync</td>
</tr>
<tr>
<td>Bit 4</td>
<td>Not Used</td>
</tr>
<tr>
<td>Bit 5</td>
<td>Not Used</td>
</tr>
<tr>
<td>Bit 6</td>
<td>Not Used</td>
</tr>
<tr>
<td>Bit 7</td>
<td>Not Used</td>
</tr>
</tbody>
</table>

Bit 0  This bit, when active, indicates that a regen buffer memory access can be made without interfering with the display.

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

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

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

Sequence of Events for Changing Modes

1. Determine the mode of operation.

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

3. Program 6845 to select mode.

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

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

<table>
<thead>
<tr>
<th>Read/Write Memory Address Space (in hex)</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Read/Write Memory</td>
</tr>
<tr>
<td>01000</td>
</tr>
<tr>
<td>A0000</td>
</tr>
<tr>
<td>B8000</td>
</tr>
<tr>
<td>BC000</td>
</tr>
<tr>
<td>CO000</td>
</tr>
</tbody>
</table>

Display Buffer (16K Bytes)

128K Reserved Regen Area
### At Standard TTL Levels

<table>
<thead>
<tr>
<th>Ground</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ground</td>
<td>2</td>
</tr>
<tr>
<td>Red</td>
<td>3</td>
</tr>
<tr>
<td>Green</td>
<td>4</td>
</tr>
<tr>
<td>Blue</td>
<td>5</td>
</tr>
<tr>
<td>Intensity</td>
<td>6</td>
</tr>
<tr>
<td>Reserved</td>
<td>7</td>
</tr>
<tr>
<td>Horizontal Drive</td>
<td>8</td>
</tr>
<tr>
<td>Vertical Drive</td>
<td>9</td>
</tr>
</tbody>
</table>

### IBM Color Display or other Direct-Drive Monitor

<table>
<thead>
<tr>
<th>Color/Graphics Direct-Drive Adapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ground 2</td>
</tr>
</tbody>
</table>

### Composite Phono Jack

Hookup to Monitor

- 1
- 2

### Composite Video Signal

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

### Color/Graphics Composite Jack
1-156 Color/Graphics Adapter
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 \( \mu \text{PD765} \) or equivalent floppy disk controller (FDC).

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

Digital-Output Register

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

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

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

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

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

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

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

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

<table>
<thead>
<tr>
<th>Bit Number</th>
<th>Name</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB0</td>
<td>FDD A Busy</td>
<td>DAB</td>
<td>FDD number 0 is in the Seek mode.</td>
</tr>
<tr>
<td>DB1</td>
<td>FDD B Busy</td>
<td>DBB</td>
<td>FDD number 1 is in the Seek mode.</td>
</tr>
<tr>
<td>DB2</td>
<td>FDD C Busy</td>
<td>DCB</td>
<td>FDD number 2 is in the Seek mode.</td>
</tr>
<tr>
<td>DB3</td>
<td>FDD D Busy</td>
<td>DDB</td>
<td>FDD number 3 is in the Seek mode.</td>
</tr>
<tr>
<td>DB4</td>
<td>FDC Busy</td>
<td>CB</td>
<td>A read or write command is in process.</td>
</tr>
<tr>
<td>DB5</td>
<td>Non-DMA Mode</td>
<td>NDM</td>
<td>The FDC is in the non-DMA mode.</td>
</tr>
<tr>
<td>DB6</td>
<td>Data Input/Output</td>
<td>DIO</td>
<td>Indicates direction of data transfer between FDC and processor.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>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 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>A0</td>
<td>Address Line 0</td>
<td>A0 controls selection of main status register (A0 = 0) or data register (A0 = 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, US1</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>
</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>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>MT</td>
<td>MF</td>
<td>SK</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Command Codes</td>
</tr>
<tr>
<td></td>
<td>W</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>
<td>Sector ID information prior to command execution.</td>
</tr>
<tr>
<td></td>
<td>W</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>W</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>W</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>W</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>W</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>W</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>W</td>
<td>DTL</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</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Data transfer between the FDD and main system.</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>ST</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Status information after command execution.</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>ST</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Sector ID information after command execution.</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>Result</td>
<td>R</td>
<td>ST</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Command Codes</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>ST</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Sector ID information prior to command execution.</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>Data transfer between the FDD and main system.</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>Status information after command execution.</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>Sector ID information after command execution.</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>
</tbody>
</table>

1-166 Diskette Adapter
<table>
<thead>
<tr>
<th>Phase</th>
<th>R/W</th>
<th>D7 D6 D5 D4 D3 D2 D1 D0</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command</td>
<td>W</td>
<td>MT MF 0 0 0 1 0 1</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 to command execution.</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>EOT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>GPL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>DTL</td>
<td></td>
</tr>
<tr>
<td>Execution</td>
<td></td>
<td></td>
<td>Data transfer between the main system and FDD.</td>
</tr>
<tr>
<td>Result</td>
<td>R</td>
<td>ST 0</td>
<td>Status information after command execution.</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>ST 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>ST 2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>C</td>
<td>Sector ID information after command execution.</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>

<p>| Command | W   | MT MF 0 0 0 1 0 0 1         | Command Codes |
|        | W   | X X X X X HD US1 US0      | Sector ID information prior to command execution. |
|        | W   | C                  |         |
|        | W   | H                  |         |
|        | W   | R                  |         |
|        | W   | N                  |         |
|        | W   | EOT                |         |
|        | W   | GPL                |         |
|        | W   | DTL                |         |
| Execution |     |                       | Data transfer between FDD and main system. |
| Result | R   | ST 0              | Status ID information after command execution. |
|        | R   | ST 1              |         |
|        | R   | ST 2              |         |
|        | R   | C                 |         |
|        | R   | H                 | Sector ID information after command execution. |
|        | R   | R                 |         |
|        | R   | N                 |         |</p>
<table>
<thead>
<tr>
<th>Phase</th>
<th>R/W</th>
<th>(D_7)</th>
<th>(D_6)</th>
<th>(D_5)</th>
<th>(D_4)</th>
<th>(D_3)</th>
<th>(D_2)</th>
<th>(D_1)</th>
<th>(D_0)</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command</td>
<td>W</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>HD</td>
<td>US1</td>
<td>US0</td>
<td>C</td>
<td>Command Codes</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>H</td>
<td>Sector ID information prior to command execution.</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>R</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>EOT</td>
<td></td>
<td></td>
<td></td>
<td>Data transfer between the FDD and main system.</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>GPL</td>
<td></td>
<td></td>
<td></td>
<td>FDC reads all of cylinder's contents from index hole to EOT.</td>
</tr>
<tr>
<td>Execution</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DTL</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Result</td>
<td>R</td>
<td>ST 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Status information after command execution.</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>ST 1</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>ST 2</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>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>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>HD</td>
<td>US1</td>
<td>US0</td>
<td>C</td>
<td>Command Codes</td>
</tr>
<tr>
<td>Execution</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>H</td>
<td>The first correct ID information on the cylinder is stored in data register.</td>
</tr>
<tr>
<td>Result</td>
<td>R</td>
<td>ST 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Status information after command execution.</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>ST 1</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>ST 2</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>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>
</tbody>
</table>
### Format a Track

<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>MF</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Command Codes</td>
</tr>
<tr>
<td></td>
<td>W</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>
<td>Bytes/Sector</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Sector/Track</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>SC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Gap 3</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>GPL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>filler byte.</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>FDC formats an entire cylinder.</td>
</tr>
</tbody>
</table>

**Execution**

<table>
<thead>
<tr>
<th>Result</th>
<th>R</th>
<th>ST 0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R</td>
<td>ST 1</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>ST 2</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>H</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>N</td>
</tr>
</tbody>
</table>

**Result**

- R ST0 after command execution.
- In this case, the ID information has no meaning.

### Scan Equal

<table>
<thead>
<tr>
<th>Command</th>
<th>W</th>
<th>MT</th>
<th>MF</th>
<th>SK</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>Command Codes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>W</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>
<td>Sector ID information prior to command execution.</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Data compared between the FDD and the main system.</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Status information after command execution.</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Sector ID information after Command execution.</td>
</tr>
</tbody>
</table>

**Execution**

**Result**

- R ST0 after command execution.
<table>
<thead>
<tr>
<th>Phase</th>
<th>R/W</th>
<th>Data Bus</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command</td>
<td>W</td>
<td>D7  D6  D5  D4  D3  D2  D1  D0</td>
<td>Command Codes</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>MT   MF   SK  1  1  0  0  1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>X    X    X    X    X    HD  US1 US0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>EOT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>GPL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>STP</td>
<td></td>
</tr>
<tr>
<td>Execution</td>
<td>R</td>
<td>ST  0</td>
<td>Data compared between the FDD and main system.</td>
</tr>
<tr>
<td>Result</td>
<td>R</td>
<td>ST  1</td>
<td>Status information after command execution.</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>ST  2</td>
<td>Sector ID information after command execution.</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>N</td>
<td></td>
</tr>
</tbody>
</table>

| Command   | W   | D7  D6  D5  D4  D3  D2  D1  D0 | Command Codes |
|           | W   | MT   MF   SK  1  1  1  0  1 |                                      |
|           | W   | X    X    X    X    X    HD  US1 US0 |                                      |
|           | W   | C    |                                      |
|           | W   | H    |                                      |
|           | W   | R    |                                      |
|           | W   | N    |                                      |
|           | W   | EOT  |                                      |
|           | W   | GPL  |                                      |
|           | W   | STP  |                                      |
| Execution | R   | ST  0 | Data compared between the FDD and main system. |
| Result    | R   | ST  1 | Status information after command execution. |
|           | R   | ST  2 | Sector ID information after command execution. |
|           | R   | C    |                                      |
|           | R   | H    |                                      |
|           | R   | R    |                                      |
|           | R   | N    |                                      |

1-170 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>W 0 0 0 0 0 1 1 1</td>
<td>Command Codes</td>
</tr>
<tr>
<td>Execution</td>
<td></td>
<td>W X X X X X 0 US1 US0</td>
<td>Head retracted to track 0</td>
</tr>
<tr>
<td>No Result Phase</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Command</td>
<td>W</td>
<td>0 0 0 0 1 0 0 0</td>
<td>Command Codes</td>
</tr>
<tr>
<td>Sense Interrupt Status</td>
<td>R</td>
<td>ST 0</td>
<td>Status information at the end of seek operation about the FDC</td>
</tr>
<tr>
<td>W</td>
<td>R</td>
<td>PCN</td>
<td></td>
</tr>
<tr>
<td>Command</td>
<td>W</td>
<td>0 0 0 0 0 0 1 1</td>
<td>Command Codes</td>
</tr>
<tr>
<td>Specify</td>
<td></td>
<td>SRT</td>
<td></td>
</tr>
<tr>
<td>Command</td>
<td>W</td>
<td>0 0 0 0 0 1 0 0</td>
<td>Command Codes</td>
</tr>
<tr>
<td>Sense Drive Status</td>
<td>R</td>
<td>HD US1 USO ST 3</td>
<td>Status information about FDD.</td>
</tr>
<tr>
<td>Command</td>
<td>W</td>
<td>W X X X X X</td>
<td>Command Codes</td>
</tr>
<tr>
<td>Seek</td>
<td></td>
<td>HD US1 USO NCN</td>
<td>Head is positioned over proper cylinder on diskette.</td>
</tr>
<tr>
<td>Command</td>
<td>W</td>
<td>0 0 0 0 1 1 1 1</td>
<td>Command Codes</td>
</tr>
<tr>
<td>Invalid</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Command</td>
<td>W</td>
<td>Invalid Codes</td>
<td>Invalid command codes (NoOp - FDC goes into standy state).</td>
</tr>
<tr>
<td>Result</td>
<td>R</td>
<td>ST 0</td>
<td>ST 0 = 80.</td>
</tr>
<tr>
<td>No.</td>
<td>Name</td>
<td>Symbol</td>
<td>Description</td>
</tr>
<tr>
<td>-----</td>
<td>-----------------------</td>
<td>--------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</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. |
| D6  |                       |        |                                                                                                                                            |
| 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**

1-172  Diskette Adapter
<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 “equal” 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>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**

<table>
<thead>
<tr>
<th>Register</th>
<th>I/O Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>FDC Data Register</td>
<td>Hex 3F5</td>
</tr>
<tr>
<td>FDC Main Status Register</td>
<td>Hex 3F4</td>
</tr>
<tr>
<td>Digital Output Register</td>
<td>Hex 3F2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 0</th>
<th>Drive</th>
<th>00: DR #A 10: DR #C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Select</td>
<td>01: DR #B 11: DR #D</td>
</tr>
<tr>
<td>2</td>
<td>Not FDC Reset</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Enable INT &amp; DMA Requests</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Drive A Motor Enable</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Drive B Motor Enable</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Drive C Motor Enable</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Drive D Motor Enable</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

1-176 Diskette Adapter
The following lines are used by this adapter.

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

+A0-9 (Adapter input, load: 1 74LS) These ten lines form an address bus by which a register is selected to receive or supply the byte transferred through lines D0-7. Bit 0 is the low-order bit.

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

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

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

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

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

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

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

Drive A and B Interface

All signals are TTL-compatible:

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

All adapter outputs are driven by 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.
<table>
<thead>
<tr>
<th>Track 0</th>
<th>The selected drive makes this line active if the read/write head is over track 0.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read Data</td>
<td>The selected drive supplies a pulse on this line for each flux change encountered on the diskette.</td>
</tr>
</tbody>
</table>
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.

At Standard TTL Levels

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

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

<table>
<thead>
<tr>
<th>Pin Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-5</td>
<td>Unused</td>
</tr>
<tr>
<td>6</td>
<td>Index</td>
</tr>
<tr>
<td>7</td>
<td>Motor Enable C</td>
</tr>
<tr>
<td>8</td>
<td>Drive Select D</td>
</tr>
<tr>
<td>9</td>
<td>Drive Select C</td>
</tr>
<tr>
<td>10</td>
<td>Motor Enable D</td>
</tr>
<tr>
<td>11</td>
<td>Direction (Stepper Motor)</td>
</tr>
<tr>
<td>12</td>
<td>Step Pulse</td>
</tr>
<tr>
<td>13</td>
<td>Write Data</td>
</tr>
<tr>
<td>14</td>
<td>Write Enable</td>
</tr>
<tr>
<td>15</td>
<td>Track 0</td>
</tr>
<tr>
<td>16</td>
<td>Write Protect</td>
</tr>
<tr>
<td>17</td>
<td>Read Data</td>
</tr>
<tr>
<td>18</td>
<td>Select Head 1</td>
</tr>
<tr>
<td>20-37</td>
<td>Ground</td>
</tr>
</tbody>
</table>
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 10⁹ (recoverable)</td>
</tr>
<tr>
<td></td>
<td>1 per 10¹² (non recoverable)</td>
</tr>
<tr>
<td></td>
<td>1 per 10⁶ (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 10⁸ passes per track</td>
</tr>
<tr>
<td>Disk Speed</td>
<td>300 rpm +/- 1.5% (long term)</td>
</tr>
<tr>
<td>Instantaneous Speed Variation</td>
<td>+/- 3.0%</td>
</tr>
<tr>
<td>Start/Stop Time</td>
<td>250 ms (maximum)</td>
</tr>
<tr>
<td>Transfer Rate</td>
<td>250K bits/sec</td>
</tr>
<tr>
<td>Recording Mode</td>
<td>MFM</td>
</tr>
<tr>
<td>Power</td>
<td>+12 Vdc +/- 0.6 V, 900 mA average</td>
</tr>
<tr>
<td></td>
<td>+5 Vdc +/- 0.25 V, 600 mA average</td>
</tr>
</tbody>
</table>

Mechanical and Electrical Specifications

1-184 Diskette Drive
Diskettes

The IBM 5-1/4" Diskette Drive uses a standard 5.25-inch (133.4-millimeter) diskette. For programming considerations, single-sided, double-density, soft-sectored diskettes are used for single-sided drives. Double-sided drives use double-sided, double-density, soft-sectored diskettes. The figure below is a simplified drawing of the diskette used with the diskette drive. This recording medium is a flexible magnetic disk enclosed in a protective jacket. The protected disk, free to rotate within the jacket, is continuously cleaned by the soft fabric lining of the jacket during normal operation. Read/write/erase head access is made through an opening in the jacket. Openings for the drive hub and diskette index hole are also provided.
Notes:

1-186 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 6 5 4 3 2 1 0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0 0 d 0 0 0 e 0</td>
</tr>
</tbody>
</table>

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

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

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

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

Sense Bytes

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

<table>
<thead>
<tr>
<th>Bits</th>
<th>7 6 5 4 3 2 1 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte 0</td>
<td>Address Valid</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>
</tbody>
</table>

Remarks

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

Byte 0  Bits 4, 5  Error type.

Byte 0  Bit 6  Set to 0 (spare).

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

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

<table>
<thead>
<tr>
<th>Error Type</th>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits: 5 4 3 2 1 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0 0 0 0</td>
<td>The controller did not detect any error during the execution of the previous operation.</td>
<td></td>
</tr>
<tr>
<td>0 0 0 0 1</td>
<td>The controller did not detect an index signal from the drive.</td>
<td></td>
</tr>
<tr>
<td>0 0 0 1 0</td>
<td>The controller did not get a seek-complete signal from the drive after a seek operation (for all non-buffered step seeks).</td>
<td></td>
</tr>
<tr>
<td>0 0 0 1 1</td>
<td>The controller detected a write fault from the drive during the last operation.</td>
<td></td>
</tr>
<tr>
<td>0 0 0 1 0</td>
<td>After the controller selected the drive, the drive did not respond with a ready signal.</td>
<td></td>
</tr>
<tr>
<td>0 0 1 0 1</td>
<td>Not used.</td>
<td></td>
</tr>
<tr>
<td>0 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>
<td></td>
</tr>
<tr>
<td>0 0 0 1 1</td>
<td>Not used.</td>
<td></td>
</tr>
<tr>
<td>0 0 1 0 0</td>
<td>The drive is still seeking. This status is reported by the Test Drive Ready command for an overlap seek condition when the drive has not completed the seek. No time-out is measured by the controller for the seek to complete.</td>
<td></td>
</tr>
</tbody>
</table>

1-190  Fixed Disk Adapter
<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 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>Error Type</td>
<td>Error Code</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td>1 0 0 0 0</td>
<td>Invalid Command: The controller has received an invalid command from the system unit.</td>
<td></td>
</tr>
<tr>
<td>1 0 0 0 1</td>
<td>Illegal Disk Address: The controller detected an address that is beyond the maximum range.</td>
<td></td>
</tr>
<tr>
<td>1 1 0 0 0</td>
<td>RAM Error: The controller detected a data error during the RAM sector-buffer diagnostic test.</td>
<td></td>
</tr>
<tr>
<td>1 1 0 0 1</td>
<td>Program Memory Checksum Error: During this internal diagnostic test, the controller detected a program-memory checksum error.</td>
<td></td>
</tr>
<tr>
<td>1 1 0 1 0</td>
<td>ECC Polynominal Error: During the controller's internal diagnostic tests, the hardware ECC generator failed its test.</td>
<td></td>
</tr>
</tbody>
</table>
Data Register

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

<table>
<thead>
<tr>
<th>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>Byte 0</td>
<td>Command Class</td>
<td>Opcode</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Byte 1</td>
<td>0</td>
<td>0</td>
<td>d</td>
<td>Head Number</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Byte 2</td>
<td>Cylinder High</td>
<td>Sector Number</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Byte 3</td>
<td>Cylinder Low</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Byte 4</td>
<td>Interleave or Block Count</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Byte 5</td>
<td>Control Field</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

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

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

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

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

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

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

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>r</td>
<td>a</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>s</td>
<td>s</td>
<td>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>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td>This drive is not specified and defaults to 3 milliseconds per step.</td>
<td></td>
</tr>
<tr>
<td>0 0 1</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>0 1 0</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>0 1 1</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>1 0 0</td>
<td>200 microseconds per step.</td>
<td></td>
</tr>
<tr>
<td>1 0 1</td>
<td>70 microseconds per step (specified by BIOS).</td>
<td></td>
</tr>
<tr>
<td>1 1 0</td>
<td>3 milliseconds per step.</td>
<td></td>
</tr>
<tr>
<td>1 1 1</td>
<td>3 milliseconds per step.</td>
<td></td>
</tr>
</tbody>
</table>

1-194 Fixed Disk Adapter
<table>
<thead>
<tr>
<th>Command</th>
<th>Data Control Block</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Drive Ready</td>
<td></td>
<td>d = drive (0 or 1) x = don’t care Bytes 2, 3, 4, 5 = don’t care</td>
</tr>
<tr>
<td>(Class 0, Opcode 00)</td>
<td>Bit 7 6 5 4 3 2 1 0</td>
<td>d = drive (0 or 1) x = don’t care Bytes 2, 3, 4, 5 = don’t care</td>
</tr>
<tr>
<td></td>
<td>Byte 0 0 0 0 0 0 0 0</td>
<td>d = drive (0 or 1) x = don’t care Bytes 2, 3, 4, 5 = don’t care</td>
</tr>
<tr>
<td></td>
<td>Byte 1 0 0 d x x x x x</td>
<td>d = drive (0 or 1) x = don’t care Bytes 2, 3, 4, 5 = don’t care</td>
</tr>
<tr>
<td>Recalibrate</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Class 0, Opcode 01)</td>
<td>Bit 7 6 5 4 3 2 1 0</td>
<td>d = drive (0 or 1) x = don’t care Bytes 2, 3, 4, 5 = don’t care</td>
</tr>
<tr>
<td></td>
<td>Byte 0 0 0 0 0 0 0 0 0 1</td>
<td>d = drive (0 or 1) x = don’t care Bytes 2, 3, 4, 5 = don’t care</td>
</tr>
<tr>
<td></td>
<td>Byte 1 0 0 d x x x x x</td>
<td>d = drive (0 or 1) x = don’t care Bytes 2, 3, 4, 5 = don’t care</td>
</tr>
<tr>
<td></td>
<td>Byte 5 r 0 0 0 0 s s</td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td></td>
<td>This Opcode is not used.</td>
</tr>
<tr>
<td>(Class 0, Opcode 02)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Request Sense Status</td>
<td></td>
<td>d = drive (0 or 1) x = don’t care Bytes 2, 3, 4, 5 = don’t care</td>
</tr>
<tr>
<td>(Class 0, Opcode 03)</td>
<td>Bit 7 6 5 4 3 2 1 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 0 0 0 0 0 0 0 0 1 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 1 0 0 d x x x x x</td>
<td></td>
</tr>
<tr>
<td>Format Drive</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Class 0, Opcode 04)</td>
<td>Bit 7 6 5 4 3 2 1 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 0 0 0 0 0 1 0 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 1 0 0 d Head Number</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 2 ch 0 0 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 3 Cylinder Low</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 4 0 0 0 Interleave</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 5 r 0 0 0 0 s s</td>
<td></td>
</tr>
<tr>
<td>Ready Verify</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Class 0, Opcode 05)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bit 7 6 5 4 3 2 1 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 1 0 0 0 0 0 1 0 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 1 0 0 d Head Number</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 2 ch 0 0 0 0 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 3 Cylinder Low</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 4 Block Count</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 5 r a 0 0 0 0 s s</td>
<td></td>
</tr>
<tr>
<td>Command</td>
<td>Data Control Block</td>
<td>Remarks</td>
</tr>
<tr>
<td>--------------------</td>
<td>--------------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>Format Track</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Class 0, Opcode 06)</td>
<td>Bit 0: 7 6 5 4 3 2 1 0</td>
<td>d = drive (0 or 1)</td>
</tr>
<tr>
<td></td>
<td>Byte 0: 0 0 0 0 0 1 1 0</td>
<td>r = retries</td>
</tr>
<tr>
<td></td>
<td>Byte 1: 0 0 d Head Number</td>
<td>s = step option</td>
</tr>
<tr>
<td></td>
<td>Byte 2: ch 0 0 0 0 0 0 0</td>
<td>ch = cylinder high</td>
</tr>
<tr>
<td></td>
<td>Byte 3: Cylinder Low</td>
<td>Interleave: 1 to 16 for 512-byte sectors</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>Format Bad Track</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Class 0, Opcode 07)</td>
<td>Bit 0: 7 6 5 4 3 2 1 0</td>
<td>d = drive (0 or 1)</td>
</tr>
<tr>
<td></td>
<td>Byte 0: 0 0 0 0 0 1 1 1</td>
<td>r = retries</td>
</tr>
<tr>
<td></td>
<td>Byte 1: 0 0 d Head Number</td>
<td>s = step option</td>
</tr>
<tr>
<td></td>
<td>Byte 2: ch 0 0 0 0 0 0 0</td>
<td>ch = cylinder high</td>
</tr>
<tr>
<td></td>
<td>Byte 3: Cylinder Low</td>
<td>Interleave: 1 to 16 for 512-byte sectors</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>Read</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Class 0, Opcode 08)</td>
<td>Bit 0: 7 6 5 4 3 2 1 0</td>
<td>d = drive (0 or 1)</td>
</tr>
<tr>
<td></td>
<td>Byte 0: 0 0 0 0 1 0 0 0</td>
<td>r = retries</td>
</tr>
<tr>
<td></td>
<td>Byte 1: 0 0 d Head Number</td>
<td>a = retry option on data ECC error</td>
</tr>
<tr>
<td></td>
<td>Byte 2: ch Sector Number</td>
<td>s = step option</td>
</tr>
<tr>
<td></td>
<td>Byte 3: Cylinder Low</td>
<td>ch = cylinder high</td>
</tr>
<tr>
<td></td>
<td>Byte 5: r a 0 0 0 s s s</td>
<td>This Opcode is not used</td>
</tr>
<tr>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Class 0, Opcode 09)</td>
<td>Bit 0: 7 6 5 4 3 2 1 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 0: 0 0 0 0 1 0 1 0</td>
<td></td>
</tr>
<tr>
<td></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 0 0 0 0 s s s</td>
<td></td>
</tr>
<tr>
<td>Write</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Class 0, Opcode 0A)</td>
<td>Bit 0: 7 6 5 4 3 2 1 0</td>
<td>d = drive (0 or 1)</td>
</tr>
<tr>
<td></td>
<td>Byte 0: 0 0 0 0 1 0 1 0</td>
<td>r = retries</td>
</tr>
<tr>
<td></td>
<td>Byte 1: 0 0 d Head Number</td>
<td>s = step option</td>
</tr>
<tr>
<td></td>
<td>Byte 2: ch Sector Number</td>
<td>ch = cylinder high</td>
</tr>
<tr>
<td></td>
<td>Byte 3: Cylinder Low</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 4: Block Count</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 5: r 0 0 0 0 s s s</td>
<td></td>
</tr>
<tr>
<td>Seek</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Class 0, Opcode 0B)</td>
<td>Bit 0: 7 6 5 4 3 2 1 0</td>
<td>d = drive (0 or 1)</td>
</tr>
<tr>
<td></td>
<td>Byte 0: 0 0 0 0 1 0 1 1</td>
<td>r = retries</td>
</tr>
<tr>
<td></td>
<td>Byte 1: 0 0 d Head Number</td>
<td>s = step option</td>
</tr>
<tr>
<td></td>
<td>Byte 2: ch 0 0 0 0 0 0 0</td>
<td>x = don't care</td>
</tr>
<tr>
<td></td>
<td>Byte 3: Cylinder Low</td>
<td>ch = cylinder high</td>
</tr>
<tr>
<td></td>
<td>Byte 4: x x x x x x x x x x</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 5: r 0 0 0 0 s s s</td>
<td></td>
</tr>
</tbody>
</table>

1-196 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 7 6 5 4 3 2 1 0 Byte 0 0 0 0 1 1 0 0</td>
<td>Bytes 1, 2, 3, 4, 5 = don’t care</td>
</tr>
<tr>
<td>Characteristics* (Class 0, Opcode 0C)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Read ECC Burst Error Length (Class 0, Opcode 0D)</td>
<td>Bit 7 6 5 4 3 2 1 0 Byte 0 0 0 0 1 1 0 1</td>
<td>Bytes 1, 2, 3, 4, 5 = don’t care</td>
</tr>
<tr>
<td>Read Data from Sector Buffer (Class 0, Opcode 0E)</td>
<td>Bit 7 6 5 4 3 2 1 0 Byte 0 0 0 0 1 1 1 0</td>
<td>Bytes 1, 2, 3, 4, 5 = don’t care</td>
</tr>
<tr>
<td>Write Data to Sector Buffer (Class 0, Opcode 0F)</td>
<td>Bit 7 6 5 4 3 2 1 0 Byte 0 0 0 0 1 1 1 1</td>
<td>Bytes 1, 2, 3, 4, 5 = don’t care</td>
</tr>
<tr>
<td>RAM Diagnostic (Class 7, Opcode 00)</td>
<td>Bit 7 6 5 4 3 2 1 0 Byte 0 1 1 1 0 0 0 0</td>
<td>Bytes 1, 2, 3, 4, 5 = don’t care</td>
</tr>
<tr>
<td>Reserved (Class 7, Opcode 01)</td>
<td></td>
<td>This Opcode is not used</td>
</tr>
<tr>
<td>Reserved (Class 7, Opcode 02)</td>
<td></td>
<td>This Opcode is not used</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>Bit: 7 6 5 4 3 2 1 0</td>
<td></td>
</tr>
<tr>
<td>(Class 7, Opcode 03)</td>
<td>Byte 0: 1 1 1 0 0 0 1 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 1: 0 0 d x x x x x x</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 2: x x x x x x x x x</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 3: x x x x x x x x x</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 4: x x x x x x x x x</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 5: r 0 0 0 0 s s s s</td>
<td></td>
</tr>
<tr>
<td>Controller Internal Diagnostics</td>
<td>Bit: 7 6 5 4 3 2 1 0</td>
<td>Bytes 1, 2, 3, 4, 5 = don't care</td>
</tr>
<tr>
<td>(Class 7, Opcode 04)</td>
<td>Byte 0: 1 1 1 0 0 1 0 0</td>
<td></td>
</tr>
<tr>
<td>Read Long*</td>
<td>Bit: 7 6 5 4 3 2 1 0</td>
<td></td>
</tr>
<tr>
<td>(Class 7, Opcode 05)</td>
<td>Byte 0: 1 1 1 0 0 1 0 1</td>
<td></td>
</tr>
<tr>
<td></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 0 0 0 0 s s s s</td>
<td></td>
</tr>
<tr>
<td>Write Long**</td>
<td>Bit: 7 6 5 4 3 2 1 0</td>
<td></td>
</tr>
<tr>
<td>(Class 7, Opcode 06)</td>
<td>Byte 0: 1 1 1 0 0 1 1 0</td>
<td></td>
</tr>
<tr>
<td></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 0 0 0 0 s s s s</td>
<td></td>
</tr>
</tbody>
</table>

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

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

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

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

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

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

The following lines are used by the disk controller:

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

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

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

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

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

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

IRQ 5   Positive true interrupt request signal that is asserted by the controller, when enabled to interrupt the system board on the return ending status byte from the controller.
DRQ 3 Positive-true DMA-request signal that is asserted by the controller when data is available for transfer to or from the controller under DMA control. This signal remains active until the system board’s DMA channel activates the DMA-acknowledge signal (−DACK 3) in response.

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

1-202  Fixed Disk Adapter

### Disk Drive Connector J1

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

### Disk Drive Connector J2 or J3

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

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

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

1-204  Fixed Disk Drive
IBM Memory Expansion Options

Three memory expansion options (32KB, 64KB, and 64/256KB) and two memory module kits (16KB and 64KB) are available for the IBM Personal Computer. Memory expansion is described in the following chart:

<table>
<thead>
<tr>
<th>Memory Option</th>
<th>Minimum Memory</th>
<th>Maximum Memory</th>
<th>Number of 16K Memory Module Kits</th>
<th>Number of 64K Memory Module Kits</th>
<th>Memory Module Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>16/64K System Board</td>
<td>16K</td>
<td>64K</td>
<td>1, 2, or 3</td>
<td></td>
<td>16K by 1 Bit, 16 pin</td>
</tr>
<tr>
<td>64/256K System Board</td>
<td>64K</td>
<td>256K</td>
<td>1, 2, or 3</td>
<td></td>
<td>64K by 1 Bit, 16 pin</td>
</tr>
<tr>
<td>64/256K Memory Option</td>
<td>64K</td>
<td>256K</td>
<td>1, 2, or 3</td>
<td></td>
<td>64K by 1 Bit, 16 pin</td>
</tr>
<tr>
<td>32K Memory Option</td>
<td>32K</td>
<td></td>
<td></td>
<td></td>
<td>16K by 1 Bit, 16 pin</td>
</tr>
<tr>
<td>64K Memory Option</td>
<td>64K</td>
<td></td>
<td></td>
<td>16K by 1 Bit, 16 pin</td>
<td>Stacked 32K by 1 Bit, 18 pin</td>
</tr>
</tbody>
</table>

The system board must be fully populated 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 the DIP switches on the option.

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

Memory Module Description

Both the 32K and the 64K options contain 18 dynamic memory modules. The 32K memory expansion option utilizes 16K by 1 bit modules, and the 64K memory expansion option utilizes 32K by 1 bit modules.
The 64/256K option has four banks of 9 pluggable sockets. Each bank will accept a 64K memory module kit, consisting of 9 (64K by 1) modules. The kits must be installed sequentially into banks 1, 2, and 3. The base 64/256K option comes with modules installed in bank 0, providing 64K of memory. One, two, or three 64K bits may be added, upgrading the option to 128K, 192K, or 256K of memory.

The 16K by 1 and the 32K by 1 modules require three voltage levels: +5 Vdc, −5 Vdc, and +12 Vdc. The 64K by 1 modules require only one voltage level of +5 Vdc. All three memory modules require 128 refresh cycles every 2 ns. Absolute maximum access times are:

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

*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).
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.

Memory Option Switch Settings

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

1-208 Memory Expansion Options
The following method can be used to determine the switch settings for the 32K memory expansion option.

Starting Address = xxxK

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

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  [xxxK]

Convert decimal value to binary

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

<table>
<thead>
<tr>
<th>Switch</th>
</tr>
</thead>
<tbody>
<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>

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.

Game Control Adapter Block Diagram
Functional Description

Address Decode

The select on the game control adapter is generated by two 74LS138s as an address decoder. AEN must be inactive while the address is hex 201 in order to generate the select. The select allows a write to fire the one-shots or a 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.

1-212 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 +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.

<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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Digital Inputs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Resistive Inputs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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>

1-214 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>
</tbody>
</table>

Button Coordinate Button Coordinate Button Coordinate Button Coordinate

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

**Joy Stick Schematic Diagram**

15-Pin Male D-Shell Connector

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>

Connector Specifications

1-216  Game Control Adapter
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
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.
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

1-222 Prototype Card
IBM Asynchronous Communications Adapter

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

The adapter is fully programmable and supports asynchronous communications only. It will add and remove start bits, stop bits, and parity bits. A programmable baud rate generator allows operation from 50 baud to 9600 baud. Five, six, seven or eight bit characters with 1, 1-1/2, or 2 stop bits are supported. A fully prioritized interrupt system controls transmit, receive, error, line status and data set interrupts. Diagnostic capabilities provide loopback functions of transmit/receive and input/output signals.

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

- Full double buffering eliminates need for precise synchronization.
- Independent receiver clock input.
- Modem control functions: clear to send (CTS), request to send (RTS), data set ready (DSR), data terminal ready (DTR), ring indicator (RI), and carrier detect.
- False-start bit detection.
- Line-break generation and detection.

All communications protocol is a function of the system microcode and must be loaded before the adapter is operational. All pacing of the interface and control signal status must be handled by the system software. The following figure is a block diagram of the asynchronous communications adapter.
Asynchronous Communications Adapter Block Diagram

Modes of Operation

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

### I/O Decodes

<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>2FA</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>

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

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

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

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

### Address Bits
Interrupts

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

The data format will be as follows:

<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>Transmit Data Marking</td>
<td>Start Bit</td>
<td>Parity Bit</td>
<td>Stop Bits</td>
<td></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
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

<table>
<thead>
<tr>
<th>Voltage (Vdc)</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>+15 Vdc</td>
<td>On</td>
</tr>
<tr>
<td>+3 Vdc</td>
<td>On</td>
</tr>
<tr>
<td>0 Vdc</td>
<td>Invalid</td>
</tr>
<tr>
<td>-3 Vdc</td>
<td>Off</td>
</tr>
<tr>
<td>-15 Vdc</td>
<td>Off</td>
</tr>
</tbody>
</table>

The signal will be considered in the “marking” condition when the voltage on the interchange circuit, measured at the interface point, is more negative than \(-3\) Vdc with respect to signal ground. The signal will be considered in the “spacing” condition when the voltage is more positive than \(+3\) Vdc with respect to signal ground. The region between \(+3\) Vdc and \(-3\) Vdc is defined as the transition region, and considered an invalid level. The voltage 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 Control Status</td>
</tr>
<tr>
<td>X</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>None</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Divisor Latch (Least Significant Bit)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Divisor Latch (Most Significant Bit)</td>
</tr>
</tbody>
</table>

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

Receiver Clock (RCLK), Pin 9: This input is the 16 x baud rate clock for the receiver section of the chip.
Serial Input (SIN), Pin 10: Serial data input from the communications link (peripheral device, modem, or data set).

Clear to Send (CTS), Pin 36: The CTS signal is a modem control function input whose condition can be tested by the processor by reading bit 4 (CTS) of the modem status register. Bit 0 (DCTS) of the modem status register indicates whether the CTS input has changed state since the previous reading of the modem status register.

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

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

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

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

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

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

VCC, Pin 40: +5 Vdc supply.

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

Output Signals

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

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

Output 1 (OUT1), 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 OUT1 signal is set high upon a master reset operation.

Output 2 (OUT2), 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 OUT2 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, Bits 3-7 are Permanently Low</td>
</tr>
<tr>
<td>Line Control Register</td>
<td>Master Reset</td>
<td>All Bits Low</td>
</tr>
<tr>
<td>Modem Control Register</td>
<td>Master Reset</td>
<td>All Bits Low</td>
</tr>
<tr>
<td>Line Status Register</td>
<td>Master Reset</td>
<td>Except Bits 5 and 6 are High</td>
</tr>
<tr>
<td>Modem Status Register</td>
<td>Master Reset</td>
<td>Bits 0-3 Low, Bits 4-7 - Input Signal</td>
</tr>
<tr>
<td>SOUT</td>
<td>Master Reset</td>
<td>High</td>
</tr>
<tr>
<td>INTRPT (RCVR Errors)</td>
<td>Read LSR/MR</td>
<td>Low</td>
</tr>
<tr>
<td>INTRPT (RCVR Data Ready)</td>
<td>Read RBR/MR</td>
<td>Low</td>
</tr>
<tr>
<td>INTRPT (RCVR Data Ready)</td>
<td>Read IIR/ Write THR/MR</td>
<td>Low</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

1-234  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.

### 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 \(16 \times \text{the baud rate} \ [\text{divisor} \# = (\text{frequency input})/(\text{baud rate} \times 16)]\). 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)
Divisor Latch Most Significant Bit (DLM)

The following figure illustrates the use of the baud rate generator with a frequency of 1.8432 MHz. For baud rates of 9600 and below, the error obtained is minimal.

**Note:** The maximum operating frequency of the baud generator is 3.1 MHz. In no case should the data rate be greater than 9600 baud.

<table>
<thead>
<tr>
<th>Desired Baud Rate</th>
<th>Divisor Used to Generate 16x Clock (Decimal)</th>
<th>Divisor Used to Generate 16x Clock (Hex)</th>
<th>Percent Error Difference Between Desired and Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>2304</td>
<td>900</td>
<td>—</td>
</tr>
<tr>
<td>75</td>
<td>1536</td>
<td>600</td>
<td>—</td>
</tr>
<tr>
<td>110</td>
<td>1047</td>
<td>417</td>
<td>0.026</td>
</tr>
<tr>
<td>134.5</td>
<td>857</td>
<td>359</td>
<td>0.058</td>
</tr>
<tr>
<td>150</td>
<td>768</td>
<td>300</td>
<td>—</td>
</tr>
<tr>
<td>300</td>
<td>384</td>
<td>180</td>
<td>—</td>
</tr>
<tr>
<td>600</td>
<td>192</td>
<td>0C0</td>
<td>—</td>
</tr>
<tr>
<td>1200</td>
<td>96</td>
<td>060</td>
<td>—</td>
</tr>
<tr>
<td>1800</td>
<td>64</td>
<td>040</td>
<td>—</td>
</tr>
<tr>
<td>2000</td>
<td>58</td>
<td>03A</td>
<td>0.69</td>
</tr>
<tr>
<td>2400</td>
<td>48</td>
<td>030</td>
<td>—</td>
</tr>
<tr>
<td>3600</td>
<td>32</td>
<td>020</td>
<td>—</td>
</tr>
<tr>
<td>4800</td>
<td>24</td>
<td>018</td>
<td>—</td>
</tr>
<tr>
<td>7200</td>
<td>16</td>
<td>010</td>
<td>—</td>
</tr>
<tr>
<td>9600</td>
<td>12</td>
<td>00C</td>
<td>—</td>
</tr>
</tbody>
</table>

Baud Rate at 1.843 MHz

1-238 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>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 7 6 5 4 3 2 1 0</td>
</tr>
</tbody>
</table>

- **Data Ready (DR)**
- **Overrun Error (OR)**
- **Parity Error (PE)**
- **Framing Error (FE)**
- **Break Interrupt (BI)**
- **Transmitter Holding Register Empty (THRE)**
- **Tx Shift Register Empty (TSRE)**

**Line Status Register (LSR)**

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

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

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

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

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

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

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

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

Interrupt Identification Register

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

<table>
<thead>
<tr>
<th>Hex Address 3FA</th>
<th>Bit 7 6 5 4 3 2 1 0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0 If Interrupt Pending</td>
</tr>
<tr>
<td></td>
<td>Interrupt ID Bit (0)</td>
</tr>
<tr>
<td></td>
<td>Interrupt ID Bit (1)</td>
</tr>
<tr>
<td></td>
<td>= 0</td>
</tr>
<tr>
<td></td>
<td>= 0</td>
</tr>
<tr>
<td></td>
<td>= 0</td>
</tr>
<tr>
<td></td>
<td>= 0</td>
</tr>
</tbody>
</table>

Interrupt Identification Register (IIR)

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

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

**Bits 3 through 7:** These five bits of the IIR are always logical 0.
<table>
<thead>
<tr>
<th>Interrupt ID Register</th>
<th>Interrupt Set and Reset Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 2</td>
<td>Bit 1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Interrupt Control Functions

1-242 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 Diagram]

Interrupt Enable Register (IER)

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

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

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

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

Modem Control Register

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

<table>
<thead>
<tr>
<th>Hex Address 3FC</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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Data Terminal Ready (DTR)</td>
<td>Request to Send (RTS)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Out 1</td>
<td>Out 2</td>
<td>Loop</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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 DTR 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 (OUT 1) signal, which is an auxiliary user-designated output. Bit 2 affects the OUT 1 output in a manner identical to that described above for bit 0.

Bit 3: This bit controls the output 2 (OUT 2) signal, which is an auxiliary user-designated output. Bit 3 affects the 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, OUT 1, and 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>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 7 6 5 4 3 2 1 0</td>
</tr>
<tr>
<td>----------------------</td>
</tr>
<tr>
<td>Delta Clear to Send (DCTS)</td>
</tr>
<tr>
<td>Delta Data Set Ready (DDSR)</td>
</tr>
<tr>
<td>Trailing Edge Ring Indicator (TERI)</td>
</tr>
<tr>
<td>Delta Rx Line Signal Detect (DRLSD)</td>
</tr>
<tr>
<td>Clear to Send (CTS)</td>
</tr>
<tr>
<td>Data Set Ready (DSR)</td>
</tr>
<tr>
<td>Ring Indicator (RI)</td>
</tr>
<tr>
<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 CTS input to the chip has changed state since the last time it was read by the processor.

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

Bit 2: This bit is the trailing edge of ring indicator (TERI) detector. Bit 2 indicates that the RI input to the chip has changed from an on (logical 1) to an off (logical 0) condition.
Bit 3: This bit is the delta received line signal detector (DRLSD) indicator. Bit 3 indicates that the RLSD input to the chip has changed state.

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

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

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

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

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

Receiver Buffer Register

The receiver buffer register contains the received character as defined below:

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

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

Transmitter Holding Register (THR)

Bit 0 is the least significant bit and is the first bit serially transmitted.
Selecting the Interface Format and Adapter Address

The voltage or current loop interface and adapter address are selected by plugging the programmed shunt modules with the locator dots up or down. See the figure below for the configurations.
### Rear Panel

At standard RS-232C Levels (with exception of current loops)

<table>
<thead>
<tr>
<th>Description</th>
<th>Pin</th>
</tr>
</thead>
<tbody>
<tr>
<td>NC</td>
<td>1</td>
</tr>
<tr>
<td>Transmitted Data</td>
<td>2</td>
</tr>
<tr>
<td>Received Data</td>
<td>3</td>
</tr>
<tr>
<td>Request to Send</td>
<td>4</td>
</tr>
<tr>
<td>Clear to Send</td>
<td>5</td>
</tr>
<tr>
<td>Data Set Ready</td>
<td>6</td>
</tr>
<tr>
<td>Signal Ground</td>
<td>7</td>
</tr>
<tr>
<td>Received Line Signal Detector</td>
<td>8</td>
</tr>
<tr>
<td>+Transmit Current Loop Data</td>
<td>9</td>
</tr>
<tr>
<td>NC</td>
<td>10</td>
</tr>
<tr>
<td>-Transmit Current Loop Data</td>
<td>11</td>
</tr>
<tr>
<td>NC</td>
<td>12</td>
</tr>
<tr>
<td>NC</td>
<td>13</td>
</tr>
<tr>
<td>NC</td>
<td>14</td>
</tr>
<tr>
<td>NC</td>
<td>15</td>
</tr>
<tr>
<td>NC</td>
<td>16</td>
</tr>
<tr>
<td>NC</td>
<td>17</td>
</tr>
<tr>
<td>+Receive Current Loop Data</td>
<td>18</td>
</tr>
<tr>
<td>NC</td>
<td>19</td>
</tr>
<tr>
<td>Data Terminal Ready</td>
<td>20</td>
</tr>
<tr>
<td>NC</td>
<td>21</td>
</tr>
<tr>
<td>Ring Indicator</td>
<td>22</td>
</tr>
<tr>
<td>NC</td>
<td>23</td>
</tr>
<tr>
<td>NC</td>
<td>24</td>
</tr>
<tr>
<td>-Receive Current Loop Return</td>
<td>25</td>
</tr>
</tbody>
</table>

Note: To avoid inducing voltage surges on interchange circuits, signals from interchange circuits shall be used to drive inductive devices, such as relay coils.

### Connector Specifications

1-250 Asynchronous Adapter
The binary synchronous communication (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.
8251A Universal Synchronous/Asynchronous Receiver/Transmitter

The 8251A operational characteristics are programmed by the system unit’s software, and it can support virtually any form of synchronous data technique currently in use. In the configuration being described, the 8251A is used for IBM’s binary synchronous communications (BSC) protocol in half-duplex mode.

Operation of the 8251A is started by programming the communications format, then entering commands to tell the 8251A what operation is to be performed. In addition, the 8251A can pass device status to the system unit by doing a Status Read operation. The sequence of events to accomplish this are mode instruction, command instruction, and status read. Mode instruction must follow a master reset operation. Commands can be issued in the data block at any time during operation of the 8251A.

A block diagram of the 8251A follows:
**Data Bus Buffer**

The system unit's data bus interfaces the 8251A through the data bus buffer. Data is transferred or received by the buffer upon execution of input or output instructions from the system unit. Control words, command words, and status information are also transferred through the data bus buffer.

**Read/Write Control Logic**

The read/write control logic controls the transfer of information between the system unit and the 8251A. It consists of pins designated as RESET, CLK, WR, RD, C/D, and CS.

**RESET:** The Reset pin is gated by Port B, bit 4 of the 8255, and performs a master reset of the 8251A. The minimum reset pulse width is 6 clock cycles. Clock-cycle duration is determined by the oscillator speed of the processor.

**CLK (Clock):** The clock generates internal device timing. No external inputs or outputs are referenced to CLK. The input is the system board's bus clock of 4.77 MHz.

**WR (Write):** An input to WR informs the 8251A that the system unit is writing data or control words to it. The input is the WR signal from the system-unit bus.

**RD (Read):** An input to RD informs the 8251A that the processing unit is reading data or status information from it. The input is the RD signal from the system-unit bus.

**C/D (Control/Data):** An input on this pin, in conjunction with the WR and RD inputs, informs the 8251A that the word on the data bus is either a data character, a control word, or status information. The input is the low-order address bit from the system board's address bus.

**CS (Chip Select):** A low on the input selects the 8251A. No reading or writing will occur unless the device is selected. An input is decoded at the adapter from the address information on the system-unit bus.
Modem Control

The 8251A has the following input and output control signals which are used to interface the transmission equipment selected by the user.

**DSR (Data Set Ready):** The DSR input port is a general-purpose, 1-bit, inverting input port. The 8251A can test its condition with a Status Read operation.

**CTS (Clear to Send):** A low on this input enables the 8251A to transfer serial data if the TxEnable bit in the command byte is set to 1. If either a TxEnable off or CTS off condition occurs while the transmitter is in operation, the transmitter will send all the data in the USART that was written prior to the TxDisable command, before shutting down.

**DTR (Data Terminal Ready):** The DTR output port is a general-purpose, 1-bit, inverting output port. It can be set low by programming the appropriate bit in the command instruction word.

**RTS (Request to Send):** The RTS output signal is a general-purpose, 1-bit, inverting output port. It can be set low by programming the appropriate bit in the Command Instruction word.

Transmitter Buffer

The transmitter buffer accepts parallel data from the data-bus buffer, converts it to a serial bit stream, and inserts the appropriate characters or bits for the BSC protocol. The output from the transmit buffer is a composite serial stream of data on the falling edge of Transmit Clock. The transmitter will begin transferring data upon being enabled, if CTS = 0 (active). The transmit data (TxD) line will be set in the marking state upon receipt of a master reset, or when transmit enable/CTS is off and the transmitter is empty (TxEmpty).
Transmitter Control

Transmitter control manages all activities associated with the transfer of serial data. It accepts and issues the following signals, both externally and internally, to accomplish this function:

**TxRDY (Transmitter Ready):** This output signals the system unit that the transmitter is ready to accept a data character. The TxRDY output pin is used as an interrupt to the system unit (Level 4) and is masked by turning off Transmit Enable. TxRDY is automatically reset by the leading edge of a WR input signal when a data character is loaded from the system unit.

**TxE (Transmitter Empty):** This signal is used only as a status register input.

**TxC (Transmit Clock):** The Transmit Clock controls the rate at which the character is to be transmitted. In synchronous mode, the bit-per-second rate is equal to the TxC frequency. The falling edge of TxC shifts the serial data out of the 8251A.

Receiver Buffer

The receiver accepts serial data, converts it to parallel format, checks for bits or characters that are unique to the communication technique, and sends an “assembled” character to the system unit. Serial data input is received on the RxD (Receive Data) pin, and is clocked in on the rising edge of RxC (Receive Clock).

Receiver Control

This control manages all receiver-related activities. The parity-toggle and parity-error flip-flop circuits are used for parity-error detection, and set the corresponding status bit.
RxRDY (Receiver Ready): This output indicates that the 8251A has a character that is ready to be received by the system unit. RxRDY is connected to the interrupt structure of the system unit (Interrupt Level 3). With Receive Enable off, RxRDY is masked and held in the reset mode. To set RxRDY, the receiver must be enabled, and a character must finish assembly and be transferred to the data output register. Failure to read the received character from the RxData output register before the assembly of the next RxData character will set an overrun-condition error, and the previous character will be lost.

RxC (Receiver Clock): The receiver clock controls the rate at which the character is to be received. The bit rate is equal to the actual frequency of RxC.

SYNDET (Synchronization Detect): This pin is used for synchronization detection and may be used as either input or output, programmable through the control word. It is reset to output-mode-low upon reset. When used as an output (internal synchronization mode), the SYNDET pin will go to 1 to indicate that the 8251A has found the synchronization character in the receive mode. If the 8251A is programmed to use double synchronization characters (bisynchronization as in this application), the SYNDET pin will go to 1 in the middle of the last bit of the second synchronization character. SYNDET is automatically reset for a Status Read operation.

8255A-5 Programmable Peripheral Interface

The 8255A-5 is used on the BSC adapter to provide an expanded modem interface and for internal gating and control functions. It has three 8-bit ports, which are defined by the system during initialization of the adapter. All levels are considered plus active unless otherwise indicated. A detailed description of the ports is in “Programming Considerations” in this section.
8253-5 Programmable Interval Timer

The 8253-5 is driven by a divided-by-two system-clock signal. Its outputs are used as clocking signals and to generate inactivity timeout interrupts. These level 4 interrupts occur when either of the timers reaches its programmed terminal counts. The 8253-5 has the following outputs:

Timer 0: Not used for synchronous-mode operation.

Timer 1: Connected to port A, bit 7 of the 8255 and Interrupt Level 4.

Timer 2: Connected to port A, bit 6 of the 8255 and Interrupt Level 4.

Operation

The complete functional definition of the BSC adapter is programmed by the system software. Initialization and control words are sent out by the system to initialize the adapter and program the communications format in which it operates. Once programmed, the BSC Adapter is ready to perform its communication functions.

Transmit

In synchronous transmission, the TxD output is continuously at a mark level until the system sends its first character, which is a synchronization character to the 8251A. When the CTS line goes on, the first character is serially transmitted. All bits are shifted out on the falling edge of TxC. When the 8251A is ready to receive another character from the system for transmission, it raises TxRDY, which causes a level-4 interrupt.
Once transmission has started, the data stream at the TxD output must continue at the TxC rate. If the system does not provide the 8251A with a data character before the 8251A transmit buffers become empty, the synchronization characters will be automatically inserted in the TxD data stream. In this case, the TxE bit in the status register is raised high to signal that the 8251A is empty and that synchronization characters are being sent out. (Note that this TxE bit is in the status register, and is not the TxE pin on the 8251A). TxE does not go low when SYNC is being shifted out. The TxE status bit is internally reset by a data character being written to the 8251A.

Receive

In synchronous reception, the 8251A will achieve character synchronization, because the hardware design of the BSC adapter is intended for internal synchronization. Therefore, the SYNDDET 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 SYNDDET bit in the status register (not the SYNDDET pin) is then set high, and is reset automatically by a Status Read.

Once synchronization has occurred, the 8251A begins to assemble received data bytes. When a character is assembled and ready to be transferred to memory from the 8251A, it raises RxRDY, causing an interrupt level 3 to the system.

If the system has not fetched a previous character by the time another received character is assembled (and an interrupt-level 3 issued by the adapter), the old character will be overwritten, and the overrun error flag will be raised. All error flags can be reset by an error reset operation.
Programming Considerations

Before starting data transmission or reception, the BSC adapter is programmed by the system unit to define control and gating ports, timer functions and counts, and the communication environment in which it is to operate.

Typical Programming Sequence

The 8255A-5 programmable peripheral interface (PPI) is initialized for the proper mode by selecting address hex 3A3 and writing the control word. This defines port A as an input, port B as an output for modem control and gating, and port C for 4-bit input and 4-bit output. The bit descriptions for the 8255A-5 are shown in the following figures. Using an output to port C, the adapter is then set to wrap mode, disallow interrupts, and gate external clocks (address=3A2H, data=ODH). 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.

1-260 BSC Adapter
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></td>
</tr>
</tbody>
</table>

**Definition of Control**

**SC** – Select Counter:

<table>
<thead>
<tr>
<th>SC1</th>
<th>SC0</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Select Counter 0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Select Counter 1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Select Counter 2</td>
<td></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></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Counter Latching operation</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Read/Load most significant byte only</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Read/Load least significant byte only</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Read/Load least significant byte first, then most significant byte</td>
<td></td>
</tr>
</tbody>
</table>

**M** – Mode:

<table>
<thead>
<tr>
<th>M2</th>
<th>M1</th>
<th>M0</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Mode 0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Terminal Count Interrupt</td>
<td></td>
</tr>
</tbody>
</table>

**BCD:**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Binary Counter 16-bits</td>
</tr>
<tr>
<td>1</td>
<td>Binary Coded Decimal (BCD) Counter (4 Decades)</td>
</tr>
</tbody>
</table>

---

**8253-5 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 communications 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 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.

```
3A9 C/D = 1
Mode Instruction 1
3A9 C/D = 1
SYNC Character 1
3A9 C/D = 1
SYNC Character 2
3A9 C/D = 1
Command Instruction
3A8 C/D = 0
Data
3A9 C/D = 1
Command Instruction
3A8 C/D = 0
Data
3A9 C/D = 1
Command Instruction
```

Typical Data Block
Mode Instruction Definition

The mode instruction defines the general operational characteristics of the 8251A. It follows a reset operation (internal or external). Once the mode instruction has been written to the 8251A by the system unit, synchronization characters or command instructions may be written to the device. The following figure shows the format for the mode instruction.

<table>
<thead>
<tr>
<th>Mode Instruction Format</th>
<th>Address: Hex 3A9 for BSC</th>
<th>Hex 389 for Alternate BSC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 7 6 5 4 3 2 1 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 0 Not used; always = 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 1 Not used; always = 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 2 These two bits are used together to define the character and 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>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 4 In the synchronous mode, parity is enabled from this bit. A 1 on this bit sets parity enable.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 5 The parity generation/check is set from this bit. For BSC, even parity is used by having bit 5 = 1.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 6 External synchronization is set by this bit. A 1 on this bit establishes synchronization detection as an input.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 7 This bit establishes the mode of character synchronization. A 0 is set on this bit to give double character synchronization.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

BSC Adapter 1-263
**Command-Instruction Format**

The command-instruction format defines a status word that is used to control the actual operation of the 8251A. Once the mode instruction has been written to the 8251A, and SYNC characters loaded, all further "Control Writes" to I/O address hex 3A9 or hex 389 will load a command instruction.

Data is transferred by accessing two I/O ports on the 8251A, ports 3A8 and 388. A byte of data can be read from port 3A8 and can be written to port 388.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Transmit Enable</td>
</tr>
<tr>
<td>6</td>
<td>Data Terminal Ready</td>
</tr>
<tr>
<td>5</td>
<td>Receive Enable</td>
</tr>
<tr>
<td>4</td>
<td>Send Break Character</td>
</tr>
<tr>
<td>3</td>
<td>Error Reset</td>
</tr>
<tr>
<td>2</td>
<td>Request to Send</td>
</tr>
<tr>
<td>1</td>
<td>Internal Reset</td>
</tr>
<tr>
<td>0</td>
<td>Enter Hunt Mode</td>
</tr>
</tbody>
</table>

**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>Bits</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.

Status Read Format

Bit 0 See the Note in the preceding chart.

Bit 1 An output on this bit means a character is ready to be received by the computers 8088 microprocessor.
Bit 2  A 1 on this bit indicates the 8251A has no characters to transmit.

Bit 3  The Parity Error bit sets a flag when errors are detected. It is reset by the error reset in the command instruction.

Bit 4  This bit sets a flag when the computers 8088 microprocessor does not read a character before another one is presented. The 8251A operation is not inhibited by this flag, but the overrun character will be lost.

Bit 5  Not used

Bit 6  SYNDET goes to 1 when the synchronization character is found in receive mode. For BSC, SYNDET goes high in the middle of the last bit of the second synchronization character.

Bit 7  The Data Set Ready bit is a one bit inverting input. It is used to check modem conditions, such as data-set ready.

Interface Signal Information

The BSC adapter conforms to interface signal levels standardized by the Electronics Industry Association (EIA) RS232C Standard. These levels are shown in the following figure.

Additional lines, not standardized by the EIA, are pins 11, 18, and 25 on the interface connector. These lines are designated as Select Standby, Test, and Test Indicate. Select Standby is used to support the switched network backup facility of a modem that provides this option. Test and Test Indicate support a modem wrap function on modems that are designated for business-machine, controlled-modem wraps.
<table>
<thead>
<tr>
<th>Driver</th>
<th>EIA RS232C/CCITT V24-V28 Signal Levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>+15 Vdc</td>
<td>Active/Data = 0</td>
</tr>
<tr>
<td>+5 Vdc</td>
<td></td>
</tr>
<tr>
<td>+5 Vdc</td>
<td>Invalid Level</td>
</tr>
<tr>
<td>-5 Vdc</td>
<td></td>
</tr>
<tr>
<td>-5 Vdc</td>
<td>Inactive/Data = 1</td>
</tr>
<tr>
<td>-15 Vdc</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Receiver</th>
<th>EIA RS232C/CCITT V24-V28 Signal Levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>+25 Vdc</td>
<td>Active/Data = 0</td>
</tr>
<tr>
<td>+3 Vdc</td>
<td></td>
</tr>
<tr>
<td>+3 Vdc</td>
<td>Invalid Level</td>
</tr>
<tr>
<td>-3 Vdc</td>
<td></td>
</tr>
<tr>
<td>-3 Vdc</td>
<td>Inactive/Data = 1</td>
</tr>
<tr>
<td>-25 Vdc</td>
<td></td>
</tr>
</tbody>
</table>

Interface Voltage Levels
Interrupt Information

Interrupt Level 4: Transmitter Ready
Counter 1
Counter 2

Interrupt Level 3: Receiver Ready

<table>
<thead>
<tr>
<th>Hex Address</th>
<th>Device</th>
<th>Register Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>3A0 380</td>
<td>8255</td>
<td>Port A Data</td>
<td>Internal/External Sensing</td>
</tr>
<tr>
<td>3A1 381</td>
<td>8255</td>
<td>Port B Data</td>
<td>External Modem Interface</td>
</tr>
<tr>
<td>3A2 382</td>
<td>8255</td>
<td>Port C Data</td>
<td>Internal Control</td>
</tr>
<tr>
<td>3A3 383</td>
<td>8255</td>
<td>Mode Set</td>
<td>8255 Mode Initialization</td>
</tr>
<tr>
<td>3A4 384</td>
<td>8253</td>
<td>Counter 0 LSB</td>
<td>Not Used in Synch Mode</td>
</tr>
<tr>
<td>3A4 384</td>
<td>8253</td>
<td>Counter 0 MSB</td>
<td>Not Used in Synch Mode</td>
</tr>
<tr>
<td>3A5 385</td>
<td>8253</td>
<td>Counter 1 LSB</td>
<td>Inactivity Time-Outs</td>
</tr>
<tr>
<td>3A5 385</td>
<td>8253</td>
<td>Counter 1 MSB</td>
<td>Inactivity Time-Outs</td>
</tr>
<tr>
<td>3A6 386</td>
<td>8253</td>
<td>Counter 2 LSB</td>
<td>Inactivity Time-Outs</td>
</tr>
<tr>
<td>3A6 386</td>
<td>8253</td>
<td>Counter 2 MSB</td>
<td>Inactivity Time-Outs</td>
</tr>
<tr>
<td>3A7 387</td>
<td>8253</td>
<td>Mode Register</td>
<td>8253 Mode Set</td>
</tr>
<tr>
<td>3A8 388</td>
<td>8251</td>
<td>Data Select</td>
<td>Data</td>
</tr>
<tr>
<td>3A9 389</td>
<td>8251</td>
<td>Command/Status</td>
<td>Mode/Command</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>USART Status</td>
</tr>
</tbody>
</table>

Device Address Summary
## Signal Name — Description

<table>
<thead>
<tr>
<th>Signal Name</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**

**BSC Adapter  1-269**
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.
8273 Protocol Controller Structure

The 8273 module consists of two major interfaces: the processor interface and the modem interface. A block diagram of the 8273 protocol controller module follows.

---

### Registers

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Txl/R</td>
<td>Rxl/R</td>
<td>Reset</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Command</td>
<td>Parameter</td>
<td>Status</td>
</tr>
</tbody>
</table>

---

### 8273 SDLC Protocol Control Block Diagram

---

**Processor Interface**

**Modem Interface**

---

SDLC Adapter 1-273
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

1-274 SDLC Adapter
### 8273 Control/Read/Write Registers

<table>
<thead>
<tr>
<th><strong>Command</strong></th>
<th>Operations are initialized by writing the appropriate command byte into this register.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Status</strong></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><strong>Parameter</strong></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><strong>Immediate Result (Result)</strong></td>
<td>Commands that execute immediately produce a result byte in this register, to be read by the processor.</td>
</tr>
<tr>
<td><strong>Transmit Interrupt Results (TxI/R)</strong></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><strong>Receiver Interrupt Results (Rx/I/R)</strong></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><strong>Reset</strong></td>
<td>This register provides a software reset function for the 8273.</td>
</tr>
</tbody>
</table>

The other elements of the C/R/W logic are the interrupt lines (RxINT and TxINT). Interrupt priorities are listed in the “Interrupt Information” table in this section. These lines signal the processor that either the transmitter or the receiver requires service (results should be read from the appropriate register), or a data transfer is required. The status of each interrupt line is also reflected by a bit in the status register, so non-interrupt driven operation is also possible by the communication software examining these bits periodically.
Data Interfaces

The 8273 supports two independent data interfaces through the data transfer logic: received data and transmitted data. These interfaces are programmable for either DMA or non-DMA data transfers. Speeds below 9600 bits-per-second may or may not require DMA, depending on the task load and interrupt response time of the processor. The processor DMA controller is used for management of DMA data transfer timing and addressing. The 8273 handles the transfer requests and actual counts of data-block lengths. DMA level 1 is used to transmit and receive data transfers. Dual DMA support is not provided.

Elements of Data Transfer Interface

TxDRQ/RxDRQ This line requests a DMA to or from memory and is asserted by the 8273.

TxDACK/RxDACK This line notifies the 8273 that a request has been granted and provides access to data regions. This line is returned by the DMA controller (DACK1 on the system unit control bus is connected to TxDACK/RxDACK on the 8273).

RD (Read) This line indicates data is to be read from the 8273 and placed in memory. It is controlled by the processor DMA controller.

WR (Write) This line indicates if data is to be written to the 8273 from memory and is controlled by the processor DMA controller.

To request a DMA transfer, the 8273 raises the DMA request line. Once the DMA controller obtains control of the system bus, it notifies the 8273 that the DRQ is granted by returning DACK, and WR or RD, for a transmit or receive operation, respectively. The DACK and WR or RD signals transfer data between the 8273 and memory, independent of the 8273 chip-select pin (CS). This “hard select” of data into the transmitter or out of the receiver alleviates the need for the normal transmit and receive data registers, addressed by a combination of address lines, CS, and WR or RD.
Modem Interface

The modem interface of the 8273 consists of two major blocks: the modem control block and the serial data timing block.

Modem Control Block

The modem control block provides both dedicated and user-defined modem control function. EIA inverting drivers and receivers are used to convert TTL levels to EIA levels.

Port A is a modem control input port. Bits PA0 and PA1 have dedicated functions.

<table>
<thead>
<tr>
<th>8273 Port A (Modem Control Input Port)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit PA</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>PA0 Clear to Send</td>
</tr>
<tr>
<td>PA1 Carrier Detect</td>
</tr>
<tr>
<td>PA2 Data Set Ready</td>
</tr>
<tr>
<td>PA3 CTS Change</td>
</tr>
<tr>
<td>PA4 DSR Change</td>
</tr>
<tr>
<td>Not Used</td>
</tr>
</tbody>
</table>

Bit PA0     This bit reflects the logical state of the clear to send (CTS) pin. The 8273 waits until CTS is active before it starts transmitting a frame. If CTS goes inactive while transmitting, the frame is aborted and the processor is interrupted. A CTS failure will be indicated in the appropriate interrupt-result register.

Bit PA1     This bit reflects the logical state of the carrier detect pin (CD). CD must be active in sufficient time for reception of a frame’s address field. If CD is lost (goes inactive) while receiving a frame, an interrupt is generated with a CD failure result.

Bit PA2     This bit is a sense bit for data set ready (DSR).

Bit PA3     This bit is a sense bit to detect a change in CTS.

SDLC Adapter 1-277
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.
8255A-5 Programmable Peripheral Interface

The 8255A-5 contains three 8-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
8255A-5 Port C Assignments

<table>
<thead>
<tr>
<th>Bit</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 = Gate Internal Clock (Output Bit)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 = Gate External Clock (Output Bit)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 = Electronic Wrap (Output Bit)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 = Gate Interrupts 3 and 4 (Output Bit)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Oscillating = Receive Data (Input Bit)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Oscillating = Timer 0 Output (Input bit)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 = Test Indicate Active (Input Bit)</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>

*Port C is defined for internal control and gating functions. It has three input and four output bits. The four output bits are defined during initialization, but only three are used.

8253-5 Programmable Interval Timer

The 8253-5 is driven by a processor clock signal divided by two. It has the following output:

Timer 0  Programmed to generate a square wave signal, used as an input to timer 2. Also connected to 8253 port C, bit 5.

Timer 1  Connected to 8255 port A, bit 7, and interrupt level 4.

Timer 2  Connected to 8255 port A, bit 6, and interrupt level 4.

Programming Considerations

The software aspects of the 8273 involve the communication of both commands from the processor to the 8273 and the return of results of those commands from the 8273 to the processor. Due to the internal processor architecture of the 8273, this system unit/8273 communication is basically a form of interprocessor communication, and must be considered when programming for the SDLC communications adapter.
The protocol for this interprocessor communication is implemented through use of handshaking supplied in the 8273 status register. The bit definitions of this register are shown below.

<table>
<thead>
<tr>
<th>8273 Status Register Format</th>
<th>Hex Address 388</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bit 0 This bit is the transmitter interrupt result available (T x I R A) bit. This bit is set when the 8273 places an interrupt-result byte in the T x I / R register, and reset when the processor reads the T x I / R register.

Bit 1 This bit is the receiver interrupt result available (R x I R A) bit. It is the corresponding result-available bit for the receiver. It is set when the 8273 places an interrupt-result byte in the R x I / R register and reset when the processor reads the register.

Bit 2 This bit is the transmitter interrupt (T x I N T) bit and reflects the state of the T x I N T pin. T x I N T 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 (R x I N T) bit and is identical to the T x I N T, except action is initiated based on receiver interrupt-sources.

Bit 4 This bit is the command result buffer full (C R B F) 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>Description</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>Description</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>Value</th>
<th>Description</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 7 6 5 4 3 2 1 0</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>8273 Serial I/O Mode Register Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit</td>
</tr>
<tr>
<td>-----</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>Not Used</td>
</tr>
<tr>
<td>Not Used</td>
</tr>
<tr>
<td>Not Used</td>
</tr>
<tr>
<td>Not Used</td>
</tr>
<tr>
<td>Not Used</td>
</tr>
<tr>
<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>8273 Data Transfer Mode Register Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit</td>
</tr>
<tr>
<td>-----</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<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 7 6 5 4 3 2 1 0</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Not Used</td>
</tr>
<tr>
<td>1 = One-Bit Delay Enable</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>64</td>
<td>Reset Mask</td>
<td>None</td>
<td>—</td>
<td>No</td>
</tr>
<tr>
<td>Reset One-Bit Delay</td>
<td>57</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>Reset 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>60</td>
<td>Reset Mask</td>
<td>None</td>
<td>—</td>
<td>No</td>
</tr>
<tr>
<td>General Receive</td>
<td>C0</td>
<td>80,81</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>80,81,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 Result</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Read Port B</td>
<td>23</td>
<td>None</td>
<td>Port Value Result</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Set Port B Bit</td>
<td>63</td>
<td>Reset 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.
- **RO** — 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.

8273 SDLC Protocol Controller Commands

SDLC Adapter 1-291
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.
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.
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.
## 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>Inactivity Time-Outs</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

SDLC Adapter 1-297
Interface Information

The SDLC communications adapter conforms to interface signal levels standardized by the Electronics Industries Association RS-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).

<table>
<thead>
<tr>
<th>Drivers</th>
<th>Receivers</th>
</tr>
</thead>
<tbody>
<tr>
<td>+15 Vdc</td>
<td>+25 Vdc</td>
</tr>
<tr>
<td>+5 Vdc</td>
<td>+3 Vdc</td>
</tr>
<tr>
<td>-5 Vdc</td>
<td>-3 Vdc</td>
</tr>
<tr>
<td>-15 Vdc</td>
<td>-25 Vdc</td>
</tr>
</tbody>
</table>

Active Level: Data = 0

Invalid Level

Inactive Level: Data = 1
### Signal Name — Description

<table>
<thead>
<tr>
<th>Pin</th>
<th>Signal Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>No Connection</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Transmitted Data</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Received Data</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Request to Send</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Clear to Send</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Data Set Ready</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Signal Ground</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Received Line Signal Detector</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>No Connection</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>No Connection</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Select Standby*</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>No Connection</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>No Connection</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>No Connection</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Transmitter Signal Element Timing</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>No Connection</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Receiver Signal Element Timing</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Test (IBM Modems Only)*</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>No Connection</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Data Terminal Ready</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>No Connection</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>Ring Indicator</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>Data Signal Rate Selector</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>No Connection</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>Test Indicate (IBM Modems Only)*</td>
<td></td>
</tr>
</tbody>
</table>

*Not standardized by EIA (Electronics Industry Association).

### Connector Specifications

25-Pin D-Shell Connector

**SDLC Adapter** 1-299
Notes:

1-300    SDLC Adapter
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 RS-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</th>
<th>Name</th>
<th>Modem Connector</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pin #</td>
<td></td>
<td>Pin #</td>
</tr>
<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>NC</td>
<td>NC</td>
</tr>
<tr>
<td>NC</td>
<td>Select Standby</td>
<td>11</td>
</tr>
<tr>
<td>NC</td>
<td>NC</td>
<td>NC</td>
</tr>
<tr>
<td>NC</td>
<td>NC</td>
<td>NC</td>
</tr>
<tr>
<td>NC</td>
<td>Transmitter Signal Element Timing</td>
<td>15</td>
</tr>
<tr>
<td>NC</td>
<td>NC</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>Data Terminal Ready</td>
<td>20</td>
</tr>
<tr>
<td>NC</td>
<td>NC</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>NC</td>
<td>NC</td>
</tr>
<tr>
<td>25</td>
<td>Test Indicate</td>
<td>25</td>
</tr>
</tbody>
</table>

Connector Specifications

1-302 Communications Cable
SECTION 2: ROM BIOS AND SYSTEM USAGE

ROM BIOS .................................................. 2-2
Keyboard Encoding and Usage ............................. 2-11
BIOS Cassette Logic ....................................... 2-21
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
MOV CX,HIGH_COUNT
MOV DX,LOW_COUNT
INT 1AH
```

; function is to set time of day.
; establish the current time.
; set the time.

To read the time of day:

```
MOV AH,0
INT 1AH
```

; function is to read time of day.
; 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>D_EOI</td>
</tr>
<tr>
<td>4-7</td>
<td>1</td>
<td>Single Step</td>
<td>D_EOI</td>
</tr>
<tr>
<td>8-8</td>
<td>2</td>
<td>Nonmaskable</td>
<td>NMI_INT</td>
</tr>
<tr>
<td>C-F</td>
<td>3</td>
<td>Breakpoint</td>
<td>D_EOI</td>
</tr>
<tr>
<td>10-13</td>
<td>4</td>
<td>Overflow</td>
<td>D_EOI</td>
</tr>
<tr>
<td>14-17</td>
<td>5</td>
<td>Print Screen</td>
<td>PRINT_SCREEN</td>
</tr>
<tr>
<td>18-1B</td>
<td>6</td>
<td>Reserved</td>
<td>D_EOI</td>
</tr>
<tr>
<td>1D-1F</td>
<td>7</td>
<td>Reserved</td>
<td>D_EOI</td>
</tr>
<tr>
<td>20-23</td>
<td>8</td>
<td>Time of Day</td>
<td>TIMER_INT</td>
</tr>
<tr>
<td>24-27</td>
<td>9</td>
<td>Keyboard</td>
<td>KB_INT</td>
</tr>
<tr>
<td>28-2B</td>
<td>A</td>
<td>Reserved</td>
<td>D_EOI</td>
</tr>
<tr>
<td>2C-2F</td>
<td>B</td>
<td>Communications</td>
<td>D_EOI</td>
</tr>
<tr>
<td>30-33</td>
<td>C</td>
<td>Communications</td>
<td>D_EOI</td>
</tr>
<tr>
<td>34-37</td>
<td>D</td>
<td>Disk</td>
<td>D_EOI</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>D_EOI</td>
</tr>
<tr>
<td>40-43</td>
<td>10</td>
<td>Video</td>
<td>VIDEO_IO</td>
</tr>
<tr>
<td>44-47</td>
<td>11</td>
<td>Equipment Check</td>
<td>EQUIPMENT</td>
</tr>
<tr>
<td>48-4B</td>
<td>12</td>
<td>Memory</td>
<td>MEMORY_SIZE_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
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>4FO-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 into segment of start of program text</th>
<th>Offset into segment of start of variables (end of program text 1-1)</th>
<th>Keyboard buffer contents if 0-no characters in buffer if 1-characters in buffer</th>
<th>Character color in graphics mode Set to 1, 2, or 3 to get text in colors 1 to 3. Do not set to 0. (Default = 3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line number of current line being executed</td>
<td>2E 2</td>
<td>6A 1</td>
<td>4E 1</td>
</tr>
<tr>
<td>Line number of last error</td>
<td>347 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Offset into segment of start of program text</td>
<td>30 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Offset into segment of start of variables</td>
<td>358 2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Example

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

Example

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

100 Hex 64 Hex 00

BASIC Workspace Variables
Starting Address in Hex

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000</td>
<td>BIOS Interrupt Vectors</td>
</tr>
<tr>
<td>00080</td>
<td>Available Interrupt Vectors</td>
</tr>
<tr>
<td>00400</td>
<td>BIOS Data Area</td>
</tr>
<tr>
<td>00500</td>
<td>User Read/Write Memory</td>
</tr>
<tr>
<td>C8000</td>
<td>Disk Adapter</td>
</tr>
<tr>
<td>F0000</td>
<td>Read Only Memory</td>
</tr>
<tr>
<td>FEO00</td>
<td>BIOS Program Area</td>
</tr>
</tbody>
</table>

BIOS Memory Map

**BIOS Programming Hints**

The BIOS code is invoked through software interrupts. The programmer should not "hard code" BIOS addresses into applications. The internal workings and absolute addresses within BIOS are subject to change without notice.

If an error is reported by the disk or diskette code, you should reset the drive adapter and retry the operation. A specified number of retries should be required on diskette reads to ensure the problem is not due to motor start-up.

When altering I/O port bit values, the programmer should change only those bits which are necessary to the current task. Upon completion, the programmer should restore the original environment. Failure to adhere to this practice may be incompatible with present and future applications.
Adapter Cards with System-Accessible ROM Modules

The ROM BIOS provides a facility to integrate adapter cards with on board ROM code into the system. During the POST, interrupt vectors are established for the BIOS calls. After the default vectors are in place, a scan for additional ROM modules takes place. At this point, a ROM routine on the adapter card may gain control. The routine may establish or intercept interrupt vectors to hook themselves into the system.

The absolute addresses hex C8000 through hex F4000 are scanned in 2K blocks in search of a valid adapter card ROM. A valid ROM is defined as follows:

Byte 0: Hex 55
Byte 1: Hex AA
Byte 2: A length indicator representing the number of 512 byte blocks in the ROM (length/512).

A checksum is also done to test the integrity of the ROM module. Each byte in the defined ROM is summed modulo hex 100. This sum must be 0 for the module to be deemed valid.

When the POST identifies a valid ROM, it does a far call to byte 3 of the ROM (which should be executable code). The adapter card may now perform its power-on initialization tasks. The feature ROM should return control to the BIOS routines by executing a far return.
Notes:
Keyboard Encoding and Usage

Encoding

The keyboard routine provided by IBM in the ROM BIOS is responsible for converting the keyboard scan codes into what will be termed “Extended ASCII.”

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

Character Codes

The following character codes are passed through the BIOS keyboard routine to the system or application program. A “-1” means the combination is suppressed in the keyboard routine. The codes are returned in AL. See Appendix C for the exact codes. Also, see “Keyboard Scan Code Diagram” in Section 1.

<table>
<thead>
<tr>
<th>Key Number</th>
<th>Base Case</th>
<th>Upper Case</th>
<th>Ctrl</th>
<th>Alt</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Esc</td>
<td>Esc</td>
<td>Esc</td>
<td>-1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>!</td>
<td>-1</td>
<td>Note 1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>@</td>
<td>Nul (000)</td>
<td>Note 1</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>#</td>
<td>-1</td>
<td>Note 1</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>$</td>
<td>-1</td>
<td>Note 1</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>%</td>
<td>-1</td>
<td>Note 1</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>^</td>
<td>RS(030)</td>
<td>Note 1</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>&amp;</td>
<td>-1</td>
<td>Note 1</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>*</td>
<td>-1</td>
<td>Note 1</td>
</tr>
<tr>
<td>10</td>
<td>9</td>
<td>(</td>
<td>-1</td>
<td>Note 1</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>)</td>
<td>-1</td>
<td>Note 1</td>
</tr>
<tr>
<td>12</td>
<td>-</td>
<td>—</td>
<td>US(031)</td>
<td>Note 1</td>
</tr>
<tr>
<td>13</td>
<td>=</td>
<td>+</td>
<td>-1</td>
<td>Note 1</td>
</tr>
<tr>
<td>14</td>
<td>Backspace (008)</td>
<td>Backspace (008)</td>
<td>Del (127)</td>
<td>-1</td>
</tr>
<tr>
<td>15</td>
<td>(009)</td>
<td>(Note 1)</td>
<td>DC1 (017)</td>
<td>Note 1</td>
</tr>
<tr>
<td>16</td>
<td>q</td>
<td>Q</td>
<td>ETB (023)</td>
<td>Note 1</td>
</tr>
<tr>
<td>17</td>
<td>w</td>
<td>W</td>
<td></td>
<td></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>~</td>
<td>~</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>42 Shift</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>43</td>
<td>\</td>
<td>\</td>
<td>FS 028</td>
<td>-1</td>
</tr>
<tr>
<td>44</td>
<td>z</td>
<td>Z</td>
<td>SUB 026</td>
<td>Note 1</td>
</tr>
<tr>
<td>45</td>
<td>x</td>
<td>X</td>
<td>CAN 024</td>
<td>Note 1</td>
</tr>
<tr>
<td>46</td>
<td>c</td>
<td>C</td>
<td>ETX 003</td>
<td>Note 1</td>
</tr>
<tr>
<td>47</td>
<td>v</td>
<td>V</td>
<td>SYN 022</td>
<td>Note 1</td>
</tr>
<tr>
<td>48</td>
<td>b</td>
<td>B</td>
<td>STX 002</td>
<td>Note 1</td>
</tr>
<tr>
<td>49</td>
<td>n</td>
<td>N</td>
<td>SO 014</td>
<td>Note 1</td>
</tr>
<tr>
<td>50</td>
<td>m</td>
<td>M</td>
<td>CR 013</td>
<td>Note 1</td>
</tr>
<tr>
<td>51</td>
<td>\</td>
<td>&lt;</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>52</td>
<td>.</td>
<td>&gt;</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>53</td>
<td>/</td>
<td>?</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>54 Shift</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>55</td>
<td>*</td>
<td>(Note 2)</td>
<td>(Note 1)</td>
<td>-1</td>
</tr>
<tr>
<td>56 Alt</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>57</td>
<td>SP</td>
<td>SP</td>
<td>SP</td>
<td>SP</td>
</tr>
<tr>
<td>58 Caps Lock</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-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>Alt</th>
<th>Ctrl</th>
</tr>
</thead>
<tbody>
<tr>
<td>71</td>
<td>7</td>
<td>Home (Note 1)</td>
<td>-1</td>
<td>Clear Screen</td>
</tr>
<tr>
<td>72</td>
<td>8</td>
<td></td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>73</td>
<td>9</td>
<td>Page Up (Note 1)</td>
<td>-1</td>
<td>Top of Text and Home</td>
</tr>
<tr>
<td>74</td>
<td>-</td>
<td></td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>75</td>
<td>4</td>
<td></td>
<td>-1</td>
<td>Reverse Word (Note 1)</td>
</tr>
<tr>
<td>76</td>
<td>5</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>77</td>
<td>6</td>
<td></td>
<td>-1</td>
<td>Advance Word (Note 1)</td>
</tr>
<tr>
<td>78</td>
<td>+</td>
<td>+</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>79</td>
<td>1</td>
<td>End (Note 1)</td>
<td>-1</td>
<td>Erase to EOL (Note 1)</td>
</tr>
<tr>
<td>80</td>
<td>2</td>
<td></td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>81</td>
<td>3</td>
<td>Page Down (Note 1)</td>
<td>-1</td>
<td>Erase to EOS (Note 1)</td>
</tr>
<tr>
<td>82</td>
<td>0</td>
<td>Ins</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>83</td>
<td>.</td>
<td>Del (Notes 1,2)</td>
<td>Note 2</td>
<td>Note 2</td>
</tr>
</tbody>
</table>

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

Extended Functions

For certain functions that cannot be represented in the standard ASCII code, an extended code is used. A character code of 000 (Nul) is returned in AL. This indicates that the system or application program should examine a second code that will indicate the actual function. Usually, but not always, this second code is the scan code of the primary key that was pressed. This code is returned in AH.

<table>
<thead>
<tr>
<th>Second Code</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Nul Character</td>
</tr>
<tr>
<td>15</td>
<td></td>
</tr>
<tr>
<td>30-38</td>
<td>Alt A, S, D, F, G, H, J, K, L</td>
</tr>
<tr>
<td>44-50</td>
<td>Alt Z, X, C, V, B, N, M</td>
</tr>
<tr>
<td>59-68</td>
<td>F1 to F10 Function Keys Base Case</td>
</tr>
<tr>
<td>71</td>
<td>Home</td>
</tr>
<tr>
<td>72</td>
<td></td>
</tr>
<tr>
<td>73</td>
<td>Page Up and Home Cursor</td>
</tr>
<tr>
<td>75</td>
<td></td>
</tr>
<tr>
<td>77</td>
<td>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 (Uppercase 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

2-16 Keyboard Encoding
Shift States

Most shift states are handled within the keyboard routine, transparent to the system or application program. In any case, the current set of active shift states are available by calling an entry point in the ROM keyboard routine. The following keys result in altered shift states:

Shift

This key temporarily shifts keys 2-13, 15-27, 30-41, 43-53, 55, and 59-68 to upper case (base case if in Caps Lock state). Also, the Shift key temporarily reverses the Num Lock or non-Num-Lock state of keys 71-73, 75, 77, and 79-83.

Ctrl

This key temporarily shifts keys 3, 7, 12, 14, 16-28, 30-38, 43-50, 55, 59-71, 73, 75, 77, 79, and 81 to the Ctrl state. Also, the Ctrl key is used with the Alt and Del keys to cause the “system reset” function, with the Scroll Lock key to cause the “break” function, and with the Num Lock key to cause the “pause” function. The system reset, break, and pause functions are described in “Special Handling” on the following pages.

Alt

This key temporarily shifts keys 2-13, 16-25, 30-38, 44-50, and 59-68 to the Alt state. Also, the Alt key is used with the Ctrl and Del keys to cause the “system reset” function described in “Special Handling” on the following pages.

The Alt key has another use. This key allows the user to enter any character code from 0 to 255 into the system from the keyboard. The user holds down the Alt key and types the decimal value of the characters desired using the numeric keypad (keys 71-73, 75-77, and 79-82). The Alt key is then released. If more than three digits are typed, a modulo-256 result is created. These three digits are interpreted as a character code and are transmitted through the keyboard routine to the system or application program. Alt is handled internal to the keyboard routine.
Caps Lock

This key shifts keys 16-25, 30-38, and 44-50 to upper case. A second depression of the Caps Lock key reverses the action. Caps Lock is handled internal to the keyboard routine.

Scroll Lock

This key is interpreted by appropriate application programs as indicating use of the cursor-control keys should cause windowing over the text rather than cursor movement. A second depression of the Scroll Lock key reverses the action. The keyboard routine simply records the current shift state of the Scroll Lock key. It is the responsibility of the system or application program to perform the function.

Shift Key Priorities and Combinations

If combinations of the Alt, Ctrl, and Shift keys are pressed and only one is valid, the precedence is as follows: the Alt key is first, the Ctrl key is second, and the Shift key is third. The only valid combination is Alt and Ctrl, which is used in the "system reset" function.

Special Handling

System Reset

The combination of the Alt, Ctrl, and Del keys will result in the keyboard routine initiating the equivalent of a "system reset" or "reboot." System reset is handled internal to the keyboard.
Break

The combination of the Ctrl and Break keys will result in the keyboard routine signaling interrupt hex 1A. Also, the extended characters (AL = hex 00, AH = hex 00) will be returned.

Pause

The combination of the Ctrl and Num Lock keys will cause the keyboard interrupt routine to loop, waiting for any key except the Num Lock key to be pressed. This provides a system- or application-transparent method of temporarily suspending list, print, and so on, and then resuming the operation. The “unpause” key is thrown away. Pause is handled internal to the keyboard routine.

Print Screen

The combination of the Shift and PrtSc (key 55) keys will result in an interrupt invoking the print screen routine. This routine works in the alphanumeric or graphics mode, with unrecognizable characters printing as blanks.

Other Characteristics

The keyboard routine does its own buffering. The keyboard buffer is large enough to support a fast typist. However, if a key is entered when the buffer is full, the key will be ignored and the “bell” will be sounded.

Also, the keyboard routine suppresses the typematic action of the following keys: Ctrl, Shift, Alt, Num Lock, Scroll Lock, Caps Lock, and Ins.
## Keyboard Usage

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Key(s)</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Home Cursor</td>
<td>Home</td>
<td>Editors; word processors</td>
</tr>
<tr>
<td>Return to outermost menu</td>
<td>Home</td>
<td>Menu driven applications</td>
</tr>
<tr>
<td>Move cursor up</td>
<td>↑</td>
<td>Full screen editor, word processor</td>
</tr>
<tr>
<td>Page up, scroll backward 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>Delete character at cursor</td>
<td>Del</td>
<td>Text, command entry</td>
</tr>
<tr>
<td>Destructive backspace</td>
<td>← Key 14</td>
<td>Text, command entry</td>
</tr>
<tr>
<td>Tab forward</td>
<td>→ I</td>
<td>Text entry</td>
</tr>
<tr>
<td>Tab reverse</td>
<td>← I</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 Prt Sc (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)**

2-20  Keyboard Encoding
<table>
<thead>
<tr>
<th>Function</th>
<th>Key(s)</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exit insert mode</td>
<td>Ins</td>
<td>Line editor</td>
</tr>
<tr>
<td>Cancel current line</td>
<td>Esc</td>
<td>Command entry, text entry</td>
</tr>
<tr>
<td>Suspend system (pause)</td>
<td>Ctrl</td>
<td>Stop list, stop program, and so on</td>
</tr>
<tr>
<td></td>
<td>Num Lock</td>
<td>Resumes on any key</td>
</tr>
<tr>
<td>Break interrupt</td>
<td>Ctrl Break</td>
<td>Interrupt current process</td>
</tr>
<tr>
<td>System reset</td>
<td>Alt Ctrl Del</td>
<td>Reboot</td>
</tr>
<tr>
<td>Top of document and home cursor</td>
<td>Ctrl PgUp</td>
<td>Editors, word processors</td>
</tr>
<tr>
<td>Standard function keys</td>
<td>F1-F10</td>
<td>Primary function keys</td>
</tr>
<tr>
<td>Secondary function keys</td>
<td>Shift F1-F10</td>
<td>Extra function keys if 10 are not sufficient</td>
</tr>
<tr>
<td></td>
<td>Ctrl F1-F10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Alt F1-F10</td>
<td></td>
</tr>
<tr>
<td>Extra function keys</td>
<td>Alt Keys 2-13 (1-9,0,-,=)</td>
<td>Used when stickers are put along top of keyboard</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>←</td>
</tr>
<tr>
<td>Line feed</td>
<td>Ctrl ←</td>
</tr>
<tr>
<td>Bell</td>
<td>Ctrl G</td>
</tr>
<tr>
<td>Home</td>
<td>Home</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>

**BASIC Screen Editor 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>← Key 14</td>
</tr>
<tr>
<td>Line feed</td>
<td>Ctrl ←</td>
</tr>
<tr>
<td>Cancel line</td>
<td>Esc</td>
</tr>
<tr>
<td>Copy character</td>
<td>F1 or ←</td>
</tr>
<tr>
<td>Copy until match</td>
<td>F2</td>
</tr>
<tr>
<td>Copy remaining</td>
<td>F3</td>
</tr>
<tr>
<td>Skip character</td>
<td>Del</td>
</tr>
<tr>
<td>Skip until match</td>
<td>F4</td>
</tr>
<tr>
<td>Enter skip 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>

**DOS Special Functions**
BIOS Cassette Logic

Software Algorithms – Interrupt Hex 15

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

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

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

Note: The carry flag will be set on any error.
Cassette Write

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

The write-block routine turns on the cassette drive motor and a synchronization bit (0) and then writes the leader (256 bytes of all 1's) to the tape. Next, the routine writes the number of data blocks specified by CX. After each data block of 256 bytes, a 2-byte cyclic redundancy check (CRC) is written. The data bytes are taken from the memory location pointed at by ES.

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

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

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

The read-block routine then searches for the leader and must detect all 1 bits for approximately 1/4 of the leader length before it can look for the sync (0) bit. After the sync bit is detected, the sync byte (ASCII character hex 16) is read. If the sync byte is read correctly, the data portion can be read. If a correct sync byte is not found, the routine goes back and searches for the leader again. The data is read a bit at a time and assembled into bytes. After each byte is assembled, it is written into memory at location ES:BX and BX is incremented by 1.

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

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

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

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

Error Recovery

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

### System ROM BIOS

<table>
<thead>
<tr>
<th>Equates</th>
<th>A-2</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>8088 Interrupt Locations</td>
<td>A-2</td>
<td>34</td>
</tr>
<tr>
<td>Stack</td>
<td>A-2</td>
<td>66</td>
</tr>
<tr>
<td>Data Areas</td>
<td>A-2</td>
<td>74</td>
</tr>
<tr>
<td>Power-On Self-Test</td>
<td>A-5</td>
<td>229</td>
</tr>
<tr>
<td>Boot Strap Loader</td>
<td>A-21</td>
<td>1493</td>
</tr>
</tbody>
</table>

### I/O Support

- Asynchronous Communications (RS-232C) A-22 | 1551
- Keyboard                                   A-26 | 1818
- Diskette                                   A-36 | 2426
- Printer                                    A-46 | 3201
- Display                                    A-47 | 3327

### System Configuration Analysis

- Memory Size Determination                  A-73 | 5177
- Equipment Determination                   A-73 | 5208
- Cassette I/O Support                      A-74 | 5253
- Graphics Character Generator              A-80 | 5769
- Time of Day                                A-82 | 5903
- Print Screen                               A-84 | 6077

### Fixed Disk ROM BIOS

<table>
<thead>
<tr>
<th>Fixed Disk I/O Interface</th>
<th>A-87</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boot Strap Loader</td>
<td>A-92</td>
<td>399</td>
</tr>
</tbody>
</table>

System BIOS A-1
LOC OBJ
LINE SOURCE

0000 (4) 78 RS232_BASE DW 4 DUP(?) ; ADDRESSES OF RS232 ADAPTERS

0008 (4) 79 PRINTER_BASE DW 4 DUP(?) ; ADDRESSES OF PRINTERS

0010 ?? 80 EQUIP_FLAG DW ? ; INSTALLED HARDWARE

0012 ?? 81 MF6_TST DB ? ; INITIALIZATION FLAG

0013 ?? 82 MEMORY_SIZE DW ? ; MEMORY SIZE IN K BYTES

0015 ?? 83 IO_RAM_SIZE DW ? ; MEMORY IN I/O CHANNEL

0017 ?? 84 ; KEYBOARD DATA AREAS

0019 ?? 85 KB_FLAG DB ?

001B ?? 86 ; SECOND BYTE OF KEYBOARD STATUS

0030 91 INS_STATE EQU $00H ; INSERT STATE IS ACTIVE

0030 92 CAPS_STATE EQU $40H ; CAPS LOCK STATE HAS BEEN Toggled

0030 93 NUM_STATE EQU $20H ; NUM LOCK STATE HAS BEEN Toggled

003B 94 SCROLL_STATE EQU $10H ; SCROLL LOCK STATE HAS BEEN Toggled

003B 95 ALT_SHIFT EQU $08H ; ALTERNATE SHIFT KEY DEPRESSED

003B 96 CTL_SHIFT EQU $04H ; CONTROL SHIFT KEY DEPRESSED

003B 97 LEFT_SHIFT EQU $02H ; LEFT SHIFT KEY DEPRESSED

003B 98 RIGHT_SHIFT EQU $01H ; RIGHT SHIFT KEY DEPRESSED

003B 99 ; SECOND BYTE OF KEYBOARD STATUS

003B 100 KB_FLAG_1 DB ?

003B 101 ; SECOND BYTE OF KEYBOARD STATUS

003B 102 INS_SHIFT EQU $00H ; INSERT KEY IS DEPRESSED

003B 103 CAPS_SHIFT EQU $40H ; CAPS LOCK KEY IS DEPRESSED

003B 104 NUM_SHIFT EQU $20H ; NUM LOCK KEY IS DEPRESSED

003B 105 SCROLL_SHIFT EQU $10H ; SCROLL LOCK KEY IS DEPRESSED

003B 106 HOLD_STATE EQU $00H ; SUSPEND KEY HAS BEEN Toggled

003B 107 ; SECOND BYTE OF KEYBOARD STATUS

003B 108 ; SECOND BYTE OF KEYBOARD STATUS

003B 109 ALT_INPUT DB ? ; STORAGE FOR ALTERNATE KEYPAD ENTRY

003B 10A BUFFER_HEAD DW ? ; POINTER TO HEAD OF KEYBOARD BUFFER

003B 10B BUFFER_TAIL DW ? ; POINTER TO TAIL OF KEYBOARD BUFFER

003B 10C KB_BUFFER DW 16 DUP(?) ; ROOM FOR 16 ENTRIES

CODE

112 KB_BUFFER_END LABEL WORD

113

114 ; ---- HEAD = TAIL INDICATES THAT THE BUFFER IS EMPTY

115

116 NUM_KEY EQU $69 ; SCAN CODE FOR NUMBER LOCK

117 SCROLL_KEY EQU $70 ; SCROLL LOCK KEY

118 ALT_KEY EQU $56 ; ALTERNATE SHIFT KEY SCAN CODE

119 CTL_KEY EQU $29 ; SCAN CODE FOR CONTROL KEY

120 CAPS_KEY EQU $56 ; SCAN CODE FOR SHIFT LOCK

121 LEFT_KEY EQU $42 ; SCAN CODE FOR LEFT SHIFT

122 RIGHT_KEY EQU $54 ; SCAN CODE FOR RIGHT SHIFT

123 INS_KEY EQU $82 ; SCAN CODE FOR INSERT KEY

124 DEL_KEY EQU $83 ; SCAN CODE FOR DELETE KEY

125 ; ---- HEAD = TAIL INDICATES THAT THE BUFFER IS EMPTY

126 ; SECOND BYTE OF KEYBOARD STATUS

127 ; DISKETTE DATA AREAS

128

129 SEEK_STATUS DB ? ; DRIVE RECALIBRATION STATUS

130 ; BIT 3-0 = DRIVE 3-0 NEEDS RECAL BEFORE

131 ; NEXT SEEK IF BIT IS = 0

132 INT_FLAG EQU $00H ; INTERRUPT OCCURRENCE FLAG

133 MOTOR_STATUS DB ? ; MOTOR STATUS

134 ; BIT 3-0 = DRIVE 3-0 IS CURRENTLY RUNNING

135 ; BIT 7 = CURRENT OP IS A WRITE, REQUIRES DELAY

136 MOTOR_COUNT DB ? ; TIME OUT COUNTER FOR DRIVE TURN OFF

137 MOTOR_WAIT EQU $37 ; THE SEC OF COUNT FOR MOTOR TURN OFF

138

139 DISKETTE_STATUS DB ? ; BYTE OF RETURN CODE INFO FOR STATUS

140 TIME_OUT EQU $00H ; ATTACHMENT FAILED TO RESPOND

141 BAD_SEEK EQU $40H ; SEEK OPERATION FAILED

142 BAD_CRC EQU $20H ; NEC CONTROLLER HAS FAILED

143 BAD_CRC_ON_DISKETTE_FAIL EQU $10H ; BAD CRC ON DISKETTE READ

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

145 BAD_DMA EQU $08H ; DMA OVERRUN ON OPERATION

146 RECORD_NOT_FOUND EQU $04H ; REQUESTED SECTOR NOT FOUND

147 WRITE_PROTECT EQU $03H ; WRITE ATTEMPTED ON WRITE PROTECT DISK

148 BAD_ADDR_MARK EQU $02H ; ADDRESS MARK NOT FOUND

Appendix A

System BIOS A-3
A-4 System BIOS
LOC OBJ LINE SOURCE

0000 217 REGEN LABEL BYTE
0000 218 REGENM LABEL WORD
0000 (16384 219 DB 16384 DUP(?)

220 VIDEO_RAM ENDS

E000 31353031343736 227 DB '1501476 COPR. IBM 1981' ; COPYRIGHT NOTICE
20434F50522EZO 32

Appendix A
System BIOS A-5
ASSUME CS:CODE, DS:NOTHING, ES:NOTHING, SS:NOTHING

SET SF, CF, ZF, AND AF FLAGS ON
DISABLE INTERRUPTS

READ/ WRITE THE 8088 GENERAL AND SEGMENTATION REGISTERS
WITH ALL ONE’S AND ZEROES’.

SET SF, CF, ZF, AND AF FLAGS ON
DISABLE INTERRUPTS

SET SF, CF, ZF, AND AF FLAGS ON
DISABLE INTERRUPTS

READ/ WRITE THE 8088 GENERAL AND SEGMENTATION REGISTERS
WITH ALL ONE’S AND ZEROES’.

READ/ WRITE THE 8088 GENERAL AND SEGMENTATION REGISTERS
WITH ALL ONE’S AND ZEROES’.

READ/ WRITE THE 8088 GENERAL AND SEGMENTATION REGISTERS
WITH ALL ONE’S AND ZEROES’.

READ/ WRITE THE 8088 GENERAL AND SEGMENTATION REGISTERS
WITH ALL ONE’S AND ZEROES’.

READ/ WRITE THE 8088 GENERAL AND SEGMENTATION REGISTERS
WITH ALL ONE’S AND ZEROES’.

READ/ WRITE THE 8088 GENERAL AND SEGMENTATION REGISTERS
WITH ALL ONE’S AND ZEROES’.

READ/ WRITE THE 8088 GENERAL AND SEGMENTATION REGISTERS
WITH ALL ONE’S AND ZEROES’.

READ/ WRITE THE 8088 GENERAL AND SEGMENTATION REGISTERS
WITH ALL ONE’S AND ZEROES’.

READ/ WRITE THE 8088 GENERAL AND SEGMENTATION REGISTERS
WITH ALL ONE’S AND ZEROES’.

READ/ WRITE THE 8088 GENERAL AND SEGMENTATION REGISTERS
WITH ALL ONE’S AND ZEROES’.

READ/ WRITE THE 8088 GENERAL AND SEGMENTATION REGISTERS
WITH ALL ONE’S AND ZEROES’.
363 I RO8 CHECKSUM TEST I
364 I DESCRIPTION I
365 I A CHECKSUM IS DONE FOR THE 8K RO8 MODULE
366 I CONTAINING POST AND BIOS.
367 I

EDAE
368 C10:
369

EDAE E6A0
370 OUT D9H,AL ; ZERO IN AL ALREADY
371 OUT B3H,AL ; DISABLE MMU INTERRUPTS
372 MOV DX,300H
373 OUT DX,AL ; INITIALIZE DMA PAGE REG
374 INC AL
375 MOV DL,080H
376 OUT DX,AL ; DISABLE B/W VIDEO, EN HIGH RES
377 MOV AL,99H ; SET 8255 A,C-INNM, B-OUTNMT
378 OUT CND,PORT,AL ; WRITE 8255 CND/MODE REG
379 MOV AL,0FCH ; DISABLE PARITY CHECKERS AND
380 OUT PORT_B,AL ; GATE SMS SMS,CASS MOTOR OFF
381 MOV AX,CS ; SETUP SS SEG REG
382 MOV SS,AX
383 MOV DS,AX
384 MOV DS,AX
385 INC SS:CODE ; SETUP RETURN ADDRESS
386 MOV BH,OE0H ; SETUP STARTING ROS ADDR (E000H)
387 MOV SP,OFFSET Cl ; SETUP RETURN ADDRESS
388 JMP ROS_CHECKSUM
389

EOE1 750A
390 C11:
391 JNE ERROR1 ; HALT SYSTEM IF ERROR
392

ED03 B004
393 MOV AL,04 ; DISABLE DMA CONTROLLER
394 E005 E608
395 OUT DMA00,AL
396

ED07 B054
397 MOV AL,54H ; SELECT TIMER 1,54B,MODE 2
398 E009 E643
399 OUT TIMER+3,AL ; SET INITIAL TIMER CNT TO 0
400 ED0B 8AC1
401 MOV AL,CL
402 ED0C E641
403 OUT TIMER+1,AL
404

ED0F B040
405 MOV AL,40H ; TIMER1_BITS_ON
406 EODF C12:
407 OUT TIMER+3,AL ; LATCH TIMER 1 COUNT
408 CMP BL,0FFH
409 JE C13 ; YES SEE IF ALL BITS GO OFF
410 IN AL,TIMER+1 ; READ TIMER 1 COUNT
411 OR BL,AL
412 MOV AL,04
413 OUT TIMER+1,AL
414 LOOP C12 ; TIMER1_BITS_ON
415 HALT ; TIMER 1 FAILURE, HALT SYS
416

EE0F 8AC3
417 MOV AL,BL ; SET TIMER 1 CNT
418 E0FI 2BC9
419 MOV CX,CX
420 ED03 E641
421 OUT TIMER+1,AL
422 ED05 B040
423 MOV AL,40H ; LATCH TIMER 1 COUNT
424 OUT TIMER+3,AL
425 ED07 E643
426 NOP ; DELAY FOR TIMER
427 ED09 90
428 MOV AL,03H
429 OUT TIMER+3,AL ; READ TIMER 1 COUNT
430 ED0B 8AC1
431 MOV AL,04
432 OUT TIMER+1,AL
433 ED0F B040
434 MOV AL,40H
435 ED07 E643
436 MOV AL,03H
437 ED09 90
438

Appendix A
System BIOS A-7
A-8 System BIOS
LOC OBJ | LINE | SOURCE
--- | --- | ---
E16F D3E0 | 518 | SHL AX, CL
E171 88C8 | 516 | MOV CX, AX
E173 FC | 517 | CIP ; SET DIR FLAG TO INC
E174 | 518 | C19:
E174 AA | 519 | STOSB ; FILL BASE RAM WITH DATA
E175 E2FD | 520 | LOOP C19 ; LOOP TIL ALL ZERO
E177 89E7204 | 521 | MOV DATA_WORD offset RESET_FLAGS, BP
E179 | 522 |
E17B D0F8 | 523 | ; ---- DETERMINE IO CHANNEL RAM SIZE
E17D E661 | 524 | MOV AL,0F8H ; ENABLE SWITCH 5
E17F E662 | 525 | OUT PORT B, AL
E181 2401 | 526 | IN AL, PORT C ; READ SWITCHES
E183 810C | 527 | MOV CL, 12D
E185 D5C0 | 528 | ROL AX, CL
E187 B0FC | 529 | MOV AL, OFCH ; DISABLE SW. 5
E189 E661 | 530 | OUT PORT B, AL
E18B E662 | 531 | IN AL, PORT C
E18D 240F | 532 | AND AL, 0FH
E18F 0AC4 | 533 | OR AL, AH ; COMBINE SWITCH VALUES
E191 0A00 | 534 | MOV BL, AL ; SAVE
E193 0B20 | 535 | MOV AH, .32
E195 F664 | 536 | MUL AH ; CALC. LENGTH
E197 A31504 | 537 | MOV DATA_WORD offset IO_RAM_SIZE1, AX ; SAVE IT
E19A 7418 | 538 | JE C21
E19C 8A010 | 539 | MOV DX, 1000H ; SEGMENT FOR I/O RAM
E19F 8A00 | 540 | MOV AH, AL
E1A1 B000 | 541 | MOV AL, 0
E1A3 8EC2 | 542 | MOV ES, 0X
E1A5 890080 | 543 | MOV CX, 0000H ; FILL 32K BYTES
E1A6 2BFF | 544 | SUB DI, DI
E1AA F3 | 545 | REP STOSB
E1AB AA | 546 | ADD DX, 8000H ; NEXT SEGMENT VALUE
E1AC 812000B | 547 | DEC BL
E1B2 75EF | 548 | JNZ C20 ; FILL IO
E1B4 | 549 |
E1B4 8013 | 550 | MOV AL, 13H ; IC51 - EDGE, SNGL, IC54
E1B6 E620 | 551 | MOV INTA00, AL
E1B8 D000 | 552 | MOV AL, 0 ; SETUP ICW2 - INT TYPE B (B-P)
E1BA 6E21 | 553 | MOV INTA01, AL
E1BC B009 | 554 | MOV AL, 9 ; SETUP ICW4 - BUFFRD.0086 MODE
E1BD 6E21 | 555 | MOV INTA01, AL
E1C0 2D0C | 556 | SUB AX, AX ; POINT ES TO BEGIN
E1C2 8EC0 | 557 | MOV ES, AX
E1C4 833000 | 558 | MOV AX, stack
E1C7 8E0D | 559 | MOV SS, AX ; GET STACK VALUE
E1C9 8C0001 | 560 | MOV SP, offset TOS ; SET THE STACK UP
E1CC 01FD3A12 | 561 | CMP BP, 1234H ; STACK IS READY TO GO
E1DE 7425 | 562 | JE C25 ; INTERNAL FLAGS TEST
E1D2 2BFF | 563 | SUB DI, DI
E1D4 8E0F | 564 | MOV DS, DI
E1D6 8B2409 | 565 | MOV BX, 24H
E1D8 C70747FF | 566 | MOV word PTR [(BX), offset D11] ; SET UP KB INTERRUPT
E1DD 43 | 567 |
E1DF 8C0F | 568 | MOV [BX], CS
E1E1 88F004 | 569 | CALL KB_RESET ; READ PB KB RESET CODE TO BL
E1E4 00F065 | 570 | CMP BL, 065H ; IS THIS MANUFACTURING TEST ??
E1E7 750E | 571 | JNZ C25 ; JUMP IF NOT MAN. TEST
E1EB B2FF | 572 | MOV DL, 255 ; READ IN TEST PROGRAM
E1EB E86204 | 573 | CALL SP_TEST
E1EE 8AC3 | 574 | MOV AL, BL
E1EF AA | 575 | STOSB ; SETUP STACK SEG AND SP
E1F4 883000 | 576 |
E1F7 8E0D | 577 |
E1F9 8C0F | 578 |
E1FB E86204 | 579 |
E1FF AA | 580 |
SET UP THE BIOS INTERRUPT VECTORS TO TEST INTERRUPTS

-----

GO TO A-10 System

LOC OBJ LINE SOURCE

E1F1 FECA 590 DEC DL ; JUMP IF NOT DONE YET
E1F3 75F6 591 JNZ C22
E1F5 C03E 592 INT 21H ; SET INTERRUPT TYPE 62 ADDRESS F8H
E1F7 C25:
594
595 ------ SET UP THE BIOS INTERRUPT VECTORS TO TEMP INTERRUPT
596
E1F7 B92000 597 MOV CX,32 ; FILL ALL 32 INTERRUPTS
E1FA 2FF 598 SUB DI,DI ; FIRST INTERRUPT LOCATION
E1FC D3:
599
E1FC B047FF 600 MOV AX,OFFSET D1L ; MOVE ADDR OF INV PROC TO TBL
E1FF AB 601 STOSW
E200 8CC6 602 MOV AX,CS ; SET ADDR OF INTR PROC SEG
E202 AB 603 STOSW
E203 E2FF 604 LOOP D3 ; VECTORS
605
606 ------ SET UP OTHER INTERRUPTS AS NECESSARY
607
E205 C7660800C308 608 MOV NMI_PTR,OFFSET NMI_INT ; NMI INTERRUPT
E208 C766140054FF 609 MOV INTS_PTR,OFFSET PRINT SCREEN ; PRINT SCREEN
E211 C76620000E06 610 MOV BASIC_PTR+2,OF606H ; SEGMENT FOR CASSETTE BASIC
611
612 "-----------------------------------------------------------------"
613 0259 INTERRUPT CONTROLLER TEST :
614 DESCRIPTION :
615 READ-WRITE THE INTERRUPT MASK REGISTER (IMR) WITH ALL
616 ONES AND ZEROS. ENABLE SYSTEM INTERRUPTS. MASK DEVICE :
617 INTERRUPTS OFF. CHECK FOR HOT INTERRUPTS (UNEXPECTED). :
618 "-----------------------------------------------------------------"
619
620 ------ TEST THE IMR REGISTER
621
E217 B8100 622 MOV DX,0001H ; POINT INTR. CHIP ADDR 21
E21A B000 623 MOV AL,0 ; SET IMR TO ZERO
E21C EE 624 OUT DX,AL
E21D EC 625 IN AL,DX ; READ IMR
E21E 0AC0 626 OR AL,AL ; IMP = 0?
E220 7515 627 JNZ D6 ; GO TO ERR ROUTINE IF NOT 0
E222 B0FF 628 MOV AL,OFFH ; DISABLE DEVICE INTERRUPTS
E224 EE 629 OUT DX,AL ; WRITE TO IMR
E225 EC 630 IN AL,DX ; READ IMR
E226 0401 631 ADD AL,1 ; ALL IMR BIT ON!
E228 750D 632 JNZ D6 ; NO - GO TO ERR ROUTINE
633
634 ------ CHECK FOR HOT INTERRUPTS
635
E22A 32E4 636 XOR AH,AH ; CLEAR AH REG
E22C FB 637 STI ; ENABLE INTERNAL INTERRUPTS
E22D B8C9 638 MOV AX,OFFSET PRINT_SCREEN ; PRINT SCREEN ROUTINE
E22F D4:
641
E22F E2FE 642 LOOP D4 ; MIGHT OCCUR
E231 D5:
643
E231 E2FE 644 LOOP D5
E233 0AE4 645 OR AH,AH ; DID ANY INTERRUPTS OCCUR?
E235 7408 646 JZ D7 ; NO - GO TO NEXT TEST
E237 D6:
647
E237 B0101 648 MOV DX,18H ; BEEP SPEAKER IF ERROR
E23A E9203 649 CALL ERR_BEEP ; GO TO BEEP SUBROUTINE
E23D FA 650 CLI
E23E F4 651 HLT ; HALT THE SYSTEM
652 "-----------------------------------------------------------------"
653 0253 TIMER CHECKOUT :
654 DESCRIPTION :
655 VERIFY THAT THE SYSTEM TIMER (0) :
656 DOESN'T COUNT TOO FAST OR TOO SLOW. :
657 "-----------------------------------------------------------------"
E23F D7:
658
E23F B0FF 659 MOV AL,OFFH ; MASK ALL INTRS EXCEPT LVL 0
E241 EE 660 OUT DX,AL ; WRITE THE 0259 IMR
E242 B010 661 MOV AL,00010000B ; SEL TIM 0, LSB, MODE 0, BINARY
E244 6465 662 OUT TIM_CTRL_AL ; WRITE TIMER CONTROL MODE REG
E246 B91600 663 MOV CX,1H ; SET FOR LOOP COUNT
E249 BAC1 664 MOV AL,EL ; SET TIMER 0 CNT REG
E24A E640 665 OUT TIMERO_AL ; WRITE TIMER 0 CNT REG

A-10 System BIOS
LOC OBJ  
LINE SOURCE

E240 666  D0: TEST AH,OFFH ; DID TIMER 0 INTERRUPT OCCUR?
E240 F4C0FF 667  JNZ D9 ; YES - CHECK TIMER 0 FOR SLOW TIME
E250 7504 668  LOOP D8 ; WAIT FOR INTR FOR SPECIFIED TIME
E252 E2F9 669  JMP D6 ; TIMER 0 INTR DIDN'T OCCUR - ERR
E256 670  D10: TEST AH,OFFH ; DID TIMER 0 INTERRUPT OCCUR?
E256 B112 671  MOV CL,18 ; SET PSM LOOP CNT
E258 B0FF 672  MOV AL,OFFH ; WRITE TIMER 0 CNT REG
E25A E640 673  OUT TIMER0,AL
E25C BAFEO0 674  MOV AX,OFFH
E25F EE 675  OUT DX,AL
E260 676  D10: TEST AH,OFFH ; DID TIMER 0 INTERRUPT OCCUR?
E263 7502 677  JNZ D6 ; YES - TIMER TIMING TOO FAST, ERR
E265 E2F9 678  LOOP D10 ; WAIT FOR INTR FOR SPECIFIED TIME

--- SETUP BIOS SUBROUTINE CALL INTERRUPT VECTORS

E267 686  PUSH DS ; SAVE POINTER TO DATA AREA
E268 B4000 687  MOV DI,OFFSET VIDEO_INT ; SETUP ADDR TO INTR AREA
E26B 688  PUSH CS
E26C 689  POP DS ; SETUP ADDR OF VECTOR TABLE
E26D B03FP0 690  MOV SI,OFFSET SECTOR_TABLE+16 ; START WITH VIDEO ENTRY
E271 891000 691  MOV CX,16
E271 692

--- SETUP TIMER 0 TO MODE 3

E274 B0FF 693  MOV AL,OFFH ; DISABLE ALL DEVICE INTERRUPTS
E276 EE 694  OUT DX,AL
E277 B036 695  MOV AL,36H ; SEL TIM 0.LSB,MSB,MODE 3
E279 6643 696  OUT TIMER+3,AL ; WRITE TIMER MODE REG
E27B B000 697  MOV AL,0
E27D E640 698  OUT TIMER,AL ; WRITE LSB TO TIMER 0 REG
E27F 699  EIA:
E27F A5 700  MOVSM ; MOVE VECTOR TABLE TO RAM
E280 47 701  INC DI ; MOVE PAST SEGMENT POINTER
E281 47 702  INC DI
E282 E2F8 703  LOOP EIA
E284 E640 704  OUT TIMER,AL ; WRITE MSB TO TIMER 0 REG
E286 IF 705  POP DS ; RECOVER DATA SEG POINTER
E286 706

--- SETUP TIMER 0 TO BLINK LED IF MANUFACTURING TEST MODE

E287 E60903 707  CALL KBD_RESET ; SEND SOFTWARE RESET TO KEYBOARD
E28A 00FBA 709  CMP BL,0A0H ; SCAN CODE 'AA' RETURNED?
E28D 741E 710  JE E6 ; YES - CONTINUE (NON MFG MODE)
E28F B03C 711  MOV AL,3CH ; EN KBD, SET KBD CLK LINE LOW
E291 E661 712  OUT PORT_B,AL
E293 90 713  NOP
E294 90 714  NOP
E295 E640 715  IN AL,PORT_A ; WAS A BIT CLOCKED IN?
E297 04FF 716  AND AL,OFFH
E299 750F 717  JNZ E2 ; YES - CONTINUE (NON MFG MODE)
E29B FE061204 718  INC DATA_AREA[OFFSET MFG_TST] ; ELSE SET SW FOR MFG TEST MODE
E29D C7062000A06E 719  MOV INT_ADDR.OFFSET BLINK_INT ; SETUP TIMER INTR TO BLINK LED
E2A5 B0FE 720  MOV AL,OFFH ; ENABLE TIMER INTERRUPT
E2A7 E621 721  OUT INTA01,AL
E2A9 E2: 722  JUMP NOT_IN:
E2A9 B0CC 723  MOV AL,6CH ; RESET THE KEYBOARD
E2AB E661 724  OUT PORT_B,AL
E2AB 725

--- INITIALIZE AND START CRT CONTROLLER (4045):

E2AD 726  CALL KBD_RESET ; SEND SOFTWARE RESET TO KEYBOARD
E2A9 E640 727  IN AL,PORT_A ; READ SENSE SWITCHES
E2AF B000 728  MOV AH,0
E2B1 A31004 729  MOV DATA_WOR[OFFSET EQUIP_FLAGS],AX ; STORE SENSE SW INFO
E2B4 730  E6A: ; DESCRIPTION:
E2B4 740  OUT BE03H,AL ; SELECT ALPHANUMERIC MODE, 40 * 25, B & M:
E2B4 741  MOV DATA_WOR[OFFSET SENSE_SW],AX ; STORE SENSE SW INFO
E2B4 742  JNZ E7 ; VIDEO SW SET TO 0?
SET MODE FOR GO TEST VIDEO STS
SET MODE FOR 40X25
A-12 System BIOS
AND HORIZONTAL SYNC LINES.

E32E 56 620 POP AX  ; GET VIDEO SENSE SW INFO
E32E 56 621 PUSH AX  ; SAVE IT
E32E 0FC30 622 CMP AX,550H  ; B/W CARD ATTACHED?
E32E BA8A03 623 MOV DX,03BAH  ; SETUP ADDR OF FN STATUS PORT
E336 7402 624 JE E11  ; YES - GO TEST LINES
E336 02DA 625 MOV DX,02DAH  ; COLOR CARD IS ATTACHED
E336 E11 626  ; LINE_TST:
E336 B408 627 MOV AH,8  ; OFLOOP_CNT:
E33C E12 628 SUB CX,CX  ; READ CRT STATUS PORT
E33C 2BC9 629 E13: AND AL,AN  ; CHECK VIDEO/HORIZ LINE
E33C EC 630 IN AL,DX  ; ERROR='1801'
E33F 22C4 631 AND AL,AN  ; TEST ADDRESS PRESENT - IF
E341 7504 632 JNZ E14  ; NO - IF DATA PATTERN
E343 E2F9 633 LOOP E13  ; ITS ON - CHECK IF IT GOES OFF
E345 E009 634 JMP SHORT E17  ; LOOP TILL ON OR TIMEOUT
E347 E14 635  ; GO PRINT ERROR MSG
E347 2BC9 636 E14: SUB CX,CX  ; CRT_ERR
E349 E15 637 MOV AX,0201H  ; GET NEXT BIT TO CHECK
E349 EC 638 IN AL,DX  ; GET NEXT BIT TO CHECK
E34A 740A 639 AND AL,AN  ; CHECK VIDEO/HORIZ LINE
E34E E2F9 640 LOOP E15  ; ITS ON - CHECK NEXT LINE
E350 E17 641 MOV DX,102H  ; LOOP IF OFF TILL IT GOES ON
E350 BA201 642 CALL ERR_BEEP  ; CRT_ERR
E353 E7902 643 JMP SHORT E16  ; NEXT_LINE
E356 E006 644 MOV DX,102H  ; GO BEEP SPEAKER
E358 E16 645 CALL ERR_BEEP  ; NX_LINE
E358 B103 646 MOV CL,5  ; GET NEXT BIT TO CHECK
E35D D2EC 647 MOV DX,0210H  ; CONTROL
E35F 750E 648 JMP E16  ; SETUP ADDR OF OTHER MODULES
E35F E10 649 SHR AH,CL  ; DISPLAY_CURSOR:
E35F 50 650 JNZ E12  ; GO CHECK HORIZONTAL LINE
E35F 50 651 POP AX  ; GET VIDEO SENSE SW (AH)
E35F B400 652 MOV AH,0  ; SET MODE AND DISPLAY CURSOR
E361 CD10 653 INT 10H  ; CALL VIDEO I/O PROCEDURE
E361 119 654 INT 10H  ; CALL VIDEO I/O PROCEDURE
E363 55 655 E10: MOV DX,0C000H  ; TOP OF VIDEO ROM AREA YET?
E363 B400C0 656 MOV DX,0C000H  ; TOP OF VIDEO ROM AREA YET?
E364 018A: 657 MOV AX,EBX  ; GET FIRST 2 LOCATIONS
E366 E0A 658 MOV DS,DX  ; GET FIRST 2 LOCATIONS
E366 2800 659 MOV BX,BX  ; GET FIRST 2 LOCATIONS
E366 6807 660 MOV BX,BX  ; GET FIRST 2 LOCATIONS
E36C 53 661 MOV AX,EBX  ; GET FIRST 2 LOCATIONS
E36D 50 662 PUSH BX  ; GET FIRST 2 LOCATIONS
E36D 50 663 POP BX  ; GET FIRST 2 LOCATIONS
E36E 3055AA 664 CMP AX,0A55H  ; PRESENT?
E371 7505 665 JNZ E10  ; IS IT ON - CHECK NEXT LINE
E373 E00E3 666 CALL ROM_CHECK  ; GET SCAN MODULE
E376 E004 667 JMP SHORT E10C  ; GET SCAN MODULE
E378 E10B 668 CALL ROM_CHECK  ; GET SCAN MODULE
E378 61C8000 669 ADD DX,0080H  ; EXPRESS ZER IN BOX
E37C E10C: 670 CMP DX,0C800H  ; EXPRESS ZER IN BOX
E37C 01FA00C8 671 TOP OF VIDEO ROM AREA YET?
E380 7CE4 672 JL E10A  ; GET DATA PATTERN
E383 7C43 673 | EXPANSION I/O BOX TEST:
E383 7C43 674 | CHECK TO SEE IF EXPANSION BOX PRESENT - IF INSTALLED, :
E383 7C43 675 | TEST DATA AND ADDRESS BUSES TO I/O BOX. :
E383 7C43 676 | ERROR='1001' :
E383 7C43 677 | EXPANSION I/O BOX TEST:
E383 7C43 678 | CHECK TO SEE IF EXPANSION BOX PRESENT - IF INSTALLED, :
E383 7C43 679 | TEST DATA AND ADDRESS BUSES TO I/O BOX. :
E383 7C43 680 | ERROR='1001' :
E383 7C43 681 | EXPANSION I/O BOX TEST:
E383 7C43 682 | CHECK TO SEE IF EXPANSION BOX PRESENT - IF INSTALLED, :
E383 7C43 683 | TEST DATA AND ADDRESS BUSES TO I/O BOX. :
E383 7C43 684 | ERROR='1001' :
E383 7C43 685 | EXPANSION I/O BOX TEST:
E383 7C43 686 | CHECK TO SEE IF EXPANSION BOX PRESENT - IF INSTALLED, :
E383 7C43 687 | TEST DATA AND ADDRESS BUSES TO I/O BOX. :
E383 7C43 688 | ERROR='1001' :
E383 7C43 689 | EXPANSION I/O BOX TEST:
E383 7C43 690 | CHECK TO SEE IF EXPANSION BOX PRESENT - IF INSTALLED, :
E383 7C43 691 | TEST DATA AND ADDRESS BUSES TO I/O BOX. :
E383 7C43 692 | ERROR='1001' :
E383 7C43 693 | EXPANSION I/O BOX TEST:
E383 7C43 694 | CHECK TO SEE IF EXPANSION BOX PRESENT - IF INSTALLED, :
E383 7C43 695 | TEST DATA AND ADDRESS BUSES TO I/O BOX. :
E383 7C43 696 | ERROR='1001' :
E396 752A
695 JNE E19
696 ; NO ANSWER=NEXT TEST
697

E39A 800B
900 MOV BX,AX
901 ; LOAD DATA REG ADDRESS
902 MOV DS:BX,AL
903 ; WRITE ADDRESS F000+BX
904 OUT DX,AL
905 ; WRITE DATA
906

E3A2 EE
907 MOV CS:BX,AL
908 ; WRITE ADDRESS F000+BX
909 OUT DX,AL
910 ; WRITE DATA
911

E3A3 90
912 MOV AL,BH
913 ; READ DATA
914

E3A4 EC
915 MOV AL,BX
916 ; READ DATA
917

E3A5 3AC7
918 CMP AL,86H
919 ; COMPARE TO HI ADDRESS
920

E3A7 7514
921 JNE EXP_ERR
922

E3A9 42
923 INC DX
924 ; DX=E25H (ADDR. HI REG)
925

E3AA EC
926 IN AL,DX
927 ; ADDR. LOW OK?
928

E3AC 3AC4
929 CMP AL,86H
930 ; COMPARE TO HI ADDRESS
931

E3AD 750E
932 JNE EXP_ERR
933

E3AF 42
934 INC DX
935 ; DX=E25H (ADDR. LOW REG)
936

E3B0 EC
937 IN AL,DX
938

E3B1 3AC4
939 CMP AL,86H
940 ; COMPARE TO HI ADDRESS
941

E3B2 750F
942 MOV AL,86H
943 ; INVERT AX
944

E3B5 F000
945 MOV AL,0AAH
946 ; BACK TO STARTING VALUE (AAAA) YET
947

E3B7 3CAA
948 CMP AL,0AAH
949 ; BACK TO STARTING VALUE (AAAA) YET
950

E3B9 7409
951 JE E19
952 ; GO ON TO NEXT TEST IF 50
953

E3BD E0DD
954 JMP EXP
955 ; LOOP BACK THROUGH WITH DATA OF 5555
956

E3B0 80E9
957 MOV SI,OFFSET F3D
958

E3C1 80F602
959 CALL P_MSG
960

E3C4 E0771B
961 CALL DDS
962

E3C7 A0100
963 MOV AL,BYTE PTR EQUIP_FLAG
964 ; GET SENSE SWS INFO
965

E3CA 240C
966 AND AL,0CH
967 ; ISOLATE RAM SIZE SWS
968

E3CC 8044
969 MOV AH,4
970

E3CE 6F64
971 MOV AL,AX
972

E3D0 0410
973 ADD AL,16
974 ; ADD BASIC 16K
975

E3D2 8000
976 MOV DX,AX
977 ; SAVE PLANAR RAM SIZE IN DX
978

E3D4 8008
979 MOV BX,AX
980 ; AND IN BX
981

E3D6 A11500
982 MOV AX,IO_RAM_SIZE
983 ; GET IO CHANNEL RAM SIZE
984

E3D9 03F040
985 CMP BX,40H
986 ; PLANAR RAM SIZE = 64K?
987

E3DC 7402
988 JE E20
989 ; YES - ADD IO CHN RAM SIZE
990

E3DE 2BC0
991 SUB AX,AX
992 ; NO - DON'T ADD ANY IO RAM
993

E3E0 E20:
994 ADD AX,0
995 ; ADD_IO_SIZE:
996

E3E2 03C3
997 ADD AX,BX
998 ; SUM TOTAL RAM SIZE
999

E3E4 A31300
1000 MOV MEMORY_SIZE,AX
1001 ; SETUP MEMORY SIZE PARM
1002

E3E5 610F3512
1003 CMP BP,1234H
1004 ; PARM INITIATED BY KBD RESET?
1005

E3E9 1E
1006 PUSH DS
1007 ; SAVE DATA SEGMENT
1008

E3EA 740F
1009 JE TST12
1010 ; YES - SKIP MEMORY TEST
1011

E3EC 500004
1012 MOV BX,400H
1013

E3EF 891000
1014 MOV CX,16
1015

E3F2
1016 MOV DX,CX
1017 ; ANY MORE STG TO BE TESTED?
1018

E3F2 3001
1019 CMP DX,CX
1020 ; ANY MORE STG TO BE TESTED?
1021

E3F4 762D
1022 JBE E23
1023 ; NO - GO TO NEXT TEST
1024

E3F6 AED0
1025 MOV DS,BX
1026 ; SETUP STG ADDR IN DS AND ES
1027

E3F8 0EC3
1028 MOV ES,BX
1029 ; INCREMENT STG BYTE COUNTER
1030

E3FA 03C110
1031 ADD CX,16
1032 ; INCREMENT STG BYTE COUNTER
1033

E3FD 81C30004
1034 ADD BX,400H
1035 ; SET POINTER TO NEXT 16K BLK
1036

E401 51
1037 PUSH CX
1038

E402 53
1039 PUSH BX
1040

E403 52
1041 PUSH DX
1042

E404 E011FC
1043 CALL STGSTST
1044 ; GO TEST A 16K BLK OF STG
1045

E407 5A
1046 POP BX
1047

A-14 System BIOS
I- ----- PRINT Failing address and xor'ed pattern if data compare error

------ KEYBOARD TEST

SETUP

CHECK FOO STUCK KEYS

--------- PRINT ERROR MSG

1000 i KEYBOARD TEST :

1001 i DESCRIPTION :

1002 i RESET THE KEYBOARD AND CHECK THAT SCAN CODE :

1003 i 'AA' IS RETURNED TO THE CPU. CHECK FOR STUCK :

1004 i KEYS. :

1005 i

1006 ASSUME DS:DATA

1007 TST12:

1008 i

1009 CMP MSG.TST.1 : MANUFACTURING TEST MODE?

1010 JE F7 : YES - SKIP KEYBOARD TEST

1011 CALL KBD.RESET : ISSUE SOFTWARE RESET TO KEYBOARD

1012 JCXZ F6 : PRINT ERR MSG IF NO INTERRUPT

1013 MOV AL,40H : ENABLE KEYBOARD

1014 OUT PORT.B,AL

1015 CMP BL,0AH : SCAN CODE AS EXPECTED?

1016 JNE F6 : NO - DISPLAY ERROR MSG

1017 i

1018 i CHECK FOR STUCK KEYS

1019 i

1020 MOV AL,0CH : CLR KBD. SET CLK LINE HIGH

1021 OUT PORT.B,AL

1022 MOV AL,4CH : ENABLE KBD.CLK IN NEXT BYTE

1023 OUT PORT.B,AL

1024 SUB CX,CX

1025 F5: KBD.WAIT

1026 LOOP F5 : DELAY FOR A WHILE

1027 IN AL,KBD_IN : CHECK FOR STUCK KEYS

1028 CMP AL,0 : SCAN CODE = 0?

1029 JE F7 : YES - CONTINUE TESTING

1030 CALL XPC.BYTE : CONVERT AND PRINT

1031 F6: MOV SI,OFFSET F1 : GET MSG ADDR

1032 CALL P.MSG : PRINT MSG ON SCREEN

1033 i

1034 i SETUP INTERRUPT VECTOR TABLE

1035 i

1036 F7: SUB AX,AX : SETUP_INT_TABLE:

1037 MOV AX,ES

1038 MOV CX,0 : GET VECTOR CNT

1039 MOV AL,EX : SAVE DATA SEGMENT

1040 PUSH DS : SETUP DS SEG REG

1041 PUSH CS

1042 MOV DS,0 : SAVE DATA SEGMENT

1043 MOV DS,0 : SAVE DATA SEGMENT

1044 MOV DS,0 : SAVE DATA SEGMENT

1045 F7A: MOV AX,80H : SKIP OVER SEGMENT

1046 MOV AX,80H : SKIP OVER SEGMENT

1047 MOV AX,80H : SKIP OVER SEGMENT

1048 MOV AX,80H : SKIP OVER SEGMENT

System BIOS A-15
A-16 System BIOS
LOOP POWER-ON-DIAGS

GO TO

LOC OBJ

E4E3 7403
E4E5 6E1D3
E4E8
E4E8 80C602
E4EB 00FEE
E4ED 7E5C
E4F0 IF

E4F1

E4F1 A0100
E4F4 A001
E4F6 750A
E4FB 803120001
E4F0 7520
E4F7 69F8
E502
E504 24BF
E506 E2C2
E508 B400
E50A 0A04
E50C CD13
E50E 7221

E510 BAF03
E513 92
E514 BD1C
E517 28C9
E519
E519 E2FE
E51D
E51B E2FE
E51D 3302
E51F B501
E521 6163E00
E525 E6509
E528 7207
E52A 5B22
E52C E6409
E52F 7307
E531 F13
E531 BEEF90
E535 E60201
E538 B00C
E53A 5A
E53B EE
E53C
E53C B8E00
E53F 89361A00
E543 89361C00
E545 8936B00
E54B 83C620
E54F 8936A200
E552 E421
E554 24FC
E556 E621
E559 B03D6E00
E55C 2B46
E55E
E55E ZE0856400

E561 1201

Appendix A

System BIOS A-17
LOC OBJ

ES62 B0AA 1202 MOV AL,0Ah
ES64 EE 1203 OUT DX,AL
ES65 52 1204 PUSH DX
ES66 EC 1205 IN AL,DX
ES67 5A 1206 POP DX
ES68 3CA 1207 CMP AL,0Ah
ES6A 7505 1208 JNE FF7
ES6C 95408 1209 MOV PRINTER_BASE+1,DX
ES6F 46 1210 INC $1
ES70 46 1211 INC SI
ES71 1212 FI7: ; NO_STORE:
ES71 45 1213 INC BP
ES72 45 1214 INC BP
ES73 8IF04326 1215 CMP BP,OFFSET FAE ; ALL POSSIBLE ADDRS CHECKED?
ES77 7505 1216 JNE FI6
ES79 2008 1217 SUB BX,BX
ES7B BAFO43 1218 MOV DX,3FH
ES7E EC 1219 IN AL,DX
ES7F 40FH 1220 TEST AL,0FH
ES81 7506 1221 JNZ FI6
ES83 C7078003 1222 MOV RS232_BASE+1,3FH ; SETUP RS232 CD #1 ADDR
ES87 43 1223 INC BX
ES88 43 1224 INC BX
ES89 1225 F18: ; BASE_END:
ES89 6062 1226 MOV DH,02H ; CHECK IF RS232 CD 2 ATTCH (AT 2FA)
ES8B EC 1227 IN AL,DX ; READ INTERRUPT ID REG
ES8C A0FH 1228 TEST AL,0FH
ES8E 7506 1229 JNZ FI9
ES90 C7078002 1230 MOV RS232_BASE+1,2FH ; SETUP RS232 CD #2
ES94 43 1231 INC BX
ES95 43 1232 INC BX
ES95 1233
ES96 1234 ;----- SETUP EQUIP FLAG TO INDICATE NUMBER OF PRINTERS AND RS232 CARDS
ES96 1235
ES96 F19: 1236 MOV AX,SI
ES96 80C6 1237 MOV CL,3
ES98 B013 1238 MOV AL,CL
ES9A 02C8 1239 XOR AL,CL ; ROTATE RIGHT 3 POSITIONS
ES9C 1240 OR AL,AL
ES9E A2100 1241 MOV BYTE PTR EQUIP_FLAGS+1,AL ; STORE AS SECOND BYTE
ESA1 B001 1242 MOV DL,01H ; DX=201
ESA3 EC 1243 IN AL,DX
ESA4 AD0F 1244 TEST AL,0FH
ESA6 7506 1245 JNZ F20
ESAB 000E110010 1246 OR BYTE PTR EQUIP_FLAGS+1,16
ESAD 1247 F20:
ESAD 1248
ESAD 1E 1249 ;----- SET DEFAULT TIMEOUT VALUES FOR PRINTER AND RS232
ESAE 07 1250
ESAF BF7000 1251 PUSH DS
ESB2 B01416 1252 POP DS
ESB3 AB 1253 MOV AX,1416H ; PRINTER DEFAULTS (COUNT=20)
ESB5 AB 1254 STOSW
ESB6 AB 1255 STOSW
ESB7 B0101 1256 MOV AX,0101H ; RS232 DEFAULTS=01
ESBD AB 1257 STOSW
ESBE AB 1258 STOSW
ESB9 AB 1259 STOSW
ESBC B000 1260 ;----- ENABLE NMI INTERRUPTS
ESDE EE0 1261 MOV AL,60H ; ENABLE NMI INTERRUPTS
ESC0 8032100001 1262 OUT DATH,AL
ESC1 7506 1263 CMP MFG_TST,1 ; MFG MODE?
ESC7 BA100 1264 JE F31
ESC9 E600200 1265 MOV DX,1
ESCA E600200 1266 CALL ERR_BEPI ; BEEP 1 SHORT TONE
ESC0 1267 F21: ; LOAD_BOOT_STRAP:
ESCD 1268 INT 1H
ESCE CO19 1269 ;----- INITIAL RELIABILITY TEST -- SUBROUTINES :
ESCH 1270
ESCH 1271
ESCH 1272
ESCH 1273
ESCH 1274
ESCH 1275
ESCH 1276
ESCH 1277
ESCH 1278 ; SUBROUTINES FOR POWER ON DIAGNOSTICS

A-18  System BIOS
FUNCTIONS

E603 B006
E604 0A65
E605 0A56
E606 0A42
E607 0A3D05
E608 0A2C4
E609 0A1E461
E60A 0A18E0
E60B 0A0C03
E60C 0A0E61
E60D 0A0C9
E60E 0A0E61
E60F 0A0C9
E610 0A0E61
E611 0A0C9
E612 0A0E61
E613 0A0E61
E614 0A0E61
E615 0A0E61
E616 0A0E61
E617 0A0E61
E618 0A0E61
E619 0A0E61
E61A 0A0E61
E61B 0A0E61
E61C 0A0E61
E61D 0A0E61
E61E 0A0E61
E61F 0A0E61
E620 0A0E61
E621 0A0E61
E622 0A0E61
E623 0A0E61
E624 0A0E61
E625 0A0E61
E626 0A0E61
E627 0A0E61
E628 0A0E61
E629 0A0E61
E62A 0A0E61
E62B 0A0E61
E62C 0A0E61
E62D 0A0E61
E62E 0A0E61
E62F 0A0E61

PARAMETERS

E603 ERR_BEEP PROC NEAR
E604 BEEP PROC NEAR
E605 BEEP END
E606 BEEP_ALLOC
E607 BEEP_ALLOC
E608 BEEP_ALLOC
E609 BEEP_ALLOC
E610 BEEP_ALLOC
E611 BEEP_ALLOC
E612 BEEP_ALLOC
E613 BEEP_ALLOC
E614 BEEP_ALLOC
E615 BEEP_ALLOC
E616 BEEP_ALLOC
E617 BEEP_ALLOC
E618 BEEP_ALLOC
E619 BEEP_ALLOC
E620 BEEP_ALLOC
E621 BEEP_ALLOC
E622 BEEP_ALLOC
E623 BEEP_ALLOC
E624 BEEP_ALLOC
E625 BEEP_ALLOC
E626 BEEP_ALLOC
E627 BEEP_ALLOC
E628 BEEP_ALLOC
E629 BEEP_ALLOC
E62A BEEP_ALLOC
E62B BEEP_ALLOC
E62C BEEP_ALLOC
E62D BEEP_ALLOC
E62E BEEP_ALLOC
E62F BEEP_ALLOC

ERROR BEEP

ESF 9C
ESF 9D
ESF 9E
ESF 9F
ESF 5F
ESF 60
ESF 61
ESF 62
ESF 63
ESF 64
ESF 65
ESF 66
ESF 67
ESF 68
ESF 69
ESF 6A
ESF 6B
ESF 6C
ESF 6D
ESF 6E
ESF 6F
ESF 50
ESF 51
ESF 52
ESF 53
ESF 54
ESF 55
ESF 56
ESF 57
ESF 58
ESF 59
ESF 5A
ESF 5B
ESF 5C
ESF 5D
ESF 5E
ESF 5F
ESF 60
ESF 61
ESF 62
ESF 63
ESF 64
ESF 65
ESF 66
ESF 67
ESF 68
ESF 69
ESF 6A
ESF 6B
ESF 6C
ESF 6D
ESF 6E
ESF 6F
ESF 70
ESF 71
ESF 72
ESF 73
ESF 74
ESF 75
ESF 76
ESF 77
ESF 78
ESF 79
ESF 7A
ESF 7B
ESF 7C
ESF 7D
ESF 7E
ESF 7F
ESF 80
ESF 81
ESF 82
ESF 83
ESF 84
ESF 85
ESF 86
ESF 87
ESF 88
ESF 89
ESF 8A
ESF 8B
ESF 8C
ESF 8D
ESF 8E
ESF 8F
ESF 90
ESF 91
ESF 92
ESF 93
ESF 94
ESF 95
ESF 96
ESF 97
ESF 98
ESF 99
ESF 9A
ESF 9B
ESF 9C
ESF 9D
ESF 9E
ESF 9F
ESF A0
ESF A1
ESF A2
ESF A3
ESF A4
ESF A5
ESF A6
ESF A7
ESF A8
ESF A9
ESF AA
ESF AB
ESF AC
ESF AD
ESF AE
ESF AF
ESF B0
ESF B1
ESF B2
ESF B3
ESF B4
ESF B5
ESF B6
ESF B7
ESF B8
ESF B9
ESF BA
ESF BB
ESF BC
ESF BD
ESF BE
ESF BF
ESF C0
ESF C1
ESF C2
ESF C3
ESF C4
ESF C5
ESF C6
ESF C7
ESF C8
ESF C9
ESF CA
ESF CB
ESF CC
ESF CD
ESF CE
ESF CF
ESF D0
ESF D1
ESF D2
ESF D3
ESF D4
ESF D5
ESF D6
ESF D7
ESF D8
ESF D9
ESF DA
ESF DB
ESF DC
ESF DD
ESF DE
ESF DF
ESF E0
ESF E1
ESF E2
ESF E3
ESF E4
ESF E5
ESF E6
ESF E7
ESF E8
ESF E9
ESF EA
ESF EB
ESF EC
ESF ED
ESF EE
ESF EF
ESF F0
ESF F1
ESF F2
ESF F3
ESF F4
ESF F5
ESF F6
ESF F7
ESF F8
ESF F9
ESF FA
ESF FB
ESF FC
ESF FD
ESF FE
ESF FF

System BIOS A-19
LOC OBJ  LINE  SOURCE

E630        1356  XLAT_PR  PROC    NEAR     | CONVERT 90-9F TO ASCII CHARACTER
E630 0640   1357    ADD     AL,09FH    | ADD FIRST CONVERSION FACTOR
E632 27     1358    DAA     | ADJUST FOR NUMERIC AND ALPHA RANGE
E635 1440   1359    ADC     AL,04FH    | ADD CONVERSION AND ADJUST LOW NIBBLE
E635 27     1360    DAA     | ADJUST HI NIBBLE TO ASCII RANGE
E636        1361  PRT_HEX  PROC    NEAR     | DISPLAY CHAR. IN AL
E636 B40E   1362    MOV     AH,14     | CALL VIDEO_ID
E638 B700   1363    MOV     BH,0     |
E63A CD10   1364    INT     10H     |
E63C C3     1365    RET     |
E63D        1366  PRT_HEX  ENDP     |
E63D        1367  XLAT_PR  ENDP     |
E63E        1368  XPC_BYTE  ENDP     |
E63F        1369

E643        1370  F4      LABEL    WORD     | PRINTER SOURCE TABLE
E64D BC03   1371    DW     38CH     |
E64F 7003   1372    DW     378H     |
E651 7802   1373    DW     278H     |
E652        1374  F4E     LABEL    WORD     |
E657        1375

E653        1376  ;    THIS PROCEDURE WILL SEND A SOFTWARE RESET TO THE KEYBOARD.
E654        1377  ;    SCAN CODE '"A' SHOULD BE RETURNED TO THE CPU.
E655        1378  ;    -----------------------------
E656        1379

E657        1380  KBD_RESET  PROC    NEAR     |
E658 B00C   1381    MOV     AL,0CH     | SET KBD CLK LINE LOW
E659        1382  G6:      LOOP    G6     | WRITE 0255 PORT B
E65A        1383    MOV     CX,10502   | HOLD KBD CLK LOW FOR 20 MS
E65B        1384  G6:      LOOP    G6     | LOOP FOR 20 MS
E65C        1385  MOV     AL,0CCCH    | SET CLK, ENABLE LINES HIGH
E65D        1386  OUT     PORT_B,AL  |
E65E        1387  SP_TEST:   PROC    NEAR     | ENTRY FOR MANUFACTURING TEST 2
E65F        1388    MOV     AL,0CH     | SET KBD CLK HIGH, ENABLE LOW
E660 B04C   1389    OUT     PORT_B,AL  |
E661        1390    MOV     AL,0FH     | ENABLE KEYBOARD INTERRUPTS
E662        1391    OUT     INTA01,AL  | WRITE 0259 IMR
E663        1392    MOV     AH,0      | ENABLE SYSTEM INTERRUPTS
E664        1393  G9:      LOOP    G9     | RESET INTERRUPT INDICATOR
E665        1394  SUB     CX,CX     | SETUP INTERRUPT TIMEOUT CNT
E666        1395  TEST     AH,OFFH    | DID A KEYBOARD INT OCCUR?
E667        1396  G9:      LOOP    G9     | NO - LOOP TILL TIMEOUT
E668        1397  IN      AL,PORT_A   | READ KEYBOARD SCAN CODE
E669 E6A0   1398    MOV     BL,AL     | SAVE SCAN CODE JUST READ
E66A B0CC   1399    MOV     AL,00CH    | CLEAR KEYBOARD
E66B E661   1400    OUT     PORT_B,AL  |
E66C 1E73   1401  G10:     LOOP    G10    | RETURN TO CALLER
E66D        1402  KBD_RESET  ENDP     |
E66E        1403

E66F        1404  BLINK_INT  PROC    NEAR     |
E670        1405  STI      |
E670 FB     1406  PUSH     AX      | SAVE AX REG CONTENTS
E671 E641   1407  IN      AL,PORT_B   | READ CURRENT VAL OF PORT B
E671 8AE0   1408  MOV     AH,AL     |
E673 6600   1409  NOT     AL      | FLIP ALL BITS
E675 2400   1410  AND     AL,01000000B | ISOLATE CONTROL BIT
E677 20EBF  1411  AND     AH,1111111B | MASK OUT OF ORIGINAL VAL
E678 0404   1412  OR      AL,AH     | OR NEW CONTROL BIT IN
E679 E6A4   1413  OUT     PORT_B,AL  |
E67A D0E0   1414  MOV     AL,EOI    |
E67B D0E0   1415  OUT     INTA00,AL  |
E67C C506   1416  PDP     AX      | RESTORE AX REG
E67D CF     1417  IRET     |
E67E        1418  BLINK_INT  ENDP     |
E67F        1419

A-20  System BIOS
System BIOS A-21

LOC OBJ

E693 2AE4
E694 044702
E695 B199
E696 D300
E697 0BC8
E698 51
E699 B104
E69A D300
E69B 0D0
E69C 59
E69D E0005
E69E 7105
E69F E651
E6A0 EB13
E6A1 03
E6A2 C60001
E6A3 28700001
E6A4 60H
E6A5 3

LINE SOURCE

1433 SUB AH,AH ; ZERO OUT AH
1434 MOV AL,(BX+2) ; SET LENGTH INDICATOR
1435 MOV CL,3AH ; MULTIPLE BY 512
1436 MOV CX,AX
1437 MOV CX,AX ; SET COUNT
1438 PUSH CX
1439 MOV CL,4
1440 SHR AX,CL
1441 ADD DX,AX ; SET POINTER TO NEXT MODULE
1442 POP CX
1443
1444 CALL ROM_CHECKSUM_CHK ; DO CHECKSUM
1445 JZ ROM_CHECK_1
1446 CALL ROM_ERR ; PRINT ERROR INFO
1447 JMP SHORT ROM_CHECK_END
1448 ROM_CHECK_1:
1449 PUSH DX ; SAVE POINTER
1450 MOV ES:IO_ROM_SEG,DS LOAD SEGMENT
1451 MOV ES:IO_ROM_SEG,DS LOAD SEGMENT
1452 CALL DWORD PTR ES:IO_ROM_INIT ; CALL INIT RTN.
1453 POP DX
1454
1455 RET
1456 JMP ROM_CHECK END
1457
1458 |--------------------------------------------------------------------------|
1459 | THIS SUBROUTINE WILL PRINT A MESSAGE ON THE DISPLAY |
1460 |
1461 | ENTRY REQUIREMENTS: |
1462 | SI = OFFSET(ADDRESS) OF MESSAGE BUFFER |
1463 | CX = MESSAGE BYTE COUNT |
1464 | MAXIMUM MESSAGE LENGTH IS 36 CHARACTERS |
1465 |
1466 P_MSG PROC NEAR
1467 CALL DDS
1468 CMP MSG_TST,1 ; MSG TEST MODE?
1469 JNE G1Z ; NO - DISPLAY ERROR MSG
1470 MOV DH,1 ; YES - SETUP TO BEEP SPEAKER
1471 JMP ERR_BEEP ; YES - BEEP SPEAKER
1472 G1Z: MOV AL,CS:[SI] ; PUT CHAR IN AL
1473 MOV AX,SI ; POINT TO NEXT CHAR
1474 MOV AX,SAVE CHAR
1475 MOV AX,DATA
1476 CALL PRT_HEX ; CALL VIDEO_IO
1477 POP AX ; RECOVER PRINT CHAR
1478 CMP AL,10 ; WAS IT LINE FEED
1479 JNE G1Z ; NO, KEEP PRINTING STRING
1480 RET
1481 P_MSG ENDP
1482
1483 F3A DB ' ROM',13,10
1484 DD
1485 D_OEO PROC NEAR
1486 PUSH AX
1487 MOV AL,2DH
1488 OUT 2DH,AL
1489 POP AX
1490 IRET
1491 D_OEO ENDP
1492
1493 |--- INT 19 ---| |--------------------------|
1494 | BOOT STRAP LOADER |
1495 | IF A 5 1/4" DISKETTE DRIVE IS AVAILABLE ON THE SYSTEM, |
1496 | TRACK 0, SECTOR 1 IS READ INTO THE BOOT LOCATION |
1497 | (SEGMENT 0, OFFSET 7C00) AND CONTROL IS TRANSFERRED |
1498 | THERE. |
1499 |
1500 | IF THERE IS NO DISKETTE DRIVE, OR IF THERE IS A |
1501 | HARDWARE ERROR CONTROL IS TRANSFERRED TO THE RESIDENT |
1502 | BASIC ENTRY POINT. |
1503 |
1504 | IPL ASSUMPTIONS: |
1505 | 0855 POINT 60H BIT 0 = 1 IF IPL FROM DISKETTE |
1506 |--------------------------------------------------------------------------|
1507 | ASSUME CS:CODE,DS:ABOS
IPL WAS 150
1200
110
CX HAS RETRY COUNT
4800
9600
BAllO MUST LOAD SYSTEM FROM DISKETTE
NONE
EVEN
2
LaC OBJ
E6E4
E706 741E
E6F3 2DC0
E6F5 6EDB
E6F2 FB
E6F2
E6F4 EA07C000
E6FD 0CE74AD0
E701 A11004
E704 A001
E706 741E
E13E C704
E13F 1528
E13F 1522
E13F 1520
E13F 1519
E13F 1518
E13F 1517
E13F 1516
E13F 1515
E13E 1514
E13E 1513
E13E 1512
E13E 1511
E13E 1510
E13D 1509
E13D 1508

**** IPL WAS SUCCESSFUL
MOV WORD PTR DISK_POINTER,OFFSET DISK_BASE
MOV WORD PTR DISK_POINTER+2,CS
MOV AX,DATA_WORD(OFFSET EQUIPJLAGl
; GET THE EQUIPMENT SWITCHES
TEST AL,1 ; ISOLATE IPL SENSE SWITCH
JZ H3 ; GO TO CASSETTE BASIC ENTRY POINT
; MUST LOAD SYSTEM FROM DISKETTE -- CX HAS RETRY COUNT
E708 B90400
E708 51
E70C 6400
E70E CD15
E710 720F
E712 B0102
E715 2002
E717 0ECC
E719 B0007C
E71C B91000
E71F CD15
E721 59
E722 75CD
E724 E1EE
E726
E726 CD10
E726

; --- IPL SYSTEM
; SAVE RETRY COUNT
; RESET THE DISKETTE SYSTEM
; JC HZ ; IF ERROR, TRY AGAIN
; READ IN THE SINGLE SECTOR
; MOV ES,DX
; MOV AX,DATA
; MOV BX,OFFSET BOOT_LOCN
; MOV CX,1 ; SECTOR 1, TRACK 0
; ; DISKETTE_ID
; ; RECOVER RETRY COUNT
; ; OF SET BY UNSUCCESSFUL READ
; DO IT FOR RETRY TIMES
; --- UNABLE TO IPL FROM THE DISKETTE
; --- INT 14-------------------------------
; PORT ACCORDING TO THE PARAMETERS:
; INITIALIZE THE COMMUNICATIONS PORT
; (AL) HAS PARAMETERS FOR INITIALIZATION
; ----- PARITY---- STOPBIT ----WORD LENGTH-----
; X0 - NONE 0 - 1 10 - 7 BITS
; 01 - ODD 1 - 2 11 - 8 BITS
; 10 - EVEN 0 - 3 12 - 16 BITS
; 11 - 600 0 - 5 13 - 256
; 100 - 1200 0 - 6 14 - 512
; 101 - 2400 0 - 7 15 - 256
; 110 - 4800 0 - 8 16 - 256
; 111 - 9600 0 - 9 17 - 512
; ON RETURN, CONDITIONS SET AS IN CALL TO COMPO STATUS (AH=3)
; SEND THE CHARACTER IN (AL) OVER THE COMPO LINE
; (AL) REGISTER IS PRESERVED
; ON EXIT, BIT 7 OF AH IS SET IF THE ROUTINE WAS UNABLE
; TO TRANSMIT THE BYTE OF DATA OVER THE LINE.
; IF BIT 7 OF AH IS NOT SET, THE REMAINDER OF AH
; IS SET AS IN A STATUS REQUEST, REFLECTING THE
; CURRENT STATUS OF THE LINE.
; ((AH)=2 RECEIVE A CHARACTER IN (AL) FROM COMPO LINE BEFORE
; RETURNING TO CALLER
; ON EXIT, AH HAS THE CURRENT LINE STATUS, AS SET BY THE
; THE STATUS ROUTINE, EXCEPT THAT THE ONLY BITS
; LEFT ON ARE THE ERROR BITS (7,A,3,2,1)
; IF AH HAS BIT 7 ON (TIME OUT) THE REMAINING
; BITS ARE NOT PREDICTABLE.
; THUS, AH IS NON ZERO ONLY WHEN AN ERROR

A-22 System BIOS
LOC OBJ LINE SOURCE

1585 ; OCCURRED.
1586 ; (AH)=3 RETURN THE COMM PORT STATUS IN (AX)
1587 ; AH CONTAINS THE LINE STATUS
1588 ; BIT 7 = TIME OUT
1589 ; BIT 6 = TRANS SHIFT REGISTER EMPTY
1590 ; BIT 5 = TRAN HOLDING REGISTER EMPTY
1591 ; BIT 4 = BREAK DETECT
1592 ; BIT 3 = FRAMING ERROR
1593 ; BIT 2 = PARITY ERROR
1594 ; BIT 1 = OVERRUN ERROR
1595 ; BIT 0 = DATA READY
1596 ; AL CONTAINS THE MODEN STATUS
1597 ; BIT 7 = RECEIVED LINE SIGNAL DETECT
1598 ; BIT 6 = RING INDICATOR
1599 ; BIT 5 = DATA SET READY
1600 ; BIT 4 = CLEAR TO SEND
1601 ; BIT 3 = DELTA RECEIVE LINE SIGNAL DETECT
1602 ; BIT 2 = TRAILING EDGE RING DETECTOR
1603 ; BIT 1 = DATA SET READY
1604 ; BIT 0 = DELTA CLEAR TO SEND
1605 ;
1606 ; (DX) = PARAMETER INDICATING WHICH RS232 CARD (0,1 ALLOWED)
1607 ;
1608 ; DATA AREA RS232_BASE CONTAINS THE BASE ADDRESS OF THE 8250 ON THE
1609 ; CARD LOCATION 400H CONTAINS UP TO 4 RS232 ADDRESSES POSSIBLE
1610 ; DATA AREA LABEL RS232_T2H_OUT (BYTE) CONTAINS OUTER LOOP COUNT
1611 ; VALUE FOR TIMEOUT (DEFAULT=11)
1612 ; OUTPUT
1613 ; AX MODIFIED ACCORDING TO PARMS OF CALL
1614 ; ALL OTHERS UNCHANGED
1615 ;-----------------------------------------------------------------------
1616 ; ASSUME CS:CODE,DS:DATA
1617
1618 E729 1704 DW 1047 ; 110 BAUD
1619 E729 1700 DW 768 ; 150
1620 E729 0000 DW 300
1621 E729 0000 DW 300
1622 E729 0000 DW 192
1623 E729 0000 DW 96
1624 E729 0000 DW 40
1625 E729 0000 DW 2400
1626 E729 0000 DW 4000
1627
1628 E739 1860 DW 12
1629
1630 ;------ VECTOR TO APPROPRIATE ROUTINE
1631
1632 E739 FR ; STI 
1633 E73A 1E PUSH DS ; INTRUPTS BACK ON
1634 E73B 52 PUSH DX ; SAVE SEGMENT
1635 E73C 56 PUSH SI
1636 E73D 57 PUSH DI
1637 E73E 51 PUSH CX
1638 E73F 53 PUSH BX
1639 E740 86F2 MOV SI,DX ; RS232 VALUE TO SI
1640 E742 88FA MOV DI,DX
1641 E744 D16 MOV SI,1 ; WORD OFFSET
1642 E746 E0F517 CALL DS5 ; CALL DS
1643 E749 8814 MOV DX,R5232_BASE[SI] ; GET BASE ADDRESS
1644 E748 0802 OR DX,DX ; TEST FOR 0 BASE ADDRESS
1645 E747 7413 JZ A3 ; RETURN
1646 E754 044 OR AH,AH
1647 E751 7416 JZ A4 ; TEST FOR (AH)=0
1648 E753 FECC DEC AH ; COPPER INIT
1649 E755 7445 JZ A5 ; SEND AL
1650 E757 FECC DEC AH ; TEST FOR (AH)=2
1651 E759 744A JE A12 ; RECEIVE INTO AL
1652 E750 1652 A2: DEC AH ; TEST FOR (AH)=3
1653 E750 FECC DEC AH
1654 E750 7503 JNZ A3
1655 E75F E98300 JMP A10 ; COMMUNICATION STATUS
1656 E762 1656 A3: ; RETURN FROM RS232
1657 E762 5B POP BX
1658 E763 59 POP CX
1659 E764 5F POP DI
1660 E765 5E POP SI
1661 E766 5A POP DX

Appendix A

System BIOS A-23
--- INITIALIZE THE COMMUNICATIONS PORT ---

1664 1665 1666

E769 8AE0 1668 1669 1670

E768 03C203 1669 1670 1671

E760 B080 1670 1671 1672

E771 8AD4 1672 1673 1674

E773 B104 1674 1675 1676

E775 02C2 1676 1677 1678

E777 01E020 1677 1678 1679

E779 BFEF 1678 1679 1680

E77E 03FA 1679 1680 1681

E782 4A 1681 1682 1683

E785 03C04 1682 1683 1684

E787 8AC4 1683 1684 1685

E790 24 1684 1685 1686

E792 EE 1685 1686 1687

E794 03 1686 1687 1688

E796 03 1687 1688 1689

E798 03 1688 1689 1690

E79A EB 1689 1690 1691

E79C 50 1690 1691 1692

E79D 03C204 1691 1692 1693

E79F B083 1692 1693 1694

E7A2 EE 1693 1694 1695

E7A4 04 1694 1695 1696

E7A6 04 1695 1696 1697

E7A8 04 1696 1697 1698

E7B0 EE 1697 1698 1699

E7C9 50 1699 1700 1701

E7CD 75 1701 1702 1703

E7D3 73 1702 1703 1704

E7D8 06 1703 1704 1705

E7DE 7400 1704 1705 1706

E7F4 49 1705 1706 1707

E7F5 0A 1706 1707 1708

E7F7 49 1707 1708 1709

E7F9 49 1708 1709 1710

E7FA 49 1709 1710 1711

E7FB 49 1710 1711 1712

E7FD 49 1711 1712 1713

E7FF 49 1712 1713 1714

E801 49 1713 1714 1715

E804 49 1714 1715 1716

E807 49 1715 1716 1717

E80A 49 1716 1717 1718

E80D 49 1717 1718 1719

E810 49 1718 1719 1720

E813 49 1719 1720 1721

E816 49 1720 1721 1722

E819 49 1721 1722 1723

E81C 49 1722 1723 1724

E81F 49 1723 1724 1725

E822 49 1724 1725 1726

E825 49 1725 1726 1727

E828 49 1726 1727 1728

E82B 49 1727 1728 1729

A-24 System BIOS
E7C7 E2000D 1739 CALL WAIT_FOR_STATUS ; TEST FOR DSR 
E7D2 750B 1740 JNZ A8 ; RETURN WITH ERROR 
E7D4 4A 1741 AI: WAIT_DSR_END 
E7D4 4A 1742 DEC DX ; LINE STATUS REGISTER 
E7D5 4B 1743 AI: WAIT_RECV 
E7D5 4701 1744 MOV BH,0 ; RECEIVE BUFFER FULL 
E7D7 E81000 1745 CALL WAIT_FOR_STATUS ; TEST FOR REC. BUFF. FULL 
E7D8 7503 1746 JNZ A8 ; SET TIME OUT ERROR 
E7DC 4B 1747 AI: GET_CHAR 
E7DC B041E 1748 AND AH,0001110B ; TEST FOR ERR CONDITIONS ON RECV CHAR 
E7DF 0814 1749 MOV DX,RS232_BASE+1I ; DATA PORT 
E7E1 EC 1750 IN AL,DX ; GET CHARACTER FROM LINE 
E7E2 E970FF 1751 JMP A3 ; RETURN 

1752 

1753 ;----- COMM PORT STATUS ROUTINE 

E7E4 8814 1754 CALL 
E7E5 8814 1755 A1: MOV DX,RS232_BASE+1I ; CONTROL PORT 
E7E7 32C7 1756 ADD DX,5 
E7E8 805 1757 IN AL,DX ; GET LINE CONTROL STATUS 
E7EA 84E0 1758 MOV AH,AL ; PUT IN AH FOR RETURN 
E7EB 42 1759 INC DX ; POINT TO MODEM STATUS REGISTER 
E7EE EC 1760 IN AL,DX ; GET MODEM CONTROL STATUS 
E7EF E970FF 1761 JMP A3 ; RETURN 

1762 

1763 ;------------------------------------- 

E7F2 8A507C 1764 ; WAIT FOR STATUS ROUTINE 
E7F7 8A507C 1765 : 

1766 ; ENTRY: 
1767 ; BH=STATUS-bit(51) TO LOOK FOR, 
1768 ; DX=ADDR. OF STATUS REG 
1769 ; EXIT: 
1770 ; ZERO FLAG ON = STATUS FOUND 
1771 ; ZERO FLAG OFF = TIMEOUT. 
1772 ; AH=LAST STATUS READ 

1773 

1774 ;------------------------------------- 

E7F2 8A507C 1775 WAIT_FOR_STATUS_PROC NEAR 
E7F5 E970FF 1776 MOV BL,RS232_TIM_OUT(1D1) ; LOAD OUTER LOOP COUNT 
E7F5 2DC9 1777 WFSO: SUB CX,CX 
E7F7 E970FF 1778 WFS1: 
E7F7 EC 1779 IN AL,DX ; GET STATUS 
E7F8 08E0 1780 MOV AH,AL ; MOVE TO AH 
E7FA 2B07 1781 AND AL,SH ; ISOLATE BITS TO TEST 
E7FC 3AC7 1782 CMP AL,BH ; EXACTLY = TO MASK 
E7FE 7408 1783 JE WFS_END ; RETURN WITH ZERO FLAG ON 
E800 E25F 1784 LOOP WFS1 ; TRY AGAIN 
E802 FEB6 1785 DEC BL 
E804 75EF 1786 JNZ WFSO 
E806 0AFF 1787 OR BH,SH ; SET ZERO FLAG OFF 
E808 8F 1788 WFS_END: 
E809 C3 1789 RET 

1790 WAIT_FOR_STATUS_ENDP 
E809 C3 1791 RS232_ID ENDP 
E809 C3 1792 

1793 ;------------------------------------- 

E809 C3 1794 ; PRINT ADDRESS AND ERROR MESSAGE FOR ROM CHECKSUM ERRORS 
E809 C3 1795 ;------------------------------------- 

E809 C3 1796 ROM_ERR_PROC NEAR 
E80A 52 1797 PUSH DX ; SAVE POINTER 
E80A 50 1798 PUSH AX 
E80B 6CDA 1799 MOV DX,05 ; GET ADDRESS POINTER 
E80D 01FAC8 1800 CMP DX,0C800H 
E811 7E13 1801 JLE ROM_ERR_BEEP ; SPECIAL ERROR INDICATION 
E813 6AC6 1802 MOV AL,DH 
E815 E0DFE 1803 CALL XPC_BYTE ; DISPLAY ADDRESS 
E816 6AC2 1804 MOV AL,DL 
E818 E0BFE 1805 CALL XPC_BYTE 
E818 E0766 1806 MOV SI,OFFSET F3A ; DISPLAY ERROR MSG 
E820 E97FE 1807 CALL P_MSG 
E823 8000 1808 ROM_ERR_END: 
E825 5A 1809 POP AX 
E825 5A 1810 POP DX 
E825 C3 1811 RET 
E826 8A02D1 1812 ROM_ERR_BEEP: MOV DX,010H ; BEEP 1 LONG, 2 SHORT 
E829 E83FD 1813 CALL ERR_BEEP 
E83C EB85 1815 JMP SHORT ROM_ERR_END
BAD OBJ

LINE SOURCE

1616 ROM_ERR ENDP
1617
1618 1:---- INT 16  -----------------------------
1619 1 KEYBOARD I/O
1620 1 THESE ROUTINES PROVIDE KEYBOARD SUPPORT
1621 1 INPUT
1622 1 (AH)=0 READ THE NEXT ASCII CHARACTER STRUCK FROM THE KEYBOARD
1623 1 RETURN THE RESULT IN (AH), SCAN CODE IN (AH)
1624 1 (AH)=1 SET THE Z FLAG TO INDICATE IF AN ASCII CHARACTER IS
1625 1 AVAILABLE TO BE READ.
1626 1 (ZF)=1 -- NO CODE AVAILABLE
1627 1 (ZF)=0 -- CODE IS AVAILABLE
1628 1 IF ZF = 0, THE NEXT CHARACTER IN THE BUFFER TO BE READ
1629 1 IS IN AX, AND THE ENTRY REMAINS IN THE BUFFER
1630 1 (AH)=2 RETURN THE CURRENT SHIFT STATUS IN AL REGISTER
1631 1 THE BIT SETTINGS FOR THIS CODE ARE INDICATED IN THE
1632 1 THE EQUATES FOR KB_FLAG
1633 1 OUTPUT
1634 1 AS NOTED ABOVE, ONLY AX AND FLAGS CHANGED
1635 1 ALL REGISTERS PRESERVED
1636 1:------------------------------------------------------------------------
1637 1 ASSUME CS:CODE,DS:DATA
1638
1639 KEYBOARD_IO PROC FAR
1640 E82E
1641 STI
1642 PUSH DS
1643 PUSH BX
1644 CALL E630
1645 OR AH, AH
1646 JZ 1636
1647 MOV AX, BUFFER_HEAD
1648 CALL K3
1649 JMP SHORT INTIO_END
1650 KEYBOARD_IO ENDP
1651
1652 1:----- READ THE KEY TO FIGURE OUT WHAT TO DO
1653
1654 KB:
1655 STI
1656 PUSH DS
1657 CALL E69A
1658 MOV BX, BUFFER_HEAD
1659 CMP BX, BUFFER_TAIL
1660 JZ 1667
1661 MOV AX, [BX]
1662 CALL K2
1663 MOV BUFFER_HEAD, BX
1664 JMP SHORT INTIO_END
1665 KEYBOARD_IO ENDP
1666
1667 1:----- ASCII STATUS
1668
1669 K2:
1670 CLI
1671 POP BX
1672 MOV BX, BUFFER_HEAD
1673 CMP BX, BUFFER_TAIL
1674 JZ 1667
1675 MOV AX, [BX]
1676 MOV AX, [BX]
1677 JMP SHORT INTIO_END
1678
1679 1:----- SHIFT STATUS
1680
1681 KB:
1682 MOV AL, KB_FLAG
1683 MOV AX, AL
1684 MOV AX, AL
1685 MOV AX, AL
1686 KEYBOARD_IO ENDP
1687
1688 1:----- INCREMENT A BUFFER POINTER
1689
1690 K:
1691 INC BX
1692 INC BX
1693 INC BX

A-26 System BIOS
<table>
<thead>
<tr>
<th>LOC OBJ</th>
<th>LINE</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>E073 301E8200</td>
<td>1093</td>
<td>CMP BX,BUFFER_END ; AT END OF BUFFER?</td>
</tr>
<tr>
<td>E077 7504</td>
<td>1094</td>
<td>JNE K5 ; NO. CONTINUE</td>
</tr>
<tr>
<td>E079 001E8000</td>
<td>1095</td>
<td>MOV BX,BUFFER_START ; YES, RESET TO BUFFER BEGINNING</td>
</tr>
<tr>
<td>E07D C3</td>
<td>1096</td>
<td>K5:</td>
</tr>
<tr>
<td>E07D 53</td>
<td>1097</td>
<td>RET</td>
</tr>
<tr>
<td>E07D C4</td>
<td>1098</td>
<td>K4:</td>
</tr>
<tr>
<td>E080 10</td>
<td>1099</td>
<td>1900 c---- TABLE OF SHIFT KEYS AND MASK VALUES</td>
</tr>
</tbody>
</table>

1901

| E08E 52 | 1902 | K6: |
| E07F 3A | 1903 | DB INS_KEY ; INSERT KEY |
| E080 45 | 1904 | DB CAPS_KEY,HMR_KEY,SCROLL_KEY,ALT_KEY,CTRL_KEY |
| E081 46 | 1905 | DB LEFT_KEY,RIGHT_KEY |
| E082 30 | 1906 | K6L EQU $-K6 |
| E083 10 | 1907 | 1908 c---- SHIFT MASK TABLE |

1909

| E086 80 | 1910 | K7: |
| E087 40 | 1911 | DB INS_SHIFT ; INSERT MODE SHIFT |
| E088 20 | 1912 | DB CAPS_SHIFT,HMR_SHIFT,SCROLL_SHIFT,ALT_SHIFT,CTRL_SHIFT |
| E089 04 | 1913 | DB LEFT_SHIFT,RIGHT_SHIFT |
| E08A 01 | 1914 | 1915 c---- SCAN CODE TABLES |

1916

| E08E 1B | 1917 | K8: |
| E08F 80 | 1918 | DB 27,-1,0,-1,-1,-1,30,-1 |
| E090 00 | 1919 | DB -1,-1,-1,31,-1,127,-1,17 |
| E091 FF | 1920 | DB 23,5,10,20,25,21,9,15 |
| E092 FF | 1921 | DB 4,6,7,8,10,11,12,-1,-1 |
| E093 FF | 1922 | DB 16,27,29,10,-1,1,19 |
| E094 1E | 1923 | DB 18,20,22,24,26,28,30,-1 |
| E095 FF | 1924 | DB 19,21,23,25,27,29,31,-1 |
| E096 FF | 1925 | DB 0,2,4,6,8,10,12,-1,-1 |
| E097 FF | 1926 | DB 14,16,18,20,22,24,26,28 |
| E098 FF | 1927 | DB 0,2,4,6,8,10,12,-1,-1 |
| E099 FF | 1928 | DB 14,16,18,20,22,24,26,28 |
| E09A FF | 1929 | DB 0,2,4,6,8,10,12,-1,-1 |
| E09B 7F | 1930 | DB 14,16,18,20,22,24,26,28 |
| E09C FF | 1931 | DB 0,2,4,6,8,10,12,-1,-1 |
| E09D 11 | 1932 | DB 14,16,18,20,22,24,26,28 |
| E09E 17 | 1933 | DB 0,2,4,6,8,10,12,-1,-1 |
| E09F 05 | 1934 | DB 14,16,18,20,22,24,26,28 |
| E0A0 12 | 1935 | DB 0,2,4,6,8,10,12,-1,-1 |
| E0A1 14 | 1936 | DB 14,16,18,20,22,24,26,28 |
| E0A2 19 | 1937 | DB 0,2,4,6,8,10,12,-1,-1 |
| E0A3 15 | 1938 | DB 14,16,18,20,22,24,26,28 |
| E0A4 09 | 1939 | DB 0,2,4,6,8,10,12,-1,-1 |
| E0A5 0F | 1940 | DB 14,16,18,20,22,24,26,28 |
| E0A6 10 | 1941 | DB 0,2,4,6,8,10,12,-1,-1 |
| E0A7 1B | 1942 | DB 14,16,18,20,22,24,26,28 |
| E0A8 1D | 1943 | DB 0,2,4,6,8,10,12,-1,-1 |
| E0A9 0A | 1944 | DB 14,16,18,20,22,24,26,28 |
| E0AA FF | 1945 | DB 0,2,4,6,8,10,12,-1,-1 |
| E0AB 01 | 1946 | DB 14,16,18,20,22,24,26,28 |
| E0AC 13 | 1947 | DB 0,2,4,6,8,10,12,-1,-1 |
| E0AD 04 | 1948 | DB 14,16,18,20,22,24,26,28 |
| E0AE 06 | 1949 | DB 0,2,4,6,8,10,12,-1,-1 |
| E0AF 07 | 1950 | DB 14,16,18,20,22,24,26,28 |
| E0B0 08 | 1951 | DB 0,2,4,6,8,10,12,-1,-1 |
| E0B1 0A | 1952 | DB 14,16,18,20,22,24,26,28 |
| E0B2 08 | 1953 | DB 0,2,4,6,8,10,12,-1,-1 |
| E0B3 0C | 1954 | DB 14,16,18,20,22,24,26,28 |
| E0B4 FF | 1955 | DB 0,2,4,6,8,10,12,-1,-1 |
| E0B5 FF | 1956 | DB 14,16,18,20,22,24,26,28 |
| E0B6 FF | 1957 | DB 0,2,4,6,8,10,12,-1,-1 |
| E0B7 FF | 1958 | DB 14,16,18,20,22,24,26,28 |
| E0B8 1C | 1959 | DB 0,2,4,6,8,10,12,-1,-1 |

Appendix A

System BIOS  A-27
LaC OBJ
LINE
SOURCE

1923  DB  14,13,1,1,1,1,1,1,1
1924  DB  ' ','-'-1
1925  DB  94,95,96,97,98,99,100,101
1926  DB  102,103,1,1,119,1,132,-1
1927  DB  115,-1,116,-1,117,-1,118,-1
1928  DB  01DH,'1E34567890-','00H,09H
1929  DB  60H,-1 ,5CH,'zxcvbnnh'.1',
1930  DB  27,'!~$%^&*()+,-.,/01234567890-='
1931  DB  21402324
1932  DB  25
1933  DB  262A28295F2B
1934  DB  2C2:E2F
1935  DB  494F507B7D
1936  DB  'QWERTYUIOP',00H,09H
1937  DB  'ASDFGHJKL';
LOC OBJ | LINE | SOURCE
---|---|---
E936 00 | 1941 | DB 07EH,-1,'خذفمن>؟،-1,0,-1',+1
E937 FF |
E938 41534446474849 | 48AC3A22 |
E943 7E | |
E944 FF |
E945 7C5A5843564245 | 484C3A22 |
E950 FF |
E951 00 |
E952 FF |
E953 20 |
E954 FF |
E955 1942 | 1943 | K12 LABEL BYTE
E956 54 |
E957 55 |
E958 56 |
E959 57 |
E960 58 |
E961 59 |
E962 5A |
E963 5B |
E964 5C |
E965 5D |
E966 5E |
E967 5F |
E968 60 |
E969 61 |
E96A 62 |
E96B 63 |
E96C 64 |
E96D 65 |
E96E 66 |
E96F 67 |
E970 68 |
E971 69 |
E972 6A |
E973 6B |
E974 6C |
E975 6D |
E976 6E |
E977 6F |
E978 70 |
E979 71 |
E97A 72 |
E97B 73 |
E97C 74 |
E97D 75 |
E97E 76 |
E97F 77 |
E980 78 |
E981 79 |
E982 7A |
E983 7B |
E984 7C |
E985 7D |
E986 7E |
E987 7F |
E988 80 |
E989 81 |
E98A 82 |
E98B 83 |
E98C 84 |
E98D 85 |
E98E 86 |
E98F 87 |
E990 88 |
E991 89 |
E992 8A |
E993 8B |
E994 8C |
E995 8D |
E996 8E |
E997 8F |
E998 90 |
E999 91 |
E99A 92 |
E99B 93 |
E99C 94 |
E99D 95 |
E99E 96 |
E99F 97 |
E9A0 98 |
E9A1 99 |
E9A2 9A |
E9A3 9B |
E9A4 9C |
E9A5 9D |
E9A6 9E |
E9A7 9F |
E9A8 A0 |
E9A9 A1 |
E9AA A2 |
E9AB A3 |
E9AC A4 |
E9AD A5 |
E9AE A6 |
E9AF A7 |
E9B0 A8 |
E9B1 A9 |
E9B2 AA |
E9B3 AB |
E9B4 AC |
E9B5 AD |
E9B6 AE |
E9B7 AF |
E9B8 B0 |
E9B9 B1 |
E9BA B2 |
E9BB B3 |
E9BC B4 |
E9BD B5 |
E9BE B6 |
E9BF B7 |
E9C0 B8 |
E9C1 B9 |
E9C2 BA |
E9C3 BB |
E9C4 BC |
E9C5 BD |
E9C6 BE |
E9C7 BF |
E9C8 C0 |
E9C9 C1 |
E9CA C2 |
E9CB C3 |
E9CC C4 |
E9CD C5 |
E9CE C6 |
E9CF C7 |
E9D0 C8 |
E9D1 C9 |
E9D2 CA |
E9D3 CB |
E9D4 CC |
E9D5 CD |
E9D6 CE |
E9D7 CF |
E9D8 D0 |
E9D9 D1 |
E9DA D2 |
E9DB D3 |
E9DC D4 |
E9DD D5 |
E9DE D6 |
E9DF D7 |
E9E0 E8 |
E9E1 E9 |
E9E2 EA |
E9E3 EB |
E9E4 EC |
E9E5 ED |
E9E6 EE |
E9E7 EF |
E9E8 F0 |
E9E9 F1 |
E9EA F2 |
E9EB F3 |
E9EC F4 |
E9ED F5 |
E9EE F6 |
E9EF F7 |
E9F0 F8 |
E9F1 F9 |
E9F2 FA |
E9F3 FB |
E9F4 FC |
E9F5 FD |
E9F6 FE |
E9F7 FF |

Appendix A

System BIOS  A-29
A-30  System BIOS
EA0E 80F010 2054 CMP AH,SCROLL_SHIFT ; IS THIS A TOGGLE KEY
EA12 731A 2055 JAE K24 ; YES, HANDLE BREAK TOGGLE
EA12 F04 2056 NOT AH ; INVERT MASK
EA12 20621700 2057 AND KB_FLAG,AH ; TURN OFF SHIFT BIT
EA29 7C80 2058 CMP AL,ALT_KEY=00H ; IS THIS ALTERNATE SHIFT RELEASE
EA29 752C 2059 JNE K26 ; INTERRUPT_RETURN
2060
EA29 20611000 2061 ------- ALTERNATE SHIFT KEY RELEASED, GET THE VALUE INTO BUFFER
EA29 2062 2062
EA29 A01900 2063 MOV AL,ALT_INPUT
EA30 B000 2064 MOV AH,0 ; SCAN CODE OF 0
EA32 0061900 2065 MOV AL,INPUT, AH ; ZERO OUT THE FIELD
EA32 3C00 2066 CMP AL,0 ; WAS THE INPUT-0?
EA32 741F 2067 JE K26 ; INTERRUPT_RETURN
EA34 E9A1D1 2068 JMP K56 ; IT MASH'T, SO PUT IN BUFFER
EA35 2069 K24: ; BREAK-TOGGLE
EA36 F04 2070 NOT AH ; INVERT MASK
EA37 20621000 2071 AND KB_FLAG_1,AH ; INDICATE NO LONGER DEPRESSED
EA43 ED14 2072 JMP SHORT K26 ; INTERRUPT_RETURN
2073
EA44 2074 2074 ------- TEST FOR HOLD STATE
EA44 2075 2075
EA44 A6 2076 K25: ; NO-SHIFT-FOUND
EA44 3C00 2077 CMP AL,00H ; TEST FOR BREAK KEY
EA44 7310 2078 JAE K26 ; NOTHING FOR BREAK CHARMS FROM HERE ON
EA44 F06110008 2079 TEST KB_FLAG_1,HOLD_STATE ; ARE WE IN HOLD STATE
EA44 7417 2080 JZ K20 ; BRANCH AROUND TEST IF NOT
EA45 3C45 2081 CMP AL,HAPM_KEY
EA45 7405 2082 JE K26 ; CAN'T END HOLD ON NUM_LOCK
EA45 00611000F7 2083 AND KB_FLAG_1,HOLD_STATE ; TURN OFF THE HOLD STATE BIT
EA45 2084
EA45 2085 K26: ; INTERRUPT-RETURN
EA45 FA 2086 CLI ; TURN OFF INTERRUPTS
EA45 D0C0 2087 MOV AL,EOI ; END OF INTERRUPT COMMAND
EA45 E220 2088 OUT 020H,AL ; SEND COMMAND TO INT CONTROL PORT
EA45 2089
EA46 208A
EA46 07 2089 POP E5
EA46 1F 2090 POP DS
EA46 6F 2091 POP D1
EA46 6E 2092 POP SI
EA46 5A 2093 POP DX
EA46 59 2094 POP CX
EA46 58 2095 POP BX
EA46 CF 2096 POP AX ; RESTORE STATE
EA46 2097 IRET ; RETURN, INTERRUPTS BACK ON
EA46 2098
EA46 2099
EA46 70 2090 I ----- NOT IN HOLD STATE, TEST FOR SPECIAL CHAR
EA46 71 2091
EA46 2092 K28: ; NO-HOLD-STATE
EA46 F06170008 2093 TEST KB_FLAG,ALT_SHIFT ; ARE WE IN ALTERNATE SHIFT
EA46 7503 2094 JNZ K29 ; JUMP IF ALTERNATE SHIFT
EA46 E9100 2095 JMP K30 ; JUMP IF NOT ALTERNATE
EA46 2096
EA46 2097
EA46 2098
EA46 2099
EA46 70 2090 I ----- TEST FOR RESET KEY SEQUENCE (CTL ALT DEL)
EA46 2091
EA46 2092 K28: ; NO-HOLD-STATE
EA46 F06170004 2093 TEST KB_FLAG,CTRL_SHIFT ; ARE WE IN CONTROL SHIFT ALSO
EA46 7503 2094 JNZ K31 ; NO_RESET
EA46 7503 2111 JZ K31 ; NO_RESET
EA46 752F 2112 CMP AL,DEL_KEY ; SHIFT STATE IS THERE, TEST KEY
EA46 752F 2113 JNE K31 ; NO_RESET
EA46 2114
EA46 2115
EA46 2116
EA46 C70672003412 2117 MOV RESET_FLAG,123AH ; SET FLAG FOR RESET FUNCTION
EA46 E98E000F0 2118 JMP RESET ; JUMP TO POWER ON DIAGNOSTICS
EA46 2119
EA46 2120
EA46 2120 I ----- ALT-INPUT-TABLE
EA46 2121 K30 LABEL BYTE
EA46 2122 K50 DB 02,79,80,61,75,76,77
EA46 2123 K30 DB 71,72,73 ; 10 NUMBERS ON KEYPAD
A-32 System BIOS
System BIOS  A-33
A-34 System BIOS
EBDS 3CF 2330 CMP AL,-1 ; IS THIS AN IGNORE CHAR
EBD7 741F 2331 JE K59 ; YES, DO NOTHING WITH IT
EBD9 60CF 2332 CMP AH,-1 ; LOOK FOR ~I PSEUDO SCAN
ESC 741A 2333 JE K59 ; CLEAR INTERRUPT_RETURN

2334 -------- HANDLE THE CAPS LOCK PROBLEM
2336 EBD0 2337 K58: TEST KB_FLAG,CAPS_STATE ; ARE WE IN CAPS LOCK STATE
2338 JZ K61 ; SKIP IF NOT
2340
2341 -------- IN CAPS LOCK STATE
2342 EBD3 2343 K59: TEST KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT ; TEST FOR SHIFT STATE
2344 JZ K60 ; IF NOT SHIFT, CONVERT LOWER TO UPPER

2345 -------- CONVERT ANY UPPER CASE TO LOWER CASE
2346 EBD6 2347 K60: CMP AL, 'A' ; FIND OUT IF ALPHABETIC
2348 JB K61 ; NOT_CAPS_STATE
2349
2350 -------- CONVERT ANY LOWER CASE TO UPPER CASE
2353 EBD9 2354 K63: SUB AL, 'A'-'A' ; CONVERT TO UPPER CASE
2355 JMP SHORT K61 ; NOT_CAPS_STATE
2356
2357 -------- CONVERT ANY LOWER CASE TO UPPER CASE
2358 EBDB 2359 K66: MOV AL, O ; ZERO ASCII CODE
2360 JMP K57 ; PUT IT INTO THE BUFFER
2361 KB_INT ENDP

2362 -------- TRANSLATE SCAN FOR PSEUDO SCAN CODES
2364 EC10 2365 K63: MOV AL, 'A' ; FIND OUT IF ALPHABETIC
2366 JB K61 ; NOT_CAPS_STATE
2367
2368 -------- TRANSLATE SCAN FOR PSEUDO SCAN CODES
2369 EC1C 2370 K64: MOV AL, 'A' ; CONVERT ORIGIN TO FUNCTION KEYS
2371
2372 -------- TRANSLATE SCAN FOR PSEUDO SCAN CODES
2375 EC1F 2376 K65: MOV AL, 'A' ; CONVERT TO LOWER CASE
2377
2378 -------- TRANSLATE SCAN FOR PSEUDO SCAN CODES
2379 EC26 2380 K66: MOV AX,AL ; PUT VALUE INTO AN
2381
2382 -------- TRANSLATE SCAN FOR PSEUDO SCAN CODES
2385 EC2A 2386 K67: MOV BX,BUFFER_TAIL ; HAS THE BUFFER WRAPPED AROUND
2387
2388 -------- TRANSLATE SCAN FOR PSEUDO SCAN CODES
238B EC2E 238C K68: MOV BX,BUFFER_HEAD ; GET THE END POINTER TO THE BUFFER
238D
238E -------- TRANSLATE SCAN FOR PSEUDO SCAN CODES
2391 EC33 2392 K69: MOV AX,BUFFER_TAIL.EX ; MOVE THE POINTER UP
2393
2394 -------- TRANSLATE SCAN FOR PSEUDO SCAN CODES
2397 EC3E 2398 K6A: MOV AX,AX ; STORE THE VALUE
2399
2400 -------- TRANSLATE SCAN FOR PSEUDO SCAN CODES
2403 EC43 2404 K6B: MOV CX,48H ; SET UP COUNT
2405
2406 -------- TRANSLATE SCAN FOR PSEUDO SCAN CODES
240B EC45 240C K6C: MOV CX,48H ; PUT IT INTO THE BUFFER
240D
240E -------- TRANSLATE SCAN FOR PSEUDO SCAN CODES
2412 EC4D 2413 K6D: MOV CX,CX ; PUT THE VALUE INTO AN
2414
2415 -------- TRANSLATE SCAN FOR PSEUDO SCAN CODES
2418 EC52 2419 K6E: MOV BX,BUFFER_TAIL ; HAS THE BUFFER WRAPPED AROUND
2420
2421 -------- TRANSLATE SCAN FOR PSEUDO SCAN CODES
2424 EC57 2425 K6F: MOV BX,BUFFER_HEAD ; GET THE END POINTER TO THE BUFFER
2426
2427 -------- TRANSLATE SCAN FOR PSEUDO SCAN CODES
242C EC62 242D K70: MOV AX,AX ; STORE THE VALUE
242E
242F -------- TRANSLATE SCAN FOR PSEUDO SCAN CODES
2434 EC6F 2435 K71: MOV AX,AX ; PUT THE VALUE INTO AN
2436
2437 -------- BUFFER IS FULL, SOUND THE DEEPER
2442 EC77 2443 K62: MOV AL,EOI ; BUFFER-FULL-BEEP
2444
2445 -------- BUFFER IS FULL, SOUND THE DEEPER
244A EC7D 244B K63: MOV AX,AL ; END OF INTERRUPT COMMAND
244C
244D -------- BUFFER IS FULL, SOUND THE DEEPER
2451 EC82 2452 K64: MOV BX,EOH ; SEND COMMAND TO INT CONTROL PORT
2453
2454 -------- BUFFER IS FULL, SOUND THE DEEPER
2459 EC87 245A K65: MOV AH,80H ; NUMBER OF CYCLES FOR 1/12 SECOND TONE
245B
245C -------- BUFFER IS FULL, SOUND THE DEEPER
2461 EC92 2462 K66: MOV BX,EOH ; BEEP-CYCLE
2463
2464 -------- BUFFER IS FULL, SOUND THE DEEPER
2469 EC97 246A K67: MOV AX,AL ; TURN OFF TIMER GATE AND SPEAKER DATA
246B
246C -------- BUFFER IS FULL, SOUND THE DEEPER
2471 EC9C 2472 K68: MOV AX,AL ; OUTPUT TO CONTROL
2473
2474 -------- BUFFER IS FULL, SOUND THE DEEPER
2479 ECAC 247A K69: MOV AL,CX ; HALF CYCLE TIME FOR TONE
247B
247C -------- BUFFER IS FULL, SOUND THE DEEPER
2481 ECD6 2482 K6A: MOV BX,CX ; SET UP COUNT
2483
2484 -------- BUFFER IS FULL, SOUND THE DEEPER
2489 ECEB 248C K6B: MOV AX,CX ; SPEAKER OFF
248D
248E -------- BUFFER IS FULL, SOUND THE DEEPER
2493 ECF1 2494 K6C: MOV AX,AL ; TURN ON SPEAKER BIT
2495
2496 -------- BUFFER IS FULL, SOUND THE DEEPER
249B ECF6 249C K6D: MOV AX,AL ; OUTPUT TO CONTROL
249D
249E -------- BUFFER IS FULL, SOUND THE DEEPER
24A3 ECFB 24A4 K6E: MOV AX,AL ; SET UP COUNT
24A5
24A6 -------- BUFFER IS FULL, SOUND THE DEEPER
24B1 ECFC 24B2 K6F: MOV AX,AL ; TOTAL TIME COUNT
24B3
24B4 
24B5 
24B6 
24B7 
24B8 
24B9 
24BA 
24BB 
24BC 
24BD 
24BE 
24BF 
24C0 
24C1 
24C2 
24C3 
24C4 
24C5 
24C6 
24C7 
24C8 
24C9 
24CA 
24CB 
24CC 
24CD 
24CE 
24CF 
24D0 
24D1 
24D2 
24D3 
24D4 
24D5 
24D6 
24D7 
24D8 
24D9 
24DA 
24DB 
24DC 
24DD 
24DE 
24DF 
24E0 
24E1 
24E2 
24E3 
24E4 
24E5 
24E6 
24E7 
24E8 
24E9 
24EA 
24EB 
24EC 
24ED 
24EE 
24EF 
24F0 
24F1 
24F2 
24F3 
24F4 
24F5 
24F6 
24F7 
24F8 
24F9 
24FA 
24FB 
24FC 
24FD 
24FE 
24FF 

System BIOS A-35
INT 13 --------------------------------------------------------------

[Listing of assembly code and comments]

A-36 System BIOS
...
LOC OBJ  LINE  SOURCE

ECE4 FB  2561  STI  ; REENABLE THE INTERRUPTS
ECE5 E02A02  2562  CALL CHK_STAT_E  ; DO SENSE INTERRUPT STATUS
ECE6 A04200  2563  ; FOLLOWING RESET
ECEB 30C0  2564  MOV AL,HEC_STATUS  ; IGNORE ERROR RETURN AND DO OWN TEST
ECEF 7406  2565  CMP AL,HCON  ; TEST FOR DRIVE READY TRANSITION
ECEF 006410020  2566  JZ  J7  ; EVERYTHING OK
ECEF  E006410020  2567  OR DISKETTE_STATUS,BAD_SEC  ; SET ERROR CODE
EFC4 C3  2568  RET
EFC4 F7  2569  ;------- SEND SPECIFY COMMAND TO DISKETTE FORMAT
EFC5  2570  J7:  ; DRIVE READY
EFC5 B043  2571  MOV AH,03H  ; SPECIFY COMMAND
EFC7 E94701  2572  CALL NEC_OUTPUT  ; OUTPUT THE COMMAND
EFC8 B00100  2573  MOV BX,1  ; FIRST BYTE PARM IN BLOCK
EFC9 E60C01  2574  CALL GET_PARM  ; TO THE NEC CONTROLLER
ED00 B00300  2575  MOV BX,3  ; SECOND BYTE PARM IN BLOCK
ED01 E66401  2576  CALL GET_PARM  ; TO THE NEC CONTROLLER
ED06  2577  J8:  ; RESET RET
ED06 C3  2578  RET  ; RETURN TO CALLER
ED07 A04100  2579  ;------- DISKETTE STATUS ROUTINE
ED07  2580  DISK_STATUS PROC NEAR
ED08 A04100  2581  MOV AL,DISKETTE_STATUS
ED0A C3  2582  RET
ED0B  2583  DISK_STATUS ENDP
ED0D B046  2584  ;------- DISKETTE READ
ED0D  2585  DISK_READ PROC NEAR
ED0E  2586  MOV AL,O46H  ; READ COMMAND FOR DMA
ED0F B00801  2587  J9:  ; DISK_READ_CONT
ED10 B466  2588  CALL DMA_SETUP  ; SET UP THE DMA
ED11 E836  2589  MOV AH,O6EH  ; SET UP RD COMMAND FOR NEC CONTROLLER
ED12 E836  2590  JMP SHORT RM_OPM  ; GO DO THE OPERATION
ED13  2591  DISK_READ ENDP
ED14 B042  2592  ;------- DISKETTE VERIFY
ED14  2593  DISK_VERF PROC NEAR
ED16 E8F5  2594  MOV AL,O42H  ; VERIFY COMMAND FOR DMA
ED16  2595  JMP J9  ; DO AS IF DISK_READ
ED17  2596  DISK_VERF ENDP
ED1E  2597  ;------- DISKETTE FORMAT
ED1E  2598  DISK_FORMAT PROC NEAR
ED1E  2599  OR MOTOR_STATUS,B0H  ; INDICATE WRITE OPERATION
ED1F B044  2600  MOV AL,O44H  ; WILL WRITE TO THE DISKETTE
ED1F E86601  2601  CALL DMA_SETUP  ; SET UP THE DMA
ED22 B040  2602  MOV AH,O40H  ; ESTABLISH THE FORMAT COMMAND
ED24 E824  2603  JMP SHORT RM_OPM  ; DO THE OPERATION
ED25  2604  J10:  ; CONTINUATION OF RM_OPM FOR FMT
ED26 B00700  2605  MOV BX,7  ; GET THE
ED27 E84001  2606  CALL GET_PARM  ; BYTES/SECTOR VALUE TO NEC
ED2C B00900  2607  MOV BX,9  ; GET THE
ED2E E83401  2608  CALL GET_PARM  ; SECTORS/TRACK VALUE TO NEC
ED32 B00F00  2609  MOV BX,15  ; GET THE
ED33 E83401  2610  CALL GET_PARM  ; GAP LENGTH VALUE TO NEC
ED38 B01100  2611  MOV BX,17  ; GET THE FILLER BYTE
ED3B E9A000  2612  JMP J16  ; TO THE CONTROLLER
ED3C  2613  DISK_FORMAT ENDP
ED3E  2614  ;------- DISKETTE WRITE ROUTINE
ED3E  2615  DISK_WRITE PROC NEAR
ED3E  2616  OR MOTOR_STATUS,B0H  ; INDICATE WRITE OPERATION
ED3F B043  2617  MOV AL,O43H  ; DMA WRITE COMMAND
ED44 E86001  2618  CALL DMA_SETUP  ; NEC COMMAND TO WRITE TO DISKETTE
ED46 B4C5  2619  MOV AH,0C5H  ; DISK_WRITE ENDP
ED47  2620  ;------- ALLOW WRITE ROUTINE TO FALL INTO RM_OPM
ED48  2621  ;-------------------------------------------------------------------

A-38 System BIOS
638 ; RH_OPN
639 ; THIS ROUTINE PERFORMS THE READ/WRITE/VERIFY OPERATION :
640 -------------------------------------------------------------
641 ED4A
642 RH_OPN PROC NEAR
643
644 ED4A 7308
645 JNC J11 ; TEST FOR DMA ERROR
646
647 ED4C C66410009
648 MOV DISKETTE_STATUS,DMA_BOUNDARY ; SET ERROR
649
650 ED51 B000
651 MOV AL,0 ; NO SECTORS TRANSFERRED
652
653 ED53 C3
654 RET ; RETURN TO MAIN ROUTINE
655
656 ED54
657 J11: ; DO_RH_OPN
658
659 ED54 59
660 PUSH AX ; SAVE THE COMMAND
661
662 ED64 69
663 1----- ON THE MOTOR AND SELECT THE DRIVE
664
665 ED55 51
666 PUSH CX ; SAVE THE T/S PARAMS
667
668 ED55 6ACA
669 MOV CL,DL ; GET DRIVE NUMBER AS SHIFT COUNT
670
671 ED55 8001
672 MOV AL,1 ; MASK FOR DETERMINING MOTOR BIT
673
674 ED5A D2E0
675 SAL AL,CL ; SHIFT THE MASK BIT
676
677 ED5C FA
678 CLI ; NO INTERRUPTS WHILE DETERMINING MOTOR STATUS
679
680 ED5D C6640000FF
681 MOV MOTOR_COUNT,OFFSET ; SET LARGE COUNT DURING OPERATION
682
683 ED62 84063F00
684 TEST AL,MOTOR_STATUS ; TEST THAT MOTOR FOR OPERATING
685
686 ED64 7331
687 JNC J14 ; IF RUNNING, SKIP THE WAIT
688
689 ED66 80263F00F0
690 AND MOTOR_STATUS,OFFSET ; TURN OFF ALL MOTOR BITS
691
692 ED6F 00643F00
693 OR MOTOR_STATUS,AL ; TURN ON THE CURRENT MOTOR
694
695 ED71 FB
696 STI ; INTERRUPTS BACK ON
697
698 ED72 8010
699 MOV AL,10H ; MASK BIT
700
701 ED76 D2E0
702 SAL AL,CL ; DEVELOP BIT MASK FOR MOTOR ENABLE
703
704 ED76 8ACA
705 OR AL,DL ; GET DRIVE SELECT BITS IN
706
707 ED77 6C0C
708 OR AL,CH ; NO RESET, ENABLE DMA/INT
709
710 ED7A 52
711 PUSH DX ; SAVE REG
712
713 ED7B BAF203
714 MOV DX,03F2H ; CONTROL PORT ADDRESS
715
716 ED7F EE
717 OUT DX,AL ; RECOVER REGISTERS
718
719 ED7F 5A
720 POP DX
721
722 ED60 F6643F0000
723 TEST MOTOR_STATUS,00H ; IS THIS A WRITE
724
725 ED65 7412
726 JZ J14 ; NO, CONTINUE WITHOUT WAIT
727
728 ED67 801400
729 MOV BX,20 ; GET THE MOTOR WAIT
730
731 ED6A E0F0
732 CALL GET_PARM ; PARAMETER
733
734 ED6D 0A44
735 OR AH,AL ; TEST FOR NO WAIT
736
737 ED6F J12: ; TEST_WAIT_TIME
738
739 ED6F 7408
740 JZ J14 ; EXIT WITH TIME EXPIRED
741
742 ED91 2BC9
743 SUB CK,CX ; SET UP 1/8 SECOND LOOP TIME
744
745 ED93
746 J13: ;
747
748 ED93 E2F3
749 LOOP J13 ; WAIT FOR THE REQUIRED TIME
750
751 ED95 F2CC
752 DEC AH ; DECREMENT TIME VALUE
753
754 ED97 EB66
755 JMP J12 ; ARE WE DONE YET
756
757 ED99 2664
758 MOV SI,OFFSET J17 ; DUMMY RETURN ON STACK FOR NEC_OUTPUT
759
760 ED9A 2662
761 POP AX
762
763 ED9A 59
764 POP CX
765
766 ED9A
767 --- DO THE SEEK OPERATION
768
769 ED9B E0F0
770 CALL SEEK ; MOVE TO CORRECT TRACK
771
772 ED9C 56
773 POP AX ; RECOVER COMMAND
774
775 ED9F 8AFC
776 MOV BM,AH ; SAVE COMMAND IN BM
777
778 ED91 B400
779 MOV DX,0 ; SET NO SECTORS READ IN CASE OF ERROR
780
781 ED93 724B
782 JC J17 ; IF ERROR, THEN EXIT AFTER MOTOR OFF
783
784 ED95 B5F0ED90
785 MOV SI.OFFSET J17 ; DUMMY RETURN ON STACK FOR NEC_OUTPUT
786
787 ED9A 2690
788 PUSH SI
789
790 ED9A 2699
791 --- SEND OUT THE PARAMETERS TO THE CONTROLLER
792
793 ED9A EA9400
794 CALL NEC_OUTPUT ; OUTPUT THE OPERATION COMMAND
795
796 ED9A 846601
797 MOV AH,(BP+1) ; GET THE CURRENT HEAD NUMBER
798
799 ED9B D0E4
800 MOV AH,1 ; MOVE IT TO BIT 2
801
802 ED9B D0E4
803 MOV AH,1 ; MOVE IT TO BIT 2
804
805 ED9B E0E404
806 MOV AH,4 ; ISOLATE THAT BIT
807
808 ED9B 0AE2
809 OR AH,DL ; OR IN THE DRIVE NUMBER
810
811 ED9B 8E5500
812 CALL NEC_OUTPUT ;
813
814 ED9A 2700
815 --- TEST FOR FORMAT COMMAND
816
817 ED9A 2711
818 --- SEND OUT THE PARAMETERS TO THE CONTROLLER
819
81A ED9A 00F404
81B CHP BH,04DH ; IS THIS A FORMAT OPERATION
81C
81D ED9A 7503
81E JNE J15 ; NO, CONTINUE WITH R/A/V
81F
820 ;
821 System BIOS
822 A-39
LOC OBJ  
LINE  
SOURCE

ECD1 E962FF  
2715  JMP  J10  
2716  J15:  MOV  AH,CH  
2717  CALL  NEC_OUTPUT  
2718  MOV  AH,CH  
2719  CALL  NEC_OUTPUT  
2720  MOV  AH,CH  
2721  MOV  AH,CH  
2722  MOV  BX,7  
2723  MOV  BX,7  
2724  CALL  GET_PARAM  
2725  MOV  BX,9  
2726  CALL  GET_PARAM  
2727  MOV  BX,11  
2728  CALL  GET_PARAM  
2729  MOV  BX,13  
2730  CALL  GET_PARAM  
2731  MOV  BX,40  
2732  MOV  BX,40  
2733  MOV  BX,40  
2734  MOV  BX,40  
2735  MOV  BX,40  
2736  MOV  BX,40  
2737  MOV  BX,40  
2738  MOV  BX,40  
2739  MOV  BX,40  
2740  MOV  BX,40  
2741  MOV  BX,40  
2742  MOV  BX,40  
2743  MOV  BX,40  
2744  MOV  BX,40  
2745  MOV  BX,40  
2746  MOV  BX,40  
2747  MOV  BX,40  
2748  MOV  BX,40  
2749  MOV  BX,40  
2750  MOV  BX,40  
2751  MOV  BX,40  
2752  MOV  BX,40  
2753  MOV  BX,40  
2754  MOV  BX,40  
2755  MOV  BX,40  
2756  MOV  BX,40  
2757  MOV  BX,40  
2758  MOV  BX,40  
2759  MOV  BX,40  
2760  MOV  BX,40  
2761  MOV  BX,40  
2762  MOV  BX,40  
2763  MOV  BX,40  
2764  MOV  BX,40  
2765  MOV  BX,40  
2766  MOV  BX,40  
2767  MOV  BX,40  
2768  MOV  BX,40  
2769  MOV  BX,40  
2770  MOV  BX,40  
2771  MOV  BX,40  
2772  MOV  BX,40  
2773  MOV  BX,40  
2774  MOV  BX,40  
2775  MOV  BX,40  
2776  MOV  BX,40  
2777  MOV  BX,40  
2778  MOV  BX,40  
2779  MOV  BX,40  
2780  MOV  BX,40  
2781  MOV  BX,40  
2782  MOV  BX,40  
2783  MOV  BX,40  
2784  MOV  BX,40  
2785  MOV  BX,40  
2786  MOV  BX,40  
2787  MOV  BX,40  
2788  MOV  BX,40  
2789  MOV  BX,40  
2790  MOV  BX,40  
2791  MOV  BX,40  

A-40  System BIOS
This document contains assembly code for handling diskette status and controller operations, likely related to System BIOS, as indicated by the System BIOS A-41 text on the right.
DRIVE IS ADDRESS TO READ!WRITE

LOC OBJ  LINE  SOURCE

EE79 1F  2069  POP DS        ; RESTORE SEGMENT

EE7A 70CS  2070  ASSUME DS:DATA

EE7C C3  2071  JC NEC_OUTPUT  ; IF FLAG SET, OUTPUT TO CONTROLLER

EE7D 2087  SEEK  PROC NEAR

EE7D B001  2088  MOV AL,1      ; ESTABLISH MASK FOR RECAL TEST

EE7F 51  2089  PUSH CX        ; SAVE INPUT VALUES

EE80 8ACA  2090  MOV CL,DL     ; GET DRIVE VALUE INTO CL

EE82 02C0  2091  ROL AL,CL     ; SHIFT IT BY THE DRIVE VALUE

EE84 59  2092  POP CX        ; RECOVER TRACK VALUE

EE85 04063E00  2093  TEST AL,SEEK_STATUS  ; TEST FOR RECAL REQUIRED

EE87 7515  2094  JNZ J28      ; NO_RECAL

EE89 00063E00  2095  OR AL,SEEK_STATUS_AL  ; TURN ON THE NO RECAL BIT IN FLAG

EE8A 0407  2096  MOV AH,06H  ; RECALIBRATE COMMAND

EE91 E04FF  2097  CALL NEC_OUTPUT  ; OUTPUT THE DRIVE NUMBER

EE94 B0E2  2098  MOV AH,DL

EE96 E04FF  2099  CALL NEC_OUTPUT  ; OUTPUT THE DRIVE NUMBER

EE9C 7229  2900  CALL CHK_STAT_2  ; GET THE INTERRUPT AND SENSE INT STATUS

EE9D 20A1  2901  JC J32  ; SEEK_ERROR

EE9E 20B5  2902  JZ J31  ; DRIVE IS IN SYNCH WITH CONTROLLER, SEEK TO TRACK

EE9F B04F  2903  MOV AH,0FH  ; SEEK COMMAND TO NEC

EEA0 E09EFF  2904  CALL NEC_OUTPUT  ; DRIVE NUMBER

EEA3 BAE2  2905  MOV AH,DL

EEA5 E09FF  2906  CALL NEC_OUTPUT  ; TRACK NUMBER

EEA8 BAE5  2907  MOV AH,CN

EEAA E094FF  2908  CALL NEC_OUTPUT

EEAD E06200  2909  CALL CHK_STAT_2  ; GET ENDING INTERRUPT AND

EEB0 9C  2910  JC J31  ; SENSE STATUS

EEB1 0B1200  2911  PUSHF

EEB4 E08FF  2912  CALL GET_PARM  ; GET HEAD SETTLE PARAMETER

EEB7 51  2913  J29:  ; HEAD_SETTLE

EEB8 092602  2914  MOV CX,550  ; 1 MS LOOP

EEB0 04E4  2915  OR AH,AH  ; TEST FOR TIME EXPIRED

EEB2 7406  2916  JZ J31

EEB5 J30:  2917  LOOP J30  ; DELAY FOR 1 MS

EEC1 FECC  2918  DEC AH  ; DECREMENT THE COUNT

EEC3 68F3  2919  JMP J29  ; DO IT SOME MORE

EEC5 94E9  2920  JZ J31

EEC7 59  2921  POP CX  ; RECOVER STATE

EEC9 90  2922  POPF

EEC7 C3  2923  RET  ; SEEK_ERROR

EEC9 2933  SEEK ENDP  ; RETURN TO CALLER

EECA 2935  ; DMA_SETUP

EEC8 S1  2936  DMA_SETUP  ; PROC NEAR

EEC9 94E9  2937  PUSH CX  ; SAVE THE REGISTER

A-42  System BIOS
DETERMINE COUNT

ERROR IS

SOURCE

LOC OBJ

LINE

CL1

OUT DMA+12,AL

1 NO MORE INTERRUPTS

SET THE FIRST/LAST F/F

OUT DMA+11,AL

OUTPUT THE MODE BYTE

GET THE ES VALUE

SHIFT COUNT

ROTATE LEFT

GET HIGHEST NYBLE OF ES TO CH

ZERO THE LOW NYBLE FROM SEGMENT

TEST FOR CARRY FROM ADDITION

CARRY MEANS HIGH 4 BITS MUST BE INC

SAVE START ADDRESS

OUTPUT LOW ADDRESS

OUTPUT HIGH ADDRESS

GET HIGH 4 BITS

OUTPUT THE HIGH 4 BITS TO

THE PAGE REGISTER

----- DETERMINE COUNT -----

NUMBER OF SECTORS

TIMES 256 INTO AX

SECTORS = 128 INTO AX

GET THE BYTES/SECTOR PARM

USE AS SHIFT COUNT (0=128, 1=256 ETC)

MULTIPLY BY CORRECT AMOUNT

-1 FOR DMA VALUE

LOW BYTE OF COUNT

HIGH BYTE OF COUNT

INTERRUPTS BACK ON

RECOVER COUNT VALUE

RECOVER ADDRESS VALUE

ADD TEST FOR 68K OVERFLOW

RECOVER REGISTER

MODE FOR 8237

INITIALIZE THE DISKETTE CHANNEL

RETURN TO CALLER,

CFL SET BY ABOVE IF ERROR

------ DMA_SETUP ENDP ------

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

; CHK_STAT_2

; THIS ROUTINE HANDLES THE INTERRUPT RECEIVED AFTER A

; RECALIBRATE, SEEK, OR RESET TO THE ADAPTER.

; THE INTERRUPT IS WAITED FOR, THE INTERRUPT STATUS SENSED;

; AND THE RESULT RETURNED TO THE CALLER.

; INPUT

; NONE

; OUTPUT

; CY = 0 SUCCESS

; CY = 1 FAILURE -- ERROR IS IN DISKETTE_STATUS

; (AX) DESTROYED

; CHK_STAT_2 PROC NEAR

; CALL WAIT_INT

; WAIT FOR THE INTERRUPT

; IF ERROR, RETURN IT

; SENSE INTERRUPT STATUS COMMAND

; READ IN THE RESULTS

; CHK2_ERROR

; GET THE FIRST STATUS BYTE

; ISOLATE THE BITS

; TEST FOR CORRECT VALUE

; IF ERROR, GO MARK IT

; GOOD RETURN

; RETURN TO CALLER

; CHK2_ERROR

System BIOS A-43
LOC OBJ
LINE SOURCE

EF3C 800E410040 3023 OR DISKETTE_STATUS.BADSEEK I ERROR RETURN CODE
EF31 F9 3024 STC RET
EF32 C3 3025

3026 CHK_STATE_Endp

3027 ; WAIT INT
3028 ; THIS ROUTINE WAITS FOR AN INTERRUPT TO OCCUR. A TIME OUT
3029 ; ROUTINE TAKES PLACE DURING THE WAIT, SO THAT AN ERROR MAY BE
3030 ; RETURNED IF THE DRIVE IS NOT READY.
3031 ; INPUT
3032 ; NONE
3033 ; OUTPUT
3034 ; CY = 0 SUCCESS
3035 ; CY = 1 FAILURE -- DISKETTE_STATUS IS SET ACCORDINGLY
3036 ; (AX) DESTROYED

3037 ;--------------------------------------------------------
3038

3039 WAIT_INT PROC NEAR I TURN ON INTERRUPTS, JUST IN CASE
3040 EF33 FB 3041 STI:
3042 PUSH BX I SAVE REGISTERS
3043 PUSH CX
3044 MOV BL,2 I CLEAR THE COUNTERS
3045 XOR CX,CX I FOR 2 SECOND WAIT
3046 J36:
3047 TEST SEEK_STATUS.INT_FLAG I TEST FOR INTERRUPT OCCURRING
3048 JNZ J37 I COUNT DOWN WHILE WAITING
3049 DEC BL I SECOND LEVEL COUNTER
3050 JNZ J36
3051 OR DISKETTE_STATUS.TIMER_OUT I NOTHING HAPPENED
3052 STC I ERROR RETURN
3053 J37:
3054 PUSHF I RE ENABLE INTERRUPTS
3055 PUSH AX
3056 POPF I RECOVER CARRY
3057 POP CX I RECOVER REGISTERS
3058 POP DX
3059 RET I GOOD RETURN CODE COMES
3060 ; FROM TEST INST

3061 WAIT_INT Endp

3062 ;--------------------------------------------------------
3063 ; DISK_INT
3064 ; THIS ROUTINE HANDLES THE DISKETTE INTERRUPT
3065 ; INPUT
3066 ; NONE
3067 ; OUTPUT
3068 ; THE INTERRUPT FLAG IS SET IS SEEK_STATUS
3069 ;--------------------------------------------------------

3070 ORG 0EFS7H
EF57

3071 DISK_INT PROC FAR I RE ENABLE INTERRUPTS
3072 STI
3073 PUSH DS
3074 PUSH AX
3075 CALL DDS I SYSTEM BIOS
3076 OR SEEK_STATUS.INT_FLAG
3077 MOV AL,20H I END OF INTERRUPT MARKER
3078 OUT 20H,AL I INTERRUPT CONTROL PORT
3079 POP AX
3080 POP DS I RECOVER SYSTEM
3081 IRET I RETURN FROM INTERRUPT

3082 DISK_INT Endp

3083 ;--------------------------------------------------------
3084 ; RESULTS
3085 ; THIS ROUTINE WILL READ ANYTHING THAT THE NEC CONTROLLER HAS
3086 ; TO SAY FOLLOWING AN INTERRUPT.
3087 ; INPUT
3088 ; NONE
3089 ; OUTPUT
3090 ; CY = 0 SUCCESSFUL TRANSFER
3091 ; CY = 1 FAILURE -- TIME OUT IN WAITING FOR STATUS
3092 ; NEC_STATUS AREA HAS STATUS BYTE LOADED INTO IT
3093 ; (AX) DESTROYED
3094 ;--------------------------------------------------------

3095 RESULTS PROC NEAR
3096 ;--------------------------------------------------------
3097 ; DISK_INT
3098 ;--------------------------------------------------------
3099 ; RESULTS PROC NEAR

A-44 System BIOS
EF66 53 3100 PUSH BX  MAX STATUS BYTES
EF70 B307 3101 MOV BL, 7
EF72 33C9 3102 J36: INPUT_LOOP
EF76 BA403S 3105 MOV DX, 03F4H STATUS PORT
EF77 3397 3106 MOV DX, 03FH MAX
EF79 A000 3107 TEST 000H MASTER READY
EF7A 750C 3108 JNZ J40A TEST_DIR
EF7C E2F9 3112 LOOP J39 WAIT_MASTER
EF7E 000E1000 3113 OR DISKETTE_STATUS, TIME_OUT
EF83 J34: RESULTS_ERROR
EF84 F9 3115 STC SET ERROR RETURN
EF84 5B 3117 POP BX
EF84 5A 3118 POP DX
EF86 59 3119 POP CX
EF87 C3 3119 RET
EF88 EC 3119
EF89 A840 3125 TEST AL, 040H TEST_DIR
EF8A FE07 3126 JHZ J40A OK TO READ STATUS
EF8D 800E4100 3128 OR DISKETTE_STATUS, BAD_NEC
EF92 EBF9 3129 JMP J40 RESULTS_ERROR
EF94 47 3137 INC IO INCREMENT THE POINTER
EF95 EC 3139 IN AL, OX GET STATUS
EF96 8805 3140 MOV [OII, AL STORE THE BYTE
EF98 47 3141 INC IO INCREMENT THE POINTER
EF99 890A00 3142 MOV CX,10 LOOP TO KILL TIME FOR NEC
EF9D E2FE 3143 LOOP J43.
EF9F 4A 3145 DEC CX POINT AT STATUS PORT
EF9F EC 3146 IN AL, DX GET STATUS
EFA0 6018 3147 TEST AL, 010H TEST FOR NEC STILL BUSY
EFA2 7046 3149 JZ J45 RESULTS DONE
EFA4 FEC0 3149 JMP J44 CHIP HAS FAILED
EFA6 75CA 3149 JNZ J38 GO BACK FOR MORE
EFA8 AEE3 3149 JMP J41 CHIP HAS FAILED
EFAA J44: }
EFAE 3150 POP BX
EFAE 5A 3150 POP DX
EFAE 59 3150 POP CX
EFAE C3 3150 RET GOOD RETURN CODE FROM TEST INST
EFAE A04500 3154 PROC NEAR
EFAE A04500 3155 THIS ROUTINE CALCULATES THE NUMBER OF SECTORS THAT
EFAE A04500 3156 WERE ACTUALLY TRANSFERRED TO/FROM THE DISKETTE:
EFAE A04500 3157 INPUT:
EFAE A04500 3158 (CH) = CYLINDER OF OPERATION:
EFAE A04500 3159 (CL) = START SECTOR OF OPERATION:
EFAE A04500 3160 OUTPUT:
EFAE A04500 3161 (AL) = NUMBER ACTUALLY TRANSFERRED:
EFAE A04500 3162 NO OTHER REGISTERS MODIFIED:
EFAE A04500 3163 ERROR CODE
EFAE A04500 3164 NEAR
EFAE A04500 3165 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
EFAE A04500 3166 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
;--------- PRINT THE CHARACTER IN (AL) -------

;------- PRINTER STATUS -------

;-------- INITIALIZE THE PRINTER PORT -------

;--- INT 10 ----------------------------------
A-48 System BIOS
RESULTS ONLY FOR CHARACTERS CONTAINED ON THE SAME ROW.

CONTINUATION TO SUCCEEDING LINES WILL NOT PRODUCE

CORRECTLY.

GRAPHICS INTERFACE

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

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

NOTE: FOR THE CURRENT COLOR CARD, THIS ENTRY POINT

HAS MEANING ONLY FOR 310200 GRAPHICS.

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 = CYAN/1/MAGENTA/2/WHITE(3)

IN 40X25 OR 40X25 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.

(DX) = ROW NUMBER.

(AX) = COLUMN NUMBER.

(IF BIT 0 OF AL = 1, THEN THE COLOR VALUE IS

EXCLUSIVE OR'D WITH THE CURRENT CONTENTS OF

THE DOT.

(AH) = 12 WRITE DOT

(AX) = COLUMN NUMBER.

(AX) = 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.

(AX) = COLUMN NUMBER.

(AX) = RETURN THE DOT.

ASCII TELETYPe ROUTINE FOR OUTPUT

(AH) = 14 WRITE TELETYPE TO ACTIVE PAGE

(AX) = 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

(AX) = CURRENT COLOR SET (SEE AH=0 FOR EXPLANATION)

(AX) = NUMBER OF CHARACTER COLUMNS ON SCREEN.

(BX) = CURRENT ACTIVE DISPLAY PAGE.

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

KILL ALL OTHERS DESTROYED


ORG 0F0SH

ORI 0F0SH

TABLE OF ROUTINES WITHIN VIDEO I/O

DW OFFSET SET_MODE

DW OFFSET SET_CTYPE

DW OFFSET SET_CPOS

DW OFFSET READ_CURSOR

DW OFFSET READ_LEN

DW OFFSET ACT_DISP_PAGE

DW OFFSET SCROLL_UP

DW OFFSET SCROLL_DOWN

DW OFFSET READ_AC_CURRENT

DW OFFSET WRITE_AC_CURRENT

DW OFFSET WRITE_C_CURRENT

DW OFFSET SET_COLOR

DW OFFSET WRITE_DOT

DW OFFSET READ_DOT

DW OFFSET WRITE_TTY

DW OFFSET VIDEO_STATE

ORG 0F0SH

VIDEO_ID PROC NEAR

STI

CLD

LDI

PUSH DS

PUSH DS

PUSH 1

PUSH 1

SAVE SEGMENT REGISTERS

INTERRUPTS BACK ON

SET DIRECTION FORWARD

System BIOS A-49
SET_MODE

This routine initializes the attachment to the selected mode. The screen is blanked.

Table for use in setting of mode:

<table>
<thead>
<tr>
<th>Mode</th>
<th>HANDLE</th>
<th>DB</th>
<th>3OH,2OH,2D0H,0AH,1FH,01H</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-9</td>
<td>4</td>
<td>DB</td>
<td>0,0,0</td>
</tr>
<tr>
<td>0-9</td>
<td>5</td>
<td>DB</td>
<td>38H,28H,2D8H,0AH,7FH,64H</td>
</tr>
</tbody>
</table>

Set up for 40x25:

<table>
<thead>
<tr>
<th>Mode</th>
<th>HANDLE</th>
<th>DB</th>
<th>3OH,2OH,2D0H,0AH,1FH,01H</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-9</td>
<td>4</td>
<td>DB</td>
<td>0,0,0</td>
</tr>
<tr>
<td>0-9</td>
<td>5</td>
<td>DB</td>
<td>38H,28H,2D8H,0AH,7FH,64H</td>
</tr>
</tbody>
</table>

Set up for 60x25:

<table>
<thead>
<tr>
<th>Mode</th>
<th>HANDLE</th>
<th>DB</th>
<th>3OH,2OH,2D0H,0AH,1FH,01H</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-9</td>
<td>4</td>
<td>DB</td>
<td>0,0,0</td>
</tr>
<tr>
<td>0-9</td>
<td>5</td>
<td>DB</td>
<td>38H,28H,2D8H,0AH,7FH,64H</td>
</tr>
</tbody>
</table>

Set up for GRAPHICS:

<table>
<thead>
<tr>
<th>Mode</th>
<th>HANDLE</th>
<th>DB</th>
<th>3OH,2OH,2D0H,0AH,1FH,01H</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-9</td>
<td>4</td>
<td>DB</td>
<td>0,0,0</td>
</tr>
<tr>
<td>0-9</td>
<td>5</td>
<td>DB</td>
<td>38H,28H,2D8H,0AH,7FH,64H</td>
</tr>
</tbody>
</table>
SET UP FOR 80X25 B&W CARD

TABLE OF REGEN LENGTHS

TABLE OF MODE SETS

ADDRESS OF COLOR CARD

ADDRESS OF B&W CARD (304)

ADDRESS OF COLOR MODE

ADDRESS OF B&W MODE

SAVE MODE IN AN

SAVE IN GLOBAL VARIABLE

SAVE ADDRESS OF BASE

SAVE POINTER TO DATA SEGMENT

SAVE OUTPUT PORT VALUE
LOC OBJ

F118 63C204 3573 ADD DX,4  
F118 8AC3 3574 MOV AL,01  
F11D 04 3575 OUT DX,AL  
F11E 5A 3576 POP DX  
F11F 2BC0 3577 SUB AX,AX  
F121 AEOB 3578 MOV DS,AX  
F123 C51E7400 3579 ASSUME DS:AX50  
F127 5F 3580 LDS BX,PARM_PTR  
F128 B91000 3581 POP AX  
F12B 6AC3 3582 ASSUME DS:CODE  
F12C 12 3583 MOV CX,MP  
F12E 7210 3584 CJM AH,2  
F130 0309 3585 ADD BX,CX  
F132 08FC04 3586 CMP AH,4  
F135 7209 3587 JC M9  
F137 0309 3588 ADD BX,CX  
F139 08FC07 3589 CMP AH,7  
F13C 720C 3590 JC M9  
F13E 0309 3591 ADD BX,CX  
F143 5953 3592 MOV CX,MP  
F144 8AC4 3593 ADD DS:AX  
F145 EE 3594 loop to correct row of initialization table  
F148 59 3595 M9:  
F14A 50 3596 POP AX  
F14C 32E4 3597 XOR AH,AH  
F14E 82F3 3598 ah will serve as register  
F150 56 3599 number during loop  
F151 1F 3600 loop through table, outputting reg address, then value from table  
F152 33FF 3601 loop  
F154 093E6E00 3602 MOV AL,AH  
F155 C506420000 3603 MOV CX,01  
F15B 090000 3604 MOV AL,IBX1  
F160 08FC04 3605 OUT DX,AL  
F162 720B 3606 INC DX  
F164 FEC4 3607 INC AH  
F169 B907 3608 MOV AL,IBX1  
F16B 08FC04 3609 OUT DX,AL  
F16D 43 3610 INC BX  
F16F 4A 3611 DEC DX  
F174 82F3 3612 LOOP M10  
F176 50 3613 POP AX  
F177 1F 3614 POP DS  
F178 33FF 3615 ASSUME DS:DATA  
F182 33FF 3616 fill regen area with blank  
F185 693E6E00 3617 fill  
F18B 093E6E00 3618 MOV AL,01  
F18C 42 3619 XOR DI,01  
F18F B90000 3620 MOV CRT_START,01  
F190 C506420000 3621 MOV ACTIVE_PAGE,0  
F196 090000 3622 MOV CX,01  
F19B 08FC04 3623 CMP AH,4  
F19F 720B 3624 JC M12  
F19F 08FC07 3625 CMP AH,7  
F1A4 7404 3626 JE M11  
F1A6 3C00 3627 XOR AX,AH  
F1A5 E605 3628 JMP SHORT M13  
F1A7 7404 3629 M11:  
F1AA B508 3630 MOV CH,08H  
F1AD 32E4 3631 M12:  
F1AF 04 3632 MOV AX,'874256  
F1B0 B02007 3633 M13:  
F1B3 7E 3634 REP STOSH  
F1B5 AD 3635 fill regen buffer with blanks  
F1B7 04 3636 enable video and correct port setting  
F1B9 7C706400000706 3637 enable  
F1BA 049000 3638 MOV AL,CURSOR_MODE  
F1BB 32E4 3639 MOV AL,CRT_MODE  
F1BC 40 3640 XOR AX,AX  
F1BD 80F0 3641 MOV SI,AX  
F1BF 00166300 3642 MOV DX,ADDR_6645  
F1C3 00C504 3643 PREPARE TO OUTPUT TO  
F1C5 44 3644 ADD DX,4  
F1C7 8E8A4040F0 3645 ADD AL,CS:[SI+OFFSET M7]  
F1C8 EE 3646 OUT DX,AL  
F1C9 A26500 3647 MOV CRT_MODE_SET,AL  
F1CA 8E 3648 save that value  

A-52 System BIOS
<table>
<thead>
<tr>
<th>LOC OBJ</th>
<th>LINE</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>3649</td>
<td>i---- DETERMINE NUMBER OF COLUMNS, BOTH FOR ENTIRE DISPLAY</td>
<td></td>
</tr>
<tr>
<td>3650</td>
<td>i---- AND THE NUMBER TO BE USED FOR TTY INTERFACE</td>
<td></td>
</tr>
<tr>
<td>3651</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3652</td>
<td>MOV</td>
<td>AL,CS:[SI + OFFSET M61]</td>
</tr>
<tr>
<td>3653</td>
<td>XOR</td>
<td>AX,AX</td>
</tr>
<tr>
<td>3654</td>
<td>MOV</td>
<td>CRt_COLS,AX</td>
</tr>
<tr>
<td>3655</td>
<td>; NUMBER OF COLUMNS IN THIS SCREEN</td>
<td></td>
</tr>
<tr>
<td>3656</td>
<td>i---- SET CURSOR POSITIONS</td>
<td></td>
</tr>
<tr>
<td>3657</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3658</td>
<td>AND</td>
<td>SI,0EH</td>
</tr>
<tr>
<td>3659</td>
<td>MOV</td>
<td>CX,CS:[SI + OFFSET M51]</td>
</tr>
<tr>
<td>3660</td>
<td>MOV</td>
<td>CRt_LEN,CX</td>
</tr>
<tr>
<td>3661</td>
<td>MOV</td>
<td>CX,8</td>
</tr>
<tr>
<td>3662</td>
<td>MOV</td>
<td>DI,OFFSET_CURSOR_MXN</td>
</tr>
<tr>
<td>3663</td>
<td>PUSH</td>
<td>DS</td>
</tr>
<tr>
<td>3664</td>
<td>POP</td>
<td>ES</td>
</tr>
<tr>
<td>3665</td>
<td>XOR</td>
<td>AX,AX</td>
</tr>
<tr>
<td>3666</td>
<td>REP</td>
<td>STOSW</td>
</tr>
<tr>
<td>3667</td>
<td></td>
<td>FILL WITH ZEROES</td>
</tr>
<tr>
<td>3668</td>
<td>i---- SET UP OVERSCAN REGISTER</td>
<td></td>
</tr>
<tr>
<td>3669</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3670</td>
<td>INC</td>
<td>DX</td>
</tr>
<tr>
<td>3671</td>
<td>MOV</td>
<td>AL,30H</td>
</tr>
<tr>
<td>3672</td>
<td>MOV</td>
<td>AL,30H</td>
</tr>
<tr>
<td>3673</td>
<td>CMP</td>
<td>CRt_MODE,A</td>
</tr>
<tr>
<td>3674</td>
<td>JNZ</td>
<td>M14</td>
</tr>
<tr>
<td>3675</td>
<td>MOV</td>
<td>AL,3FH</td>
</tr>
<tr>
<td>3676</td>
<td>MOV</td>
<td>AL,3FH</td>
</tr>
<tr>
<td>3677</td>
<td>MOV</td>
<td>AL,3FH</td>
</tr>
<tr>
<td>3678</td>
<td>MOV</td>
<td>AL,3FH</td>
</tr>
<tr>
<td>3679</td>
<td>MOV</td>
<td>AL,3FH</td>
</tr>
<tr>
<td>3680</td>
<td>i---- NORMAL RETURN FROM ALL VIDEO RETURNS</td>
<td></td>
</tr>
<tr>
<td>3681</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3682</td>
<td>VIDEO_RETURN</td>
<td></td>
</tr>
<tr>
<td>3683</td>
<td>POP</td>
<td>DI</td>
</tr>
<tr>
<td>3684</td>
<td>POP</td>
<td>SI</td>
</tr>
<tr>
<td>3685</td>
<td>POP</td>
<td>BX</td>
</tr>
<tr>
<td>3686</td>
<td>M1S</td>
<td></td>
</tr>
<tr>
<td>3687</td>
<td>MOV</td>
<td>CX</td>
</tr>
<tr>
<td>3688</td>
<td>MOV</td>
<td>AX</td>
</tr>
<tr>
<td>3689</td>
<td>MOV</td>
<td>DS</td>
</tr>
<tr>
<td>3690</td>
<td>MOV</td>
<td>ES</td>
</tr>
<tr>
<td>3691</td>
<td>INT</td>
<td></td>
</tr>
<tr>
<td>3692</td>
<td>SET_MODE</td>
<td>ENDP</td>
</tr>
<tr>
<td>3693</td>
<td>i----</td>
<td></td>
</tr>
<tr>
<td>3694</td>
<td>i SET_CTYPE</td>
<td></td>
</tr>
<tr>
<td>3695</td>
<td>THIS ROUTINE SETS THE CURSOR VALUE</td>
<td></td>
</tr>
<tr>
<td>3696</td>
<td>i INPUT</td>
<td></td>
</tr>
<tr>
<td>3697</td>
<td>(CX) HAS CURSOR VALUE CH-START LINE, CL-STOP LINE</td>
<td></td>
</tr>
<tr>
<td>3698</td>
<td>i OUTPUT</td>
<td></td>
</tr>
<tr>
<td>3699</td>
<td>i NONE</td>
<td></td>
</tr>
<tr>
<td>3700</td>
<td>i----</td>
<td></td>
</tr>
<tr>
<td>3701</td>
<td>FICD</td>
<td></td>
</tr>
<tr>
<td>3702</td>
<td>SET_CTYPE</td>
<td>PROC HEAR</td>
</tr>
<tr>
<td>3703</td>
<td>FICD B40A</td>
<td></td>
</tr>
<tr>
<td>3704</td>
<td>FICF E00200</td>
<td></td>
</tr>
<tr>
<td>3705</td>
<td>FID0 EED</td>
<td></td>
</tr>
<tr>
<td>3706</td>
<td>FID0</td>
<td></td>
</tr>
<tr>
<td>3707</td>
<td>i----</td>
<td></td>
</tr>
<tr>
<td>3708</td>
<td>i----</td>
<td></td>
</tr>
<tr>
<td>3709</td>
<td>M16:</td>
<td></td>
</tr>
<tr>
<td>3710</td>
<td>MOV</td>
<td>AX,ADDR_6045</td>
</tr>
<tr>
<td>3711</td>
<td>MOV</td>
<td>AL,AX</td>
</tr>
<tr>
<td>3712</td>
<td>MOV</td>
<td>AL,AX</td>
</tr>
<tr>
<td>3713</td>
<td>MOV</td>
<td>AL,AX</td>
</tr>
<tr>
<td>3714</td>
<td>MOV</td>
<td>AL,AX</td>
</tr>
<tr>
<td>3715</td>
<td>MOV</td>
<td>AL,AX</td>
</tr>
<tr>
<td>3716</td>
<td>DEC</td>
<td>DX</td>
</tr>
<tr>
<td>3717</td>
<td>MOV</td>
<td>AL,AX</td>
</tr>
<tr>
<td>3718</td>
<td>MOV</td>
<td>AL,AX</td>
</tr>
<tr>
<td>3719</td>
<td>MOV</td>
<td>AL,AX</td>
</tr>
<tr>
<td>3720</td>
<td>MOV</td>
<td>AL,AX</td>
</tr>
<tr>
<td>3721</td>
<td>MOV</td>
<td>AL,AX</td>
</tr>
<tr>
<td>3722</td>
<td>MOV</td>
<td>AL,AX</td>
</tr>
<tr>
<td>3723</td>
<td>MOV</td>
<td>AL,AX</td>
</tr>
<tr>
<td>3724</td>
<td>SET_CTYPE</td>
<td>ENDP</td>
</tr>
</tbody>
</table>

Appendix A

System BIOS A-53
A-54 System BIOS
LOC OBJ
F23E 885750 3602  MOV  DX,(BL+OFFSET CURSOR_POSH)
F242 8806E00 3603  MOV  CX,CURSOR_MODE
F246 6F 3604  POP  SI
F247 5E 3605  POP  DI
F248 5B 3606  POP  BX
F249 58 3607  POP  AX  ; DISCARD SAVED CX AND DX
F24A 5B 3608  POP  AX
F24C 07 3609  POP  DS
F24D CF 3610  POP  ES
3611  IN    ET
3612  READ_CURSOR ENDP
3613
3614  ; SET_COLOR
3615  ; THIS ROUTINE WILL ESTABLISH THE BACKGROUND COLOR; THE OVERSCAN
3616  ; COLOR, AND THE FOREGROUND COLOR SET FOR MEDIUM RESOLUTION
3617  ; GRAPHICS
3618  ; INPUT
3619  ; (BH) HAS COLOR ID
3620  ; IF BH=0, THE BACKGROUND COLOR VALUE IS SET
3621  ; FROM THE LOW BITS OF BL (0-31)
3622  ; IF BH=1, THE PALETTE SELECTION IS MADE
3623  ; BASED ON THE LOW BIT OF BL
3624  ; 0=GREEN, RED, YELLOW FOR COLORS 1,2,3
3625  ; 1=BLUE, CYAN, MAGENTA FOR COLORS 1,2,3
3626  ; (BL) HAS THE COLOR VALUE TO BE USED
3627  ; OUTPUT
3628  ; THE COLOR SELECTION IS UPDATED
3629
3630  ; SET_COLOR  PROC  NEAR
3631  ; I/O PORT FOR PALETTE
3632  ; OVERSCAN PORT
3633  ; MOV  AL,CRT_PALETTE
3634  ; OR  BL,BH
3635  ; IS THIS COLOR 0?
3636
3637  ;----- HANDLE COLOR 0 BY SETTING THE BACKGROUND COLOR
3638
3639  ; AND  AL,OEOH
3640  ; TURN OFF LOW 5 BITS OF CURRENT
3641  ; COLOR
3642  ; OR  AL,BL
3643  ; PUT VALUE INTO REGISTER
3644  ; OUT  DX,AL
3645  ; OUTPUT COLOR SELECTION TO I/O PORT
3646
3647  ;----- HANDLE COLOR 1 BY SELECTING THE PALETTE TO BE USED
3648
3649  ; M10:
3650  ; AND  AL,0DFH
3651  ; TURN OFF PALETTE SELECT BIT
3652  ; SHR  BL,1
3653  ; TEST THE LOW ORDER BIT OF BL
3654  ; JNC  M19
3655  ; ALREADY DONE
3656  ; OR  AL,BL
3657  ; TURN ON PALETTE SELECT BIT
3658  ; JMP  M19
3659  ; GO DO IT
3660
3661  ; SET_COLOR ENDP
3662
3663  ; VIDEO_STATE  PROC  NEAR
3664  ; GET NUMBER OF COLUMNS
3665  ; CURRENT MODE
3666  ; GET CURRENT ACTIVE PAGE
3667  ; RECOVER REGISTERS
3668  ; DISCARD SAVED BX
3669  ; RETURN TO CALLER
3670
3671  ; VIDEO_STATE ENDP
3672
3673  ; POSITION
3674  ; THIS SERVICE ROUTINE CALCULATES THE REGEN
3675  ; INPUT
3676  ; AX = ROW, COLUMN POSITION
3677  ; OUTPUT
3678
3679
HANDLE COMMON SCROLL SET UP

A-56 System BIOS
F329 005E400003 3956 CMP CRT_MODE,3
F2EE 7711 3957 JA N9
F350 1------ 6825 COLOR CARD SCROLL 3958
F2F0 52 3961 PUSH DX
F2F1 B0A03 3962 MOV DX,30AH ; GUARANTEED TO BE COLOR CARD HERE
F2F4 50 3963 PUSH AX
F2F5 N9: 3964
F2F6 A00B 3965 TEST AL,0 ; WAIT FOR VERTICAL RETRACE
F2F7 74F6 3966 JZ H8 ; WAIT_DISP_ENABLE
F2F8 74F5 3967
F2F9 B025 3968 PUSH DX,30AH ; GUARANTEED TO BE COLOR CARD HERE
F300 5A 3972 POP DX ; DURING VERTICAL RETRACE
F301 N9: 3973
F302 E081FF 3974 CALL POSITION ; CONVERT TO REGEN POINTER
F304 E00A-A000 3975 ADD AX,CRT_START ; OFFSET OF ACTIVE PAGE
F305 00F6 3976 MOV DI,AX ; TO ADDRESS FOR SCROLL
F306 00F0 3977 MOV SI,AX ; FROM ADDRESS FOR SCROLL
F307 20D1 3978 SUB DI,CX ; DX = ROWS, BCOLS IN BLOCK
F308 0EC6 3979 INC DH ; INCREMENT FOR D ORIGIN
F310 32ED 3980 INC DL
F312 32ED 3981 XOR CH,CH ; SET HIGH BYTE OF COUNT TO ZERO
F314 80244000 3982 MOV BP,CRT_COLS ; GET NUMBER OF COLUMNS IN DISPLAY
F316 03D0 3983 ADD DP,DP ; TIMES 2 FOR ATTRIBUTE BYTE
F318 04C3 3984 MOV AL,DL ; GET LINE COUNT
F31A F6264000 3985 MOV BYTE PTR CRT_COLS ; DETERMINE OFFSET TO FROM ADDRESS
F320 03C0 3986 ADD AX,AH ; #2 FOR ATTRIBUTE BYTE
F322 06 3987 PUSH ES ; ESTABLISH ADDRESSES TO REGEN BUFFER
F323 1F 3988 POP DS ; FOR BOTH POINTERS
F324 0F0000 3989 CMP BL,0 ; 0 SCROLL MEANS BLANK FIELD
F327 F3 3990 RET ; RETURN WITH FLAGS SET
F328 1------ MOVE_ROW 3991 SCROLL_POSITION ENDP
F329 3992 SCROll_POSITION ENDP
F32A 3993 ; INPUT
F32B 3994 N10 PROC NEAR
F32C 3995 MOV CL,DL ; GET # OF COLS TO MOVE
F32D 56 3996 MOV CH,CL
F32E S7 3997 PUSH SI
F32F F3 3998 PUSH DI ; SAVE START ADDRESS
F330 F5 3999 REP MOVSW ; MOVE THAT LINE ON SCREEN
F331 AS 399A POP CX
F332 5F 4000 POP DI
F333 5E 4001 POP SI ; RECOVER ADDRESSES
F334 C3 4002 RET ; OUTPUT
F335 N10 ENDP 4003
F336 5F 4004 1------ CLEAR_ROW 4005 SCROLL_DOWN PROC NEAR
F337 399C MOV CL,DL ; GET # COLUMNS TO CLEAR
F338 57 4007 N11 PROC NEAR
F339 S7 4008 MOV CL,DL
F33A 57 4009 PUSH DI
F33B F3 4010 REP STOSW ; STORE THE FILL CHARACTER
F33C AB 4011 POP DI
F33D 5F 4012 RET
F33E C5 4013 N11 ENDP 4014
F33F 4015 SCROLL_DOWN PROC NEAR
F340 4016 THIS ROUTINE MOVES THE CHARACTERS WITHIN A
F341 4017 DEFINED BLOCK DOWN ON THE SCREEN, FILLING THE
F342 4018 TOP LINES WITH A DEFINED CHARACTER
F343 4019 ; INPUT
F344 4020 ; (AH) = CURRENT CRT MODE
F345 4021 ; (AL) = NUMBER OF LINES TO SCROLL
F346 4022 ; (CX) = UPPER LEFT CORNER OF REGION
F347 4023 ; (DX) = LOWER RIGHT CORNER OF REGION
F348 4024 ; (BH) = FILL CHARACTER
F349 4025 ; (DS) = DATA SEGMENT
F34A 4026 ; (ES) = REGEN SEGMENT
F34B 4027 ; OUTPUT
F34C 4028 ; NONE -- SCREEN IS SCROLLED
F34D 4029 1-----------------------------------------------------------------
F34E 4030 SCROLL_DOWN PROC NEAR

System BIOS A-57
READ_AC_CURRENT PROC NEAR
F374 80FC04 4081 CMP AH,4 ; IS THIS GRAPHICS
F377 7208 4082 JC P1 ; AT THE CURRENT CURSOR POSITION AND RETURNS THEM :
F379 60FC07 4083 CMP AH,7 ; TO THE CALLER :
F37C 7403 4084 JE P1 ; INPUT :
F37E E9A0D 4085 JMP GRAPHICS_READ ; AL) = CURRENT CRT MODE :
F381 28F3 4086 POP AX ; (BH) = DISPLAY PAGE ( ALPHA MODES ONLY ) :
F386 ESIA00 4087 CALL FIND_POSITION ; (DS) = DATA SEGMENT :
F394 D603 4088 MOV SI,DX ; IWAIT FOR RETRACE LOW :
F399 80166300 4089 MOV DX,ADDR_6845 ; IESTABLISH ADDRESSING IN SI :
F39A B3C206 4090 ADD DX,6 ; IWEIGHT FOR RETRACE LOW :
F3B0 06 4091 PUSH ES ; IGET STATUS :
F3B3 IF 4092 POP DS ; IGET STATUS:
F3B5 EC 4093 MOV DX,ADDR_6445 ; IGET BASE ADDRESS:
F3B9 E9A01 4094 ADD DX,6 ; IPOINT AT STATUS PORT:
F3BE P2: 4095 POP DS ; IGET SEGMENT FOR QUICK ACCESS:
F3C2 75FB 4096 MOV AX,SI ; IWAIT FOR RETRACE LOW:
F3C4 FA 4097 IN AL,DX ; IGET STATUS:
F3C8 EC 4098 TEST AL,1 ; INCREASE RETRACE LOW:
F3CE A001 4099 JNZ P2 ; IWAIT UNTIL IT IS:
F3D4 FA 409A CLI ; IGET INTERRUPTS:
F3D8 P3: 409B MOV AX,SI ; IWAIT FOR RETRACE HIGH:
F3DD EC 409C IN AL,DX ; IGET STATUS:
F3E0 A001 409D TEST AL,1 ; IS IT HIGH:
F3E2 74FB 409E JZ P3 ; IWAIT UNTIL IT IS:
F3EB AD 409F LOOP SW ; IGET THE CHAR/ATTR:
F3F0 EB27FE 4100 JMP VIDEO_RETURN ; IGET THE CHAR/ATTR:

A-58 System BIOS
APPENDIX A

System BIOS  A-59
A-60  System BIOS

LOC OBJ  LINE  SOURCE

F3EC 4185  WRITE_DOT PROC NEAR
F3EC 4186  AL = CHAR TO WRITE
F3EC 4187  DS = DATA SEGMENT
F3EC 4188  ES = REGEN SEGMENT
F3EC 4189  none
F3EC 4190  NEAR
F3EC 4191 -----------------------------------------------
F3EC 4192  CMP AX,AH
F3EC 4193  JC  P10
F3EC 4194  CMP AX,AH
F3EC 4195  JE  P10
F3EC 4196  JMP GRAPHICS_WRITE
F3EC 4197  P10:
F3EC 4198  PUSH AX
F3EC 4199  PUSH CX
F3EC 4200  CALL FIND_POSITION
F3EC 4201  MOV DI,DX
F3EC 4202  POP CX
F3EC 4203  POP BX
F3EC 4204  P11:
F3EC 4205  WRITE_LOOP
F3EC 4206  ------ WAIT FOR HORIZONTAL RETRACE
F3EC 4207
F3EC 4208  DB166E300
F3EC 4209  MOV DX,ADDR$645
F3EC 4210  ADD DX,6
F3EC 4211  P12:
F3EC 4212  IN AL,DX
F3EC 4213  TEST AL,1
F3EC 4214  CLI
F3EC 4215  P13:
F3EC 4216  IN AL,DX
F3EC 4217  TEST AL,1
F3EC 4218  JZ P13
F3EC 4219  MOV AL,0
F3EC 4220  STOSB
F3EC 4221  STI
F3EC 4222  IA47
F3EC 4223  LOOP P11
F3EC 4224  JMP VIDEO_RETURN
F3EC 4225  WRITE_C_CURRENT ENDP
F3EC 4226
F3EC 4227  READ DOT -- WRITE DOT
F3EC 4228  THESE ROUTINES WILL WRITE A DOT, OR READ THE DOT AT:
F3EC 4229  THE INDICATED LOCATION
F3EC 4230  ENTRY --
F3EC 4231  DX = ROM (10-199)  (THE ACTUAL VALUE DEPENDS ON THE MODE)
F3EC 4232  = COLUMN (0-639)  (THE VALUES ARE NOT RANGE CHECKED)
F3EC 4233  AL = DOT VALUE TO WRITE (1,2 OR 4 BITS DEPENDING ON MODE)
F3EC 4234  REQ'D FOR WRITE DOT ONLY, RIGHT JUSTIFIED
F3EC 4235  BIT 7 OF AL=1 INDICATES XOR THE VALUE INTO THE LOCATION
F3EC 4236  DS = DATA SEGMENT
F3EC 4237  ES = REGEN SEGMENT
F3EC 4238
F3EC 4239  EXIT
F3EC 4240  AL = DOT VALUE READ, RIGHT JUSTIFIED, READ ONLY
F3EC 4241  assume cs:CODE,ds:DATA,es:DATA
F3EC 4242
F41E 4243  READ_DOT PROC NEAR
F41E 4244  CALL R3
F41E 4245  MOV AL,[ES][SI]
F41E 4246  AND AL
F41E 4247  SHR AL
F41E 4248  MOV CL
F41E 4249  ROL AL
F41E 4250  JMP VIDEO_RETURN
F41E 4251  READ_DOT ENDP
F42F 4252
F42F 4253  WRITE_DOT PROC NEAR
F42F 4254  PUSH AX
F42F 4255
F430 4256  CALL R3
F430 4257  SHR AL
F430 4258  AND AL
F430 4259
F433 425A  MOV [CL][ES][SI]
F433 425B  RECOVER XOR FLAG
F433 425C
F43C 425D

A-60  System BIOS
<table>
<thead>
<tr>
<th>LOC OBJ</th>
<th>LINE</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>F43F 750D</td>
<td>4262</td>
<td>JNZ R2 ; YES, XOR THE DOT</td>
</tr>
<tr>
<td>F441 6D04</td>
<td>4263</td>
<td>NOT AH ; SET THE MASK TO REMOVE THE</td>
</tr>
<tr>
<td>F443 22CC</td>
<td>4264</td>
<td>AND CL, AH ; INDICATED BITS</td>
</tr>
<tr>
<td>F445 0AC1</td>
<td>4265</td>
<td>OR AL,CL ; OR IN THE NEW VALUE OF THOSE BITS</td>
</tr>
<tr>
<td>F447</td>
<td>4266</td>
<td>R1: ; FINISH_DOT</td>
</tr>
<tr>
<td>F447 26004</td>
<td>4267</td>
<td>MOV ES:SI:AL ; RESTORE THE BYTE IN MEMORY</td>
</tr>
<tr>
<td>F44A 50</td>
<td>4268</td>
<td>POP AX</td>
</tr>
<tr>
<td>F44B E977FD</td>
<td>4269</td>
<td>JMP VIDEO_RETURN ; RETURN FROM VIDEO ID</td>
</tr>
<tr>
<td>F44C</td>
<td>4270</td>
<td>R2: ; XOR_DOT</td>
</tr>
<tr>
<td>F44D 301C1</td>
<td>4271</td>
<td>XOR AL,CL ; EXCLUSIVE OR THE DOTS</td>
</tr>
<tr>
<td>F450 EBF5</td>
<td>4272</td>
<td>JMP R1 ; FINISH UP THE WRITING</td>
</tr>
<tr>
<td>F453</td>
<td>4273</td>
<td>WRITE_DOT ENDP</td>
</tr>
<tr>
<td>F455 B028</td>
<td>4274</td>
<td>PROC HEAR</td>
</tr>
<tr>
<td>F456 53</td>
<td>4275</td>
<td>PUSH BX ; SAVE BX DURING OPERATION</td>
</tr>
<tr>
<td>F457 50</td>
<td>4276</td>
<td>PUSH AX ; WILL SAVE AL DURING OPERATION</td>
</tr>
<tr>
<td>F458</td>
<td>4277</td>
<td>----------- DETERMINE 1ST BYTE IN INDICATED ROW BY MULTIPLYING ROW VALUE BY 40</td>
</tr>
<tr>
<td>F459 B028</td>
<td>4278</td>
<td>MOV AL, 40</td>
</tr>
<tr>
<td>F45A 52</td>
<td>4279</td>
<td>PUSH DX ; SAVE ROW VALUE</td>
</tr>
<tr>
<td>F45B 0601FE</td>
<td>4280</td>
<td>AND DL, OFEN ; STRIP OFF ODD/EVEN BIT</td>
</tr>
<tr>
<td>F45A F6EE</td>
<td>4281</td>
<td>MUL DL ; AX HAS ADDRESS OF 1ST BYTE</td>
</tr>
<tr>
<td>F45C 5A</td>
<td>4282</td>
<td>POP DX ; OF INDICATED ROW</td>
</tr>
<tr>
<td>F45D 66C201</td>
<td>4283</td>
<td>TEST DL, 1 ; TEST FOR EVEN/ODD</td>
</tr>
<tr>
<td>F45F 7003</td>
<td>4284</td>
<td>JZ P6 ; JUMP IF EVEN ROW</td>
</tr>
<tr>
<td>F461 000200</td>
<td>4285</td>
<td>ADD AX, 2000H ; OFFSET TO LOCATION OF ODD ROWS</td>
</tr>
<tr>
<td>F462 5020</td>
<td>4286</td>
<td>J1M</td>
</tr>
<tr>
<td>F465</td>
<td>4287</td>
<td>R4: ; EVEN_ROW</td>
</tr>
<tr>
<td>F466 8F</td>
<td>4288</td>
<td>MOV SI, AX ; MOVE POINTER TO SI</td>
</tr>
<tr>
<td>F467 5B</td>
<td>4289</td>
<td>POP AX ; RECOVER AL VALUE</td>
</tr>
<tr>
<td>F468 0DD1</td>
<td>4290</td>
<td>MOV DX, CX ; COLUMN VALUE TO DX</td>
</tr>
<tr>
<td>F469 4DD6</td>
<td>4291</td>
<td>MOV AL, 40</td>
</tr>
<tr>
<td>F46A BDC002</td>
<td>4292</td>
<td>MOV BX, 200H</td>
</tr>
<tr>
<td>F46B 900003</td>
<td>4293</td>
<td>MOV CX, 300H ; SET PARMS FOR MED RES</td>
</tr>
<tr>
<td>F46C 8000096</td>
<td>4294</td>
<td>CMP CRT_MODE, 6 ; HANDLE IF MED ARES</td>
</tr>
<tr>
<td>F46E 7266</td>
<td>4295</td>
<td>JC RS ; HANDLES</td>
</tr>
<tr>
<td>F470 B00001</td>
<td>4296</td>
<td>MOV BX, 180H</td>
</tr>
<tr>
<td>F47A 900307</td>
<td>4297</td>
<td>MOV CX, 703H ; SET PARMS FOR HIGH RES</td>
</tr>
<tr>
<td>F47B 22EA</td>
<td>4298</td>
<td>AND CH, DL ; ADDRESS OF PEL WITHIN BYTE TO CH</td>
</tr>
<tr>
<td>F47C</td>
<td>4299</td>
<td>----------- DETERMINE BIT OFFSET IN BYTE FROM COLUMN MASK</td>
</tr>
<tr>
<td>F47D</td>
<td>429A</td>
<td>RS:</td>
</tr>
<tr>
<td>F47D 22EA</td>
<td>429B</td>
<td>AND CH, DL ; ADDRESS OF PEL WITHIN BYTE TO CH</td>
</tr>
<tr>
<td>F47E</td>
<td>429C</td>
<td>----------- DETERMINE BYTE OFFSET FOR THIS LOCATION IN COLUMN</td>
</tr>
<tr>
<td>F47F D3EA</td>
<td>429D</td>
<td>SHR DX, CL ; SHIFT BY CORRECT AMOUNT</td>
</tr>
<tr>
<td>F481 0F32</td>
<td>429E</td>
<td>ADD SI, DX ; INCREMENT THE POINTER</td>
</tr>
<tr>
<td>F483 8AF7</td>
<td>429F</td>
<td>MOV DH, DH ; GET THE # OF BITS IN RESULT TO DH</td>
</tr>
<tr>
<td>F484 8435</td>
<td>4300</td>
<td>----------- MULTIPLY BH (VALID BITS IN BYTE) BY CH (BIT OFFSET)</td>
</tr>
<tr>
<td>F485 2AC9</td>
<td>4301</td>
<td>SUB CL, CL ; ZERO INTO STORAGE LOCATION</td>
</tr>
<tr>
<td>F487</td>
<td>4302</td>
<td>R6:</td>
</tr>
</tbody>
</table>
LOC OBJ LINE SOURCE

F487 DDC0 4339 ROR AL,1 ; LEFT JUSTIFY THE VALUE

F489 D2CD 4341 ADD CL,CH ; ADD IN THE BIT OFFSET VALUE
F489 FECF 4342 DEC BH ; LOOP CONTROL
F480 75F8 4343 JNZ R6 ; ON EXIT, CL HAS SHIFT COUNT

F48F 8AE3 4345 MOV AH,AL ; GET MASK TO AH
F491 D2EC 4346 SHR AH,CL ; MOVE THE MASK TO CORRECT LOCATION

F493 50 4347 POP BX ; RECOVER REG
F494 C3 4348 RET ; RETURN WITH EVERYTHING SET UP

R3 ENDP

F499 0002 4349 4350 4351 ; SCROLL UP
4352 ; THIS ROUTINE SCROLLS UP THE INFORMATION ON THE CRT
4353 ; ENTRY
4354 ; CH,CL = UPPER LEFT CORNER OF REGION TO SCROLL
4355 ; DH,DL = LOWER RIGHT CORNER OF REGION TO SCROLL
4356 ; BOTH OF THE ABOVE ARE IN CHARACTER POSITIONS
4357 ; BH = FILL VALUE FOR BLANKED LINES
4358 ; AL = # LINES TO SCROLL (AL=0 MEANS BLANK THE ENTIRE
4359 ; FIELD)
4360 ; DS = DATA SEGMENT
4361 ; ES = REGEN SEGMENT
4362 ; EXIT
4363 ; NOTHING, THE SCREEN IS SCROLLED

F498 4365 4366 4367 4368 4369 4370 4371 4372 4373 4374 4375 4376 4377 4378 4379 4380 4381 4382 4383 4384 4385 4386 4387 4388 4389 4390 4391 4392 4393 4394 4395 4396 4397 4398 4399 4400 4401 4402 4403 4404 4405 4406 4407 4408 4409 4410 4411 4412 4413 4414 4415

GRAPHICS_UP PROC NEAR

F498 0AD8 4396 MOV BL,AL ; SAVE LINE COUNT IN BL
F497 B8C1 4397 MOV AX,CX ; GET UPPER LEFT POSITION INTO AX REG

F499 E6902 4398 ; USE CHARACTER SUBROUTINE FOR POSITIONING
F49C E6F8 4399 ; ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE

F49E 0371 439A ; NOTHING, THE SCREEN IS SCROLLED

F498 439B 439C 439D 439E 439F 4400 4401 4402 4403 4404 4405 4406 4407 4408

PROC

F4A8 0364 90026 4409 4410

F4AD 7304 4411 ; FIND_SOURCE

F493 4394 4395 4396 4397 4398 4399 4400 4401 4402 4403 4404 4405 4406 4407 4408 4409 4410 4411 4412 4413 4414 4415

F403 306 439B 439C 439D 439E 439F 4400 4401 4402 4403 4404 4405 4406 4407 4408 4409 4410 4411 4412 4413 4414 4415

F4C5 03F0 4413 4414 4415

F4C7 2AE6 4416

F4C9 2AE3 4417

F4C9 4418

F4CD 4419

F4CE 80000 441A

F4CE 01EEBF01F 441B

F4D4 441C

R8: ; ROM_LOOP

F4D8 441D

F4DE 441E

F4F4 441F

F4F4 4420

F4F4 4421

A-62 System BIOS
LOC OBJ   LINE   SOURCE
F406  F8CC  4416  DEC   AH   ; NUMBER OF ROWS TO MOVE
F406  75F1  4417  JNZ   RB   ; CONTINUE TILL ALL MOVED
F418
F420  ----- FILL IN THE VACATED LINE(S)
F419
F44A
F44A  BAC7  4422  MOV   AL,0H   ; CLEAR_ENTRY
F44C  E000  4424  CALL   R10   ; ATTRIBUTE TO FILL WITH
F40F  D1FF001F  4425  SUB   DI,200H-00   ; POINT TO NEXT LINE
F44E  FCB  4426  DEC   BL   ; NUMBER OF LINES TO FILL
F44F  75F5  4427  JNZ   R10   ; CLEAR_LOOP
F457  75DC  4428  JMP   VIDEO_RETURN   ; EVERYTHING DONE
F457  75DE  4429  MOV   BL,0H   ; BLANK_FIELD
F457  75EE  4430  MOV   BL,0H   ; SET BLANK COUNT TO
F457  75F0  4431  MOV   BL,0H   ; EVERYTHING IN FIELD
F457  75F1  4432  JNP   R9   ; CLEAR THE FIELD
F457  75F2  4433  GRAPHICS_UP END
F457  75F3  4434  ; SCROLL DOWN
F457  75F4  4435  ; THIS ROUTINE SCROLLS DOWN THE INFORMATION ON THE CRT
F457  75F5  4436  ; ENTRY
F457  75F6  4437  ; CH.CL = UPPER LEFT CORNER OF REGION TO SCROLL
F457  75F7  4438  ; DL,DL = LOWER RIGHT CORNER OF REGION TO SCROLL
F457  75F8  4439  ; BOTH OF THE ABOVE ARE IN CHARACTER POSITIONS
F457  75F9  4440  ; DH = FILL VALUE FOR BLANKED LINES
F457  75FA  4441  ; AL = # LINES TO SCROLL (AL=0 MEANS BLANK THE ENTIRE FIELD)
F457  75FB  4442  ; DS = DATA SEGMENT
F457  75FC  4443  ; ES = REGEN SEGMENT
F457  75FD  4444  ; EXIT
F457  75FE  4445  ; NOTHING, THE SCREEN IS SCROLLED
F457  75FF  4446  ; -------------------------------------------
F457
F4EE  GRAPHICS_DONG PROC NEAR
F4EE  F0  4449  STD   ; SET DIRECTION
F4EF  0A06  4450  MOV   BL,AL   ; SAVE LINE COUNT IN BL
F4F1  0B2C  4451  MOV   AX,DX   ; GET LOWER RIGHT POSITION INTO AX REG
F4F3  00F02  4452  CALL   GRAPH_POSN
F4F6  00F0  4453  MOV   DI,AX   ; SAVE RESULT AS DESTINATION ADDRESS
F4F9
F4FB  2001  4454  SUB   DX,CX   ; DETERMINE SIZE OF WINDOW
F4FC  D006  4455  ADD   DX,101H   ; ADJUST VALUES
F4FE  D006  4456  SAL   DH,1   ; MULTIPLY # ROWS BY 4
F500  D006  4457  SAL   DH,1   ; SINCE 8 VERT DOTS/CHAR
F502  D006  4458  SAL   DH,1   ; AND EVEN/ODD ROWS
F504
F506  80349006  4459  CMP   CRT_MODE,6   ; TEST FOR MEDIUM RES
F507  7305  4460  JNC   R12   ; FIND_SOURCE_DOWN
F50F  7305  4461  JNC   R12   ; MEDIUM RES DOWN
F513
F51E  01E7  4462  SAL   DI,1   ; # COLUMNS = 2, SINCE
F51E  0247  4463  INC   DI   ; 2 BYTES/CHAR (OFFSET OR)
F51F  047  4464  INC   DI   ; OFFSET = 2 SINE 2 BYTES/CHAR
F520  047  4465  INC   DI   ; POINT TO LAST BYTE
F524
F526  R12:  4466  JNC   R12   ; FIND_SOURCE_DOWN
F527  R12:  4467  PUSH   ES   ; BOTH SEGMENTS TO REGEN
F528  R12:  4468  POP   DS   ; ZER0 TO HIGH OF COUNT REG
F529  R12:  4469  ADD   DI,240   ; POINT TO LAST ROW OF PIXELES
F52A  R12:  446A  SAL   BL,1   ; MULTIPLY NUMBER OF LINES BY 4
F52B  R12:  446B  JZ   R16   ; IF ZERO, THEN BLANK ENTIRE FIELD
F52C  R12:  446C  MOV   AL,0L   ; GET NUMBER OF LINES IN AL
F52D  R12:  446D  MOV   AH,0H   ; 08 BYTES/ROW
F52E  R12:  446E  MUL   AH   ; DETERMINE OFFSET TO SOURCE
F52F  R12:  446F  JNZ   R12   ; DETERMINE THE SOURCE ADDRESS IN THE BUFFER

System BIOS   A-63
A-64 System BIOS
ASSUME CS:CODE,DS:DATA,ES:DATA

GRAPHICS_WRITE PROC NEAR

MOV AH,0 ; ZERO TO HIGH OF CODE POINT
PUSH AX ; SAVE CODE POINT VALUE

;----- DETERMINE POSITION IN REGEN BUFFER TO PUT CODE POINTS

CALL S26 ; FIND LOCATION IN REGEN BUFFER
MOV DI,AX ; REGEN POINTER IN DI

;----- DETERMINE REGION TO GET CODE POINTS FROM

POP AX ; RECOVER CODE POINT

JAE AX,AL ; ZERO ORIGIN FOR SECOND HALF
PUSH DS ; SAVE DATA SEGMENT
SUB SI,SI ; ESTABLISH VECTOR ADDRESSING
ASSUME DS:DATA
IDS SI,EXT_PTR ; GET THE OFFSET OF THE TABLE
MOV OX,OS ; GET THE SEGMENT OF THE TABLE
ASSUME DS:DATA
POP DS ; RECOVER DATA SEGMENT

;----- DETERMINE GRAPHICS MODE IN OPERATION

SAL AX,1 ; MULTIPLY CODE POINT
SAL AX,1 ; VALUE BY 8
MOV DS,SI,AX ; SI HAS OFFSET OF DESIRED CODES
ADD SI,AX
CMP CR T_MODE,6
POP DS ; RECOVER TABLE POINTER SEGMENT
JC 57 ; TEST FOR MEDIUM RESOLUTION MODE

;----- HIGH RESOLUTION MODE

JS HF ; HIGH_CHAR

;----- DETERMINE GRAPHICS MODE IN OPERATION

SAL AX,1 ; MULTIPLY CODE POINT
SAL AX,1 ; VALUE BY 8
MOV DS,SI,AX ; SI HAS OFFSET OF DESIRED CODES
ADD SI,AX
CMP CR T_MODE,6
POP DS ; RECOVER TABLE POINTER SEGMENT
JC 57 ; TEST FOR MEDIUM RESOLUTION MODE

;----- HIGH RESOLUTION MODE

JS HF ; HIGH_CHAR
LOC OBJ  LINE  SOURCE
F5AE  4643  S4:  | GET BYTE FROM CODE POINTS
F5AF AC  4644  LODSB  | SHOULD WE USE THE FUNCTION
F5AF F6C500  4645  TEST DL,80H  | TO PUT CHAR IN
F5B2 7516  4646  JNZ S6  | STORE IN RECEIVE BUFFER
F5B4 4A  4647  STOSB  | STORE IN SECOND HALF
F5B5 AC  4648  LODSB  | MOVE TO NEXT ROW IN REGEN
F5B6  4649  S5:  | DONE WITH LOOP
F5B6 268005FF00  4650  MDV ES:[DI+2000H-1].AL  | RECOVER REGEN POINTER
F5BB B3C74F  4651  ADD DI,79  | POINT TO NEXT CHAR POSITION
F5BE F6CE  4652  DEC DH  | MORE CHAR TO WRITE
F5C0 75EC  4653  JNZ S4  | EXCLUSIVE OR WITH CURRENT
F5C2 5E  4654  POP SI  | STORE THE CODE POINT
F5C3 5F  4655  POP DI  | AGAIN FOR OLD FIELD
F5C4 47  4656  INC DI  | BACK TO MAINSTREAM
F5C5 E2E3  4657  LOOP S3  | BACK TO MAINSTREAM
F5C7 E9F8FB  4658  JMP VIDEO_RETURN  | TO NEXT LOCATION
F5CA  4659  S6:  | STORE IN SECOND HALF
F5CA 263205  4660  XOR AL,ES:[DI]  | STORE IN SECOND HALF
F5CD AA  4661  STOSB  | MOVE TO NEXT ROW IN REGEN
F5CE AC  4662  LODSB  | DONE WITH LOOP
F5CF 263205FF00  4663  XOR AL,ES:[DI+2000H-1]  | RECOVER REGEN POINTER
F5D0 4BE0  4664  JMP S5  | POINT TO NEXT CHAR POSITION
F5D0  4665  ; MEDIUM RESOLUTION WRITE
F5D0 4666  S7:  |меди_рес_писать
F5D2 26A3  4668  MDV DL,BL  | SAVE HIGH COLOR BIT
F5DB 01E7  4670  SAL DI,1  | OFFSET#2 SINCE 2 BYTES/CHAR
F5DA E01100  4671  CALL S19  | EXPAND BL TO FULL WORD OF COLOR
F5DD  4672  S8:  | МЕД_ЧАР
F5DD  4673  PUSH DI  | STORE REGEN POINTER
F5DE 5E  4674  PUSH SI  | AGAIN FOR OLD FIELD
F5DF 8604  4675  MDV DH,4  | BACK TO MAINSTREAM
F5E1  4676  S9:  | TO NEXT LOCATION
F5E2 E0E0  4677  LODSB  | TO NEXT LOCATION
F5E3 23C3  4678  CALL S21  | TO NEXT LOCATION
F5E5 26C200  4679  AND AX,8X  | DOUBLE UP ALL THE BITS
F5E6  4680  ; CONV_DM TO FO  | CONVERT THEM TO FOREGROUND
F5E7 F6C200  4681  TEST DL,0AH  | COLOR ( 0 BACK )
F5E8 7407  4682  JZ S10  | NO, STORE IT IN AS IT IS
F5EA 263225  4683  XOR AH,ES:[DI]  | DO FUNCTION WITH HALF
F5EF 26324501  4684  XOR AL,ES:[DI+1]  | AND WITH OTHER HALF
F5F3  4685  S10:  | GET CODE POINT
F5F3 26A025  4686  MDV ES:[DI].AH  | STORE FIRST BYTE
F5F6 26A4501  4687  MDV ES:[DI+1].AL  | STORE SECOND BYTE
F5FA AC  4688  LODSB  | GET CODE POINT
F5FB E5C500  4689  CALL S21  | GET CODE POINT
F5FE 23C3  4690  AND AX,8X  | CONVERT TO COLOR
F600 F6C200  4691  TEST DL,0AH  | AGAIN, THIS XOR FUNCTION
F602 740A  4692  JZ S11  | NO, JUST STORE THE VALUES
F603 2632A50020  4693  XOR AH,ES:[DI+2000H]  | FUNCTION WITH FIRST HALF
F604 2632B501020  4694  XOR AL,ES:[DI+2001H]  | AND WITH SECOND HALF
F60F  4695  S11:  | STORE IN SECOND PORTION OF BUFFER
F60F 26A0A00020  4696  MDV ES:[DI+2000H].AH  | STORE IN SECOND PORTION OF BUFFER
F614 26A6050100  4697  MDV ES:[DI+2001H].AL  | STORE IN SECOND PORTION OF BUFFER
F619 63C750  4698  ADD DI,80  | STORE IN SECOND PORTION OF BUFFER
F61C FECE  4699  DEC DH  | STORE IN SECOND PORTION OF BUFFER
F61E 75C1  4700  JNZ S9  | STORE IN SECOND PORTION OF BUFFER
F620 5E  4701  POP SI  | STORE IN SECOND PORTION OF BUFFER
F621 5F  4702  POP DI  | STORE IN SECOND PORTION OF BUFFER
F622 47  4703  INC DI  | STORE IN SECOND PORTION OF BUFFER
F623 47  4704  INC DI  | STORE IN SECOND PORTION OF BUFFER
F624 E0E7  4705  LOOP S8  | STORE IN SECOND PORTION OF BUFFER
F626 E9F0FB  4706  JMP VIDEO_RETURN  | STORE IN SECOND PORTION OF BUFFER
F627  4707  GRAPHICS_WRITE ENDP  | STORE IN SECOND PORTION OF BUFFER
F629  4708  |-------------------------
F629  4709  | GRAPHICS_READ:
F62A  4710  |-------------------------
F629  4711  | GRAPHICS_READ PROC NEAR
F629  E00600  4712  CALL S26  | CONVERTED TO OFFSET IN REGEN
F62C 86F0  4713  MDV SI,AX  | SAVE IN SI
F62E 83C500  4714  SUB SP,B  | ALLOCATE SPACE TO SAVE THE
F630 80EC  4715  MOV BP,SP  | READ CODE POINT
F631 80EC  4716  MOV BP,SP  | POINTER TO SAVE AREA
F632  4717  ; DETERMINE GRAPHICS MODES
F633  4718  ;-------------------------------
F634  4719  ;-------------------------------

A-66  System BIOS
Appendix A
LOC OBJ  LINE  SOURCE
F6A6 E8D2  4796  JMP S16  ; GO BACK AND TRY FOR IT
F6A6  4797  ASSUME DS:DATA
F6A6  4798  i----- CHARACTER IS FOUND ( AL=0 IF NOT FOUND )
F6A6  4800
F6A8  4801  S16:  ADD SP,8  ; READJUST THE STACK, THROW AWAY SAVE
F6A8 B3C408  4802  JMP VIDEO_RETURN  ; ALL DONE
F6A8 E17FB  4803  GRAPHICS_READ ENDP
F6A8  4804  i-----------------------------------------------------------------
F6A8  4805  i EXPAND_MED_COLOR
F6A8  4806  i THIS ROUTINE EXPANDS THE LOW 2 BITS IN BL TO
F6A8  4807  i FILL THE ENTIRE BX REGISTER
F6A8  4808  i ENTRY
F6A8  4809  i BL = COLOR TO BE USED ( LOW 2 BITS )
F6A8  4810  i EXIT
F6A8  4811  i BX = COLOR TO BE USED ( 8 REPLICATIONS OF THE
F6A8  4812  i 2 COLOR BITS )
F6A8  4813  i-----------------------------------------------------------------
F6A8  4814
F6A8  4815  S19  PROC NEAR
F6A8 B0E303  4816  AND BL,3  ; ISOLATE THE COLOR BITS
F6A8  4817  MOV AL,BL  ; COPY TO AL
F6A8  4818  PUSH CX  ; SAVE REGISTER
F6A8 B93000  4819  MOV CX,3  ; NUMBER OF TIMES TO DO THIS
F6A8  4820  S20:  SAL AL,1  ; LEFT SHIFT BY 2
F6A8  4821  SAL AL,1  ; LEFT SHIFT BY 2
F6A8  4822  CMP BL,AL  ; ANOTHER COLOR VERSION INTO BL
F6A8  4823  LOOP S20  ; FILL ALL OF BL
F6A8  4824  MOV BH,BL  ; FILL UPPER PORTION
F6A8  4825  FOC1 S9  POP CX  ; REGISTER BACK
F6A8  4826  FOC2 C3  RET  ; ALL DONE
F6A8  4827  S19 ENDP
F6A8  4828
F6A8  4829  ; EXPAND_BYTE
F6A8  4830  ; THIS ROUTINE TAKES THE BYTE IN AL AND DOUBLES
F6A8  4831  ; ALL OF THE BITS, TURNING THE 0 BITS INTO
F6A8  4832  ; 16 BITS. THE RESULT IS LEFT IN AX
F6A8  4833  ;-----------------------------------------------------------------
F6A8  4834
F6A8  4835  S21  PROC NEAR
F6A8  4836  PUSH DX  ; SAVE REGISTERS
F6A8  4837  PUSH CX
F6A8  4838  PUSH BX
F6A8  4839  SUB DX,DX  ; RESULT REGISTER
F6A8 B90100  4840  MOV CX,1  ; MASK REGISTER
F6A8  4841  S22:  MOV BX,AX  ; BASE INTO TEMP
F6A8  4842  MOV BX,CX
F6A8  4843  AND BX,CX  ; USE MASK TO EXTRACT A BIT
F6A8  4844  OR DX,BX  ; PUT INTO RESULT REGISTER
F6A8 D1E9  4845  SHL AX,1  ; SHIFT BASE AND MASK BY 1
F6A8  4846  SHL CX,1  ; SHIFT BASE AND MASK BY 1
F6A8  4847  MOV DX,AX  ; BASE TO TEMP
F6A8  4848  AND DX,CX  ; EXTRACT THE SAME BIT
F6A8  4849  OR DX,BX  ; PUT INTO RESULT
F6A8 D1E1  4850  SHL CX,1  ; SHIFT ONLY MASK NOW,
F6A8  4851  MOV TO NEXT BASE
F6A8  4852  JNC S22  ; USE MASK BIT COMING OUT TO TERMINATE
F6A8  4853  MOV AX,DX  ; RESULT TO PARM REGISTER
F6A8  4854  RET  ; RECOVER REGISTERS
F6A8  4855  FOC3  FA  POP DX  ; ALL DONE
F6A8  4856  FOC4 C3  RET  ; ALL DONE
F6A8  4857  S21 ENDP
F6A8  4858
F6A8  4859  ; MED_READ_BYTE
F6A8  4860  ; THIS ROUTINE WILL TAKE 2 BYTES FROM THE REGEN
F6A8  4861  ; BUFFER, COMPARE AGAINST THE CURRENT FOREGROUND
F6A8  4862  ; COLOR, AND PLACE THE CORRESPONDING ON/OFF BIT
F6A8  4863  ; PATTERN INTO THE CURRENT POSITION IN THE SAVE
F6A8  4864  ; AREA
F6A8  4865  ; ENTRY
F6A8  4866  ; SJDS = POINTER TO REGEN AREA OF INTEREST
F6A8  4867  ; BX = EXPANDED FOREGROUND COLOR
F6A8  4868  ; BP = POINTER TO SAVE AREA
F6A8  4869  ; EXIT
F6A8  4870  ; BP IS INCREMENT AFTER SAVE
F6A8  4871
F6A8  4872

A-68  System BIOS
FA5
F65 0A24
F67 8A4401
F6A B90000
F6D B200
F6F
F6F C51
F6F1 F8
F6F2 7401
F6F4 F9
F6F5 0002
F6F7 01E9
F6F9 01E9
F6FB 73F2
F6F0 005600
F70 45
F701 CS
F702
F702 A5000
F705
F706 0800
F708 1AC4
F70A 626400
F70C 01E0
F710 01E0
F712 3AFF
F714 03C3
F716 5B
F717 C3
F718
F719
F71A B403
F71C 0A36200
F720 C10
F722 5B

4873 523 PROC NEAR
MDV AX,[SI] ; GET FIRST BYTE
MDV AL,[SI+1] ; GET SECOND BYTE
MDV CX,0000H ; 2 BIT MASK TO TEST THE ENTRIES
MDV DL,0 ; RESULT REGISTER

4878 524:
TEST AX,CX
IS THIS SECTION BACKGROUND?

4879 CLC
CLEAR CARRY IN HOPES THAT IT IS

4880 JZ 525 ; IF ZERO, IT IS BACKGROUND

4881 STC
ISN'T IT, SO SET CARRY

4882 RCL D.L,1
MOVE THAT BIT INTO THE RESULT

4883 SHR CX,1

4884 SHR CX,1
MOVE THE MASK TO THE RIGHT BY 2 BITS

4885 MOV IBP,DL ; STORE RESULT IN SAVE AREA

4886 JNC 524 ; DO IT AGAIN IF MASK DIDN'T FALL OUT

4887 MOV BP,AX ; ADJUST POINTER

4888 RET

4889 523 ENDP

4891 -----------------------------------------------

4892 | V4_POSITION |

4893 THIS ROUTINE TAKES THE CURSOR POSITION.

4894 CONTAINED IN THE MEMORY LOCATION, AND

4895 CONVERTS IT INTO AN OFFSET INTO THE

4896 REGEN BUFFER, ASSUMING ONE BYTE/CHAR.

4897 FOR MEDIUM RESOLUTION GRAPHICS.

4898 THE MOUNT MUST BE DOUBLED.

4899 ENTRY

4900 NO REGISTERS, MEMORY LOCATION

4901 CURSOR_POSN IS USED

4902 EXIT

4903 AX CONTAINS OFFSET INTO REGEN BUFFER

4904 ********************************************

4905 526 PROC NEAR

4906 MDV AX,CURSOR_POSN ; GET CURRENT CURSOR

4907 GRAPH_POSN LABEL NEAR ; GET CURRENT CURSOR

4908 PUSH BX ; SAVE REGISTER

4909 MDV BX,AX ; SAVE A COPY OF CURRENT CURSOR

4910 MDV AL,AH ; GET ROWS TO AL

4911 MUL BYTE PTR CRT_COLS ; MULTIPLY BY BYTES/COLUMN

4912 SHL AX,1 ; MULTIPLY * 4 SINCE 4 REGS/BYTE

4913 SHL AX,1

4914 SUB AH,BH ; ISOLATE COLUMN VALUE

4915 ADD AX,BX ; DETERMINE OFFSET

4916 POP BX ; RECOVER POINT

4917 RET

4918 526 ENDP

4919 -------

4920 | WRITE_TTY |

4921 THIS INTERFACE PROVIDES A TELETYPE LIKE INTERFACE TO THE VIDEO

4922 CARD. THE INPUT CHARACTER IS WRITTEN TO THE CURRENT CURSOR

4923 POSITION, AND THE CURSOR IS MOVED TO THE NEXT POSITION. IF THE

4924 CURSOR LEAVES THE LAST COLUMN OF THE FIELD, THE COLUMN IS SET

4925 TO ZERO, AND THE ROW VALUE IS INCREMENTED. IF THE ROW VALUE

4926 LEAVES THE FIELD, THE CURSOR IS PLACED ON THE LAST ROW, FIRST

4927 COLUMN, AND THE ENTIRE SCREEN IS SCROLLED UP ONE LINE. WHEN

4928 THE SCREEN IS SCROLLED UP, THE ATTRIBUTE FOR FILLING THE NON-NEWLY

4929 BLANKED LINE IS READ FROM THE CURSOR POSITION ON THE PREVIOUS

4930 LINE BEFORE THE SCROLL, IN CHARACTER MODE, IN GRAPHICS MODE,

4931 THE 0 COLOR IS USED.

4932 ENTRY

4933 (AH) = CURRENT CRT MODE

4934 (AL) = CHARACTER TO BE WRITTEN

4935 NOTE THAT SPACE, CAR RET, BEL AND LINE FEED ARE HANDLED

4936 AS COMMANDS RATHER THAN AS DISPLAYABLE GRAPHICS

4937 (BL) = FOREGROUND COLOR FOR CHAR WRITE IF CURRENTLY IN A

4938 GRAPHICS MODE

4939 EXIT

4940 ALL REGISTERS SAVED

4941 -------

4942 | WRITE_TTY | PROC NEAR |

4943 ASSUME CS:CODE,DS:DATA

4944 PUSH AX ; SAVE REGISTERS

4945 PUSH AX ; SAVE CHAR TO WRITE

4946 MDV AH,3 ; GET THE CURRENT ACTIVE PAGE

4947 MDV BH,ACTIVE_PAGE

4948 INT 10H ; READ THE CURRENT CURSOR POSITION

4949 POP AX ; RECOVER CHAR

System BIOS  A-69
A-70  System BIOS
LOC OBJ  
LINE  
SOURCE

5027  I------ LINE FEED FOUND
5028
F786
5029  U10:
F786  B0FE18
5030  CMP DH,24  ; BOTTOM OF SCREEN
5031  JNE U6  ; YES, SCROLL THE SCREEN
5032  JMP UI  ; NO, JUST SET THE CURSOR
5033
5034  I------ BELL FOUND
5035
F788  B0E8
5036  UI1:
5037  MOV BL,0  ; SET UP COUNT FOR BEEP
5038  CALL BEEP  ; SOUND THE PROD BELL
5039  JMP US  ; TTY_RETURN
5040
5041  WRITE_TTY  ; NDF
5042  LIGHT PEN  
5043  THIS ROUTINE TESTS THE LIGHT PEN SWITCH AND THE LIGHT PEN TRIGGER. IF BOTH ARE SET, THE LOCATION OF THE LIGHT PEN Is DETERMINED. OTHERWISE, A RETURN WITH NO INFORMATION IS MADE.
5044  INFORMATION IS MADE.
5045
5046  ON EXIT:
5047  ; (AH) = 0 IF NO LIGHT PEN INFORMATION IS AVAILABLE
5048  ; (AH) = 1 IF LIGHT PEN INFORMATION IS AVAILABLE
5049  ; (AH) = 1 IF LIGHT PEN INFORMATION IS AVAILABLE
5050  ; (AH) = 1 IF LIGHT PEN INFORMATION IS AVAILABLE
5051  ; (AH) = 1 IF LIGHT PEN INFORMATION IS AVAILABLE
5052
5053  ; POSITION
5054  ; (CH) = RASTER POSITION
5055
5056  ; ASSUME CS:CODE,DS:DATA
5057  I------ SUBTRACT_TABLE
5058
F794
5059  VI  ; LABEL BYTE
5060  DB 3,3,5,5,3,3,3,4

F794  03
F795  03
F796  05
F797  05
F798  03
F799  03
F79A  03
F79B  04
F79C
5060  READ_LPEN  ; PROC NEAR
5061   PROC Z80
5062  I------ WAIT FOR LIGHT PEN TO BE DEPRESSES
5063
F79D  B000
5064  MOV AH,0  ; SET NO LIGHT PEN RETURN CODE
F79E  08166300
5065  MOV DX,ADDR_6845  ; GET BASE ADDRESS OF 6845
5066
F7A0  03C106
5067  ADD DX,6  ; POINT TO STATUS REGISTER
F7A1  EC
5068  IN AL,DX  ; GET STATUS REGISTER
F7A2  A004
5069  TEST AL,4  ; TEST LIGHT PEN SWITCH
F7A3  75E7
5070  JNZ V6  ; NOT SET, RETURN
F7A4  090100
5071
F7A5  A002
5072  TEST AL,2  ; TEST LIGHT PEN TRIGGER
F7A6  7503
5073  JNZ V7A  ; RETURN WITHOUT RESETTING TRIGGER
F7A7  E90100
5074
F7A8  B410
5075  V7A:
5076
F7A9  B000
5077  MOV AH,16  ; LIGHT PEN REGISTERS ON 6845
5078  MPC Z80
5079  INPUT_REGS POINTED TO BY AH, AND CONVERT TO ROW COLUMN IN DX
F7BA  B0166300
5080  MOV DX,ADDR_6845  ; ADDRESS REGISTER FOR 6845
F7BB  8AC4
5081  MOV AL,AH  ; REGISTER TO READ
F7BC  B4E
5082  MOV AL,DX  ; GET IT UP
F7BE  A4
5083  MOV CH,AL  ; SAVE IN CX
F7BF  4A
5084  DEC DX  ; ADDRESS REGISTER
F7C0  EC4
5085  INC AH  ; SECOND DATA REGISTER
F7C1  BAC4
5086  MOV AL,AH  ; SECOND DATA REGISTER
F7C2  42
5087  OUT DX,AL  ; POINT TO DATA REGISTER
F7C3  EC
5088  IN AL,DX  ; GET SECOND DATA VALUE
F7C4  8AE5
5089  MOV AH,CH  ; AX HAS INPUT VALUE
F7C6  B410
5090
5091
5092
5093
5094
5095
5096

System BIOS  A-71
; DETERMINE ALPHA MODE
LOC OBJ

; DETERMINE ALPHA
LINE SOURCE

5097 5098 ; AX HAS THE VALUE READ IN FROM THE 6845

F7C8 8A1E49D0
F7CC 2AFF
F7CE 2E4A9F4F7
F7D3 2BC3
F7D5 3B1E4600
F7D9 D1E8
F7DB 2BC3
F7DD 3702
F7DF 2BC0

5099

; DETERMINE GRAPHICS MODE

5099

5102 MOV BL,CRT_MODE ; DETERMINE AMOUNT TO SUBTRACT
5103 SUB AX,AH ; TAKE IT AWAY
5104 MOV BX,CRT_START
5105 SHR BX,1
5106 SUB AX,BX
5107 JHS V2 ; IF POSITIVE, DETERMINE MODE
5108 SUB AX,AH ; <0 PLAYS AS 0
5109

5110 ; DETERMINE MODE OF OPERATION
5111

F7E1 3B03
F7E3 303E490004
F7EB 722A
F7EA 303E490007
F7EF 7423

5112 V2: ; DETERMINE_MODE
5113 MOV CL,3 ; SET #5 SHFT COUNT
5114 CMP CRT_MODE,4 ; DETERMINE IF GRAPHICS OR ALPHA
5115 JB V4 ; ALPHA_PEN
5116 CMP CRT_MODE,7 ; DETERMINE MEDIUM OR HIGH RES
5117 JE V4 ; ALPHA_PEN

5118 ; GRAPHICS MODE

5119

F7F1 B228
F7F3 F02

5120 DIV DL ; DETERMINE ROW(AX) AND COLUMN(AH)

5121

5122 ; AL RANGE 0-59, AH RANGE 0-39
5123

5124 ; DETERMINE GRAPHIC ROW POSITION

5125 F7F5 8AEB
F7F7 8EE0
F7F9 8ADC
F7FB 2AFF
F7FD 303E490006
F802 7504
F806 B104
F80E D0E4
F808 33E3

5126 MOV CH,AL ; SAVE ROW VALUE IN CH
5127 ADD CH,CH ; #2 FOR EVEN/ODD FIELD
5128 MOV BL,AH ; COLUMN VALUE TO BX
5129 SUB BH,BH ; MULTIPLY BY 8 FOR MEDIUM RES
5130 CMP CRT_MODE,6 ; DETERMINE MEDIUM OR HIGH RES
5131 MOV CL,4 ; SHIFT VALUE FOR HIGH RES
5132 JNE V3 ; NOT_HIGH_RES
5133 MOV CL,4 ; SHIFT VALUE FOR HIGH RES
5134 SAL AH,1 ; COLUMN VALUE TIMES 2 FOR HIGH RES
5135 JE ; NOT_HIGH_RES
5136 SHL BX,CL ; MULTIPLY #16 FOR HIGH RES

5137 ; DETERMINE ALPHA CHAR POSITION

5139 F80A 8A04
F80C 8A00
F80E D0EE
F810 D0EE
F812 EB12

5140 MOV DL,AH ; COLUMN VALUE FOR RETURN
5141 MOV DH,AL ; ROW VALUE
5142 SHR DH,1 ; DIVIDE BY 4
5143 SHR DH,1 ; FOR VALUE IN 0-24 RANGE
5144 JMP SHORT V5 ; LIGHT_PEN_RETURN_SET

5145 ; ALPHA MODE ON LIGHT PEN

5146 F814
F814 F6364400
F818 8A00
F81A 8A04
F81C 8E02
F81E 8A09
F820 8A0C
F822 32FF
F824 33E3
F826
F826 B061
F828 8A01
F82B 52
F82B 8A01
F82D 3C2C07
F82D EE
F831 8A
F832 5F
F833 5E
F834 1F
F835 1F
F836 1F
F837 1F
F838 07

5147 V4: ; ALPHA_PEN
5148 DIV BYTE PTR CRT_COLS ; DETERMINE ROW,COLUMN VALUE
5149 MOV DH,AL ; ROWS TO DH
5150 MOV DL,AL ; COLS TO DL
5151 SAL AL,CL ; MULTIPLY ROWS * 6
5152 MOV CH,AL ; GET RASTER VALUE TO RETURN REG
5153 MOV BL,AL ; COLUMN VALUE
5154 XOR BH,BH ; TO BX
5155 SAL BX,CL ; LIGHT_PEN_RETURN_SET

5156 ; LIGHT_PEN_RETURN_SET
5157 MOV AH,1 ; INDICATE EVERYTHING SET
5158 ; LIGHT_PEN_RETURN

5159 V5: ; LIGHT_PEN_RETURN
5160 PUSH DX ; SAVE RETURN VALUE (IN CASE)
5161 MOV DX,ADDR_6845 ; GET BASE ADDRESS
5162 ADD DX,7 ; POINT TO RESET PARM
5163 OUT DX,AL ; ADDRESS, NOT DATA, IS IMPORTANT
5164 POP DX ; RECOVER VALUE
5165 POP DX ; RETURN_NO_RESET

5166 ; POP DX
5167 ; POP SI
5168 ; POP DS
5169 ; POP DS
5170 ; POP DS
5171 ; POP DS
5172 ; POP DS
5173 ; POP ES

A-72 System BIOS
;--- INT 12 ------------------------------

; THIS ROUTINE DETERMINES THE AMOUNT OF MEMORY IN THE SYSTEM
; AS REPRESENTED BY THE SWITCHES ON THE PLANAR. NOTE THAT THE
; SYSTEM MAY NOT BE ABLE TO USE I/O MEMORY UNLESS THERE IS A FULL
; COMPLEMENT OF 64K BYTES ON THE PLANAR.

; INPUT
; NO REGISTERS

; THE MEMORY_SIZE VARIABLE IS SET DURING POWER ON DIAGNOSTICS

; ACCORDING TO THE FOLLOWING HARDWARE ASSUMPTIONS:

; PORT 60 Bits 3,2 = 00 - 16K BASE RAM
; 01 - 32K BASE RAM
; 10 - 48K BASE RAM
; 11 - 64K BASE RAM

; PORT 62 Bits 3-0 INDICATE AMOUNT OF I/O RAM IN 32K INCREMENTS

; E.G., 0000 - NO RAM IN I/O CHANNEL
; 0010 - 64K RAM IN I/O CHANNEL, ETC.

; OUTPUT
; (AX) = NUMBER OF CONTIGUOUS 1K BLOCKS OF MEMORY

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

ASSUME CS:CODE,DS:DATA

ORG 0F841H

MEMORY_SIZE_DET PROC FAR

STI ; INTERRUPTS BACK ON

PUSH DS ; SAVE SEGMENT

CALL DDS

MOV AX,MEMORY_SIZE ; GET VALUE

POP DS ; RECOVER SEGMENT

IRET ; RETURN TO CALLER

MEMORY_SIZE_DET ENDP

;--- INT 11 ----------------------------------------

; THIS ROUTINE ATTEMPTS TO DETERMINE WHAT OPTIONAL
; DEVICES ARE ATTACHED TO THE SYSTEM.

; INPUT
; NO REGISTERS

; THE EQUIP_FLAG VARIABLE IS SET DURING THE POWER ON
; DIAGNOSTICS USING THE FOLLOWING HARDWARE ASSUMPTIONS:

; PORT 60 = LOW ORDER BYTE OF EQUIPMENT
; PORT 3F4 = INTERRUPT 10 REGISTER OF 8250
; BITS 7-3 ARE ALWAYS 0

; PORT 378 = OUTPUT PORT OF PRINTER -- 8255 PORT THAT
; CAN BE READ AS WELL AS WRITTEN

; (AX) IS SET, BIT SIGNIFICANT, TO INDICATE ATTACHED I/O

; BIT 15,14 = NUMBER OF PRINTERS ATTACHED
; BIT 13 NOT USED
; BIT 12 = GAME I/O ATTACHED
; BIT 11,10,9 = NUMBER OF RS232 CARDS ATTACHED
; BIT 8 UNUSED
; BIT 7,6 = NUMBER OF DISKETTE DRIVES
; BIT 5,4 = INITIAL VIDEO MODE
; BIT 3,2 = PLANAR RAM SIZE (00=16K,01=32K,10=48K,11=64K)
; BIT 1 NOT USED
; BIT 0 = IPL FROM DISKETTE -- THIS BIT INDICATES THAT
; THERE ARE DISKETTE DRIVES ON THE SYSTEM

; NO OTHER REGISTERS AFFECTED

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

ASSUME CS:CODE,DS:DATA

ORG 0F840H

EQUIPMENT PROC FAR

STI ; INTERRUPTS BACK ON

PUSH DS ; SAVE SEGMENT

CALL DDS

MOV AX,EQUIP_FLAG ; GET THE CURRENT SETTINGS

POP DS ; RECOVER SEGMENT

IRET ; RETURN TO CALLER

EQUIPMENT ENDP

; ASSUME CS:CODE,DS:DATA

ORG 0F840H

EQUIPMENT PROC FAR

STI ; INTERRUPTS BACK ON

PUSH DS ; SAVE SEGMENT

CALL DDS

MOV AX,EQUIP_FLAG ; GET THE CURRENT SETTINGS

POP DS ; RECOVER SEGMENT

IRET ; RETURN TO CALLER

EQUIPMENT ENDP

System BIOS A-73
ORG 0F859H

CASSETTE_IO PROC FAR

STI

PUSH OS

CALL DDS

AND BIOS_BREAK.7FH

POP OS

RET

CASSETTE_IO ENDP

PURPOSE:

TO CALL APPROPRIATE ROUTINE DEPENDING ON REG AH

F069 FB

CASSETTE_IO PROC

PUSH AX

MOV AX,CASSETTE_IO

INT 15H

POP AX

RETN

ENDP

F084 AREA

PURPOSE:

TO TURN ON CASSETTE MOTOR

F086 7413

JZ MOTOR_ON

F086 FECC

F070 741B

JZ MOTOR_OFF

F087 FECC

F072 FECC

F074 741A

JZ READ_BLOCK

F076 FECC

F078 7503

JNZ W2

F07A E92401

JMP WRITE_BLOCK

F07D 8440

JMP COMMAND_NOT_DEFINED

F07D B460

MOV AH,000H

F07F 99

STC

F085 C5

RET

F081 D1

ENDP

F081 E461

IN AL,PORT_B

F083 24F7

AND AL,NOT 08H

F085 E661

OUT PORT_B,AL

F087 2A44

SUB AH,AH

F089 C3

RET

F08A

ENDP

A-74 System BIOS
System BIOS  A-75
A-76 System BIOS
**Appendix A**

**System BIOS A-77**
A-78 System BIOS
LOC OBJ

LINE

SOURCE

5635

5636
5637

f9E2 A16900

WHICH IS CHECKED FOR CORRECTNESS WHEN THE BLOCK IS READ :

; REG AX IS MODIFIED

1------------------------------------------------------_________ _

5638

MOV

AX,CRC_REG

NOT

AX

; FOR I' 5 COMPLEMENT

5639
F9E5 F700

I WRITE THE ONE'S COMPLEMENT OF THE
;

5640

TWO BYTE CRC TO TAPE

f9E7 50

5641

PUSH

AX

; SAVE IT

f9E6 86EO

5642

XCHG

AH,AL

; WRITE MS BYTE FIRST

F9EA E82300

5643

CALL

WRITE_BYTE

I WRITE IT

POP

AX

; GET IT BACK

CALL

WRITE_BYTE

I NOW WRITE LS BYTE

OR

ex,cx

; IS BYTE COUNT EXHAUSTED?

JNZ

WR_BLoeK

; JUMP IF NOT DONE YET

PUSH

ex

HOV

ex, 32

F9ED 58

5644

F9EE E8lFOO

5645

F9F 1 OBC9
F9F3 7507

5646
5647

f9F5 51

5648

F9F6 Bnooo

5649

F9F9

5650

F9F9 F9

5651

; SAVE REG CX
; WRITE OUT TRAILER BITS

10126:

; TRAI L - LOOP
STC

f9FA E82AOO

5652

CAll

WRITE_BIT

F9FD EZFA

5653

LOOP

10126

; WRITE UNTIL TRAILER WRITTEN

F9Ff 59

5654

POP

CX

; RESTORE REG CX

AL, OBOH

; TURN TIHER2 OFF

FAOO BOBO

5655

HOV

FADZ E643

5656

OUT

FA04680100

5657

MOV

AX, 1

FAD7 E83300

5658

CAll

10131

FADA EB70FE

5659

CALL

MOTOR_OFF

FAOD 28CO

5660

SUB

AX,AX

FAOF C3

SET_TIMER
I

RET

5661

TURN MOTOR OFF
NO ERRORS REPORTED ON WRITE OP
FINISHED

5662

ENOP

5663
5664

; -------------------------------; WRITE A BYTE TO CASSETTE.

5665

; BYTE TO WRITE IS IN REG AL.

5666

FAID

5667

FAID 51

5668

FAll 50

5669

PUSH

AX

FAl2 8AE8

5670

MOV

CH,AL

; AL=BYTE TO WRITE.

HOV

CL,8

; FOR 8 DATA BITS IN BYTE.

PROC
PUSH

NEAR

CX

I SAVE REGS CX,AX

5671
FAl4 B106

(MS BIT WRITTEN FIRSTi

5672
5673

NOTE: TWO EDGES PER BIT
W27:

fAl6

5674

FA16 0005

5675

RCl

FAIS 9C

5676

PUSHF

I

CH,I

DISASSEMBLE THE DATA BIT

I ROTATE HS BIT INTO CARRY

; SAVE FLAGS.
NOTE: DATA BIT IS IN CARRY

5677
FA19 E80BOO

5678

F Ale 90

5679

; WRITE DATA BIT
POPf

; RESTORE CARRY FOR CRC CALC

FAID E82400

5680

CAll

FAZO FEC9

5681

DEC

Cl

I LOOP TIll All 8 BITS DONE

FAZZ 75F2

5682

JNZ

10127

; JUHP IF NOT DONE YET

FA24 58

5683

POP

AX

; RESTORE REGS AX,ex

FA25 59

5684

POP

CX

FA26 C3

5685

RET

; WE ARE FINISHED

5686
5687
5688

; COMPUTE CRC ON DATA BIT

CRC_GEN

ENOP

; ----------------------------------------- --------------PURPOSE:

5689

TO WRITE A DATA BIT TO CASSETTE

5690

CARRY FLAG CONTAINS DATA BIT

5691

I.E. IF SET DATA BIT IS A ONE

5692

IF CLEAR DATA BIT IS A ZERO

5693
5694

NOTE: TWO EDGES ARE WRITTEN PER BIT

5695

ONE BIT

5696

HAS 500 USEC BETWEEN EDGES

FOR A 1000 USEC PERIOD (1 MIlLISEC)

5697
ZERO BIT HAS 250 USEC BETWEEN EDGES

5698
5699
5700
5701
FAZ7

FOR A

500 USEC PERIOD (.5 MIllISEC)

; CARRY flAG IS DATA BIT
; --------- ______________________________________________ _

5702

PROC

NEAR

5703

1 ASSUME IT'S A 'I'

FAZ7 B8AO04

5704

HOV

FAZA 1203

5705

JC

fAze 685002

5706

MOV

fA2F

5707

FAZf 50

5708

FA30

5709

FA30 E462
FA32 2420

AX,1l84

",.

AX,592

W28!

I

SET AX TO NOMINAL ONE SIZE

; JUMP IF ONE BIT
; NO, SET TO NOMINAL ZERO SIZE
; WRITE-BIT-AX

PUSH

AX

5710

IN

Al,PORT_C

5711

AND

Al,020H

; WRITE BIT WITH PERIOD EQ TO VALUE AX

10129:
INPUT TIHER_ 0 OUTPUT

System BIOS

A-79


FA34 704A 5712  JZ  W29  ; LOOP TILL HIGH
FA36 5713  W30:  ; NOW WAIT TILL TIMER’S OUTPUT IS LOW
FA36 E642 5714  IN  AL,PORT_C  
FA35 2042 5715  AND  AL,020H  
FA3A 75FA 5716  JNZ  W30  
FA37 5717  ; RELOAD TIMER WITH PERIOD
FA3C 5718  ; FOR NEXT DATA BIT
FA3D 5719  POP  AX  ; RESTORE PERIOD COUNT
FA3D E642 5720  W31:  ; SET TIMER
FA3D D4E2 5721  OUT  042H, AL  ; SET LOW BYTE OF TIMER 2
FA3F 8AC4 5722  MOV  AL, AH  
FA41 E642 5723  OUT  042H, AL  ; SET HIGH BYTE OF TIMER 2
FA43 C3 5724  SET
FA44 5725  WRITE_BIT  ENDP
FA45 5726  ;-----------------------------------------------
FA46 5727  ; UPDATE CRC REGISTER WITH NEXT DATA BIT  
FA47 5728  ; CRC IS USED TO DETECT READ ERRORS  
FA48 5729  ; ASSUMES DATA BIT IS IN CARRY  
FA49 5730  ;  
FA4A 5731  ; REG AX IS MODIFIED  
FA4B 5732  ; FLAGS ARE MODIFIED  
FA4C 5733  ;-----------------------------------------------
FA44 5734  CRC_GEN  PROC  NEAR
FA44 A16900 5735  MOV  AX,CRC_REG
FA45 5736  ; THE FOLLOWING INSTRUCTIONS
FA46 5737  ; WILL SET THE OVERFLOW FLAG
FA47 5738  ; IF CARRY AND MS BIT OF CRC
FA48 5739  ; ARE UNEQUAL
FA49 5740  RCR  AX,1  
FA4A 5741  RCL  AX,1  ; CLEAR CARRY
FA4B 5742  CLC  
FA4C 7104 5743  JNO  W32  ; SKIP IF NO OVERFLOW
FA4D 5744  ; IF DATA BIT XORED WITH
FA4E 5745  ; CRC REG BIT IS ONE
FA4F 5746  XOR  AX,0810H  ; THEN XOR CRC REG WITH 0810H
FA50 5747  STC  ; SET CARRY
FA51 5748  W32:  ; ROTATE CARRY (DATA BIT)
FA52 5749  RCL  AX,1  
FA53 5750  INTO CRC REG
FA54 5751  MOV  CRC_REG,AX  ; UPDATE CRC_REG
FA55 5752  RET  ; FINISHED
FA56 5753  CRC_GEN  ENDP
FA57 5754  ;-----------------------------------------------
FA58 5755  BEGIN_OP  PROC  NEAR
FA58 E62FE 5756  CALL  MOTOR_ON  ; START TAPE AND DELAY
FA58 B342 5757  MOV  BL,42H  ; TURN ON MOTOR
FA59 5758  IDelay  FOR TAPE DRIVE
FA5A 5759  ; TO GET UP TO SPEED (1/2 SEC)
FA5B 575A  W33:  ; CLEAR CARRY DATA
FA5C 575B  MOV  CX,700H  ; REG BIT 15
FA5D 890007 575C  LOOP  W34  ; INNER LOOP= APPROX. 10 MILISEC
FA5E 575D  575E  DEC  BL  
FA5F 7577 575F  JNZ  W33  
FA60 5760  CRC_REG  AX  
FA61 C3 5761  RET  
FA62 5762  BEGIN_OP  ENDP
FA63 5763  ;-----------------------------------------------
FA67 20323031 5764  E1  DB  ’201’,13,10
FA6B 5765  ;-----------------------------------------------
FA6C 5766  ;-----------------------------------------------
FA6D 5767  ;-----------------------------------------------
FA6E 5768  ;-----------------------------------------------
FA6E 5769  ;-----------------------------------------------
FA6F 5770  ;-----------------------------------------------
FA6F 5771  ;-----------------------------------------------
FA6F 5772  ORG  0F66H  
FA6F 5773  CRT_CHAR_GEN  LABEL  BYTE
FA6F 0000000000000000 5774  DB  00H,000H,000H,000H,000H,000H,000H,000H  D_00
FA6F 7E81A550D0F9E17E 5775  DB  07EH,010H,065H,051H,067H,069H,081H,07EH  D_01
FA6F 7EFDB0FCC3E7FFFE 5776  DB  07EH,0FFH,008H,0FFH,03CH,07EH,0FFH,07EH  D_02
FA6F 6CFEEFE7C361000 5777  DB  06CH,0FEH,0FEH,0FEH,07CH,030H,0100H,000H  D_03
FA6F 10307CFE7C301000 5778  DB  010H,030H,07CH,0FEH,06FH,07CH,030H,0100H  D_04
FA6F 96C03C06FEFC307C0 5779  DB  030H,07CH,030H,0FEH,0FEH,07CH,030H,07CH  D_05
FA6F 10103C06FEC307C0 5780  DB  010H,010H,030H,07CH,07CH,030H,07CH,07CH  D_06
FA6F 0000103C06180000 5781  DB  000H,000H,010H,03CH,030H,010H,000H,000H  D_07
FA6F 8FFFFFFEC3C3E7FFFF 5782  DB  0FFH,0FFH,07EH,0CH,03CH,07EH,0FFH,0FFH  D_08
FA6F 06C03664242663C0 5783  DB  000H,03CH,066H,042H,042H,066H,03CH,000H  D_09
FA6F FFC39900009FC3FF 5784  DB  0FFH,03CH,09FH,000H,000H,09FH,03CH,0FFH  D_0A
FA6F 0000970F0DCC7C70 5785  DB  00FH,007H,09FH,070H,070H,000H,03CH,000H  D_0B
FA6F 3C666665C10718 5786  DB  03CH,066H,066H,066H,03CH,018H,07EH,010H  D_0C

A-80 System BIOS
; THIS ROUTINE ALL OWS
; TIME OF DAY
; = HIGH
; HOURS SINCE LAST
; INTERRUPTED
; IF TIMER
; PASS
; ATOMIC
; PORTION OF COUNT
; PER SECOND
; PICOSECOND
; COUNT
; IF TIMER
; AL = 0 IF TIMER HAS NOT PASSED
; AL = 1
; 24 HOURS SINCE LAST READ
; 48 SECONDS SINCE LAST READ
; NOTE: COUNTS OCCUR AT THE RATE OF
; 193180/65536 COUNTS/SECOND
; (AH) = 1
; GET THE CURRENT CLOCK
; (AH) = 0
; READ THE CURRENT CLOCK SETTING
; INPUT
; READ SEGMENT
; IF INTERRUPTS BACK ON
; ON ANOTHER DAY
; SET TIME
; TOO RETURN
; INTERRUPTS BACK ON
; RECOVER SEGMENT
; RETURN TO CALLER
; READ TIME
; NO TIMER INTERRUPT WHILE READING
; SET OVERFLOW, AND RESET THE FLAG

; TIME_OF_DAY
; ORG 0F66H

; PROC FAR

; INTERRUPTS BACK ON
; SAVE SEGMENT

; READ SEGMENT
; AH = 0
; READ TIME
; AH = 1
; RETURN TO CALLER
; READ TIME
; NO TIMER INTERRUPT WHILE READING
; SET OVERFLOW, AND RESET THE FLAG

; TIME_OF_DAY
; ORG 0F66H

; PROC FAR

; INTERRUPTS BACK ON
; SAVE SEGMENT

; READ SEGMENT
; AH = 0
; READ TIME
; AH = 1
; RETURN TO CALLER
; READ TIME
; NO TIMER INTERRUPT WHILE READING
; SET OVERFLOW, AND RESET THE FLAG

LOC OBJ | LINE | SOURCE
--- | --- | ---
FE08 EBEA | 5941 | JMP T1 ; TOO RETURN
FE09 | 5942 | T5: ; TOO RETURN
FE0A | 5943 | CLI ; NO INTERRUPTS WHILE WRITING
FE0B | 5944 | MOV TIMER_LOW,DX
FE0C | 5945 | MOV TIMER_HIGH,CX
FE0D | 5946 | MOV TIMER_OFT,O,0
FE0E | 5947 | JMP T1
FE0F | 5948 | TIME_OF_DAY ENDP
FE0F | 5949 | 
FE10 | 5950 | 
FE11 | 5951 | THIS ROUTINE HANDLES THE TIMER INTERRUPT FROM
FE12 | 5952 | CHANNEL 0 OF THE 8253 TIMER. INPUT FREQUENCY
FE13 | 5953 | IS 1.19318 MHZ AND THE DIVISOR IS 65536, RESULTING
FE14 | 5954 | IN APPROX. 18.2 INTERRUPTS EVERY SECOND.
FE15 | 5955 |  | 
FE16 | 5956 | THE INTERRUPT HANDLER MAINTAINS A COUNT OF INTERRUPTS
FE17 | 5957 | SINCE POWER ON TIME WHICH MAY BE USED TO ENSURE
FE18 | 5958 | THE INTERRUPT HANDLER ALSO DECREMENTS THE MOTOR
FE19 | 5959 | COUNT OF DISKETTE, AND WHEN IT EXPIRES.
FE1A | 5960 | WILL TURN OFF DISKETTE MOTOR. AND RESET THE
FE1B | 5961 |  | 
FE1C | 5962 | MOTOR RUNNING FLAGS.
FE1D | 5963 | THE INTERRUPT HANDLER WILL ALSO INVOKE A USER ROUTINE
FE1E | 5964 | THROUGH INTERRUPT ICH AT EVERY TIME. THE USER
FE1F | 5965 | MUST CODE A ROUTINE AND PLACE THE CORRECT ADDRESS IN
FE20 | 5966 | THE VECTOR TABLE.
FE21 | 5967 | -----------
FE22 | 5968 | ORG OFEASH
FE23 | 5969 | TIMER_INT PROC FAR ; INTERRUPTS BACK ON
FE24 | 5970 | STI
FE25 | 5971 | PUSH DS
FE26 | 5972 | PUSH AX
FE27 | 5973 | PUSH DX
FE28 | 5974 | CALL ODS
FE29 | 5975 | INC TIMER_LOW ; INCREMENT TIME
FE2A | 5976 | JNZ T5
FE2B | 5977 | INC TIMER_HIGH ; INCREMENT HIGH WORD OF TIME
FE2C | 5978 | T5: ; TEST_DAY
FE2D | 5979 | CMP TIMER_HIGH,018H ; TEST FOR COUNT EQUALING 24 HOURS
FE2E | 5980 | JNZ T5 ; DISKETTE_CTL
FE2F | 5981 | CMP TIMER_LOW,008H
FE30 | 5982 | JZ T5
FE31 | 5983 | DISKETTE_CTL
FE32 | 5984 | ------ TEST FOR DISKETTE TIME OUT
FE33 | 5985 | ------ TEST FOR DISKETTE TIME OUT
FE34 | 5986 | SUB AX,AX
FE35 | 5987 | MOV TIMER_HIGH,AX
FE36 | 5988 | MOV TIMER_LOW,AX
FE37 | 5989 | MOV TIMER_OFT,0
FE38 | 5990 |  | 
FE39 | 5991 | ------ TEST FOR DISKETTE TIME OUT
FE3A | 5992 | T5: ; DISKETTE_CTL
FE3B | 5993 | DEC MOTOR_COUNT
FE3C | 5994 | JNZ T6
FE3D | 5995 | AND MOTOR_STATUS,0FH
FE3E | 5996 | MOV AL,0CH
FE3F | 5997 | MOV DX,03F2H
FE40 | 5998 | FOC CTL PORT
FE41 | 5999 | OUT DX,AL
FE42 | 6000 | T6: ; TIMER@return
FE43 | 6001 | INT 1CH
FE44 | 6002 | MOV AL,EDI
FE45 | 6003 | OUT 020H,AL
FE46 | 6004 | POP AX
FE47 | 6005 | POP DS
FE48 | 6006 | RET
FE49 | 6007 | IRET
FE4A | 6008 | TIMER_INT ENDP
FE4B | 6009 | 
FE4C | 6010 | FAR DB '1801',13,10
FE4D | 6011 | 
FE4E | 6012 | 
FE4F | 6013 | THESE ARE THE VECTORS WHICH ARE MOVED INTO
FE50 | 6014 | THE 8066 INTERRUPT AREA DURING POWER ON.
FE51 | 6015 | ONLY THE OFFSETS ARE DISPLAYED HERE. CODE SEGMENT

System BIOS A-83
A-84 System BIOS
ASSUME CS:CODE,DS:OSDATA

ORG 0FF54H

; PRINT_SCREEN PROC FAR

FF54 FB 6095 STI 1; MUST RUN WITH INTERRUPTS ENABLED
FF55 1E 6096 PUSH DS 1; MUST USE 50:0 FOR DATA AREA STORAGE
FF56 50 6097 PUSH AX
FF57 53 6098 PUSH BX
FF58 51 6099 PUSH CX ; WILL USE THIS LATER FOR CURSOR LIMITS
FF59 52 6000 PUSH DX 1; WILL HOLD CURRENT CURSOR POSITION
FF5A B55000 6001 MOV AX,XXDATA 1; HEX 50
FF5B 803EOOOO1 6002 MOV DS,AX
FF5F 03C0000001 6003 CMP STATUS_BYTE,1 1; SEE IF PRINT ALREADY IN PROGRESS
FF64 74F8 6004 JZ EXIT 1; JUMP IF PRINT ALREADY IN PROGRESS
FF66 C600000001 6005 MOV STATUS_BYTE,1 1; INDICATE PRINT NOW IN PROGRESS
FF68 B40F 6006 MOV AH,15 1; WILL REQUEST THE CURRENT SCREEN MODE
FF6D C10 6007 INT 10H 1; [AH]=MODE

; AT THIS POINT WE KNOW THE COLUMNS/LINE ARE IN
; [AX] AND THE PAGE IF APPLICABLE IS IN [BH]. THE STACK ; HAS DS,AX,BX,CX,DX PUSHED. [AL] HAS VIDEO MODE

FF6F BACC 6011 MOV CL,AH ; WILL MAKE USE OF [CX] REGISTER TO
FF71 B519 6016 MOV CH,25 1; CONTROL ROW & COLUMNS
FF73 E05500 6017 CALL CRLF 1; CARRIAGE RETURN LINE FEED ROUTINE
FF76 51 6018 PUSH CX ; SAVE SCREEN BOUNDS
FF77 0403 6019 MOV AH,3 ; WILL NOW READ THE CURSOR.
FF79 C10 6020 INT 10H ; AND PRESERVE THE POSITION
FF7B 59 6021 POP CX 1; RECALL SCREEN BOUNDS
FF7C 52 6022 PUSH DX 1; RECALL [BH]=VIRTUAL PAGE
FF7D 33D2 6023 XOR DX,DX ; WILL SET CURSOR POSITION TO [0,0]

; THE LOOP FROM PRI10 TO THE INSTRUCTION PRIOR TO PRI20
; IS THE LOOP TO READ EACH CURSOR POSITION FROM THE
; SCREEN AND PRINT. ;

FF7F 629 6029 PRI10:

FF7F B402 6030 MOV AH,2 1; TO INDICATE CURSOR SET REQUEST
FF61 C10 6031 INT 10H 1; NEW CURSOR POSITION ESTABLISHED
FF63 B408 6032 MOV AH,8 1; TO INDICATE READ CHARACTER
FF65 C10 6033 INT 10H 1; CHARACTER NOW IN [AL]
FF67 0400 6034 OR AL,AL 1; SEE IF VALID CHAR
FF69 7502 6035 JNZ PRI15 1; JUMP IF VALID CHAR
FF6D B20 6036 MOV AL,10 1; MAKE A BLANK

FF6D 637 6037 PRI15:

FF6D 33D2 6038 MOV AH,DX 1; SAVE CURSOR POSITION
FF6E 33D2 6039 XOR DX,DX 1; INDICATE PRINTED 1
FF90 32E4 6040 XOR AH,AH 1; TO INDICATE PRINT CHAR IN [AL]
FF92 C17 6041 INT 17H 1; POINT THE CHARACTER
FF94 5A 6042 POP DX 1; RECALL CURSOR POSITION
FF95 F6C25 6043 TEST AH,5AH 1; TEST FOR PRINTER ERROR
FF98 7521 6044 JNZ ERR10 1; JUMP IF ERROR DETECTED
FF9A FEC2 6045 INC DL 1; ADVANCE TO NEXT COLUMN
FF9C 3A4A 6046 CMP CL,DL 1; SEE IF AT END OF LINE
FF9E 750F 6047 JNZ PRI15 1; IF NOT PROCEED
FFA0 33D2 6048 XOR DL,DL 1; BACK TO COLUMN 0
FFA2 0AEE 6049 MOV AH,DL 1; [AH]=0
FFA4 52 6050 PUSH DX 1; SAVE NEW CURSOR POSITION
FFA5 E03300 6051 CALL CRLF 1; LINE FEED CARRIAGE RETURN
FFA8 5A 6052 POP DX 1; RECALL CURSOR POSITION
FFA9 FEC6 6053 INC DH 1; ADVANCE TO NEXT LINE
FFAD 35EE 6054 CMP CH,DH 1; FINISHED?
FFAD 7500 6055 JNZ PRI10 1; IF NOT CONTINUE

FFAD 656 6056 PRI20:

FFAD 5A 6057 POP DX 1; RECALL CURSOR POSITION
FFBB B402 6058 MOV AH,2 1; TO INDICATE CURSOR SET REQUEST
FFB2 C10 6059 INT 10H 1; CURSOR POSITION RESTORED
FFB4 C600000000 605A MOV STATUS_BYTE,0 1; INDICATE FINISHED
FFEB 1000 605B JMP SHORT EXIT 1; EXIT THE ROUTINE

ERR10:

System BIOS  A-85
PRINT_SCREEN ENDP

CRLF PROC
XOR DX,DX
XOR AH,AH
MOV AL,12H
INT 17H
XOR AH,AH
MOV AL,15H
INT 17H
CRLF ENDP

D1 DB 'PARITY CHECK 2',13,10
F3 DB '601',13,10
CODE ENDS

END VECTOR SEGMENT AT OFFFH

JMP RESET
DB '10/27/82'

VECTOR ENDS

END
**Fixed Disk BIOS**

**FOR IBM DISK CONTROLLER**

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. APPLYING ADDRESSES WITHIN THE CODE SEGMENT VIOLATE THE STRUCTURE AND DESIGN OF BIOS.**

---

**INPUT (AH = HEX VALUE)**

- **AH=00** Reset Disk (DL = 80H, 81H) - Diskette
- **AH=01** Read the Status of the Last Disk Operation into (AL)
  - NOTE: DL < 80H - Diskette
  - DL > 80H - Disk
- **AH=02** Read the desired sectors into memory
- **AH=03** Write the desired sectors from memory
- **AH=04** Verify the desired sectors
- **AH=05** Format the desired track
- **AH=06** Format the desired track and set bad sector flags
- **AH=07** Format the drive starting at the desired track
- **AH=08** Return the current drive parameters
- **AH=09** Initialize drive pair characteristics
- **AH=0A** Read 41 points to data block
- **AH=0B** Write long
- **AH=0C** Seek
- **AH=0D** Alternate disk reset (see DL)
- **AH=0E** Read sector buffer
- **AH=0F** Write sector buffer
  - (Recommended practice before formatting)
- **AH=10** Test drive ready
- **AH=11** Recalibrate
- **AH=12** Controller RAM diagnostic
- **AH=13** Drive diagnostic
- **AH=14** Controller internal diagnostic

**REGISTERS USED FOR FIXED DISK OPERATIONS**

- **(DL)** - Drive number (80H-07H for disk, value checked)
- **(DH)** - Head number (0-7 allowed, not value checked)
- **(CH)** - Cylinder number (0-1023, not value checked; see CL)
- **(CL)** - Sector number (1-17, not value checked)

**NOTE:** High 2 bits of cylinder number are placed in the high 2 bits of the CL register

**INPUT (AL)** - Number of sectors (maximum possible range 1-80H, for read/write long 1-79H)

**ES:BX** - Address of buffer for reads and writes, (not required for verify)

**OUTPUT**

- **AH = Status of current operation**
  - STATUS BITS ARE DEFINED IN THE EQUATES BELOW
  - **CY = 0** Successful operation (AH=0 on return)
  - **CY = 1** Failed operation (AH HAS ERROR REASON)

**NOTE:** ERROR 11H indicates that the data read had a recoverable error which was corrected by the ECC algorithm. The data is probably good, however the BIOS routine indicates an error to allow the controlling program a chance to decide for itself. The error may not recur if the data is
Fixed Disk BIOS
<table>
<thead>
<tr>
<th>Line</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>0320</td>
<td>169</td>
</tr>
<tr>
<td>0328</td>
<td>169</td>
</tr>
<tr>
<td>0329</td>
<td>169</td>
</tr>
<tr>
<td>0330</td>
<td>169</td>
</tr>
<tr>
<td>0331</td>
<td>169</td>
</tr>
<tr>
<td>0332</td>
<td>HF_PORT EQU 0320H ; DISK PORT</td>
</tr>
<tr>
<td>0333</td>
<td>RI_BUSY EQU 00001000B ; DISK PORT BUSY BIT</td>
</tr>
<tr>
<td>0334</td>
<td>RI_BUSY EQU 00001000B ; COMMAND/DATA BIT</td>
</tr>
<tr>
<td>0335</td>
<td>RI_MODE EQU 00000111B ; MODE BIT</td>
</tr>
<tr>
<td>0336</td>
<td>RI_REQ EQU 00000001B ; REQUEST BIT</td>
</tr>
<tr>
<td>0337</td>
<td>DMA_READ EQU 01000111B ; CHANNEL 3 (047H)</td>
</tr>
<tr>
<td>0338</td>
<td>DMA_WRITE EQU 01010101B ; CHANNEL 3 (048H)</td>
</tr>
<tr>
<td>0339</td>
<td>DMA EQU 0 ; DMA ADDRESS</td>
</tr>
<tr>
<td>0340</td>
<td>DMA_HIGH EQU 080H ; PORT FOR HIGH 4 BITS OF DMA</td>
</tr>
<tr>
<td>1789</td>
<td>TST_RDONLY_CMD EQU 00000000B ; CNTLR READY (00H)</td>
</tr>
<tr>
<td>1790</td>
<td>RECALS_CMD EQU 00000011B ; RECALS (01H)</td>
</tr>
<tr>
<td>1791</td>
<td>SENSE_CMD EQU 00000011B ; SENSE (03H)</td>
</tr>
<tr>
<td>1792</td>
<td>FMTDRV_CMD EQU 00000100B ; DRIVE (04H)</td>
</tr>
<tr>
<td>1793</td>
<td>CHK_TRK_CMD EQU 00000101B ; T CHK (05H)</td>
</tr>
<tr>
<td>1794</td>
<td>FMTTRK_CMD EQU 00000110B ; TRACK (06H)</td>
</tr>
<tr>
<td>1795</td>
<td>FMTBAD_CMD EQU 00000111B ; BAD (07H)</td>
</tr>
<tr>
<td>1796</td>
<td>READ_CMD EQU 00001000B ; READ (08H)</td>
</tr>
<tr>
<td>1797</td>
<td>WRITE_CMD EQU 00001010B ; WRITE (0AH)</td>
</tr>
<tr>
<td>1798</td>
<td>SEEK_CMD EQU 00001101B ; SEEK (0BH)</td>
</tr>
<tr>
<td>1799</td>
<td>INIT_DRV_CMD EQU 00001100B ; INIT (0CH)</td>
</tr>
<tr>
<td>1800</td>
<td>RD_ECC_CMD EQU 00001111B ; BURST (0DH)</td>
</tr>
<tr>
<td>1801</td>
<td>RD_BUFF_CMD EQU 00001111B ; BUFFR (0EH)</td>
</tr>
<tr>
<td>1802</td>
<td>WR_BUFF_CMD EQU 00001111B ; BUFFR (0FH)</td>
</tr>
<tr>
<td>1803</td>
<td>RAM_DIAG_CMD EQU 11100000B ; RAM (08H)</td>
</tr>
<tr>
<td>1804</td>
<td>CHK_DRV_CMD EQU 11100001B ; DRV (09H)</td>
</tr>
<tr>
<td>1805</td>
<td>CNTLR_DIAG_CMD EQU 11100010B ; CNTLR (0AH)</td>
</tr>
<tr>
<td>1806</td>
<td>BD_LONG_CMD EQU 11100011B ; BOLD (0BH)</td>
</tr>
<tr>
<td>1807</td>
<td>BD_SHORT_CMD EQU 11100100B ; BDLO (0CH)</td>
</tr>
<tr>
<td>1808</td>
<td>BD_ECC_CMD EQU 11100101B ; BD (0DH)</td>
</tr>
<tr>
<td>1809</td>
<td>BD_BUFF_CMD EQU 11100110B ; BDUFFR (0EH)</td>
</tr>
<tr>
<td>1810</td>
<td>200</td>
</tr>
<tr>
<td>1811</td>
<td>INT_CTL_PORT EQU 20H ; 8259 CONTROL PORT</td>
</tr>
<tr>
<td>1812</td>
<td>201</td>
</tr>
<tr>
<td>1813</td>
<td>204</td>
</tr>
<tr>
<td>1814</td>
<td>5_MAX_FILE EQU 6</td>
</tr>
<tr>
<td>1815</td>
<td>S_MAX_FILE EQU 2</td>
</tr>
<tr>
<td>205</td>
<td>ASSUME CS:CODE</td>
</tr>
<tr>
<td>206</td>
<td>ORG 00H</td>
</tr>
<tr>
<td>207</td>
<td>GENERIC BIOS HEADER</td>
</tr>
<tr>
<td>208</td>
<td>DB 05H</td>
</tr>
<tr>
<td>209</td>
<td>DB 0E0H</td>
</tr>
<tr>
<td>210</td>
<td>DB 160</td>
</tr>
<tr>
<td>211</td>
<td>FIXED DISK I/O SETUP</td>
</tr>
<tr>
<td>212</td>
<td>ESTABLISH TRANSFER VECTORS FOR THE FIXED DISK</td>
</tr>
<tr>
<td>213</td>
<td>PERFORM POFER ON DIAGNOSTICS</td>
</tr>
<tr>
<td>214</td>
<td>SHOULD AN ERROR OCCUR A &quot;1701&quot; MESSAGE IS DISPLAYED</td>
</tr>
<tr>
<td>215</td>
<td>FIXED DISK I/O SETUP</td>
</tr>
<tr>
<td>216</td>
<td>ESTABLISH TRANSFER VECTORS FOR THE FIXED DISK</td>
</tr>
<tr>
<td>217</td>
<td>PERFORM POFER ON DIAGNOSTICS</td>
</tr>
<tr>
<td>218</td>
<td>SHOULD AN ERROR OCCUR A &quot;1701&quot; MESSAGE IS DISPLAYED</td>
</tr>
<tr>
<td>219</td>
<td>FIXED DISK I/O SETUP</td>
</tr>
<tr>
<td>220</td>
<td>FIXED DISK I/O SETUP</td>
</tr>
<tr>
<td>221</td>
<td>FIXED DISK I/O SETUP</td>
</tr>
<tr>
<td>222</td>
<td>FIXED DISK I/O SETUP</td>
</tr>
<tr>
<td>223</td>
<td>FIXED DISK I/O SETUP</td>
</tr>
<tr>
<td>224</td>
<td>FIXED DISK I/O SETUP</td>
</tr>
<tr>
<td>225</td>
<td>FIXED DISK I/O SETUP</td>
</tr>
<tr>
<td>226</td>
<td>FIXED DISK I/O SETUP</td>
</tr>
<tr>
<td>227</td>
<td>FIXED DISK I/O SETUP</td>
</tr>
</tbody>
</table>

**Appendix A**

**Fixed Disk BIOS**

A-89
A-90  Fixed Disk BIOS

LOC OBJ  SOURCE

0027 FA  226  CLI
0028 A14C00  229  MOV AX,WORD PTR ORG_VECTOR ; GET DISKETTE VECTOR
002B A30001  230  MOV WORD PTR DISK_VECTOR,AX ; INTO INT 40H
002E A14E00  231  MOV AX,WORD PTR ORG_VECTOR+2
0031 A30201  232  MOV WORD PTR DISK_VECTOR+2,AX
0034 C7064C005602  233  MOV WORD PTR ORG_VECTOR,OFFSET DISK_IO ; HDISK HANDLER
0036 C6CE4E00  234  MOV WORD PTR ORG_VECTOR+2,CS
003E D60607  235  MOV AX,OFFSET HD_INT ; HDISK INTERRUPT
0041 A33000  236  MOV WORD PTR HDISK_INT,AX
0044 A5C00000  237  MOV WORD PTR HDISK_INT+2,CS
0046 C70664006001  238  MOV WORD PTR BOOT_VEC,OFFSET BOOT_STRIP ; BOOTSTRAP
0048 B6CE4600  239  MOV WORD PTR BOOT_VEC+2,CS
0050 C7064041E703  240  MOV WORD PTR HF_TBL_VEC,OFFSET FD_TBL ; PARAMETER TBL
0058 C6CE4601  241  MOV WORD PTR HF_TBL_VEC+2,CS
005C FB  242  STM
243
244  ASSUME DS:DATA
245  MOV AX,DATA ; ESTABLISH SEGMENT
246  MOV DS,AX
247  MOV DISK_STATUS,0 ; RESET THE STATUS INDICATOR
248  MOV HF_MM,0 ; ZERO COUNT OF DRIVES
249  MOV CMD_BLOCK+1,0 ; DRIVE ZERO, SET VALUE IN BLOCK
250  MOV PORT_Off,0 ; ZERO CARD OFFSET
251
252  MOV CX,Z8H ; RETRY COUNT
253  L4: CALL HD_RESET_L ; RESET CONTROLLER
254  JNC L7 ; TRY RESET AGAIN
255  JNC L7
256  MOV L4
257  JMP ERROR_EX
258  L7:
259  MOV CX,1
260  MOV DX,80H
261
262  MOV AX,1200H ; CONTROLLER DIAGNOSTICS
263  INT 13H
264  JNC P7
265  JMP ERROR_EX
266
267  MOV AX,1400H ; CONTROLLER DIAGNOSTICS
268  INT 13H
269  JNC P9
270  JMP ERROR_EX
271
272  MOV TIMER_LOW,0 ; ZERO TIMER
273  MOV AX,RESET_FLAG
274  CMP AX,AX1234H ; KEYBOARD RESET
275  JNC P0
276  MOV TIMER_LOW+100
277  JMP ERROR_EX
278
279  IN AL,02IH ; TIMER
280  AND AL,0FH
281  OUT DX,AL
282  OUT DX,AL
283  MOV AX,1000H
284  JMP ERROR_EX
285  PLEASE PRESS A KEY
286
287  CALL HD_RESET_1 ; READY
288  JNC P2
289
290  IN AL,02IH
291  AND AL,0FH
292  OUT DX,AL
293  MOV AX,1000H
294  JMP ERROR_EX
295
296  MOV AX,1100H ; RECALIBRATE
297  JNC P2
298
299  MOV AX,1100H
29A  JMP ERROR_EX
29B
29C  MOV AX,1100H
29D  JMP ERROR_EX
29E
29F  MOV AX,1100H
300  JMP ERROR_EX
301  MOV AX,1100H
302  MOV AX,1100H
303
304  MOV AH,0CO0H ; DMA TO BUFFER
305

A-90  Fixed Disk BIOS
0065 6EC0 305  MOV  ES:AX  ; SET SEGMENT
0067 200B 306  SUB  BX,BX
0069 BD00FF 307  MOV  AX,0F00H  ; WRITE SECTOR BUFFER
006C CD13 308  INT  13H
006E 7E52 309  JC  ERROR_EX
0070 FE067500 310  INC  HF_NUM  ; DRIVE ZERO RESPONDED
0074 E1302 311  MOV  DX,213H  ; EXPANSION BOX
0077 B000 312  MOV  AL,0
0079 EE 313  OUT  DX,AL  ; TURN BOX OFF
007A FA2103 314  MOV  DX,321H  ; TEST IF CONTROLLER IS IN THE SYSTEM UNIT
007D EC0 315  IN  AL,DX
007F 246F 316  AH0,AL07H
0080 2C0F 317  CMP  AL,0FH
0082 7A06 318  JE  BOX_ON
0084 C706C00A401 319  MOV  TIMER_LOW,420D  ; CONTROLLER IS IN SYSTEM UNIT
0086 0B0A 320  MOV  DX,213H  ; EXPANSION BOX
0088 B0FF 321  MOV  AL,OFFH  ; TURN BOX ON
008A EE 322  OUT  DX,AL
008C BD00 323  MOV  AX,0F00H  ; INITIALIZE CHARACTERISTICS
008E D9100 324  MOV  CX,1  ; ATTEMPT NEXT DRIVES
0090 E1302 325  MOV  DX,001H
0092 316  MOV  AX,AX  ; RESET
0094 CD13 327  INT  13H  ; POD_DONE
0096 7D40 328  JC  POD_DONE
0098 BB0011 329  MOV  AX,01100H  ; RECAL
009A CD13 330  INT  13H  ; POD_DONE
009C 730B 331  JNC  PS
009E A600C0 332  MOV  AX,TIMER_LOW  ; 25 SECONDS
00A0 30001 333  CMP  AL,446D
00A2 72EB 334  JB  P3
00A4 62F90 335  JMP  POD_DONE
00A6 0B0009 336  MOV  AX,0900H
00A8 CD13 337  INT  13H  ; POD_DONE
00A0 7227 338  JC  POD_DONE
00A2 FE067500 339  INC  HF_NUM  ; TALLY ANOTHER DRIVE
00A4 9F48100 340  CMP  DX,(EOH + S_MAX_FILE - 1)
00A6 731D 341  JAE  POD_DONE
00A8 42 342  IN  AL,021H  ; BE SURE TIMER IS DISABLED
00A9 E0F4 343  LOOP  OUT_CH
00AA 344  JMP  P3
00AB 345  OUT_CH:  ; POD_ERROR
00AC 346  JMP  POD_DONE:
00AD 347  ERROR_EX:
00AE 348  ERROR_FLAG
00AF 349  PAGE ZERO
00B0 350  GET_BYTE
00B1 351  VIDEO OUT
00B2 352  DISPLAY CHARACTER
00B3 353  NEXT Char
00B4 354  DO MORE
00B5 355  OUT_CH:
00B6 356  POD_DONE:
00B7 357  CLI
00B8 358  AL01H
00B9 359  OR  AL,01H
00BA 360  OUT  01H,AL
00BB 361  CALL  DDOS
00BC 362  RET
00BD 363  OUT  01H,AL
00BE 364  CALL  DDOS
00BF 365  RET
00C0 366  FLH  DB  '1701',0DH,0AH
A-92  Fixed Disk BIOS
0180 59 450 H2:  POP CX 1 RECOVER RETRY COUNT
018C 730A 451 JNC H4 CF SET BY UNSUCCESSFUL READ
018E 80F500 452 CMP AH,00H IF TIME OUT, NO RETRY
01C1 740A 453 JZ H5 TRY FIXED DISK
01C3 E20E 454 LOOP H1 DO IT FOR RETRY TIMES
01C5 8B0690 455 JMP H5 UNABLE TO IPL FROM THE DISKETTE
01CB 456 H4: IPL WAS SUCCESSFUL
01CB EAD7C0000 457 JMP BOOT_LOC
01C9 458----- ATTEMPT BOOTSTRAP FROM FIXED DISK
01CD 459 H5: SUB AX,AX RESET DISKETTE
01CF 7E0D02 460 SUB DX,DX
01D1 CD13 461 INT 13H
01D3 890300 462 MOV CX,3 SET RETRY COUNT
01D6 45E H6: I PL_SYSTEM
01D7 BA0000 464 MOV DX,0000H FIXED DISK ZERO
01DA 2E0C 465 SUB AX,AX RESET THE FIXED DISK
01DA CD13 466 INT 13H FILE ID CALL
01DC 7212 467 JC H7 IF ERROR, TRY AGAIN
01DE 00102 468 MOV AX,0010H READ IN THE SINGLE SECTOR
01E3 2E0B 469 SUB BX,BX
01E5 0EC3 46A MOV ES,ES
01E7 80070C 46B MOV BX,OFFSET BOOT_LOC TO THE BOOT LOCATION
01E8 BA0000 46C MOV DS,0000H DRIVE HARDER
01E9 090100 46D MOV CX,1 SECTOR 1, TRACK 0
01E0 CD13 46E INT 13H FILE ID CALL
01F0 25F 46F JP CX RECOVER RETRY COUNT
01F2 7200 470 JC H8
01F3 8FED70 471 MOV AX,WORD PTR BOOT_LOC+5100
01F8 3D55AA 472 CMP AX,A0A55H TEST FOR GENERIC BOOT BLOCK
01FB 74CB 473 JC H4
01F0 460 H6: LOOP H6 DO IT FOR RETRY TIMES
01FD E207 465 MOV AX,000H SAVE OFFSET
01FF CD10 467------ UNABLE TO IPL FROM THE DISKETTE OR FIXED DISK
0200 468 INT 10H RESIDENT BASIC
0200 469----- DISKETTE_TBL:
0201 CF 470 DB 1100111B SRT=6, HD UNLOAD=OFF-1ST SPEC BYTE
0202 82 471 DB 2 HD LOAD=1, MODE=DMA-2ND SPEC BYTE
0203 25 472 DB 25H MAST AFTER OPM 1ST MOTOR OFF
0204 02 473 DB 2 512 BYTES PER SECTOR
0205 00 474 DB 8 EOT (LAST SECTOR ON TRACK)
0206 2A 475 DB 02AH GAP LENGTH
0207 FF 476 DB OFFH DTL
0208 50 477 DB 050H GAP LENGTH FOR FORMAT
0209 F6 478 DB 0F6H FILL BYTE FOR FORMAT
020A 19 479 DB 25H HEADING TIME (MILLISECONDS)
020B 04 47A DB 4 MOTOR START TIME (1/6 SECOND)
020C 47B----- MAKE SURE THAT ALL HOUSEKEEPING IS DONE BEFORE EXIT
020C 507 DSBL PROC NEAR
020C 508 ASSUME DS:DATA
020D 8D0000 509 PUSH DS SAVE SEGMENT
020E 0E00 510 MOV AX,DATA
0210 0D00 511 MOV DS,AX
0212 0A067700 512
0216 513 MOV AH,PORT_OFF
0216 514 PUSH AX SAVE OFFSET
0217 C606770000 515 MOV PORT_OFF,04
021C E8905 516 CALL PORT_3
021F 2AC0 517 SUB AL,AL
0221 50 518 OUT DX,AL
0222 C606770004 519 MOV PORT_OFF,04H
0227 E85505 520 CALL PORT_3
022A 2AC0 521 SUB AL,AL
022F 2AC0 522 OUT DX,AL
0231 C606770008 523 MOV PORT_OFF,08H
0232 E85505 524 CALL PORT_3
0235 2AC0 525 SUB AL,AL

Fixed Disk BIOS A-93
LOC OBJ  |  LINE  |  SOURCE
---|---|---
0257 EE | 527 | OUT DX,AL  |  \[I\]  \[RESET INT/DMA MASK\]
0258 C64677000C | 528 | MOV PORT_OFFSET,0CH |  
025D 0E0405 | 529 | CALL PORT_5 |  
025A 0AC7 | 530 | SUB AL,AL |  
0242 EE | 531 | OUT DX,AL |  \[I\]  \[RESET INT/DMA MASK\]
0243 0007 | 532 | MOV AL,07H |  
0245 E30A | 533 | OUT DMA+10,AL |  \[I\]  \[SET DMA MODE TO DISABLE\]
0247 FA | 534 | CLI |  \[I\]  \[DISABLE INTERRUPTS\]
024A 0E41 | 535 | IN AL,021H |  
024A 0E20 | 536 | OR AL,020H |  
024C 0E61 | 537 | OUT 021H,AL |  \[I\]  \[DISABLE INTERRUPT 5\]
025E FB | 538 | SFI |  \[I\]  \[ENABLE INTERRUPTS\]
025F 0B | 539 | POP AX |  \[I\]  \[RESTORE OFFSET\]
0259 80567700 | 540 | MOV PORT_OFFSET,AH |  \[I\]  \[RESTORE SEGMENT\]
0259 1F | 541 | POP DS |  
0255 C3 | 542 | RET |  
0256 | 543 | DS:SI |  
0546 | 545 | i--GENERAL>Description--i |  
0546 | 546 | i--GENERAL>Description--i |  
0546 | 547 | i--GENERAL>Description--i |  
0256 | 548 | DISK_ID_PROC FAR |  
0559 | 549 | DISK_ID_PROC FAR |  
0256 0FFA0 | 550 | ASSUME DS:NOTHING;ES:NOTHING |  
0259 7306 | 551 | CMP DL,80H |  \[I\]  \[TEST FOR FIXED DISK DRIVE\]
025D CD40 | 552 | JAE FAILED_DISK |  \[I\]  \[YES, HANDLE HERE\]
025D | 553 | INT 40H |  \[I\]  \[DISKETTE HANDLER\]
025D RET-2 | 554 | RET 2 |  \[I\]  \[BACK TO CALLER\]
025D | 555 |  |  
0260 | 556 | HARD_DISK: |  
0557 | 557 | ASSUME DS:DATA |  \[I\]  \[ENABLE INTERRUPTS\]
0256 | 558 | SFI |  \[I\]  \[ENABLE INTERRUPTS\]
0559 | 559 | OR AH,AH |  
0253 7509 | 560 | JNZ A3 |  
0255 CD40 | 561 | INT 40H |  \[I\]  \[RESET NEC WHEN AH=0\]
0257 | 562 | SUB AH,AH |  
0259 0FFA01 | 563 | CMP DL:(80H + MAX_FILE - 1) |  
025C 771F | 564 | JA RET-2 |  
026E | 565 | A3: |  
026E | 566 | CMP AH,0B |  \[I\]  \[GET PARAMETERS IS A SPECIAL CASE\]
0271 | 567 | JNZ A2 |  
0275 | 568 | JMP GET_PARM_H |  
0276 | 569 | A2: |  
0276 | 570 | PUSH BX |  \[I\]  \[SAVE REGISTERS DURING OPERATION\]
0277 | 571 | PUSH CX |  
0279 | 572 | PUSH DS |  
027A | 573 | PUSH ES |  
027B | 574 | PUSH SI |  
027C | 575 | PUSH DI |  
027D | 576 | CALL DISK_ID_CONT |  \[I\]  \[PERFORM THE OPERATION\]
027D | 577 | CALL DISK_ID_CONT |  \[I\]  \[PERFORM THE OPERATION\]
0280 50 | 578 | PUSH AX |  \[I\]  \[BE SURE DISABLING OCCURRED\]
0281 | 579 | CALL DSBL |  \[I\]  \[BE SURE DISABLING OCCURRED\]
0284 | 580 | MOV AX,DATA |  
0287 | 581 | MOV DS:AX |  \[I\]  \[ESTABLISH SEGMENT\]
0289 | 582 | POP AX |  
029A | 583 | MOV AH,DISK_STATUS |  \[I\]  \[GET STATUS FROM OPERATION\]
029E | 584 | CMP AH,1 |  \[I\]  \[SET THE CARRY FLAG TO INDICATE\]
029F | 585 | CMP AH,1 |  \[I\]  \[SUCCESS OR FAILURE\]
029C | 586 | CMP AH,1 |  \[I\]  \[SUCCESS OR FAILURE\]
029D | 587 | CMP AH,1 |  \[I\]  \[SUCCESS OR FAILURE\]
029E | 588 | POP DI |  \[I\]  \[RESTORE REGISTERS\]
029F | 589 | POP SI |  
02A4 | 590 | POP DS |  
02A1 | 591 | POP ES |  
02A2 | 592 | POP CX |  
02A4 | 593 | POP DX |  
02A9 | 594 | POP BX |  \[I\]  \[THROW AWAY SAVED FLAGS\]
02A9 | 595 | RET 2 |  
029C | 596 | DS:SI |  
029C | 597 | DS:SI |  
029C | 598 | DS:SI |  
029C | 599 | DW DISK_RESET |  \[I\]  \[FUNCTION TABLE\]
029E | 600 | DW RETURN_STATUS |  \[I\]  \[FUNCTION TABLE\]
02A0 | 601 | DW DISK_READ |  \[I\]  \[FUNCTION TABLE\]
02A2 | 602 | DW DISK_WRITE |  \[I\]  \[FUNCTION TABLE\]
02A4 | 603 | DW DISKVERIFY |  \[I\]  \[FUNCTION TABLE\]

**A-94 Fixed Disk BIOS**
LOC OBJ  LINE  SOURCE
02AA 7203  604  DW  FNT_TRK  1 005H
02AB 7903  605  DW  FNT_BAD  1 006H
02AA 8003  606  DW  FNT_DRV  1 007H
02AC 5003  607  DW  BAD_COMMAND  1 008H
02AB 2704  608  DW  INT_DRV  1 009H
020B CF04  609  DW  RD_LONG  1 00AH
0202 D904  610  DW  WR_LONG  1 00BH
0204 F204  611  DW  DISK.Seek  1 00CH
0206 5803  612  DW  DISK.RESET  1 00DH
0208 F904  613  DW  RD_BUFF  1 00EH
020A 7005  614  DW  WR_BUFF  1 00FH
020C 1505  615  DW  TST_ROY  1 010H
020E 1C05  616  DW  HDISK_RECAL  1 011H
020C 2505  617  DW  RAM_DIAG  1 012H
020C 2A05  618  DW  CMN_DRV  1 013H
020C 3105  619  DW  CHITR_DIAG  1 014H
002A 620  MI  EQU  8-M1
02C6 622  SETUP.A PROC  NEAR
02CE 623  ;------- CALCULATE THE PORT OFFSET
02CC 624  MOV  DISK_STATUS,0 1  RESET THE STATUS INDICATOR
02CD 51  625  PUSH  CX  1  SAVE CX
02C8 626  ;------- FILE-_CALCULATE THE PORT OFFSET
02CC 627  MOV  CH,DL  1  SAVE DL
02CE 628  DEC  DL,1 1  GENERATE OFFSET
02D1 629  MOV  DL,0 1  STORE OFFSET
02D3 62A5  630  MOV  PORT_OFF,DL 1  STORE OFFSET
02D5 62D5  631  MOV  DL,CH 1  RESTORE DL
02D7 62F5  632  AND  DL,1 1  HEAD NUMBER
02DE 637  MOV  CL,5 1  SHIFT COUNT
02DD 638  SHR  CL,1 1  DRIVE NUMBER (0,1)
02E0 639  OR  DL,0H 1  HEAD NUMBER
02E2 640  MOV  CMD_BLOCK+1,DL 1  HEAD NUMBER
02E4 641  POP  CX 1  HEAD NUMBER
02E3 642  RET 1  HEAD NUMBER
02EA 643  MOV  CS: [SI,OFFSET HI 1  CONVERT DRIVE NUMBER TO 0 BASED RANGE
02EB 648  MOV  AX,DATA 1  ESTABLISH SEGMENT
02E6 649  MOV  AX,DATA 1  ESTABLISH SEGMENT
02E0 650  CMP  AH,01H 1  RETURN STATUS
02E4 651  JNZ  AX 1  RETURN STATUS
02E7 652  JMP  RETURN_STATUS 1  RETURN STATUS
02EF 653  MOV  AX,0 1  RETURN STATUS
02FF 654  MOV  DL,0H 1  CONVERT DRIVE NUMBER TO 0 BASED RANGE
02FC 655  MOV  DL,MAX_FILE 1  LEGAL DRIVE TEST
02FD 656  JMP  BAD_COMMAND 1  LEGAL DRIVE TEST
02FC 657  CALL  SETUP.A 1  LEGAL DRIVE TEST
0301 658  CALL  SETUP.A 1  LEGAL DRIVE TEST
0304 660  SETUP.A ENG 1  CALL SETUP.A 1  LEGAL DRIVE TEST
0306 663  MOV  CMD_BLOCK+0,0 1  SECTORS 0-16 FOR CONTROLLER
0308 664  MOV  CMD_BLOCK+2,CL 1  SECTOR AND HIGH 2 BITS CYLINDER
030F 665  MOV  CMD_BLOCK+3,CH 1  CYLINDER
0313 666  MOV  CMD_BLOCK+4,AL 1  INTERLEAVE / BLOCK COUNT
0316 667  MOV  AL,CONTROL_BYTE 1  CONTROL BYTE (STEP OPTION)
031A 668  MOV  CMD_BLOCK+5,AL 1  CONTROL BYTE (STEP OPTION)
031C 669  MOV  AX,DATA 1  ESTABLISH SEGMENT
031D 670  MOV  AL,AH 1  GET INTO LOW BYTE
031F 671  MOV  AH,AL.672  MOV  AX,1 1  ZERO HIGH BYTE
0321 673  MOV  AX,1.674  MOV  AX,1 1  ** FOR TABLE LOOKUP
0323 675  MOV  AL,AX 1  ** FOR TABLE LOOKUP
0325 676  JMP  CMD,AX,MIL 1  TEST WITHIN RANGE
0328 677  MOV  AX,0 1  RESTORE AX
0329 678  JMP  CDITR_COMMAND 1  RESTORE AX
032B 679  JMP  WORD PTR CS:[SI + OFFSET HI 1  RESTORE AX
0330 680  BAD_COMMAND 1  BAD_COMMAND
0330 681  MOV  DISK_STATUS,BAD_CHM 1  COMMAND ERROR
0330 682  MOV  AL,0 1  COMMAND ERROR

Fixed Disk BIOS  A-95
0330 DISK_RESET PROC NEAR
0331 CALL PORT_1 ; RESET PORT
0332 OUT DX,AL ; ISSUE RESET
0333 CALL PORT_1 ; CONTROLLER HARDWARE STATUS
0334 IN AL,DX ; GET STATUS
0335 AND AL,2 ; ERROR BIT
0336 JZ DRI
0337 MOV DISK_STATUS,BAD_RESET
0338 RET
0339 DRI:
0340 JMP INIT_DRV ; SET THE DRIVE PARAMETERS
0341 DISK_RESET ENDP

034D RETURN_STATUS PROC NEAR
034E MOV AL,DISK_STATUS ; OBTAIN PREVIOUS STATUS
034F MOV DISK_STATUS,0 ; RESET STATUS
0350 RET
0351 RETURN_STATUS ENDP

0356 DISK_READ PROC NEAR
0357 MOV AL,DMA_READ ; MODE BYTE FOR DMA READ
0358 MOV CMD_BLOCK+0,READ_CMD
0359 JMP DMA_DPH
0360 DISK_READ ENDP

036D DISK_WRITE PROC NEAR
036E MOV AL,DMA_WRITE ; MODE BYTE FOR DMA WRITE
0370 MOV CMD_BLOCK+0,WRITE_CMD
0371 JMP DMA_DPH
0372 DISK_WRITE ENDP

036A DISK_VERIFY PROC NEAR
036B MOV CMD_BLOCK+0,CHK_TRK_CMD
036C JMP DMA_DPH
036D DISK_VERIFY ENDP

0372 FMT_TRK PROC NEAR ; FORMAT TRACK (AH = 005H)
0373 MOV CMD_BLOCK,FMTTRK_CMD
0374 JMP SHORT FMT_CONT
0375 FMT_TRK ENDP

0379 FMT_BAD PROC NEAR ; FORMAT BAD TRACK (AH = 006H)
037A MOV CMD_BLOCK,FMTBAD_CMD
037B JMP SHORT FMT_CONT
037C FMT_BAD ENDP

0380 FMT_DRV PROC NEAR ; FORMAT DRIVE (AH = 007H)
0381 MOV CMD_BLOCK,FMTDRV_CMD
0382 JMP SHORT FMT_CONT
0383 FMT_DRV ENDP

A-96 Fixed Disk BIOS
LOC OBJ    LINE    SOURCE
0358 758    FMT_CONT:
0355 A94400 759    MOV AL,CMD_BLOCK+2 ; ZERO OUT SECTOR FIELD
0358 24CD 760    AND AL,11000000B
035A A24400 761    MOV CMD_BLOCK+2,AL
035D E9A601 762    JMP MDAarhus
763
764 i----------------------------------------
765 i GET PARAMETERS (AH = 0)
766 i----------------------------------------
767
0390 768 GET_PARAM_N  ; GET DRIVE PARAMETERS
0390 769 GET_PARAM  ; SAVE REGISTERS
0390 770 PUSH DS
0391 771 PUSH ES
0392 53
0393 772 PUSH BX
0394 773 ASSUME DS:SUMMY
0395 775 SUB AX,AX  ; ESTABLISH ADDRESSING
0396 776 MOV DS,AX
0397 777 LES BX,DF_TBL_VEC
0398 778 ASSUME SS:DATA
039B B04000 779 MOV AX,DATA
039E 780 MOV DS,AX
03A0 DEA00 781 SUB DL,80H
03A3 60A00 782 CMP DL,MAX_FILE  ; TEST WITHIN RANGE
03A6 732F 783 JAE G4
03A8 7D1FF 784 CALL SETUP_A
03B0 7D0F3 785 CALL SN2_OFFS
03AE 7277 786 JC G4
03B0 7D05 787 ADD BX,AX
03B2 260007 788 MOV AX,[0:BX]  ; MAX NUMBER OF CYLINDERS
03B5 D02000 789 SUB AX,2  ; ADJUST FOR O-N
03BB 8AE8 790 MOV CH,AL  ; AND RESERVE LAST TRACK
03B0 250003 791 ADR AH,0300H  ; HIGH TWO BITS OF CYL
03BD 0160 792 SHR AX,1
03BF 01E0 793 SHR AX,1
03C1 0C11 794 OR AL,011H  ; SECTORS
03C3 8A08 795 MOV CL,AL
03C5 26A7702 796 MOV DH,ES:[BX][1]  ; HEADS
03C9 FEE 797 DEC DH  ; 0-N RANGE
03CB 0167500 798 MOV DL,DF_TBL_VEC  ; DRIVE COUNT
03CF 20C0 799 SUB AX,AX
03D1 806 802 MOV BX,[G]:ES:[BX][2]  ; RESTORE REGISTERS
03D2 5B 803 POP BX
03D2 07 804 POP ES
03D3 1F 805 POP DS
03D4 60200 806 RET E
03D7 81F 807 MOV AX,[G]:ES:[BX][3]  ; SET ERROR FLAG
03D7 C067400007 808 MOV DISK_STATUS,INIT_FAIL ; OPERATION FAILED
03DC 8047 809 MOV AH,INIT_FAIL
03DE 2AC0 810 SUB AL,AL
03E0 2002 811 SUB DX,DX
03E2 20C9 812 SUB CX,CX
03E4 F9 813 JMP GS
03E5 EB0A 814 GET_PARAM EHD
03E9 819 820
03E9 81A 821
03E9 81B 822 i INITIALIZE DRIVE CHARACTERISTICS
03E9 81C 823 i
03E9 81D 824 i FIXED DISK PARAMETER TABLE
03E9 81E 825 i
03E9 81F 826 i THE TABLE IS COMPOSED OF A BLOCK DEFINED AS:
03E9 827 i
03E9 828 i THE TABLE IS COMPOSED OF A BLOCK DEFINED AS:
03E9 829 i
03E9 82A 830 i (1 WORD) - MAXIMUM NUMBER OF CYLINDERS
03E9 82B 831 i (1 BYTE) - MAXIMUM NUMBER OF HEADS
03E9 82C 832 i (1 WORD) - STARTING REDUCED WRITE CURRENT CYL
03E9 82D 833 i (1 WORD) - STARTING WRITE PRECOMPENSATION CYL
03E9 82E 834 i (1 BYTE) - MAXIMUM ECC DATA BURST LENGTH
03E9 82F 835 i (1 BYTE) - CONTROL BYTE (DRIVE STEP OPTION)
03E9 830 836 i BIT 7 DISABLE DISK-ACCESS RETRIES
03E9 831 837 i BIT 6 DISABLE ECC RETRIES

Appendix A

Fixed Disk BIOS  A-97
A-98  Fixed Disk BIOS
LOC OBJ LINE  SOURCE
0412 28 912 DB 020H; CHECK DRIVE
0413 00000000 913 DB 0.0.0.0
914
914 i----- DRIVE TYPE 03
915
916 0417 3201 917 DW 03060
0419 04 918 DB 040
041A 3201 919 DW 03060
041C 0000 920 DW 00000
041E 0D 921 DB 0DH
041F 05 922 DB 05H; STANDARD
0420 0C 923 DB 0CH; FORMAT DRIVE
0421 B4 924 DB 09H; CHECK DRIVE
0422 28 925 DB 020H; CHECK DRIVE
0423 00000000 927

0427 928 INIT_DRV PROC NEAR
929
930 i----- DO DRIVE ZERO
931
0427 C60642000C 932 MOV CMD_BLOCK+0.INIT_DRV_CMP
042C C606430000 933 MOV CMD_BLOCK+1.0
0431 E61000 934 CALL INIT_DRV_R
0434 7200 935 JC INIT_DRV_OUT
936
0437 937 i----- DO DRIVE ONE
938
0436 C60642000C 939 MOV CMD_BLOCK+0.INIT_DRV_CMP
043B C606430000 940 MOV CMD_BLOCK+1.0000000B
0440 E60100 941 CALL INIT_DRV_R
0443 942 INIT_DRV_OUT:
0443 C3 943 RET
944
0446 945 INIT_DRV ENDP
946
0449 946 INIT_DRV_R PROC NEAR
947
044A 947 ASSUME ES:CODE
044C 948 SUB AL,AL; ISSUE THE COMMAND
044E E61901 949 CALL COMMAND
0450 7391 950 JC B1
0452 C3 951 RET
952
0454 953 EB0C 954 SUB AL,AL; SAVE SEGMENT
0456 E60100 955 CALL COMMAND
0458 7391 956 JC B1
0458 C3 957 RET
958
045A 959 MOV DS,AX; ESTABLISH SEGMENT
045E C41E0401 960 LES BX,HF_TBL_VEC
0461 7245 961 JC B2
0461 962 CALL SNCR Área
0463 7257 963 JC B3
0465 E3FO03 964 ADD BX,AX
0466 72D8 965 CALL INIT_DRV_S
966
0467 966 SEND DRIVE PARAMETERS MOST SIGNIFICANT BYTE FIRST
967
0468 BF0100 966 MOV DI.1
046A E65F00 967 CALL INIT_DRV_S
0463 7240 968 JC B3
969
046B BF0000 970 MOV DI.0
046D E65700 971 CALL INIT_DRV_S
046F 7245 972 JC B3
973
046D BF0200 974 MOV DI.2
0470 E64F00 975 CALL INIT_DRV_S
0473 7230 976 JC B3
977
0475 BF0400 978 MOV DI.4
0477 E64700 979 CALL INIT_DRV_S
0479 7225 980 JC B3
981
047D BF0300 982 MOV DI.3
047F E63F00 983 CALL INIT_DRV_S
0483 7220 984 JC B3
985
0485 BF0600 986 MOV DI.6
0487 E63700 987 CALL INIT_DRV_S
0488 7225 988 JC B3
989
A-100  Fixed Disk BIOS
Fixed Disk BIOS  A-101
1141 I-----------------------------------------------
1142 | SUPPORT ROUTINES                      |
1143 I-----------------------------------------------
1144 0536
1145 0536 B002
1146 0538 E07200
1147 CALL COMMAND ; ISSUE THE COMMAND
1148 0538 7221
1149 JC G11
1150 0539 EB16
1151 JMP SHORT G3
1152 053F
1153 G3:
1154 053F C60674009
1155 MOV DISK_STATUS,DMA_BOUNDARY
1156 0544 C3
1157 RET
1158 0545
1159 DMA_DYN:
1160 0545 E05781
1161 CALL DMA_SETUP ; SET UP FOR DMA OPERATION
1162 0546 72F5
1163 JC G8
1164 054A B003
1165 MOV AL,03H
1166 054C E81300
1167 CALL COMMAND ; ISSUE THE COMMAND
1168 054E 7220
1169 JC G11
1170 0551 B003
1171 MOV AL,03H
1172 0553 E06A0
1173 OUT DMA+10,AL ; INITIALIZE THE DISK CHANNEL
1174 0555
1175 0555 E421
1176 IN AL,0E0H
1177 0557 E40F
1178 AND AL,00FH
1179 0559 E421
1180 OUT 0E0H,AL
1181 055B E0AA01
1182 CALL WAIT_INT
1183 055E
1184 G11:
1185 CALL ERRORCHK
1186 0561 C3
1187 RET
1188 I-----------------------------------------------
1189 | COMMAND                                      |
1190 I-----------------------------------------------
1191 1191 0562 E04200
1192 MOV SI,OFFSET CMD_BLOCK
1193 0565 E01002
1194 CALL PORT_E
1195 0568 EE
1196 OUT DX,AL ; CONTROLLER SELECT PULSE
1197 0569 E01C02
1198 CALL PORT_F
1199 056C EE
1200 OUT DX,AL
1201 056D 2BC9
1202 SUB CX,CX ; WAIT COUNT
1203 056F E00C02
1204 CALL PORT_1
1205 0572
1206 WAIT_BUST:
1207 0572 EC
1208 IN AL,DX ; GET STATUS
1209 0573 E40F
1210 AND AL,00FH
1211 0575 SC00
1212 CMP AL,R1_BUSY OR R1_BUS OR R1_REQ
1213 0577 7409
1214 JE C1
1215 0579 E2F7
1216 LOOP WAIT_BUST
1217 057B C606740080
1218 MOV DISK_STATUS,TIME_OUT
1219 0580 F9
1220 STC
1221 0581 C3
1222 RET ; ERROR RETURN
1223 0582
1224 C1:
1225 CLD
1226 0583 E00600
1227 MOV CX,6 ; BYTE COUNT
1228 0586
1229 EMS:
1230 0586 E0E801
1231 CALL PORT_D
1232 0589 AC
1233 LODSB ; GET THE NEXT COMMAND BYTE
1234 058A EE
1235 OUT DX,AL ; OUT IT GOES
1236 058B E0F9
1237 LOOP CM3 ; DO MORE
1238 058D E0EE01
1239 CALL PORT_1 ; STATUS
1240 0590 EC
1241 IN AL,DX
1242 0591 E601
1243 TEST AL,R1_REQ
1244 0593 7406
1245 JE CH
1246 0595 C606740020
1247 MOV DISK_STATUS,READ_CTLR
1248 059A F9
1249 STC
1250 059B
1251 CH7:
1252 RET
1253 059B C3
1254 RET
1255 COMMAND MSK
1256 1191 I-----------------------------------------------
1257 | SENSE STATUS BYTES                             |
1258 I-----------------------------------------------
1259 1215 I-----------------------------------------------
1260 1216 I-----------------------------------------------
1261 1217 I-----------------------------------------------

A-102  Fixed Disk BIOS
1218 BIT 7 ADDRESS VALID, WHEN SET
1219 BIT 6 SPARE, SET TO ZERO
1220 BITS 5-4 ERROR TYPE
1221 BITS 3-0 ERROR CODE
1222
1223 BYTE 1
1224 BITS 7-6 ZERO
1225 BIT 5 DRIVE (0-1)
1226 BITS 4-0 HEAD NUMBER
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

059C
1236 ERROR_CHK PROC NEAR
1237 ASSUME ES:DATA
1238
059C A07000
1239 MOV AL,DISK_STATUS ; CHECK IF THERE WAS AN ERROR
1240 059F DAC0
1241 OR AL,AL
1242 05A1 7501
1243 JNZ G21
1244
1245 05A4 059C
1246 G21:
1247 05A4 B00000
1248 MOV AX,DATA
1249 05A7 2800
1250 MOV ES,AX
1251 05A9 280C
1252 SUB AX,AX
1253 05AB 080F
1254 MOV DI,AX
1255 05AD C064200003
1256 MOV CMD_BLOCK+0,SENSE_CMD
1257 05D2 2AC0
1258 SUB AL,AL
1259 05D4 EAA0FF
1260 CALL COMMAND ; ISSUE SENSE STATUS COMMAND
1261 05D7 7223
1262 JC SENSE_ABORT ; I CANNOT RECOVER
1263 05D9 990400
1264 MOV CX,4
1265 05DC
1266 G22:
1267 05DC E00000
1268 CALL HD_WAIT_REQ
1269 05DF 7220
1270 JC G24
1271 05C1 F00100
1272 CALL PORT_0
1273 05C4 EC
1274 IN AL,DX
1275 05C5 26304542
1276 MOV ES:HD_ERROR[DI],AL ; STORE AWAY SENSE BYTES
1277 05C9 47
1278 JNC G1
1279 05CA E00100
1280 CALL PORT_1
1281 05CD E2ED
1282 LOOP G22
1283 05CF E00000
1284 CALL HD_WAIT_REQ
1285 05D0 7200
1286 JC G24
1287 05D4 E9A001
1288 CALL PORT_0
1289 05D7 EC
1290 IN AL,DX
1291 05DB A002
1292 TEST AL,2
1293 05DA 740F
1294 JZ STAT_ERR
1295 05DC
1296 SENSE_ABORT:
1297 05DC C0647400FF
1298 MOV DISK_STATUS,SENSE_FAIL
1299 05E1 F9
1230 STC
1231 05E2 C3
1232 RET
1233 05E3 IAOE
1234 T_0 DW TYPE_0
1235 05E5 2706
1236 T_1 DW TYPE_1
1237 05E7 6A0E
1238 T_2 DW TYPE_2
1239 05E9 7706
1240 T_3 DW TYPE_3
1241 05EB
1242 STAT_ERR:
1243 05EB 268A1E4200
1244 MOV BLES:HD_ERROR ; GET ERROR BYTE
1245 05F0 8AC3
1246 MOV AL,BL
1247 05F2 240D
1248 AND AL,0FH
1249 05F4 0E330
1250 AND BL,BL01100000B ; ISOLATE TYPE
1251 05F7 ZAFF
1252 SUB DH,DH
1253 05F9 0103
1254 MOV CL,3
1255 05FB D804
1256 SHR BX,CL ; ADJUST
1257 05FD 2E7F1E350
1258 JMP WORD PTR CS:[BX + OFFSET T_0]
1259 05FC
1260 ASSUME ES:NOTHING
1261 05F2
1262 0592
1263 TYPEO_TABLE LABEL BYTE
LOC OBJ     LINE     SOURCE
0602 0020020000020 1295  DB  $BAD_CNTLR,BAD_SEEK,BAD_CNTLR,TIME_OUT,0,BAD_CNTLR
0607 0040  1296  DB  $BAD_SEEK
0607 0040  1297  TYPE0_TABLE EQU $-TYPE0_TABLE
0607 0102  1298  DB  BAD_CNTLR
060B 10100200004  1299  DB  BAD_ECC,BAD_ADDR_MARK,0,RECORD_NOT_FMD
0610 4000001100B  1300  DB  BAD_SEEK,0,0,DATA_CORRECTED,BAD_TRACK
060A 0602  1301  TYPE1_LEN EQU $-TYPE1_TABLE
0615 0102  1302  TYPE1_TABLE EQU $-TYPE1_TABLE
0615 1010020004  1303  DB  BAD_ADDR_MARK
060D 0002  1304  TYPE2_LEN EQU $-TYPE2_TABLE
0617 0602  1305  TYPE2_TABLE EQU $-TYPE2_TABLE
0617 1010020010  1306  DB  BAD_CNTLR,BAD_CNTLR,BAD_ECC
0617 0003  1307  TYPE3_LEN EQU $-TYPE3_TABLE
0619 0609  1308  DB  BAD_CNTLR
061A 0609  1309  i------ TYPE 0 ERROR
061B 0610  1310  i------ TYPE 1 ERROR
061C 0621  1311  TYPE0: MOV BX,OFFSET TYPE0_TABLE
061D 0611  1312  MOV AL,TYPE0_LEN ; CHECK IF ERROR IS DEFINED
061D 0611  1313  CMP AL,TYPE0_LEN ; CHECK IF ERROR IS DEFINED
061E 0611  1314  JAE UNDEF_ERR_L
061F 061E  1315  XLAT CS:TYPE0_TABLE ; TABLE LOOKUP
0620 0622  1316  MOV DISK_STATUS,AX ; SET ERROR CODE
0622 0622  1317  RET
0623 0627  1318  i------ TYPE 2 ERROR
0624 0627  1319  TYPE2: MOVBX,OFFSET TYPE2_TABLE
0625 0627  1320  MOV AL,TYPE2_LEN ; CHECK IF ERROR IS DEFINED
0626 0627  1321  CMP AL,TYPE2_LEN ; CHECK IF ERROR IS DEFINED
0627 0627  1322  JAE UNDEF_ERR_L
0628 0629  1323  XLAT CS:TYPE2_TABLE ; TABLE LOOKUP
0629 0629  1324  MOV DISK_STATUS,AL ; SET ERROR CODE
062A 0629  1325  AND CL,00H ; CORRECTED ECC
062A 062A  1326  CMP CL,00H
062B 062B  1327  JZ OBTAIN ECC ERROR BURST LENGTH
062C 062C  1328  MOV CHD_BLOCK+0,RD_ECC_CID
062D 062D  1329  SUB AL,AL
062E 062E  1330  CALL COMMAND
062F 062F  1331  JC 3.3.
0630 0630  1332  CALL HD_WRITE_REQ
0631 0631  1333  JC 3.3.
0632 0632  1334  CALL PORT_O
0633 0633  1335  AL,DX
0634 0634  1336  TEST AL,01H
0635 0635  1337  JZ 3.3.
0636 0636  1338  MOV BAD_CNTLR
0637 0637  1339  G30:
0638 0638  1340  MOV AL,CL
0639 0639  1341  RET
063A 063A  1342  i------ TYPE 3 ERROR
063B 063B  1343  TYPE3: MOV BX,OFFSET TYPE3_TABLE
063C 063C  1344  MOV AL,TYPE3_LEN ; CHECK IF ERROR IS DEFINED
063D 063C  1345  CMP AL,TYPE3_LEN ; CHECK IF ERROR IS DEFINED
063E 063C  1346  JAE UNDEF_ERR_L
063F 063F  1347  XLAT CS:TYPE3_TABLE ; TABLE LOOKUP
0640 0640  1348  MOV DISK_STATUS,AL ; SET ERROR CODE
0641 0641  1349  RET
0642 0642  1350  i------ TYPE 2 ERROR
0643 0642  1351  TYPE2: MOV BX,OFFSET TYPE2_TABLE
0644 0642  1352  MOV AL,TYPE2_LEN ; CHECK IF ERROR IS DEFINED
0645 0642  1353  CMP AL,TYPE2_LEN ; CHECK IF ERROR IS DEFINED
0646 0642  1354  JAE UNDEF_ERR_L
0647 0647  1355  XLAT CS:TYPE2_TABLE ; TABLE LOOKUP
0648 0648  1356  MOV DISK_STATUS,AL ; SET ERROR CODE
0649 0649  1357  RET
064A 064A  1358  i------ TYPE 3 ERROR
064B 064B  1359  TYPE3: MOV BX,OFFSET TYPE3_TABLE
064C 064C  1360  MOV AL,TYPE3_LEN ; CHECK IF ERROR IS DEFINED
064D 064C  1361  JAE UNDEF_ERR_L
064E 064E  1362  XLAT CS:TYPE3_TABLE
064F 064F  1363  RET

LaC
OBJ LINE
SOURCE

0660 A27400
1372 MOV DISK_STATUS,AL

0683 C3
1373 RET

0684 1375 UNDEF_ERR_L:
0684 C60674000B
1376 MOV DISK_STATUS,UNDEF_ERR

0689 C3
1377 RET

068A 1378 HD_WAIT_REQ PROC NEAR
068A 51
1379 PUSH CX

068B 0E09
1380 SUB CX,CX
068D EEE00
1381 CALL PORT_1

0690 1382 LI: 1383 CALL PORT_1
0691 A01
1384 TEST AL,R1_REQ

0693 7508
1385 JHZ L2
0695 E9F9
1386 LOOP LI

0697 C606740080
1388 HOV DISK_STATUS,TIME_OUT

069C F9
1389 STC
069D 1390 LZ:
069E C3 1391 RET

069F 1392 HD_WAIT_REQ ENDP

1393 1394--------------------------------------------------------
1393 1395 ; DHA_SETUP
1393 1396 DNA_SETUP PROC NEAR
1394 1397 PUSH AX
1395 1398 MOV AL,CHO_BLOCK+4
1396 eHP AL,Slh ; BLOCK COUNT OUT OF RANGE
1397 POP AX
1398 JB J1
1399 STe
1400 RET
1401 J1:
1402 PUSH ex , SAVE THE REGISTER
1403 POP AX ; NO MORE INTERRUPTS
1404 MOV AL,01H ; SET THE FIRST/LAST FF
1405 POP AX
1406 OUT DMA+12,AL ; OUTPUT THE MODE BYTE
1407 JB J1
1408 JB J1
1409 JB J1
1410 STC
1411 RET

1412 J1:
1413 PUSH CX ; SAVE THE REGISTER
1414 CLI ; NO MORE INTERRUPTS
1415 OUT DMA+12,AL ; SET THE FIRST/LAST FF
1416 PUSH AX
1417 POP AX
1418 MOV AX,0H ; GET THE ES VALUE
1419 JB J1
1420 MOV CL,4 ; SHIFl COUNT
1421 ROL AX,CL ; SELECT LEFT
1422 MOV CH,AL ; MOVE TO CH
1423 AND AL,0FH ; ZERO THE LOW 4 BITS FROM SEGMENT
1424 ADD AX,0X ; TEST FOR CARRY FROM ADDITION
1425 JNC J3
1426 INC CH ; CARRY MEANS HIGH 4 BITS MUST BE INC
1427 J3:
1428 PUSH AX ; SAVE START ADDRESS
1429 OUT DMA+6,AL ; OUTPUT LOW ADDRESS
1430 MOV AL,AM
1431 OUT DMA+6,AL ; OUTPUT HIGH ADDRESS
1432 MOV AL,CH
1433 AND AL,0FH
1434 OUT DMA+12,AL ; OUTPUT THE HIGH 4 BITS TO PAGE REG
1435 I----- DETERMINE COUNT
1436 1437 MOV AL,CMD_BLOCK+4
1438 MOV AL,CMD_BLOCK+4
1439 MOV AL,AM
1440 MOV AL,CH
1441 MOV AL,AN
1442 MOV AL,AN
1443 I----- HANDLE READ AND WRITE LONG (5160 BYTE BLOCKS)
1444 1445 PUSH AX ; SAVE REGISTER
1445 MOV AL,CMD_BLOCK+4
1446 MOV AL,CMD_BLOCK+4
1447 MOV AL,CMD_BLOCK+4
1448 MOV AL,CMD_BLOCK+4
1449 MOV AL,CMD_BLOCK+4
1450 MOV AL,CMD_BLOCK+4
1451 MOV AL,CMD_BLOCK+4
1452 MOV AL,CMD_BLOCK+4
1453 MOV AL,CMD_BLOCK+4
1454 MOV AL,CMD_BLOCK+4
1455 MOV AL,CMD_BLOCK+4
1456 MOV AL,CMD_BLOCK+4
1457 MOV AL,CMD_BLOCK+4
1458 MOV AL,CMD_BLOCK+4
1459 MOV AL,CMD_BLOCK+4
1460 MOV AL,CMD_BLOCK+4
1461 MOV AL,CMD_BLOCK+4
1462 MOV AL,CMD_BLOCK+4
1463 MOV AL,CMD_BLOCK+4
1464 MOV AL,CMD_BLOCK+4
1465 MOV AL,CMD_BLOCK+4
1466 MOV AL,CMD_BLOCK+4
1467 MOV AL,CMD_BLOCK+4
1468 MOV AL,CMD_BLOCK+4

Appendix A

Fixed Disk BIOS  A-105
A-106  Fixed Disk BIOS
```assembly
0740 E02300 1527 CALL PORT_0
074E EC 1528 IN AL,DX
074F 2402 1529 AND AL,2 ; ERROR BIT
0751 0067400 1530 OR DISK_STATUS,AL ; SAVE
0755 E03000 1531 CALL PORT_3 ; INTERRUPT MASK REGISTER
0756 3C0 1532 XOR AL,AL ; ZERO
075A E008 1533 OUT DX,AL ; ERROR MASK
075E 58 1534 POP SI ; RESTORE REGISTERS
0760 0749 1535 RET

0760 074A 1541 HD_INT PROC NEAR
0761 BA2003 1542 PUSH AX
0762 2C4 1543 MOV AL,0DI
0764 E620 1544 OUT INT_CTL_PORT,AL ; END OF INTERRUPT
0765 B007 1545 MOV AL,0TH ; SET DMA MODE TO DISABLE
0766 E60A 1546 OUT DMA+10,AL
0767 E021 1547 IN AL,021H
0768 EC20 1548 ORD,AL,020H
0769 E621 1549 OUT 021H,AL
076A 50 1550 POP AX
0770 CF 1551 IRET

0771 1552 HD_INT ENDP

0771 1553 PORT_0 PROC NEAR
0772 BA2003 1554 MOV DX,HF_PORT ; BASE VALUE
0773 50 1555 PUSH AX
0774 2C4 1556 MOV AH,AL
0776 A07700 1557 MOV AL,PORT_OFF ; ADD IN THE OFFSET
0777 0D00 1558 ADD DX,AX
0778 050 1559 POP AX
0779 C3 1560 RET

0780 1561 PORT_0 ENDP

0780 1562 PORT_1 PROC NEAR
0781 E0F0FF 1563 CALL PORT_0
0782 42 1564 INC DX ; INCREMENT TO PORT ONE
0783 C3 1565 RET

0784 1566 PORT_1 ENDP

0784 1567 PORT_2 PROC NEAR
0785 E0F0FF 1568 CALL PORT_1
0786 42 1569 INC DX ; INCREMENT TO PORT TWO
0787 C3 1570 RET

0788 1571 PORT_2 ENDP

0789 1572 PORT_3 PROC NEAR
078A E0F0FF 1573 CALL PORT_2
078B 42 1574 INC DX ; INCREMENT TO PORT THREE
078C C3 1575 RET

078D 1576 PORT_3 ENDP

078D 1577 |-----------------------------------------------|
0791 1578 | | SH2_OFFSETS |
0794 1579 | | DETERMINE PARAMETER TABLE OFFSET |
0797 1580 | | USING CONTROLLER PORT TWO AND |
0799 1581 | | DRIVE IDENTIFIER (0-1) |
079B 1582 | | ||

079D 1583 |-----------------------------------------------|
079E 1584 1595 SH2_OFFSETS PROC NEAR
079F E0F3FF 1596 CALL PORT_2
07A0 EC 1597 IN AL,DX ; READ PORT 2
07A1 50 1598 PUSH AX
07A2 E0E9FF 1599 CALL PORT_1
07A5 EC 1600 IN AL,DX
07A6 2402 1601 AND AL,2 ; CHECK FOR ERROR
07A7 58 1602 POP AX
07A8 7516 1603 JNZ SH2_OFFSETS_ERR
07A9 0A264300 1604 MOV AH,CMD_BLOCK+1

Appendix A

Fixed Disk BIOS A-107
<table>
<thead>
<tr>
<th>LOC OBJ</th>
<th>LINE</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>079F 00E4Z0</td>
<td>1605</td>
<td>AND AH,00100000B</td>
</tr>
<tr>
<td>07A2 7504</td>
<td>1606</td>
<td>JNZ SW2_AND</td>
</tr>
<tr>
<td>07A4 D0E8</td>
<td>1607</td>
<td>SHR AL,1</td>
</tr>
<tr>
<td>07A5 D0E8</td>
<td>1608</td>
<td>SHR AL,1</td>
</tr>
<tr>
<td>07A8 D0E8</td>
<td>1609</td>
<td>SHL AL,1</td>
</tr>
<tr>
<td>07A8 24D3</td>
<td>1610</td>
<td>AND AL,011B</td>
</tr>
<tr>
<td>07AA B104</td>
<td>1611</td>
<td>MOV CL,4</td>
</tr>
<tr>
<td>07AC D2E0</td>
<td>1612</td>
<td>SHL AL,CL</td>
</tr>
<tr>
<td>07AE 2A44</td>
<td>1613</td>
<td>SUB AH,AL</td>
</tr>
<tr>
<td>07B0 C3</td>
<td>1614</td>
<td>RET</td>
</tr>
<tr>
<td>07B1</td>
<td>1615</td>
<td>SW2_OFFS_ERR:</td>
</tr>
<tr>
<td>07B1 F9</td>
<td>1616</td>
<td>STC</td>
</tr>
<tr>
<td>07B2 C3</td>
<td>1617</td>
<td>RET</td>
</tr>
<tr>
<td>07B3 30382F31362F38</td>
<td>1618</td>
<td>SW2_OFFS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'08/16/82'</td>
</tr>
<tr>
<td>0783 30382F31362F38</td>
<td>1619</td>
<td>32</td>
</tr>
<tr>
<td>07BB</td>
<td>1621</td>
<td>END_ADDRESS</td>
</tr>
<tr>
<td>----</td>
<td>1622</td>
<td>CODE ENDS</td>
</tr>
<tr>
<td>1623</td>
<td></td>
<td>END</td>
</tr>
</tbody>
</table>

A-108  Fixed Disk BIOS
APPENDIX B: 8088 ASSEMBLY INSTRUCTION SET REFERENCE
### 8088

#### Register Model

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

- **Stack Pointer**
- **Base Pointer**
- **Source Index**
- **Destination Index**

<table>
<thead>
<tr>
<th>IP</th>
<th>Instruction Pointer</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLAGSH</td>
<td>Status Flags</td>
</tr>
<tr>
<td>FLAGSL</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CS</th>
<th>Code Segment</th>
</tr>
</thead>
<tbody>
<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 use the symbol FLAGS to represent the file:

\[
\begin{array}{cccccccccccccccccccc}
15 & X & X & X & X & OF & DF & IF & TF & SF & ZF & X & AF & X & PF & X & CF \\
\end{array}
\]

- **x** = Don't Care

#### Flags

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

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

---

8080 Flags

8088 Flags

**B-2 8088 Instruction Reference**
Operand Summary

"reg field Bit Assignments:

<table>
<thead>
<tr>
<th>16-Bit (w=1)</th>
<th>8-Bit (w=0)</th>
<th>Segment</th>
</tr>
</thead>
<tbody>
<tr>
<td>000 AX</td>
<td>000 AL</td>
<td>00 ES</td>
</tr>
<tr>
<td>001 CX</td>
<td>001 CL</td>
<td>01 CS</td>
</tr>
<tr>
<td>010 DX</td>
<td>010 DL</td>
<td>10 SS</td>
</tr>
<tr>
<td>011 BX</td>
<td>011 BL</td>
<td>11 DS</td>
</tr>
<tr>
<td>100 SP</td>
<td>100 AH</td>
<td></td>
</tr>
<tr>
<td>101 BP</td>
<td>101 CH</td>
<td></td>
</tr>
<tr>
<td>110 SI</td>
<td>110 DH</td>
<td></td>
</tr>
<tr>
<td>111 DI</td>
<td>111 BH</td>
<td></td>
</tr>
</tbody>
</table>

Second Instruction Byte Summary

<table>
<thead>
<tr>
<th>mod</th>
<th>Displacement</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>DISP=0*, disp-low and disp-high are absent</td>
</tr>
<tr>
<td>01</td>
<td>DISP=disp-low sign-extended to 16-bits, disp-high is absent</td>
</tr>
<tr>
<td>10</td>
<td>DISP=disp-high: disp-low</td>
</tr>
<tr>
<td>11</td>
<td>r/m is treated as a &quot;reg&quot; field</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>r/m</th>
<th>Operand Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>(BX) + (SI) + DISP</td>
</tr>
<tr>
<td>001</td>
<td>(BX) + (DI) + DISP</td>
</tr>
<tr>
<td>010</td>
<td>(BP) + (SI) + DISP</td>
</tr>
<tr>
<td>011</td>
<td>(BP) + (DI) + DISP</td>
</tr>
<tr>
<td>100</td>
<td>(SI) + DISP</td>
</tr>
<tr>
<td>101</td>
<td>(DI) + DISP</td>
</tr>
<tr>
<td>110</td>
<td>(BP) + DISP*</td>
</tr>
<tr>
<td>111</td>
<td>(BX) + DISP</td>
</tr>
</tbody>
</table>

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

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

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

Immediate to register/memory

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

Immediate to register

```
1 0 1 1 w reg data data if w=1
```

Memory to accumulator

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

Accumulator to memory

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

Register/memory to segment register

```
1 0 0 0 1 1 1 0 mod 0 reg r/m
```

Segment register to register/memory

```
1 0 0 0 1 1 0 0 mod 0 reg r/m
```

**PUSH** = Push
Register/memory

```
1 1 1 1 1 1 1 1 mod 1 1 0 r/m
```

Register

```
0 1 0 1 0 reg
```

Segment register

```
0 0 0 reg 1 1 0
```

**POP** = Pop
Register/memory

```
1 0 0 0 1 1 1 1 mod 0 0 0 r/m
```

Register

```
0 1 0 1 1 reg
```

Segment register

```
0 0 0 reg 1 1 1
```
XCHG = Exchange
Register/memory with register

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

Register with accumulator

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

IN = Input to AL/AX from
Fixed port

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

Variable port (DX)

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

OUT = Output from AL/AX to
Fixed port

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

Variable port (DX)

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

XLAT = Translate byte to AL

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

LEA = Load EA to register

```
1 0 0 0 1 1 1 0 1  mod  reg  r/m
```

LDS = Load pointer to DS

```
1 1 0 0 0 1 0 1  mod  reg  r/m
```

LES = Load pointer to ES

```
1 1 0 0 0 1 0 0  mod  reg  r/m
```

LAHF = Load AH with flags

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

SAHF = Store AH into flags

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

PUSHF = Push flags

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

POPF = Pop flags

```
1 0 0 1 1 1 0 1
```
## Arithmetic

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

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Mod</th>
<th>Reg</th>
<th>R/M</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000</td>
<td>d w</td>
<td>mod reg</td>
<td>r/m</td>
</tr>
</tbody>
</table>

Immediate to register/memory

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Mod</th>
<th>Reg</th>
<th>R/M</th>
<th>Data</th>
<th>Data if s:w=01</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000000</td>
<td>s w</td>
<td>mod 0 0 0</td>
<td>r/m</td>
<td>data</td>
<td>data if s:w=01</td>
</tr>
</tbody>
</table>

Immediate to accumulator

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Mod</th>
<th>Reg</th>
<th>R/M</th>
<th>Data</th>
<th>Data if w=1</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000100</td>
<td>w</td>
<td>mod</td>
<td>r/m</td>
<td>data</td>
<td>data if w=1</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Mod</th>
<th>Reg</th>
<th>R/M</th>
</tr>
</thead>
<tbody>
<tr>
<td>00010000</td>
<td>d w</td>
<td>mod reg</td>
<td>r/m</td>
</tr>
</tbody>
</table>

Immediate to register/memory

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Mod</th>
<th>Reg</th>
<th>R/M</th>
<th>Data</th>
<th>Data if s:w=01</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000000</td>
<td>s w</td>
<td>mod 0 1 0</td>
<td>r/m</td>
<td>data</td>
<td>data if s:w=01</td>
</tr>
</tbody>
</table>

Immediate to accumulator

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Mod</th>
<th>Reg</th>
<th>R/M</th>
<th>Data</th>
<th>Data if w=1</th>
</tr>
</thead>
<tbody>
<tr>
<td>00010100</td>
<td>w</td>
<td>mod</td>
<td>r/m</td>
<td>data</td>
<td>data if w=1</td>
</tr>
</tbody>
</table>

**INC** = Increment
Register/memory

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Mod</th>
<th>Reg</th>
<th>R/M</th>
</tr>
</thead>
<tbody>
<tr>
<td>11111111</td>
<td>w</td>
<td>mod 0 0 0</td>
<td>r/m</td>
</tr>
</tbody>
</table>

Register

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Reg</th>
</tr>
</thead>
<tbody>
<tr>
<td>010000</td>
<td>reg</td>
</tr>
</tbody>
</table>

**AAA** = ASCII adjust for add

<table>
<thead>
<tr>
<th>Opcode</th>
</tr>
</thead>
<tbody>
<tr>
<td>001110111</td>
</tr>
</tbody>
</table>

**DAA** = Decimal adjust for add

<table>
<thead>
<tr>
<th>Opcode</th>
</tr>
</thead>
<tbody>
<tr>
<td>00100111</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Mod</th>
<th>Reg</th>
<th>R/M</th>
</tr>
</thead>
<tbody>
<tr>
<td>00101010</td>
<td>d w</td>
<td>mod reg</td>
<td>r/m</td>
</tr>
</tbody>
</table>

Immediate from register/memory

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Mod</th>
<th>Reg</th>
<th>R/M</th>
<th>Data</th>
<th>Data if s:w=01</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000000</td>
<td>s w</td>
<td>mod 1 0 1</td>
<td>r/m</td>
<td>data</td>
<td>data if s:w=01</td>
</tr>
</tbody>
</table>

Immediate from accumulator

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Mod</th>
<th>Reg</th>
<th>R/M</th>
<th>Data</th>
<th>Data if w=1</th>
</tr>
</thead>
<tbody>
<tr>
<td>00101110</td>
<td>w</td>
<td>data</td>
<td>data if w=1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SBB = Subtract with borrow  
Register/memory and register to either

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

Immediate from accumulator

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

DEC = Decrement  
Register/memory

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

Register

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

NEG = Change sign

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

CMP = Compare  
Register/memory and register

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

Immediate with accumulator

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

AAS = ASCII adjust for subtract

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

MUL = Multiply (unsigned)

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

IMUL = Integer multiply (signed)

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

AAM = ASCII adjust for multiply

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

DIV = Divide (unsigned)

<table>
<thead>
<tr>
<th>1 1 1 1 0</th>
<th>1</th>
<th>1</th>
<th>w</th>
<th>mod</th>
<th>1 1 0</th>
<th>r/m</th>
</tr>
</thead>
</table>
IDIV = Integer divide (signed)
1 1 1 1 0 1 1 w mod 1 1 1 r/m

AAD = ASCII adjust for divide
1 1 0 1 0 1 0 1 0 0 0 0 1 0 1 0

CBW = Convert byte to word
1 0 0 1 1 0 0 0

CWD = Convert word to double word
1 0 0 1 1 0 0 1

Logic

NOT = Invert
1 1 1 1 0 1 1 w mod 0 1 0 r/m

SHL/SAL = Shift logical/arithmetic left
1 1 0 1 0 0 v w mod 1 0 0 r/m

SHR = Shift logical right
1 1 0 1 0 0 v w mod 1 0 1 r/m

SAR = Shift arithmetic right
1 1 0 1 0 0 v w mod 1 1 1 r/m

ROL = Rotate left
1 1 0 1 0 0 v w mod 0 0 0 r/m

ROR = Rotate right
1 1 0 1 0 0 v w mod 0 0 1 r/m

RCL = Rotate through carry left
1 1 0 1 0 0 v w mod 0 1 0 r/m

RCR = Rotate through carry right
1 1 0 1 0 0 v w mod 0 1 1 r/m

AND = And
Register/memory and register to either
0 0 1 0 0 0 d w mod reg r/m

Immediate to register/memory
1 0 0 0 0 0 0 w mod 1 0 0 r/m data data if w=1

Immediate to accumulator
0 0 1 0 0 1 0 w data data if w=1
**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 |

**MOV$** = 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

\[ \text{seg-low} \ \text{seg-high} \]

Indirect intersegment

\[ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \] mod 0 1 1 r/m

**JMP** = Unconditional Jump

Direct within segment

\[ 1 \ 1 \ 1 \ 0 \ 1 \ 0 \ 0 \ 1 \] disp-low disp-high

Direct within segment-short

\[ 1 \ 1 \ 1 \ 0 \ 1 \ 0 \ 1 \ 1 \] disp

Indirect within segment

\[ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \] mod 1 0 0 r/m

Direct intersegment

\[ 1 \ 1 \ 1 \ 0 \ 1 \ 0 \ 1 \ 0 \] offset-low offset-high

\[ \text{seg-low} \ \text{seg-high} \]

Indirect intersegment

\[ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \] mod 1 0 1 r/m
**RET** = Return from CALL

Within segment

| 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |

Within segment adding immediate to SP

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

Intersegment

| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |

Intersegment, adding immediate to SP

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

**JE/JZ** = Jump on equal/zero

| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | disp |

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

| 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | disp |

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

| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | disp |

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

| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | disp |

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

| 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | disp |

**JP/JPE** = Jump on parity/parity even

| 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | disp |

**JO** = Jump on overflow

| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | disp |

**JS** = Jump on sign

| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | disp |

**JNE/JNZ** = Jump on not equal/not zero

| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | disp |

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

<p>| 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | disp |</p>
<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
<th>Opcode</th>
<th>Disp.</th>
</tr>
</thead>
<tbody>
<tr>
<td>JNLE/JG</td>
<td>Jump on not less or equal/greater</td>
<td>01111111</td>
<td>disp</td>
</tr>
<tr>
<td>JNB/JAE</td>
<td>Jump on not below/above or equal</td>
<td>01110011</td>
<td>disp</td>
</tr>
<tr>
<td>JNBE/JA</td>
<td>Jump on not below or equal/above</td>
<td>01110111</td>
<td>disp</td>
</tr>
<tr>
<td>JNP/JPO</td>
<td>Jump on not parity/parity odd</td>
<td>01111011</td>
<td>disp</td>
</tr>
<tr>
<td>JNO</td>
<td>Jump on not overflow</td>
<td>01111001</td>
<td>disp</td>
</tr>
<tr>
<td>JNS</td>
<td>Jump on not sign</td>
<td>01111101</td>
<td>disp</td>
</tr>
<tr>
<td>LOOP</td>
<td>Loop CX times</td>
<td>11110010</td>
<td>disp</td>
</tr>
<tr>
<td>LOOPZ/LOOPE</td>
<td>Loop while zero/equal</td>
<td>11110001</td>
<td>disp</td>
</tr>
<tr>
<td>LOOPNZ/LOOPNE</td>
<td>Loop while not zero/not equal</td>
<td>11110000</td>
<td>disp</td>
</tr>
<tr>
<td>JCXZ</td>
<td>Jump on CX zero</td>
<td>11110011</td>
<td>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>&quot;equal&quot; or &quot;zero&quot;</td>
</tr>
<tr>
<td>JL or JNGE</td>
<td>(SF xor OF) = 1</td>
<td>&quot;less&quot; or &quot;not greater or equal&quot;</td>
</tr>
<tr>
<td>JLE or JNG</td>
<td>((SF xor OF) or ZF) = 1</td>
<td>&quot;less or equal&quot; or &quot;not greater&quot;</td>
</tr>
<tr>
<td>JB or JNAE or JC</td>
<td>CF = 1</td>
<td>&quot;below&quot; or &quot;not above or equal&quot;</td>
</tr>
<tr>
<td>JBE or JNA</td>
<td>(CF or ZF) = 1</td>
<td>&quot;below or equal&quot; or &quot;not above&quot;</td>
</tr>
<tr>
<td>JP or JPE</td>
<td>PF = 1</td>
<td>&quot;parity&quot; or &quot;parity even&quot;</td>
</tr>
<tr>
<td>JO</td>
<td>OF = 1</td>
<td>&quot;overflow&quot;</td>
</tr>
<tr>
<td>JS</td>
<td>SF = 1</td>
<td>&quot;sign&quot;</td>
</tr>
<tr>
<td>JNE or JNZ</td>
<td>ZF = 0</td>
<td>&quot;not equal&quot; or &quot;not zero&quot;</td>
</tr>
<tr>
<td>JNL or JGE</td>
<td>(SF xor OF) = 0</td>
<td>&quot;not less or greater or equal&quot;</td>
</tr>
<tr>
<td>JNLE or JG</td>
<td>((SF xor OF) or ZF) = 0</td>
<td>&quot;not less or equal&quot; or &quot;greater&quot;</td>
</tr>
<tr>
<td>JNB or JAE or JNC</td>
<td>CF = 0</td>
<td>&quot;not below&quot; or &quot;above or equal&quot;</td>
</tr>
<tr>
<td>JNBE or JA</td>
<td>(CF or ZF) = 0</td>
<td>&quot;not below or equal&quot; or &quot;above&quot;</td>
</tr>
<tr>
<td>JNP or JPO</td>
<td>PF = 0</td>
<td>&quot;not parity&quot; or &quot;parity odd&quot;</td>
</tr>
<tr>
<td>JNO</td>
<td>OF = 0</td>
<td>&quot;not overflow&quot;</td>
</tr>
<tr>
<td>JNS</td>
<td>SF = 0</td>
<td>&quot;not sign&quot;</td>
</tr>
</tbody>
</table>

**"Above" and "below" refer to the relation between two unsigned values, while "greater" and "less" refer to the relation between two signed values.**

**INT** = Interrupt  
Type specified

```
1 1 0 0 1 1 0 1 type
```

**Type 3**

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

**INTO** = Interrupt on overflow

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

**IRET** = Interrupt return

```
1 1 0 0 1 1 1 1
```
<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Binary Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLC</td>
<td>Clear carry</td>
<td>0111110000</td>
</tr>
<tr>
<td>CMC</td>
<td>Complement carry</td>
<td>011110101</td>
</tr>
<tr>
<td>CLD</td>
<td>Clear direction</td>
<td>011111100</td>
</tr>
<tr>
<td>CLI</td>
<td>Clear interrupt</td>
<td>0111111010</td>
</tr>
<tr>
<td>HLT</td>
<td>Halt</td>
<td>011110100</td>
</tr>
<tr>
<td>LOCK</td>
<td>Bus lock prefix</td>
<td>011110000</td>
</tr>
<tr>
<td>STC</td>
<td>Set carry</td>
<td>01111001</td>
</tr>
<tr>
<td>NOP</td>
<td>No operation</td>
<td>0010000000</td>
</tr>
<tr>
<td>STD</td>
<td>Set direction</td>
<td>011111101</td>
</tr>
<tr>
<td>STI</td>
<td>Set interrupt</td>
<td>011111011</td>
</tr>
<tr>
<td>ESC</td>
<td>Escape (to external device)</td>
<td>11011xx</td>
</tr>
</tbody>
</table>

Footnotes:
if \(d = 1\) then "to"; if \(d = 0\) then "from"
if \(w = 1\) then word instruction; if \(w = 0\) then byte instruction
if \(s:w = 01\) then 16 bits of immediate data from the operand
if \(s:w = 11\) then an immediate data byte is sign extended to form the 16-bit operand
if \(v = 1\) then "count" = 1; if \(v = 1\) then "count" in (CL)
\(x = \text{don't care}\)
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></td>
<td>ADD</td>
<td>ADD</td>
<td>ADD</td>
<td>ADD</td>
<td>ADD</td>
<td>PUSH</td>
<td>POP</td>
<td>ES</td>
</tr>
<tr>
<td>0</td>
<td>b,t,r/m</td>
<td>w,t,r/m</td>
<td>b,t,r/m</td>
<td>w,t,r/m</td>
<td>b,ia</td>
<td>w,ia</td>
<td>ES</td>
<td>POP</td>
</tr>
<tr>
<td>1</td>
<td>ADC</td>
<td>ADC</td>
<td>ADC</td>
<td>ADC</td>
<td>ADC</td>
<td>PUSH</td>
<td>POP</td>
<td>SS</td>
</tr>
<tr>
<td>2</td>
<td>AND</td>
<td>AND</td>
<td>AND</td>
<td>AND</td>
<td>AND</td>
<td>SEG</td>
<td>DAA</td>
<td>ES</td>
</tr>
<tr>
<td>3</td>
<td>XOR</td>
<td>XOR</td>
<td>XOR</td>
<td>XOR</td>
<td>SEG</td>
<td>AAA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>INC</td>
<td>INC</td>
<td>INC</td>
<td>INC</td>
<td>INC</td>
<td>INC</td>
<td>INC</td>
<td>INC</td>
</tr>
<tr>
<td>5</td>
<td>PUSH</td>
<td>PUSH</td>
<td>PUSH</td>
<td>PUSH</td>
<td>PUSH</td>
<td>PUSH</td>
<td>PUSH</td>
<td>PUSH</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</td>
<td>JNO</td>
<td>JB/ JNBE</td>
<td>JNB/ JAE</td>
<td>JE/ JNE</td>
<td>JBE/ JNA</td>
<td>JNO</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Immed</td>
<td>Immed</td>
<td>Immed</td>
<td>Immed</td>
<td>TEST</td>
<td>TEST</td>
<td>XCHG</td>
<td>XCHG</td>
</tr>
<tr>
<td>9</td>
<td>NOP</td>
<td>XCHG</td>
<td>XCHG</td>
<td>XCHG</td>
<td>XCHG</td>
<td>XCHG</td>
<td>XCHG</td>
<td>XCHG</td>
</tr>
<tr>
<td></td>
<td>AX</td>
<td>PUSH</td>
<td>CX</td>
<td>PUSH</td>
<td>CX</td>
<td>PUSH</td>
<td>CX</td>
<td>PUSH</td>
</tr>
<tr>
<td>A</td>
<td>MOV</td>
<td>MOV</td>
<td>MOV</td>
<td>MOV</td>
<td>MOV</td>
<td>CMPS</td>
<td>CMPS</td>
<td>CMPS</td>
</tr>
<tr>
<td>B</td>
<td>MOV</td>
<td>MOV</td>
<td>MOV</td>
<td>MOV</td>
<td>MOV</td>
<td>MOV</td>
<td>MOV</td>
<td>MOV</td>
</tr>
<tr>
<td>C</td>
<td>RET</td>
<td>RET</td>
<td>LES</td>
<td>LDS</td>
<td>MOV</td>
<td>MOV</td>
<td>MOV</td>
<td>MOV</td>
</tr>
<tr>
<td>D</td>
<td>Shift</td>
<td>Shift</td>
<td>Shift</td>
<td>AAM</td>
<td>AAD</td>
<td>XLAT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>LOOPNZ/ LOOPNE</td>
<td>LOOPZ/ LOOPE</td>
<td>LOOP</td>
<td>JCXZ</td>
<td>IN</td>
<td>OUT</td>
<td>OUT</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>LOCK</td>
<td>REP</td>
<td>REP</td>
<td>HLT</td>
<td>CMC</td>
<td>Grp 1</td>
<td>Grp 1</td>
<td>w,r/m</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>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 b,f,r/m</td>
<td>OR b,w,r/m</td>
<td>OR b,i</td>
<td>OR w.i</td>
<td>PUSH CS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>SBB b,f,r/m</td>
<td>SBB b,t,r/m</td>
<td>SBB w,t,r/m</td>
<td>SBB b,i</td>
<td>SBB w.i</td>
<td>PUSHD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>SUB b,f,r/m</td>
<td>SUB b,t,r/m</td>
<td>SUB w,t,r/m</td>
<td>SUB b,i</td>
<td>SUB w.i</td>
<td>SEG=CS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>CMP b,f,r/m</td>
<td>CMP b,t,r/m</td>
<td>CMP w,t,r/m</td>
<td>CMP b,i</td>
<td>CMP w.i</td>
<td>SEG=CS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>DEC AX</td>
<td>DEC CX</td>
<td>DEC DX</td>
<td>DEC BX</td>
<td>DEC SP</td>
<td>DEC BP</td>
<td>DEC SI</td>
<td>DEC DI</td>
</tr>
<tr>
<td>5</td>
<td>POP AX</td>
<td>POP CX</td>
<td>POP DX</td>
<td>POP BX</td>
<td>POP SP</td>
<td>POP BP</td>
<td>POP SI</td>
<td>POP 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>JS</td>
<td>JNS</td>
<td>JP/ JPE</td>
<td>JNP/ JPO</td>
<td>JL/ JNGE</td>
<td>JNL/ JGE</td>
<td>JLE/ JNG</td>
<td>JNLE/ JG</td>
</tr>
<tr>
<td>8</td>
<td>MOV b,f,r/m</td>
<td>MOV b,t,r/m</td>
<td>MOV w,t,r/m</td>
<td>MOV s,r,t/r/m</td>
<td>LEA</td>
<td>MOV s,r,f,r/m</td>
<td>POP r/m</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>CBW</td>
<td>CWD</td>
<td>CALL l,d</td>
<td>WAIT</td>
<td>PUSHF</td>
<td>POPF</td>
<td>SAHF</td>
<td>LAHF</td>
</tr>
<tr>
<td>A</td>
<td>TEST b,i</td>
<td>TEST w,i</td>
<td>STOS b</td>
<td>STOS w</td>
<td>LODS b</td>
<td>LODS w</td>
<td>SCAS b</td>
<td>SCAS w</td>
</tr>
<tr>
<td>B</td>
<td>MOV AX</td>
<td>MOV i CX</td>
<td>MOV i DX</td>
<td>MOV i BX</td>
<td>MOV i SP</td>
<td>MOV i BP</td>
<td>MOV i SI</td>
<td>MOV i DI</td>
</tr>
<tr>
<td>C</td>
<td>RET l,(i+SP)</td>
<td>RET l</td>
<td>INT Type 3</td>
<td>INTO</td>
<td>IRET</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>ESC 0</td>
<td>ESC 1</td>
<td>ESC 2</td>
<td>ESC 3</td>
<td>ESC 4</td>
<td>ESC 5</td>
<td>ESC 6</td>
<td>ESC 7</td>
</tr>
<tr>
<td>E</td>
<td>CALL d</td>
<td>JMP d</td>
<td>JMP l,d</td>
<td>JMP si,d</td>
<td>IN v,b</td>
<td>IN v.w</td>
<td>OUT v,b</td>
<td>OUT v.w</td>
</tr>
<tr>
<td>F</td>
<td>CLC</td>
<td>STC</td>
<td>CLI</td>
<td>STI</td>
<td>CLD</td>
<td>STD</td>
<td>Grp 2 b,f,r/m</td>
<td>Grp 2 w,r/m</td>
</tr>
</tbody>
</table>

where:

<table>
<thead>
<tr>
<th>mod</th>
<th>000</th>
<th>001</th>
<th>010</th>
<th>011</th>
<th>100</th>
<th>101</th>
<th>110</th>
<th>111</th>
</tr>
</thead>
<tbody>
<tr>
<td>Immed</td>
<td>ADD</td>
<td>OR</td>
<td>ADC</td>
<td>SBB</td>
<td>AND</td>
<td>SUB</td>
<td>XOR</td>
<td>CMP</td>
</tr>
<tr>
<td>Shift</td>
<td>ROL</td>
<td>ROR</td>
<td>RCL</td>
<td>RCR</td>
<td>SHL/SAL</td>
<td>SHR</td>
<td>—</td>
<td>SAR</td>
</tr>
<tr>
<td>Grp 1</td>
<td>TEST</td>
<td>—</td>
<td>NOT</td>
<td>NEG</td>
<td>MUL</td>
<td>IMUL</td>
<td>DIV</td>
<td>IDIV</td>
</tr>
<tr>
<td>Grp 2</td>
<td>INC</td>
<td>DEC</td>
<td>CALL</td>
<td>CALL l,id</td>
<td>JMP</td>
<td>JMP l,id</td>
<td>PUSH</td>
<td>—</td>
</tr>
</tbody>
</table>

---

8088 Instruction Reference B-17
<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Page</th>
<th>Mnemonic</th>
<th>Page</th>
<th>Mnemonic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAA</td>
<td>B-7</td>
<td>JG</td>
<td>B-13</td>
<td>MOV</td>
<td>B-5</td>
</tr>
<tr>
<td>AAD</td>
<td>B-9</td>
<td>JGE</td>
<td>B-12</td>
<td>MOVVS</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>JNS</td>
<td>B-13</td>
<td>PUSH</td>
<td>B-5</td>
</tr>
<tr>
<td>CLI</td>
<td>B-15</td>
<td>JNG</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>

8088 Instruction Reference
## APPENDIX C: OF CHARACTERS, KEYSTROKES, AND COLOR

<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></td>
<td></td>
<td></td>
<td>Background</td>
<td>Foreground</td>
</tr>
<tr>
<td>00</td>
<td>0</td>
<td>Blank (Null)</td>
<td>Ctrl 2</td>
<td>Black</td>
</tr>
<tr>
<td>01</td>
<td>1</td>
<td>Control A</td>
<td></td>
<td>Black</td>
</tr>
<tr>
<td>02</td>
<td>2</td>
<td>Control B</td>
<td>Ctrl C</td>
<td>Black</td>
</tr>
<tr>
<td>03</td>
<td>3</td>
<td>Control C</td>
<td></td>
<td>Black</td>
</tr>
<tr>
<td>04</td>
<td>4</td>
<td>Control D</td>
<td>Ctrl E</td>
<td>Black</td>
</tr>
<tr>
<td>05</td>
<td>5</td>
<td>Control E</td>
<td>Black</td>
<td>Magenta</td>
</tr>
<tr>
<td>06</td>
<td>6</td>
<td>Control F</td>
<td>Black</td>
<td>Brown</td>
</tr>
<tr>
<td>07</td>
<td>7</td>
<td>Control G</td>
<td>Black</td>
<td>Light Grey</td>
</tr>
<tr>
<td>08</td>
<td>8</td>
<td>Control H, Backspace, Shift Backspace</td>
<td>Black</td>
<td>Dark Grey</td>
</tr>
<tr>
<td>09</td>
<td>9</td>
<td>Control I</td>
<td>Black</td>
<td>Light Blue</td>
</tr>
<tr>
<td>0A</td>
<td>10</td>
<td>Control J, Shift Control J</td>
<td>Black</td>
<td>Light Green</td>
</tr>
<tr>
<td>0B</td>
<td>11</td>
<td>Control K</td>
<td>Black</td>
<td>Light Green</td>
</tr>
<tr>
<td>0C</td>
<td>12</td>
<td>Control L, Shift Control L</td>
<td>Black</td>
<td>Light Red</td>
</tr>
<tr>
<td>0D</td>
<td>13</td>
<td>Control M, Shift Control M</td>
<td>Black</td>
<td>Light Magenta</td>
</tr>
<tr>
<td>0E</td>
<td>14</td>
<td>Control N</td>
<td>Black</td>
<td>Yellow</td>
</tr>
<tr>
<td>0F</td>
<td>15</td>
<td>Control O</td>
<td>Black</td>
<td>White</td>
</tr>
<tr>
<td>10</td>
<td>16</td>
<td>Control P</td>
<td>Blue</td>
<td>Black</td>
</tr>
<tr>
<td>11</td>
<td>17</td>
<td>Control Q</td>
<td>Blue</td>
<td>Blue</td>
</tr>
<tr>
<td>12</td>
<td>18</td>
<td>Control R</td>
<td>Blue</td>
<td>Green</td>
</tr>
<tr>
<td>13</td>
<td>19</td>
<td>Control S</td>
<td>Blue</td>
<td>Cyan</td>
</tr>
<tr>
<td>14</td>
<td>20</td>
<td>Control T</td>
<td>Blue</td>
<td>Red</td>
</tr>
<tr>
<td>15</td>
<td>21</td>
<td>Control U</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>22</td>
<td>Control V</td>
<td>Blue</td>
<td>Brown</td>
</tr>
<tr>
<td>17</td>
<td>23</td>
<td>Control W</td>
<td>Blue</td>
<td>Light Grey</td>
</tr>
<tr>
<td>Value</td>
<td>As Characters</td>
<td>As Text Attributes</td>
<td>IBM Monochrome Monitor Adapter</td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>---------------</td>
<td>--------------------</td>
<td>-------------------------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Color/Graphics Monitor Adapter</td>
<td>Display Adapter</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Background</td>
<td>Foreground</td>
<td>HighIntensity</td>
</tr>
<tr>
<td>18</td>
<td>24</td>
<td>†</td>
<td>Ctrl X</td>
<td>Blue</td>
</tr>
<tr>
<td>19</td>
<td>25</td>
<td></td>
<td>Ctrl Y</td>
<td>Blue</td>
</tr>
<tr>
<td>1A</td>
<td>26</td>
<td></td>
<td>Ctrl Z</td>
<td>Blue</td>
</tr>
<tr>
<td>1B</td>
<td>27</td>
<td></td>
<td>Ctrl [, Shift Esc, Ctrl Esc</td>
<td>Blue</td>
</tr>
<tr>
<td>1C</td>
<td>28</td>
<td></td>
<td>Ctrl \</td>
<td>Blue</td>
</tr>
<tr>
<td>1D</td>
<td>29</td>
<td></td>
<td>Ctrl ]</td>
<td>Blue</td>
</tr>
<tr>
<td>1E</td>
<td>30</td>
<td>▲</td>
<td>Ctrl 6</td>
<td>Blue</td>
</tr>
<tr>
<td>1F</td>
<td>31</td>
<td></td>
<td>Ctrl —</td>
<td>Blue</td>
</tr>
<tr>
<td>20</td>
<td>32</td>
<td>Blank Space</td>
<td>Space Bar, Shift, Space, Ctrl Space, Alt Space</td>
<td>Green</td>
</tr>
<tr>
<td>21</td>
<td>33</td>
<td>!</td>
<td>Shift</td>
<td>Green</td>
</tr>
<tr>
<td>22</td>
<td>34</td>
<td>&quot;</td>
<td>Shift</td>
<td>Green</td>
</tr>
<tr>
<td>23</td>
<td>35</td>
<td>#</td>
<td>Shift</td>
<td>Green</td>
</tr>
<tr>
<td>24</td>
<td>36</td>
<td>$</td>
<td>Shift</td>
<td>Green</td>
</tr>
<tr>
<td>25</td>
<td>37</td>
<td>%</td>
<td>Shift</td>
<td>Green</td>
</tr>
<tr>
<td>26</td>
<td>38</td>
<td>&amp;</td>
<td>Shift</td>
<td>Green</td>
</tr>
<tr>
<td>27</td>
<td>39</td>
<td>'</td>
<td>Shift</td>
<td>Green</td>
</tr>
<tr>
<td>28</td>
<td>40</td>
<td>(</td>
<td>Shift</td>
<td>Green</td>
</tr>
<tr>
<td>29</td>
<td>41</td>
<td>)</td>
<td>Shift</td>
<td>Green</td>
</tr>
<tr>
<td>2A</td>
<td>42</td>
<td>*</td>
<td>Shift</td>
<td>Green</td>
</tr>
<tr>
<td>2B</td>
<td>43</td>
<td>+</td>
<td>Shift</td>
<td>Green</td>
</tr>
<tr>
<td>2C</td>
<td>44</td>
<td></td>
<td>Shift</td>
<td>Green</td>
</tr>
<tr>
<td>2D</td>
<td>45</td>
<td></td>
<td>Shift</td>
<td>Green</td>
</tr>
<tr>
<td>2E</td>
<td>46</td>
<td></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>Keystrokes</th>
<th>Modes</th>
<th>Background</th>
<th>Foreground</th>
<th>IBM Monitor Adapter</th>
<th>IBM Monochrome Display Adapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>2F</td>
<td>47</td>
<td>/</td>
<td>/</td>
<td>Note 3</td>
<td>Green</td>
<td>White</td>
<td>High Intensity</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>48</td>
<td>0</td>
<td>0</td>
<td>Note 3</td>
<td>Cyan</td>
<td>Black</td>
<td>Normal</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>49</td>
<td>1</td>
<td>1</td>
<td>Note 3</td>
<td>Cyan</td>
<td>Blue</td>
<td>Underline</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>50</td>
<td>2</td>
<td>2</td>
<td>Note 3</td>
<td>Cyan</td>
<td>Green</td>
<td>Normal</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>51</td>
<td>3</td>
<td>3</td>
<td>Note 3</td>
<td>Cyan</td>
<td>Cyan</td>
<td>Normal</td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>52</td>
<td>4</td>
<td>4</td>
<td>Note 3</td>
<td>Cyan</td>
<td>Red</td>
<td>Normal</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>53</td>
<td>5</td>
<td>5</td>
<td>Note 3</td>
<td>Cyan</td>
<td>Magenta</td>
<td>Normal</td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>54</td>
<td>6</td>
<td>6</td>
<td>Note 3</td>
<td>Cyan</td>
<td>Brown</td>
<td>Normal</td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>55</td>
<td>7</td>
<td>7</td>
<td>Note 3</td>
<td>Cyan</td>
<td>Light Grey</td>
<td>Normal</td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>56</td>
<td>8</td>
<td>8</td>
<td>Note 3</td>
<td>Cyan</td>
<td>Dark Grey</td>
<td>High Intensity</td>
<td></td>
</tr>
<tr>
<td>39</td>
<td>57</td>
<td>9</td>
<td>9</td>
<td>Note 3</td>
<td>Cyan</td>
<td>Light Blue</td>
<td>High Intensity Underline</td>
<td></td>
</tr>
<tr>
<td>3A</td>
<td>58</td>
<td>:</td>
<td>:</td>
<td>Shift</td>
<td>Cyan</td>
<td>Light Green</td>
<td>High Intensity</td>
<td></td>
</tr>
<tr>
<td>3B</td>
<td>59</td>
<td>;</td>
<td>;</td>
<td>Shift</td>
<td>Cyan</td>
<td>Light Cyan</td>
<td>High Intensity</td>
<td></td>
</tr>
<tr>
<td>3C</td>
<td>60</td>
<td>&lt;</td>
<td>&lt;</td>
<td>Shift</td>
<td>Cyan</td>
<td>Light Red</td>
<td>High Intensity</td>
<td></td>
</tr>
<tr>
<td>3D</td>
<td>61</td>
<td>=</td>
<td>=</td>
<td>Shift</td>
<td>Cyan</td>
<td>Light Magenta</td>
<td>High Intensity</td>
<td></td>
</tr>
<tr>
<td>3E</td>
<td>62</td>
<td>&gt;</td>
<td>&gt;</td>
<td>Shift</td>
<td>Cyan</td>
<td>Yellow</td>
<td>High Intensity</td>
<td></td>
</tr>
<tr>
<td>3F</td>
<td>63</td>
<td>?</td>
<td>?</td>
<td>Shift</td>
<td>Cyan</td>
<td>White</td>
<td>High Intensity</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>64</td>
<td>@</td>
<td>@</td>
<td>Shift</td>
<td>Red</td>
<td>Black</td>
<td>Normal</td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>65</td>
<td>A</td>
<td>A</td>
<td>Note 4</td>
<td>Red</td>
<td>Blue</td>
<td>Underline</td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>66</td>
<td>B</td>
<td>B</td>
<td>Note 4</td>
<td>Red</td>
<td>Green</td>
<td>Normal</td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>67</td>
<td>C</td>
<td>C</td>
<td>Note 4</td>
<td>Red</td>
<td>Cyan</td>
<td>Normal</td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>68</td>
<td>D</td>
<td>D</td>
<td>Note 4</td>
<td>Red</td>
<td>Red</td>
<td>Normal</td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>69</td>
<td>E</td>
<td>E</td>
<td>Note 4</td>
<td>Red</td>
<td>Magenta</td>
<td>Normal</td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>70</td>
<td>F</td>
<td>F</td>
<td>Note 4</td>
<td>Red</td>
<td>Brown</td>
<td>Normal</td>
<td></td>
</tr>
<tr>
<td>47</td>
<td>71</td>
<td>G</td>
<td>G</td>
<td>Note 4</td>
<td>Red</td>
<td>Light Grey</td>
<td>Normal</td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>72</td>
<td>H</td>
<td>H</td>
<td>Note 4</td>
<td>Red</td>
<td>Dark Grey</td>
<td>High Intensity</td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>73</td>
<td>I</td>
<td>I</td>
<td>Note 4</td>
<td>Red</td>
<td>Light Blue</td>
<td>High Intensity Underline</td>
<td></td>
</tr>
<tr>
<td>4A</td>
<td>74</td>
<td>J</td>
<td>J</td>
<td>Note 4</td>
<td>Red</td>
<td>Light Green</td>
<td>High Intensity</td>
<td></td>
</tr>
</tbody>
</table>

Of Characters, Keystrokes, and Colors  C-3
<table>
<thead>
<tr>
<th>Value</th>
<th>As Characters</th>
<th>Keystrokes</th>
<th>Modes</th>
<th>As Text Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Color/Graphics Monitor Adapter</strong></td>
<td><strong>IBM Monochrome Display Adapter</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4B 75</td>
<td>K K Note 4</td>
<td>Red Light Cyan High Intensity</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4C 76</td>
<td>L L Note 4</td>
<td>Red Light Red High Intensity</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4D 77</td>
<td>M M Note 4</td>
<td>Red Light Magenta High Intensity</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4E 78</td>
<td>N N Note 4</td>
<td>Red Yellow High Intensity</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4F 79</td>
<td>O O Note 4</td>
<td>Red White High Intensity</td>
<td></td>
<td></td>
</tr>
<tr>
<td>50 80</td>
<td>P P Note 4</td>
<td>Magenta Black Normal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>51 81</td>
<td>Q Q Note 4</td>
<td>Magenta Blue Underline</td>
<td></td>
<td></td>
</tr>
<tr>
<td>52 82</td>
<td>R R Note 4</td>
<td>Magenta Green Normal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>53 83</td>
<td>S S Note 4</td>
<td>Magenta Cyan Normal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>54 84</td>
<td>T T Note 4</td>
<td>Magenta Red Normal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>55 85</td>
<td>U U Note 4</td>
<td>Magenta Magenta Normal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>56 86</td>
<td>V V Note 4</td>
<td>Magenta Brown Normal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>57 87</td>
<td>W W Note 4</td>
<td>Magenta Light Grey Normal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>58 88</td>
<td>X X Note 4</td>
<td>Magenta Dark Grey High Intensity</td>
<td></td>
<td></td>
</tr>
<tr>
<td>59 89</td>
<td>Y Y Note 4</td>
<td>Magenta Light Blue High Intensity Underline</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5A 90</td>
<td>Z Z Note 4</td>
<td>Magenta Light Green High Intensity</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5B 91</td>
<td>[ ] Magenta Light Cyan High Intensity</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5C 92</td>
<td>\ \ Magenta Light Red High Intensity</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5D 93</td>
<td>} } Magenta Light Magenta High Intensity</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5E 94</td>
<td>^ ^ Shift Magenta Yellow High Intensity</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5F 95</td>
<td>— — Shift Magenta White High Intensity</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60 96</td>
<td>. . Yellow Black Normal</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>61 97</td>
<td>a a Note 5</td>
<td>Yellow Blue Underline</td>
<td></td>
<td></td>
</tr>
<tr>
<td>62 98</td>
<td>b b Note 5</td>
<td>Yellow Green Normal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>63 99</td>
<td>c c Note 5</td>
<td>Yellow Cyan Normal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>64 100</td>
<td>d d Note 5</td>
<td>Yellow Red Normal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>65 101</td>
<td>e e Note 5</td>
<td>Yellow Magenta Normal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>66 102</td>
<td>f f Note 5</td>
<td>Yellow 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>As Text Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Hex</strong></td>
<td><strong>Dec</strong></td>
<td><strong>Symbol</strong></td>
</tr>
<tr>
<td>67</td>
<td>103</td>
<td>g</td>
</tr>
<tr>
<td>68</td>
<td>104</td>
<td>h</td>
</tr>
<tr>
<td>69</td>
<td>105</td>
<td>i</td>
</tr>
<tr>
<td>6A</td>
<td>106</td>
<td>j</td>
</tr>
<tr>
<td>6B</td>
<td>107</td>
<td>k</td>
</tr>
<tr>
<td>6C</td>
<td>108</td>
<td>l</td>
</tr>
<tr>
<td>6D</td>
<td>109</td>
<td>m</td>
</tr>
<tr>
<td>6E</td>
<td>110</td>
<td>n</td>
</tr>
<tr>
<td>6F</td>
<td>111</td>
<td>o</td>
</tr>
<tr>
<td>70</td>
<td>112</td>
<td>p</td>
</tr>
<tr>
<td>71</td>
<td>113</td>
<td>q</td>
</tr>
<tr>
<td>72</td>
<td>114</td>
<td>r</td>
</tr>
<tr>
<td>73</td>
<td>115</td>
<td>s</td>
</tr>
<tr>
<td>74</td>
<td>116</td>
<td>f</td>
</tr>
<tr>
<td>75</td>
<td>117</td>
<td>u</td>
</tr>
<tr>
<td>76</td>
<td>118</td>
<td>v</td>
</tr>
<tr>
<td>77</td>
<td>119</td>
<td>w</td>
</tr>
<tr>
<td>78</td>
<td>120</td>
<td>x</td>
</tr>
<tr>
<td>79</td>
<td>121</td>
<td>y</td>
</tr>
<tr>
<td>7A</td>
<td>122</td>
<td>z</td>
</tr>
<tr>
<td>7B</td>
<td>123</td>
<td></td>
</tr>
<tr>
<td>7C</td>
<td>124</td>
<td></td>
</tr>
<tr>
<td>7D</td>
<td>125</td>
<td></td>
</tr>
<tr>
<td>7E</td>
<td>126</td>
<td></td>
</tr>
<tr>
<td>7F</td>
<td>127</td>
<td></td>
</tr>
<tr>
<td>Value</td>
<td>As Characters</td>
<td>As Text Attributes</td>
</tr>
<tr>
<td>-------</td>
<td>---------------</td>
<td>--------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td><img src="image" alt="Table of Values" /></td>
</tr>
</tbody>
</table>

**C-6** Of Characters, Keystrokes, and Colors
<table>
<thead>
<tr>
<th>Value</th>
<th>As Characters</th>
<th>As Text Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>9B 155</td>
<td>$</td>
<td>Hex Dec</td>
</tr>
<tr>
<td>9C 156</td>
<td>£</td>
<td>156</td>
</tr>
<tr>
<td>9D 157</td>
<td>¥</td>
<td>157</td>
</tr>
<tr>
<td>9E 158</td>
<td>Pt</td>
<td>158</td>
</tr>
<tr>
<td>9F 159</td>
<td>¥</td>
<td>159</td>
</tr>
<tr>
<td>A0 160</td>
<td>á</td>
<td>160</td>
</tr>
<tr>
<td>A1 161</td>
<td>i</td>
<td>161</td>
</tr>
<tr>
<td>A2 162</td>
<td>ó</td>
<td>162</td>
</tr>
<tr>
<td>A3 163</td>
<td>ú</td>
<td>163</td>
</tr>
<tr>
<td>A4 164</td>
<td>ñ</td>
<td>164</td>
</tr>
<tr>
<td>A5 165</td>
<td>Ñ</td>
<td>165</td>
</tr>
<tr>
<td>A6 166</td>
<td>a</td>
<td>166</td>
</tr>
<tr>
<td>A7 167</td>
<td>ö</td>
<td>167</td>
</tr>
<tr>
<td>A8 168</td>
<td>ç</td>
<td>168</td>
</tr>
<tr>
<td>A9 169</td>
<td>¨</td>
<td>169</td>
</tr>
<tr>
<td>AA 170</td>
<td>→</td>
<td>170</td>
</tr>
<tr>
<td>AB 171</td>
<td>½</td>
<td>171</td>
</tr>
<tr>
<td>AC 172</td>
<td>¼</td>
<td>172</td>
</tr>
<tr>
<td>AD 173</td>
<td>ⅈ</td>
<td>173</td>
</tr>
<tr>
<td>AE 174</td>
<td>&lt;&lt;</td>
<td>174</td>
</tr>
<tr>
<td>AF 175</td>
<td>&gt;&gt;</td>
<td>175</td>
</tr>
<tr>
<td>B0 176</td>
<td>⏯</td>
<td>176</td>
</tr>
<tr>
<td>B1 177</td>
<td>⏰</td>
<td>177</td>
</tr>
<tr>
<td>B2 178</td>
<td>⏰</td>
<td>178</td>
</tr>
<tr>
<td>B3 179</td>
<td>⏯</td>
<td>179</td>
</tr>
<tr>
<td>B4 180</td>
<td></td>
<td>180</td>
</tr>
<tr>
<td>B5 181</td>
<td></td>
<td>181</td>
</tr>
<tr>
<td>B6 182</td>
<td></td>
<td>182</td>
</tr>
<tr>
<td>Value</td>
<td>As Characters</td>
<td>As Text Attributes</td>
</tr>
<tr>
<td>-------</td>
<td>--------------</td>
<td>--------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B7 183</td>
<td>Alt 183</td>
<td>Note 6</td>
</tr>
<tr>
<td>B8 184</td>
<td>Alt 184</td>
<td>Note 6</td>
</tr>
<tr>
<td>B9 185</td>
<td>Alt 185</td>
<td>Note 6</td>
</tr>
<tr>
<td>BA 186</td>
<td>Alt 186</td>
<td>Note 6</td>
</tr>
<tr>
<td>BB 187</td>
<td>Alt 187</td>
<td>Note 6</td>
</tr>
<tr>
<td>BC 188</td>
<td>Alt 188</td>
<td>Note 6</td>
</tr>
<tr>
<td>BD 189</td>
<td>Alt 189</td>
<td>Note 6</td>
</tr>
<tr>
<td>BE 190</td>
<td>Alt 190</td>
<td>Note 6</td>
</tr>
<tr>
<td>BF 191</td>
<td>Alt 191</td>
<td>Note 6</td>
</tr>
<tr>
<td>C0 192</td>
<td>Alt 192</td>
<td>Note 6</td>
</tr>
<tr>
<td>C1 193</td>
<td>Alt 193</td>
<td>Note 6</td>
</tr>
<tr>
<td>C2 194</td>
<td>Alt 194</td>
<td>Note 6</td>
</tr>
<tr>
<td>C3 195</td>
<td>Alt 195</td>
<td>Note 6</td>
</tr>
<tr>
<td>C4 196</td>
<td>Alt 196</td>
<td>Note 6</td>
</tr>
<tr>
<td>C5 197</td>
<td>Alt 197</td>
<td>Note 6</td>
</tr>
<tr>
<td>C6 198</td>
<td>Alt 198</td>
<td>Note 6</td>
</tr>
<tr>
<td>C7 199</td>
<td>Alt 199</td>
<td>Note 6</td>
</tr>
<tr>
<td>C8 200</td>
<td>Alt 200</td>
<td>Note 6</td>
</tr>
<tr>
<td>C9 201</td>
<td>Alt 201</td>
<td>Note 6</td>
</tr>
<tr>
<td>CA 202</td>
<td>Alt 202</td>
<td>Note 6</td>
</tr>
<tr>
<td>CB 203</td>
<td>Alt 203</td>
<td>Note 6</td>
</tr>
<tr>
<td>CC 204</td>
<td>Alt 204</td>
<td>Note 6</td>
</tr>
<tr>
<td>CD 205</td>
<td>Alt 205</td>
<td>Note 6</td>
</tr>
<tr>
<td>CE 206</td>
<td>Alt 206</td>
<td>Note 6</td>
</tr>
<tr>
<td>CF 207</td>
<td>Alt 207</td>
<td>Note 6</td>
</tr>
<tr>
<td>D0 208</td>
<td>Alt 208</td>
<td>Note 6</td>
</tr>
</tbody>
</table>

C-8 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></td>
<td></td>
<td></td>
<td>Background</td>
<td>Foreground</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hex</td>
<td>Dec</td>
<td>Symbol</td>
<td>Keystrokes</td>
<td>Modes</td>
</tr>
<tr>
<td>D1</td>
<td>209</td>
<td>Alt 209</td>
<td>Note 6</td>
<td>Magenta</td>
</tr>
<tr>
<td>D2</td>
<td>210</td>
<td>Alt 210</td>
<td>Note 6</td>
<td>Magenta</td>
</tr>
<tr>
<td>D3</td>
<td>211</td>
<td>Alt 211</td>
<td>Note 6</td>
<td>Magenta</td>
</tr>
<tr>
<td>D4</td>
<td>212</td>
<td>Alt 212</td>
<td>Note 6</td>
<td>Magenta</td>
</tr>
<tr>
<td>D5</td>
<td>213</td>
<td>Alt 213</td>
<td>Note 6</td>
<td>Magenta</td>
</tr>
<tr>
<td>D6</td>
<td>214</td>
<td>Alt 214</td>
<td>Note 6</td>
<td>Magenta</td>
</tr>
<tr>
<td>D7</td>
<td>215</td>
<td>Alt 215</td>
<td>Note 6</td>
<td>Magenta</td>
</tr>
<tr>
<td>D8</td>
<td>216</td>
<td>Alt 216</td>
<td>Note 6</td>
<td>Magenta</td>
</tr>
<tr>
<td>D9</td>
<td>217</td>
<td>Alt 217</td>
<td>Note 6</td>
<td>Magenta</td>
</tr>
<tr>
<td>DA</td>
<td>218</td>
<td>Alt 218</td>
<td>Note 6</td>
<td>Magenta</td>
</tr>
<tr>
<td>DB</td>
<td>219</td>
<td>Alt 219</td>
<td>Note 6</td>
<td>Magenta</td>
</tr>
<tr>
<td>DC</td>
<td>220</td>
<td>Alt 220</td>
<td>Note 6</td>
<td>Magenta</td>
</tr>
<tr>
<td>DD</td>
<td>221</td>
<td>Alt 221</td>
<td>Note 6</td>
<td>Magenta</td>
</tr>
<tr>
<td>DE</td>
<td>222</td>
<td>Alt 222</td>
<td>Note 6</td>
<td>Magenta</td>
</tr>
<tr>
<td>DF</td>
<td>223</td>
<td>Alt 223</td>
<td>Note 6</td>
<td>Magenta</td>
</tr>
<tr>
<td>E0</td>
<td>224</td>
<td>(\alpha)</td>
<td>Alt 224</td>
<td>Note 6</td>
</tr>
<tr>
<td>E1</td>
<td>225</td>
<td>(\beta)</td>
<td>Alt 225</td>
<td>Note 6</td>
</tr>
<tr>
<td>E2</td>
<td>226</td>
<td>(\Gamma)</td>
<td>Alt 226</td>
<td>Note 6</td>
</tr>
<tr>
<td>E3</td>
<td>227</td>
<td>(\pi)</td>
<td>Alt 227</td>
<td>Note 6</td>
</tr>
<tr>
<td>E4</td>
<td>228</td>
<td>(\Sigma)</td>
<td>Alt 228</td>
<td>Note 6</td>
</tr>
<tr>
<td>E5</td>
<td>229</td>
<td>(\sigma)</td>
<td>Alt 229</td>
<td>Note 6</td>
</tr>
<tr>
<td>E6</td>
<td>230</td>
<td>(\mu)</td>
<td>Alt 230</td>
<td>Note 6</td>
</tr>
<tr>
<td>E7</td>
<td>231</td>
<td>(\tau)</td>
<td>Alt 231</td>
<td>Note 6</td>
</tr>
<tr>
<td>E8</td>
<td>232</td>
<td>(\phi)</td>
<td>Alt 232</td>
<td>Note 6</td>
</tr>
<tr>
<td>E9</td>
<td>233</td>
<td>(\theta)</td>
<td>Alt 233</td>
<td>Note 6</td>
</tr>
<tr>
<td>EA</td>
<td>234</td>
<td>(\Omega)</td>
<td>Alt 234</td>
<td>Note 6</td>
</tr>
<tr>
<td>EB</td>
<td>235</td>
<td>(\delta)</td>
<td>Alt 235</td>
<td>Note 6</td>
</tr>
<tr>
<td>Value</td>
<td>As Characters</td>
<td>Color/Graphics Monitor Adapter</td>
<td>IBM Monochrome Monitor Adapter</td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>---------------</td>
<td>-------------------------------</td>
<td>--------------------------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Background</td>
<td>Foreground</td>
<td>Display Adapter</td>
</tr>
<tr>
<td>Hex</td>
<td>Dec</td>
<td>Symbol</td>
<td>Keystrokes</td>
<td>Modes</td>
</tr>
<tr>
<td>EC</td>
<td>236</td>
<td>$\infty$</td>
<td>Alt 236</td>
<td>Note 6</td>
</tr>
<tr>
<td>ED</td>
<td>237</td>
<td>$\phi$</td>
<td>Alt 237</td>
<td>Note 6</td>
</tr>
<tr>
<td>EE</td>
<td>238</td>
<td>$\varepsilon$</td>
<td>Alt 238</td>
<td>Note 6</td>
</tr>
<tr>
<td>EF</td>
<td>239</td>
<td>$\wedge$</td>
<td>Alt 239</td>
<td>Note 6</td>
</tr>
<tr>
<td>F0</td>
<td>240</td>
<td>$\equiv$</td>
<td>Alt 240</td>
<td>Note 6</td>
</tr>
<tr>
<td>F1</td>
<td>241</td>
<td>$\pm$</td>
<td>Alt 241</td>
<td>Note 6</td>
</tr>
<tr>
<td>F2</td>
<td>242</td>
<td>$\geq$</td>
<td>Alt 242</td>
<td>Note 6</td>
</tr>
<tr>
<td>F3</td>
<td>243</td>
<td>$\leq$</td>
<td>Alt 243</td>
<td>Note 6</td>
</tr>
<tr>
<td>F4</td>
<td>244</td>
<td>$\int$</td>
<td>Alt 244</td>
<td>Note 6</td>
</tr>
<tr>
<td>F5</td>
<td>245</td>
<td>$\smallint$</td>
<td>Alt 245</td>
<td>Note 6</td>
</tr>
<tr>
<td>F6</td>
<td>246</td>
<td>$\div$</td>
<td>Alt 246</td>
<td>Note 6</td>
</tr>
<tr>
<td>F7</td>
<td>247</td>
<td>$\approx$</td>
<td>Alt 247</td>
<td>Note 6</td>
</tr>
<tr>
<td>F8</td>
<td>248</td>
<td>$\supset$</td>
<td>Alt 248</td>
<td>Note 6</td>
</tr>
<tr>
<td>F9</td>
<td>249</td>
<td>$\bullet$</td>
<td>Alt 249</td>
<td>Note 6</td>
</tr>
<tr>
<td>FA</td>
<td>250</td>
<td>$\cdot$</td>
<td>Alt 250</td>
<td>Note 6</td>
</tr>
<tr>
<td>FB</td>
<td>251</td>
<td>$\sqrt{}$</td>
<td>Alt 251</td>
<td>Note 6</td>
</tr>
<tr>
<td>FC</td>
<td>252</td>
<td>$\eta$</td>
<td>Alt 252</td>
<td>Note 6</td>
</tr>
<tr>
<td>FD</td>
<td>253</td>
<td>$2$</td>
<td>Alt 253</td>
<td>Note 6</td>
</tr>
<tr>
<td>FE</td>
<td>254</td>
<td>$\mathbb{1}$</td>
<td>Alt 254</td>
<td>Note 6</td>
</tr>
<tr>
<td>FF</td>
<td>255</td>
<td>BLANK</td>
<td>Alt 255</td>
<td>Note 6</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 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>0</th>
<th>16</th>
<th>32</th>
<th>48</th>
<th>64</th>
<th>80</th>
<th>96</th>
<th>112</th>
</tr>
</thead>
<tbody>
<tr>
<td>DECIMAL VALUE</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>BLANK (NULL)</td>
<td>0</td>
<td>@</td>
<td>P</td>
<td>′</td>
<td>p</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BLANK (SPACE)</td>
<td>0</td>
<td>1</td>
<td>A</td>
<td>Q</td>
<td>a</td>
<td>q</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>B</td>
<td>R</td>
<td>b</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>C</td>
<td>S</td>
<td>c</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>$</td>
<td>D</td>
<td>t</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>%</td>
<td>E</td>
<td>u</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>&amp;</td>
<td>F</td>
<td>v</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>′</td>
<td>G</td>
<td>w</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>(</td>
<td>H</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>)</td>
<td>I</td>
<td>y</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>A</td>
<td>*</td>
<td>J</td>
<td>Z</td>
<td>j</td>
<td>z</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>B</td>
<td>+</td>
<td>K</td>
<td>{</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>,</td>
<td>L</td>
<td>\</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>D</td>
<td>=</td>
<td>M</td>
<td>}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>E</td>
<td>&gt;</td>
<td>N</td>
<td>^</td>
<td>n</td>
<td>~</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>F</td>
<td>/</td>
<td>O</td>
<td>_</td>
<td>o</td>
<td>Δ</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

C-12 Of Characters, Keystrokes, and Colors
### Character Set (80-FF) Quick Reference

<table>
<thead>
<tr>
<th>DECIMAL VALUE</th>
<th>HEXA DECIMAL 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>Ė</td>
<td>Ėá</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>ü</td>
<td>æ</td>
<td>í</td>
<td>ë</td>
<td>ë</td>
<td>ë</td>
<td>ë</td>
<td>ë</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>é</td>
<td>Æ</td>
<td>ò</td>
<td>ò</td>
<td>ò</td>
<td>ò</td>
<td>ò</td>
<td>ò</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>á</td>
<td>à</td>
<td>û</td>
<td>û</td>
<td>û</td>
<td>û</td>
<td>û</td>
<td>û</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>ä</td>
<td>ï</td>
<td>ï</td>
<td>ï</td>
<td>ï</td>
<td>ï</td>
<td>ï</td>
<td>ï</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>à</td>
<td>ò</td>
<td>ò</td>
<td>ò</td>
<td>ò</td>
<td>ò</td>
<td>ò</td>
<td>ò</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>â</td>
<td>â</td>
<td>â</td>
<td>â</td>
<td>â</td>
<td>â</td>
<td>â</td>
<td>â</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>ç</td>
<td>ç</td>
<td>ç</td>
<td>ç</td>
<td>ç</td>
<td>ç</td>
<td>ç</td>
<td>ç</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>ê</td>
<td>ë</td>
<td>ë</td>
<td>ë</td>
<td>ë</td>
<td>ë</td>
<td>ë</td>
<td>ë</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>ë</td>
<td>ë</td>
<td>ë</td>
<td>ë</td>
<td>ë</td>
<td>ë</td>
<td>ë</td>
<td>ë</td>
</tr>
<tr>
<td>10</td>
<td>A</td>
<td>e</td>
<td>Ü</td>
<td>Ü</td>
<td>Ü</td>
<td>Ü</td>
<td>Ü</td>
<td>Ü</td>
<td>Ü</td>
</tr>
<tr>
<td>11</td>
<td>B</td>
<td>í</td>
<td>c</td>
<td>½</td>
<td>½</td>
<td>½</td>
<td>½</td>
<td>½</td>
<td>½</td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>'l'</td>
<td>£</td>
<td>¼</td>
<td>¼</td>
<td>¼</td>
<td>¼</td>
<td>¼</td>
<td>¼</td>
</tr>
<tr>
<td>13</td>
<td>D</td>
<td>'i'</td>
<td>¥</td>
<td>'i'</td>
<td>'i'</td>
<td>'i'</td>
<td>'i'</td>
<td>'i'</td>
<td>'i'</td>
</tr>
<tr>
<td>14</td>
<td>E</td>
<td>Ä</td>
<td>P</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>f</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 (16/64K) .................................. D-2
System Board (64/256K) .................................. D-12
Keyboard – Type 1 ........................................ D-22
Keyboard – Type 2 ........................................ D-24
Expansion Board ........................................ D-25
Extender Card ........................................... D-26
Receiver Card ........................................... D-29
Printer .................................................. D-32
Printer Adapter .......................................... D-35
Monochrome Display Adapter ....................... D-36
Color/Graphics Monitor Adapter .................... D-46
Color Display .......................................... D-52
Monochrome Display .................................. D-54
5–1/4 Inch Diskette Drive Adapter ................... D-55
5–1/4 Inch Diskette Drive – Type 1 ................. D-59
5–1/4 Inch Diskette Drive – Type 2 ................. D-62
Fixed Disk Drive Adapter ............................... D-64
Fixed Disk Drive – Type 1 ............................. D-70
Fixed Disk Drive – Type 2 ............................. D-73
32K Memory Expansion Option ....................... D-76
64K Memory Expansion Option ....................... D-79
64/256K Memory Expansion Option .................. D-82
Game Control Adapter .................................. D-86
Prototype Card .......................................... D-87
Asynchronous Communications Adapter .......... D-88
Binary Synchronous Communications Adapter .... D-89
SDLC Communications Adapter ....................... D-91
Logic Diagrams D-3

16/64K System Board (Sheet 2 of 10)
Logic Diagrams
16/64K System Board (Sheet 7 of 10)
D-10 Logic Diagrams
16/64K System Board (Sheet 10 of 10)
64/256K System Board (Sheet 7 of 10)
64/256K System Board (Sheet 8 of 10)
D-20  Logic Diagrams
D-30 Logic Diagrams
D-36 Logic Diagrams

Monochrome Display Adapter (Sheet 1 of 10)
Monochrome Display Adapter (Sheet 4 of 10)
D-40 Logic Diagrams

Monochrome Display Adapter (Sheet 5 of 10)
Monochrome Display Adapter (Sheet 6 of 10)
Monochrome Display Adapter (Sheet 7 of 10)
Monochrome Display Adapter (Sheet 10 of 10)
D-46 Logic Diagrams
D-50 Logic Diagrams

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-52 Logic Diagrams
DANGER
HAZARDOUS VOLTAGES
UP TO 450 VOLTS EXIST
ON THE PRINTED
CIRCUIT BOARDS

NOTES:
1. RESISTOR VALUES ARE IN OHMS K = 1000 OHMS.
2. ALL RESISTORS ARE 1/2 WATT EXCEPT WHERE
   OTHERWISE INDICATED.
3. CAPACITOR VALUES ARE IN μF UNLESS OTHERWISE
   INDICATED P = PF.
4. ALL CAPACITORS ARE 50 VOLTS UNLESS OTHERWISE
   INDICATED.

Color Display (Sheet 1 of 1)
DANGER
HAZARDOUS VOLTAGES
UP TO 450 VOLTS EXIST ON THE PRINTED CIRCUIT BOARDS

Monochrome Display (Sheet 1 of 1)
Logic Diagrams

NOTES:
1. SIGNALS ON DRIVE PINS 10 THRU 16 ARE SWAPPED BY THE DRIVE CABLE BETWEEN DRIVES 1 & 2 (AND 3 & 4) AS FOLLOWS: 10 TO 16
2. ALL DRIVES ARE JUMPERED FOR MULTIPLEX OPERATION. HEAD LOAD WITH DRIVE SELECT AND DRIVE SELECT VIA INPUT PIN 12. TERMINATING R-PACS ARE LEFT IN DRIVES 1 & 3 ONLY
3. A01/FP SHOULD BE ADJACENT TO MODULES MC3407, 7406, 7404, 7414, 16 WWII (SSP-0, MC6404, MC4524, 74LS361 & 74LS390) R-000 CAPS SHOULD BE NEAR ASSOCIATED F-F PINS.
4. ALL SIGNAL LINES HIGHER THAN OR EQUAL TO I-I SHOULD BE KEPT TO THE SHORTEST POSSIBLE LENGTH. THIS IS A PRIMARY DESIGN GOAL
5. MAKE NO CONNECTION TO UNUSED PINS ON THE VCO, CHARGE PUMP & DATA SEPARATOR MODULES
6. ALL VOLTS AND GROUND CONNECTIONS TO THE VCO, CHARGE PUMP AND ASSOCIATED DISCRETE COMPONENTS SHOULD BE SEPARATE FROM OTHER CIRCUITS AND THEN JOINED TO THE OTHER CIRCUITS AT ONE POINT

5-1/4 Inch Diskette Drive Adapter (Sheet 1 of 4)
5-1/4 Inch Diskette Drive Adapter (Sheet 3 of 4)
5-1/4 Inch Diskette Drive Adapter (Sheet 4 of 4)
5-1/4 Inch Diskette Drive Type 1 (Sheet 1 of 3)
NOTES: UNLESS OTHERWISE SPECIFIED
1. ALL RESISTORS ARE IN OHMS. 1/4 W, 5%.
2. ALL CAPS ARE IN uF.
3. ALL DIODES ARE IN 4468.
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, ±5%, 1/4W
2. ±1% RESISTORS ARE 1/8W
3. CAPACITORS ARE IN µF, ±20%, 25V.

Logic Diagrams D-61
Fixed Disk Drive Adapter (Sheet 5 of 6)
Fixed Disk Drive - Type 1 (Sheet 3 of 3)
32K Memory Expansion Option (Sheet 3 of 3)
64K Memory Expansion Option (Sheet 1 of 3)
64K Memory Expansion Option (Sheet 3 of 3)
Game Control Adapter (Sheet 1 of 1)

D-86 Logic Diagrams
Asynchronous Communications Adapter (Sheet 1 of 1)
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:
  20.9 lb (9.5 kg)  Without a diskette drive unit
  25.0 lb (11.4 kg) With one diskette drive unit

Power Cable:
  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:
  1083 BTU/hr

Noise Level:
  56 dB Without printer
  66 dB With printer

Electrical:
  Nominal--120 Vac
  Minimum--104 Vac
  Maximum--127 Vac
  kVA--0.3175 (maximum)

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 Cable:
- 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 Cable:
- 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 (0.914 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--22 AWG

Heat Output:
  341 BTU/hr (maximum)

Electrical:
  Nominal--120 Vac
  Minimum--104 Vac
  Maximum--127 Vac
Front View (Component Side)

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) Center 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:

Data Terminal Equipment

Data Communications Equipment

Communications Line

Adapter

Cable Conforming To RS-232C Standards

EIA/CCITT Line Name

- Protective Ground
- Signal Ground
- Transmitted Data
- Received Data
- Request to Send
- Clear to Send
- Data Set Ready
- Data Terminal Ready
- Connect Data Set to Line
- Received Line Signal Detector
- Speed Select
- Transmit Signal Element Timing
- Receive Signal Element Timing
- Select Standby
- Ring Indicator
- Test

Telephone Co. Lead Number

1 AA/101
7 AB/102
2 BA/103
3 BB/104
4 CA/105
5 CB/106
6 CC/107
20 CD/108.2
20**/108.1
8 CF/109
23 CH/111
15 DB/114
17**/115
11 DD/116
22 DE/125
18**/*

External Modem Cable Connector

13 12 11 10 9 8 7 6 5 4 3 2 1

25 24 23 22 21 20 19 18 17 16 15 14

Modem

Data Terminal Equipment

(Modem) DCE Data Communications Equipment

Pin Number

*Not used when business machine clocking is used.
**Not standardized by EIA (Electronics Industry Association).
***Not standardized by CCITT

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.

### Switched Timing Sequence
- Data Terminal Ready
- Data Set Ready
- Request to Send
- Clear to Send
- Transmitted Data

### Nonswitched Timing Sequence
- Data Terminal Ready
- Data Set Ready
- Request to Send
- Clear to Send
- Transmitted Data

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 1 and 6.

2. Normally the ‘data set ready’ lines 2 and 9 from the modems are active whenever the modems are powered on.

3. Terminal A activates the ‘request to send’ line 3, which causes the modem at terminal A to generate a carrier signal.

4. Modem B detects the carrier, and activates the ‘received line signal detector’ line (sometimes called data carrier detect) 9. Modem B also activates the ‘receiver signal element timing’ line (sometimes called receive clock) 11 to send receive clock signals to the terminal. Some modems activate the clock signals whenever the modem is powered on.

5. After a specified delay, modem A activates the ‘clear to send’ line 4, which indicates to terminal A that the modem is ready to transmit data.

6. Terminal A serializes the data to be transmitted (through the serdes) and transmits the data one bit at a time (synchronized by the transmit clock) onto the ‘transmitted data’ line 6 to the modem.

7. The modem modulates the carrier signal with the data and transmits it to the modem B 5.

8. Modem B demodulates the data from the carrier signal and sends it to terminal B on the ‘received data’ line 6.

9. Terminal B deserializes the data (through the serdes) using the receive clock signals (on the ‘receiver signal element timing’ line) 11 from the modem.

10. After terminal A completes its transmission, it deactivates the ‘request to send’ line 3, which causes the modem to turn off the carrier and deactivate the ‘clear to send’ line 4.

11. Terminal A and modem A now become receivers and wait for a response from terminal B, indicating that all data has reached terminal B. Modem A begins an echo delay (50 to 150 milliseconds) to ensure that all echoes on the line have diminished before it begins receiving. An echo is a reflection of the transmitted signal. If the transmitting modem changed to receive too soon, it could receive a reflection (echo) of the signal it just transmitted.

12. Modem B deactivates the ‘received line signal detector’ line 10 and, if necessary, deactivates the receive clock signals on the ‘receiver signal element timing’ line 11.

13. Terminal B now becomes the transmitter to respond to the request from terminal A. To transmit data, terminal B activates the ‘request to send’ line 13, which causes modem B to transmit a carrier to modem A.

14. 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.

15. After the echo delay at modem A, modem A senses the carrier from modem B (the carrier was activated in step 13 when terminal B activated the ‘request to send’ line) and activates the ‘received line signal detector’ line 7 to terminal A.

16. 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.

2. Since the 'request to send' line and, therefore, the modem carrier, is active continuously 1, the modem immediately modulates the carrier with the address, and thus, the address is transmitted to all modems on the line.

3. All tributary modems, including the modem for station A, demodulate the address and send it to their terminals on the 'received data' line 5.

4. Only station A responds to the address; the other stations ignore the address and continue monitoring their 'received data' line. To respond to the poll, station A activates its 'request to send' line 6, which causes the modem to begin transmitting a carrier signal.

5. The control station's modem receives the carrier and activates the 'received line signal detector' line 3 and the 'receiver signal element timing' line 4 (to send clock signals to the control station). Some modems activate the clock signals as soon as they are powered on.

6. After a short delay to allow the control station modem to receive the carrier, the tributary modem activates the 'clear to send' line 7.

7. When station A detects the active 'clear to send' line, it transmits its response. (For this example, assume that station A has no data to send; therefore, it transmits an EOT 8.)

8. After transmitting the EOT, station A deactivates the 'request to send' line 6. This causes the modem to deactivate the carrier and the 'clear to send' line 7.

9. When the modem at the control station (host) detects the absence of the carrier, it deactivates the 'received line signal detector' line 3.

10. Tributary station A is now in receive mode waiting for the next poll or select transmission from the control station.
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.
The following switch settings are divided between two groups. The first group contains the switch settings for the 16/64K system board. The second group contains the 64/256K system board switch settings.

Determine the system board type and refer to the appropriate group of switch settings for all applications.

Switch Settings (16KB-64KB CPU) ....................... G-3
Switch Settings (64KB-256KB CPU) ....................... G-29
# Switch Settings (16KB-64KB CPU)

<table>
<thead>
<tr>
<th>Setting</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Board Switch Settings</td>
<td>G-5</td>
</tr>
<tr>
<td><strong>System Board Switch Settings</strong></td>
<td>G-5</td>
</tr>
<tr>
<td>5-1/4” Diskette Drives Switch Settings</td>
<td>G-6</td>
</tr>
<tr>
<td>Display Type Switch Settings</td>
<td>G-6</td>
</tr>
<tr>
<td>Math Coprocessor Switch Settings</td>
<td>G-7</td>
</tr>
<tr>
<td><strong>Memory Option Switch Settings</strong></td>
<td>G-8</td>
</tr>
<tr>
<td>16K Total Memory</td>
<td>G-8</td>
</tr>
<tr>
<td>32K Total Memory</td>
<td>G-8</td>
</tr>
<tr>
<td>48K Total Memory</td>
<td>G-8</td>
</tr>
<tr>
<td>64K Total Memory</td>
<td>G-8</td>
</tr>
<tr>
<td>96K Total Memory</td>
<td>G-9</td>
</tr>
<tr>
<td>128K Total Memory</td>
<td>G-10</td>
</tr>
<tr>
<td>160K Total Memory</td>
<td>G-11</td>
</tr>
<tr>
<td>192K Total Memory</td>
<td>G-12</td>
</tr>
<tr>
<td>224K Total Memory</td>
<td>G-13</td>
</tr>
<tr>
<td>256K Total Memory</td>
<td>G-14</td>
</tr>
<tr>
<td>288K Total Memory</td>
<td>G-15</td>
</tr>
<tr>
<td>320K Total Memory</td>
<td>G-16</td>
</tr>
<tr>
<td>352K Total Memory</td>
<td>G-17</td>
</tr>
<tr>
<td>384K Total Memory</td>
<td>G-18</td>
</tr>
<tr>
<td>416K Total Memory</td>
<td>G-19</td>
</tr>
<tr>
<td>448K Total Memory</td>
<td>G-20</td>
</tr>
<tr>
<td>480K Total Memory</td>
<td>G-21</td>
</tr>
<tr>
<td>512K Total Memory</td>
<td>G-22</td>
</tr>
<tr>
<td>544K Total Memory</td>
<td>G-23</td>
</tr>
<tr>
<td>576K Total Memory</td>
<td>G-24</td>
</tr>
<tr>
<td>608K Total Memory</td>
<td>G-25</td>
</tr>
<tr>
<td>640K Total Memory</td>
<td>G-26</td>
</tr>
<tr>
<td><strong>Extender Card Switch Settings</strong></td>
<td>G-27</td>
</tr>
</tbody>
</table>

Switch Settings  G-3
Switch Setting Charts

System Board Switches

WARNING: Before you change any switch settings, make a note of how the switches are presently set.

Switch Block 1

<table>
<thead>
<tr>
<th>Switch</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,7,8</td>
<td>Number of 5-1/4 inch diskette drives installed</td>
</tr>
<tr>
<td>2</td>
<td>Math Coprocessor</td>
</tr>
<tr>
<td>3,4</td>
<td>System board memory switches</td>
</tr>
<tr>
<td>5,6</td>
<td>Type(s) of display(s) connected</td>
</tr>
</tbody>
</table>

Switch Block 2

<table>
<thead>
<tr>
<th>Switch</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,2,3,4,5</td>
<td>Amount of memory options installed</td>
</tr>
<tr>
<td>6,7,8</td>
<td>Always in the Off position</td>
</tr>
</tbody>
</table>
Number of 5-1/4 Inch Diskette Drives Installed

<table>
<thead>
<tr>
<th>Number of Drives</th>
<th>Switch Block 1</th>
<th>Switch Block 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 – Drives</td>
<td>![Diagram]</td>
<td>![Diagram]</td>
</tr>
<tr>
<td>1 – Drive</td>
<td>![Diagram]</td>
<td>![Diagram]</td>
</tr>
<tr>
<td>2 – Drives</td>
<td>![Diagram]</td>
<td>![Diagram]</td>
</tr>
</tbody>
</table>

Type(s) of display(s) connected

WARNING: If an IBM Monochrome Display is connected to your system. Switch Block 1, switches 5 and 6, must always be Off. Damage to your display can result with any other switch settings.

Switch Block 1 Switch Block 2

IBM Monochrome Display (or IBM Monochrome Display plus another display)

Switch Block 1 Switch Block 2

Color Display (Do not use if an IBM Monochrome Display is connected)

Switch Block 1 Switch Block 2

40x25 Mode

80x25 Mode

Note: The 40x25 mode means there will be 40 characters across the screen and 25 lines down the screen. The 80x25 mode means there will be 80 characters across the screen and 25 lines down the screen. The 80x25 mode, when used with home televisions and various displays, can cause loss of character quality.

G-6 Switch Settings
Math Coprocessor

Switch Block 1  Switch Block 2

With Math Coprocessor

Without Math Coprocessor
Memory Switch Settings
(16KB-64KB CPU) System Board

16K Total Memory

System Board Switches | Switch Block 1 | Switch Block 2
--- | --- | ---

32K Total Memory

System Board Switches | Switch Block 1 | Switch Block 2
--- | --- | ---

48K Total Memory

System Board Switches | Switch Block 1 | Switch Block 2
--- | --- | ---

64K Total Memory

System Board Switches | Switch Block 1 | Switch Block 2
--- | --- | ---
<table>
<thead>
<tr>
<th>System Board Switches</th>
<th>Switch Block 1</th>
<th>Switch Block 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>![Switch Block 1 Image]</td>
<td>![Switch Block 2 Image]</td>
</tr>
</tbody>
</table>

### Memory Configuration

- **Total Memory**: 96K
- **System Board Memory**: 32K + (64K on System Board)

### Switch Settings

<table>
<thead>
<tr>
<th>Option</th>
<th>Switch Block 1</th>
<th>Switch Block 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - 32K option</td>
<td>![Switch Block 1 Image]</td>
<td>![Switch Block 2 Image]</td>
</tr>
</tbody>
</table>

### Options

- **64/256K Option**
- **64K Option**
- **32K Option**
<table>
<thead>
<tr>
<th>System Board Switches</th>
<th>Switch Block 1</th>
<th>Switch Block 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - 64/256K option with 64K installed</td>
<td><img src="image" alt="64/256K Option Card Switches" /></td>
<td><img src="image" alt="32K Option Card Switches" /></td>
</tr>
<tr>
<td>1 - 64K option</td>
<td><img src="image" alt="64K Option Card Switches" /></td>
<td></td>
</tr>
<tr>
<td>2 - 32K options</td>
<td></td>
<td><img src="image" alt="32K Option Card Switches" /></td>
</tr>
</tbody>
</table>
### System Board Switches

<table>
<thead>
<tr>
<th>Switch Block 1</th>
<th>Switch Block 2</th>
</tr>
</thead>
</table>

**64/256K Option**

<table>
<thead>
<tr>
<th>1 - 64/256K option with 64K installed</th>
<th>1 - 32K option</th>
</tr>
</thead>
</table>

**64K Option**

<table>
<thead>
<tr>
<th>1 - 64K option</th>
<th>1 - 32K option</th>
</tr>
</thead>
</table>

**32K Option**

<table>
<thead>
<tr>
<th>3 - 32K options</th>
</tr>
</thead>
</table>

---

**160K Total Memory**

96K + (64K on System Board)
### System Board Switches

<table>
<thead>
<tr>
<th>Switch Block 1</th>
<th>Switch Block 2</th>
</tr>
</thead>
</table>

#### 64/256K Option Card Switches

<table>
<thead>
<tr>
<th>Setting</th>
<th>Switch Block 1</th>
<th>Switch Block 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - 64/256K option with 64K option installed</td>
<td><img src="image1" alt="Switch Block 1" /></td>
<td><img src="image2" alt="Switch Block 2" /></td>
</tr>
<tr>
<td>1 - 64K option</td>
<td><img src="image3" alt="Switch Block 1" /></td>
<td><img src="image4" alt="Switch Block 2" /></td>
</tr>
<tr>
<td>2 - 64K options</td>
<td><img src="image5" alt="Switch Block 1" /></td>
<td><img src="image6" alt="Switch Block 2" /></td>
</tr>
</tbody>
</table>

#### 64K Option Card Switches

<table>
<thead>
<tr>
<th>Setting</th>
<th>Switch Block 1</th>
<th>Switch Block 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - 64/256K option with 64K installed</td>
<td><img src="image7" alt="Switch Block 1" /></td>
<td><img src="image8" alt="Switch Block 2" /></td>
</tr>
<tr>
<td>2 - 32K options</td>
<td><img src="image9" alt="Switch Block 1" /></td>
<td><img src="image10" alt="Switch Block 2" /></td>
</tr>
</tbody>
</table>

#### 32K Option Card Switches

<table>
<thead>
<tr>
<th>Setting</th>
<th>Switch Block 1</th>
<th>Switch Block 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - 64/256K option with 128K installed</td>
<td><img src="image11" alt="Switch Block 1" /></td>
<td><img src="image12" alt="Switch Block 2" /></td>
</tr>
<tr>
<td>1 - 64K option</td>
<td><img src="image13" alt="Switch Block 1" /></td>
<td><img src="image14" alt="Switch Block 2" /></td>
</tr>
<tr>
<td>2 - 32K options</td>
<td><img src="image15" alt="Switch Block 1" /></td>
<td><img src="image16" alt="Switch Block 2" /></td>
</tr>
</tbody>
</table>

---

**192K Total Memory**

**128K + (64K on System Board)**
## System Board Switches

<table>
<thead>
<tr>
<th>Switch Block 1</th>
<th>Switch Block 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>64/256K Option Card Switches</strong></td>
<td><strong>64K Option Card Switches</strong></td>
</tr>
<tr>
<td>1 - 64/256K option with 64K installed</td>
<td>1 - 64K option</td>
</tr>
<tr>
<td>1 - 64K option</td>
<td></td>
</tr>
<tr>
<td>1 - 32K option</td>
<td></td>
</tr>
<tr>
<td>2 - 64K options</td>
<td>2 - 64K options</td>
</tr>
<tr>
<td>1 - 32K option</td>
<td>1 - 32K option</td>
</tr>
<tr>
<td>1 - 64/256K option with 128K installed</td>
<td>1 - 64/256K option with 128K installed</td>
</tr>
<tr>
<td>1 - 32K option</td>
<td>1 - 32K option</td>
</tr>
</tbody>
</table>

**224K Total Memory**

**160K + (64K on System Board)**
## 256K Total Memory
### 192K + (64K on System Board)

<table>
<thead>
<tr>
<th>System Board Switches</th>
<th>Switch Block 1</th>
<th>Switch Block 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>64/256K Option Card Switches</strong></td>
<td><strong>64K Option Card Switches</strong></td>
<td><strong>32K Option Card Switches</strong></td>
</tr>
<tr>
<td>1 - 64/256K option with 192K installed</td>
<td><img src="image1.png" alt="Switch Settings" /></td>
<td><img src="image2.png" alt="Switch Settings" /></td>
</tr>
<tr>
<td>1 - 64/256K option with 128K installed</td>
<td><img src="image3.png" alt="Switch Settings" /></td>
<td><img src="image4.png" alt="Switch Settings" /></td>
</tr>
<tr>
<td>1 - 64K option</td>
<td><img src="image5.png" alt="Switch Settings" /></td>
<td><img src="image6.png" alt="Switch Settings" /></td>
</tr>
<tr>
<td>1 - 64/256K option with 64K installed</td>
<td><img src="image7.png" alt="Switch Settings" /></td>
<td><img src="image8.png" alt="Switch Settings" /></td>
</tr>
<tr>
<td>2 - 64K options</td>
<td><img src="image9.png" alt="Switch Settings" /></td>
<td><img src="image10.png" alt="Switch Settings" /></td>
</tr>
<tr>
<td>3 - 64K options</td>
<td><img src="image11.png" alt="Switch Settings" /></td>
<td><img src="image12.png" alt="Switch Settings" /></td>
</tr>
<tr>
<td>1 - 64/256K option with 128K installed</td>
<td><img src="image13.png" alt="Switch Settings" /></td>
<td><img src="image14.png" alt="Switch Settings" /></td>
</tr>
<tr>
<td>2 - 32K options</td>
<td><img src="image15.png" alt="Switch Settings" /></td>
<td><img src="image16.png" alt="Switch Settings" /></td>
</tr>
</tbody>
</table>
### Switch Settings

#### System Board Switches

<table>
<thead>
<tr>
<th>Switch Block 1</th>
<th>Switch Block 2</th>
</tr>
</thead>
</table>

#### Switch Block 1

- 64/256K Option
- 64K Option
- 32K Option

#### Switch Block 2

- 64/256K Option
- 64K Option
- 32K Option

#### 288K Total Memory

- 224K + (64K on System Board)

#### Card Switches

- 1 - 64/256K option with 192K installed
- 1 - 32K option

- 1 - 64/256K option with 128K installed
- 1 - 64K option
- 1 - 32K option
### 320K Total Memory

**256K + (64K on System Board)**

<table>
<thead>
<tr>
<th>System Board Switches</th>
<th>Switch Block 1</th>
<th>Switch Block 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>64/256K Option</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Card Switches</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 - 64/256K option with 128K installed</td>
<td><img src="image" alt="Switch Block 1" /></td>
<td><img src="image" alt="Switch Block 2" /></td>
</tr>
<tr>
<td>2 - 64K options</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>64K Option</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Card Switches</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 - 64/256K option with 192K installed</td>
<td><img src="image" alt="Switch Block 1" /></td>
<td><img src="image" alt="Switch Block 2" /></td>
</tr>
<tr>
<td>1 - 64K option</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>32K Option</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Card Switches</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 - 64/256K option with 192K installed</td>
<td><img src="image" alt="Switch Block 1" /></td>
<td><img src="image" alt="Switch Block 2" /></td>
</tr>
<tr>
<td>2 - 32K options</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 - 64/256K option with 256K installed</td>
<td><img src="image" alt="Switch Block 1" /></td>
<td><img src="image" alt="Switch Block 2" /></td>
</tr>
</tbody>
</table>
### 352K Total Memory
288K + (64K on System Board)

<table>
<thead>
<tr>
<th>System Board Switches</th>
<th>Switch Block 1</th>
<th>Switch Block 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - 64/256K option with 192K installed</td>
<td><img src="image1.png" alt="Switch Block 1" /></td>
<td><img src="image2.png" alt="Switch Block 2" /></td>
</tr>
<tr>
<td>1 - 64K option</td>
<td><img src="image3.png" alt="Switch Block 1" /></td>
<td><img src="image4.png" alt="Switch Block 2" /></td>
</tr>
<tr>
<td>1 - 32K option</td>
<td><img src="image5.png" alt="Switch Block 1" /></td>
<td><img src="image6.png" alt="Switch Block 2" /></td>
</tr>
<tr>
<td>1 - 64/256K option with 256K installed</td>
<td><img src="image7.png" alt="Switch Block 1" /></td>
<td><img src="image8.png" alt="Switch Block 2" /></td>
</tr>
<tr>
<td>1 - 32K option</td>
<td><img src="image9.png" alt="Switch Block 1" /></td>
<td><img src="image10.png" alt="Switch Block 2" /></td>
</tr>
</tbody>
</table>

**Switch Settings**

G-17
### System Board Switches

<table>
<thead>
<tr>
<th>Switch Block 1</th>
<th>Switch Block 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Switch Block 1" /></td>
<td><img src="image2" alt="Switch Block 2" /></td>
</tr>
</tbody>
</table>

### 64/256K Option Card Switches

<table>
<thead>
<tr>
<th>Card Switches</th>
<th>64K Option Card Switches</th>
<th>32K Option Card Switches</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image3" alt="Switch Block" /></td>
<td><img src="image4" alt="Switch Block" /></td>
<td><img src="image5" alt="Switch Block" /></td>
</tr>
</tbody>
</table>

### Memory Options

1. **64/256K option with 192K installed**
   - 1 - 64/256K option with 192K installed
   - 2 - 64K options

2. **64/256K option with 256K installed**
   - 1 - 64/256K option with 256K installed
   - 1 - 64/256K option with 64K installed

3. **64/256K option with 256K installed**
   - 1 - 64/256K option with 256K installed
   - 1 - 64K option

4. **64/256K option with 256K installed**
   - 1 - 64/256K option with 256K installed
   - 2 - 32K options
416K Total Memory
352K + (64K on System Board)

System Board Switches

Switch Block 1

Switch Block 2

64/256K Option Card Switches

64K Option Card Switches

32K Option Card Switches

1 - 64/256K option with 256K installed
1 - 64/256K option with 64K installed
1 - 32K option

1 - 64/256K option with 256K installed
1 - 64K option
1 - 32K option

Appendix G
### System Board Switches

<table>
<thead>
<tr>
<th>Switch Block 1</th>
<th>Switch Block 2</th>
<th>64/256K Option Card Switches</th>
<th>64K Option Card Switches</th>
<th>32K Option Card Switches</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - 64/256K option with 256K installed</td>
<td></td>
<td><img src="image1" alt="Switch Setting" /></td>
<td><img src="image2" alt="Switch Setting" /></td>
<td><img src="image3" alt="Switch Setting" /></td>
</tr>
<tr>
<td>1 - 64/256K option with 64K installed</td>
<td></td>
<td><img src="image1" alt="Switch Setting" /></td>
<td><img src="image2" alt="Switch Setting" /></td>
<td><img src="image3" alt="Switch Setting" /></td>
</tr>
<tr>
<td>1 - 64K option</td>
<td></td>
<td><img src="image1" alt="Switch Setting" /></td>
<td><img src="image2" alt="Switch Setting" /></td>
<td><img src="image3" alt="Switch Setting" /></td>
</tr>
<tr>
<td>2 - 64K options</td>
<td></td>
<td><img src="image1" alt="Switch Setting" /></td>
<td><img src="image2" alt="Switch Setting" /></td>
<td><img src="image3" alt="Switch Setting" /></td>
</tr>
<tr>
<td>1 - 64/256K option with 256K installed</td>
<td></td>
<td><img src="image1" alt="Switch Setting" /></td>
<td><img src="image2" alt="Switch Setting" /></td>
<td><img src="image3" alt="Switch Setting" /></td>
</tr>
<tr>
<td>1 - 64/256K option with 128K installed</td>
<td></td>
<td><img src="image1" alt="Switch Setting" /></td>
<td><img src="image2" alt="Switch Setting" /></td>
<td><img src="image3" alt="Switch Setting" /></td>
</tr>
</tbody>
</table>
### 480K Total Memory

416K + (64K on System Board)

<table>
<thead>
<tr>
<th>System Board Switches</th>
<th>Switch Block 1</th>
<th>Switch Block 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### 64/256K Option Card Switches

1. 64/256K option with 256K installed
2. 64/256K option with 128K installed
3. 32K option

#### 64K Option Card Switches

#### 32K Option Card Switches
512K Total Memory
448K + (64K on System Board)

<table>
<thead>
<tr>
<th>System Board Switches</th>
<th>Switch Block 1</th>
<th>Switch Block 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>64/256K Option Card Switches</td>
<td>64K Option Card Switches</td>
<td>64K Option Card Switches</td>
</tr>
</tbody>
</table>

1 - 64/256K option with 256K installed
1 - 64/256K option with 128K installed
1 - 64K option

1 - 64/256K option with 256K installed
1 - 64/256K option with 192K installed
### System Board Switches

<table>
<thead>
<tr>
<th>Switch Block 1</th>
<th>Switch Block 2</th>
</tr>
</thead>
</table>

#### 64/256K Option Card Switches

- 1 - 64/256K option with 256K installed
- 1 - 64/256K option with 192K installed
- 1 - 32K option

#### 64K Option Card Switches

#### 32K Option Card Switches

---

**544K Total Memory**

480K + (64K on System Board)
576K Total Memory
512K + (64K on System Board)

System Board Switches

<table>
<thead>
<tr>
<th>Switch Block 1</th>
<th>Switch Block 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - 64/256K option with 256K installed</td>
<td></td>
</tr>
<tr>
<td>1 - 64/256K option with 192K installed</td>
<td></td>
</tr>
<tr>
<td>1 - 64K option</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>2 - 64/256K option with 256K installed</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
608K Total Memory
544K + (64K on System Board)

<table>
<thead>
<tr>
<th>System Board Switches</th>
<th>Switch Block 1</th>
<th>Switch Block 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><img src="image" alt="Switch Block 1" /></td>
<td><img src="image" alt="Switch Block 2" /></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>2 - 64/256K option with 256K installed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 - 32K option</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Switch Settings
G-25

Appendix G
### System Board Switches

<table>
<thead>
<tr>
<th>Switch Block 1</th>
<th>Switch Block 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Switch Block 1 Diagram" /></td>
<td><img src="image2.png" alt="Switch Block 2 Diagram" /></td>
</tr>
</tbody>
</table>

### 640K Total Memory

576K + (64K on System Board)

<table>
<thead>
<tr>
<th>System Board Switches</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>2 - 64/256K option with 256K installed</td>
<td><img src="image3.png" alt="64/256K Option Card Switches Diagram" /></td>
<td><img src="image4.png" alt="64K Option Card Switches Diagram" /></td>
<td><img src="image5.png" alt="32K Option Card Switches Diagram" /></td>
</tr>
<tr>
<td>1 - 64K option</td>
<td><img src="image3.png" alt="64/256K Option Card Switches Diagram" /></td>
<td><img src="image4.png" alt="64K Option Card Switches Diagram" /></td>
<td><img src="image5.png" alt="32K Option Card Switches Diagram" /></td>
</tr>
</tbody>
</table>

### 640K Total Memory

576K + (64K on System Board)

<table>
<thead>
<tr>
<th>System Board Switches</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>2 - 64/256K option with 256K installed</td>
<td><img src="image3.png" alt="64/256K Option Card Switches Diagram" /></td>
<td><img src="image4.png" alt="64K Option Card Switches Diagram" /></td>
<td><img src="image5.png" alt="32K Option Card Switches Diagram" /></td>
</tr>
<tr>
<td>1 - 64/256K option with 64K installed</td>
<td><img src="image3.png" alt="64/256K Option Card Switches Diagram" /></td>
<td><img src="image4.png" alt="64K Option Card Switches Diagram" /></td>
<td><img src="image5.png" alt="32K Option Card Switches Diagram" /></td>
</tr>
</tbody>
</table>
### Extender Card Switch Settings

<table>
<thead>
<tr>
<th>System Memory</th>
<th>Extender Card Switch Block</th>
<th>Memory Segment</th>
</tr>
</thead>
<tbody>
<tr>
<td>16K to 64K</td>
<td><img src="image" alt="16K to 64K Switch Block" /></td>
<td>1</td>
</tr>
<tr>
<td>96K to 128K</td>
<td><img src="image" alt="96K to 128K Switch Block" /></td>
<td>2</td>
</tr>
<tr>
<td>160K to 192K</td>
<td><img src="image" alt="160K to 192K Switch Block" /></td>
<td>3</td>
</tr>
<tr>
<td>224K to 256K</td>
<td><img src="image" alt="224K to 256K Switch Block" /></td>
<td>4</td>
</tr>
<tr>
<td>288K to 320K</td>
<td><img src="image" alt="288K to 320K Switch Block" /></td>
<td>5</td>
</tr>
<tr>
<td>352K to 384K</td>
<td><img src="image" alt="352K to 384K Switch Block" /></td>
<td>6</td>
</tr>
<tr>
<td>416K to 448K</td>
<td><img src="image" alt="416K to 448K Switch Block" /></td>
<td>7</td>
</tr>
<tr>
<td>480K to 512K</td>
<td><img src="image" alt="480K to 512K Switch Block" /></td>
<td>8</td>
</tr>
<tr>
<td>544K to 576K</td>
<td><img src="image" alt="544K to 576K Switch Block" /></td>
<td>9</td>
</tr>
<tr>
<td>608K to 640K</td>
<td><img src="image" alt="608K to 640K Switch Block" /></td>
<td>A</td>
</tr>
</tbody>
</table>
Switch Settings (64KB-256KB CPU)

System Board Switch Settings .................................. G-31
  System Board Switch Settings .................................. G-31
  5-1/4" Diskette Drives Switch Settings ..................... G-32
  Display Type Switch Settings ................................. G-32
  Math Coprocessor Switch Settings ............................ G-32

Memory Option Switch Settings .................................. G-34
  64K Total Memory ............................................... G-34
  128K Total Memory .............................................. G-34
  192K Total Memory .............................................. G-34
  256K Total Memory .............................................. G-34
  288K Total Memory .............................................. G-34
  320K Total Memory .............................................. G-36
  352K Total Memory .............................................. G-37
  384K Total Memory .............................................. G-38
  416K Total Memory .............................................. G-39
  448K Total Memory .............................................. G-40
  480K Total Memory .............................................. G-41
  512K Total Memory .............................................. G-42
  544K Total Memory .............................................. G-43
  576K Total Memory .............................................. G-44
  608K Total Memory .............................................. G-45
  640K Total Memory .............................................. G-46

Extender Card Switch Settings .................................. G-47
Switch Setting Charts

System Board Switches

WARNING: Before you change any switch settings, make a note of how the switches are presently set.

Switch Block 1

<table>
<thead>
<tr>
<th>Switch</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,7,8</td>
<td>Number of 5-1/4 inch diskette drives installed</td>
</tr>
<tr>
<td>2</td>
<td>Math Coprocessor</td>
</tr>
<tr>
<td>3,4</td>
<td>System board memory switches</td>
</tr>
<tr>
<td>5,6</td>
<td>Type(s) of display(s) connected</td>
</tr>
</tbody>
</table>

Switch Block 2

<table>
<thead>
<tr>
<th>Switch</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,2,3,4,5</td>
<td>Amount of memory options installed</td>
</tr>
<tr>
<td>6,7,8</td>
<td>Always in the Off position</td>
</tr>
</tbody>
</table>
Number of 5-1/4 Inch Diskette Drives Installed

<table>
<thead>
<tr>
<th>Number of Drives</th>
<th>Switch Block 1</th>
<th>Switch Block 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td><img src="image1" alt="Switch Block 1" /></td>
<td><img src="image2" alt="Switch Block 2" /></td>
</tr>
<tr>
<td>1</td>
<td><img src="image3" alt="Switch Block 1" /></td>
<td><img src="image4" alt="Switch Block 2" /></td>
</tr>
<tr>
<td>2</td>
<td><img src="image5" alt="Switch Block 1" /></td>
<td><img src="image6" alt="Switch Block 2" /></td>
</tr>
</tbody>
</table>

Type(s) of display(s) connected

**WARNING:** If an IBM Monochrome Display is connected to your system. Switch Block 1, switches 5 and 6, must always be Off. Damage to your display can result with any other switch settings.

<table>
<thead>
<tr>
<th>Display Type</th>
<th>Switch Block 1</th>
<th>Switch Block 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM Monochrome Display (or IBM Monochrome Display plus another display)</td>
<td><img src="image7" alt="Switch Block 1" /></td>
<td><img src="image8" alt="Switch Block 2" /></td>
</tr>
<tr>
<td>Color Display (Do not use if an IBM Monochrome Display is connected)</td>
<td><img src="image9" alt="Switch Block 1" /></td>
<td><img src="image10" alt="Switch Block 2" /></td>
</tr>
</tbody>
</table>

**Note:** The 40x25 mode means there will be 40 characters across the screen and 25 lines down the screen. The 80x25 mode means there will be 80 characters across the screen and 25 lines down the screen. The 80x25 mode, when used with home televisions and various displays, can cause loss of character quality.

G-32 Switch Settings
Math Coprocessor

Switch Block 1

With Math Coprocessor

Without Math Coprocessor

Switch Block 2
Memory Switch Settings
(64KB-256KB CPU) System Board

64K Total Memory

System Board Switches | Switch Block 1 | Switch Block 2

128K Total Memory

System Board Switches | Switch Block 1 | Switch Block 2

192K Total Memory

System Board Switches | Switch Block 1 | Switch Block 2

256K Total Memory

System Board Switches | Switch Block 1 | Switch Block 2
### 288K Total Memory
32K + (256K on System Board)

<table>
<thead>
<tr>
<th>System Board Switches</th>
<th>Switch Block 1</th>
<th>Switch Block 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><img src="image1" alt="Switch Block 1" /></td>
<td><img src="image2" alt="Switch Block 2" /></td>
</tr>
</tbody>
</table>

<p>| 64/256K Option       | 64K Option       | 32K Option       |</p>
<table>
<thead>
<tr>
<th>Card Switches</th>
<th>Card Switches</th>
<th>Card Switches</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - 32K option</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Appendix G*

**Switch Settings**
G-35
### 320K Total Memory

64K + (256K on System Board)

<table>
<thead>
<tr>
<th>System Board Switches</th>
<th>Switch Block 1</th>
<th>Switch Block 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1 - 64/256K option with 64K installed</strong></td>
<td><img src="#" alt="Switch Block 1 Diagram" /></td>
<td><img src="#" alt="Switch Block 2 Diagram" /></td>
</tr>
<tr>
<td><strong>1 - 64K option</strong></td>
<td><img src="#" alt="Switch Block 1 Diagram" /></td>
<td><img src="#" alt="Switch Block 2 Diagram" /></td>
</tr>
<tr>
<td><strong>2 - 32K options</strong></td>
<td><img src="#" alt="Switch Block 1 Diagram" /></td>
<td><img src="#" alt="Switch Block 2 Diagram" /></td>
</tr>
</tbody>
</table>
### 352K Total Memory
96K + (256K on System Board)

<table>
<thead>
<tr>
<th>System Board Switches</th>
<th>1 - 64/256K option with 64K installed</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 - 32K option</td>
<td><img src="image1.png" alt="Switch Block 1" /></td>
<td><img src="image2.png" alt="Switch Block 1" /></td>
<td><img src="image3.png" alt="Switch Block 1" /></td>
<td><img src="image4.png" alt="Switch Block 1" /></td>
</tr>
<tr>
<td>1 - 64K option</td>
<td><img src="image5.png" alt="Switch Block 1" /></td>
<td><img src="image6.png" alt="Switch Block 1" /></td>
<td><img src="image7.png" alt="Switch Block 1" /></td>
<td><img src="image8.png" alt="Switch Block 1" /></td>
</tr>
<tr>
<td>1 - 32K option</td>
<td><img src="image9.png" alt="Switch Block 1" /></td>
<td><img src="image10.png" alt="Switch Block 1" /></td>
<td><img src="image11.png" alt="Switch Block 1" /></td>
<td><img src="image12.png" alt="Switch Block 1" /></td>
</tr>
<tr>
<td>3 - 32K options</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 384K Total Memory
128K + (256K on System Board)

<table>
<thead>
<tr>
<th>System Board Switches</th>
<th>Switch Block 1</th>
<th>Switch Block 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>64/256K Option Card Switches</strong></td>
<td><strong>64K Option Card Switches</strong></td>
<td><strong>32K Option Card Switches</strong></td>
</tr>
<tr>
<td>1 - 64/256K option with 64K option installed</td>
<td><img src="image1" alt="Diagram" /></td>
<td><img src="image2" alt="Diagram" /></td>
</tr>
<tr>
<td>1 - 64K option</td>
<td><img src="image3" alt="Diagram" /></td>
<td><img src="image4" alt="Diagram" /></td>
</tr>
<tr>
<td>2 - 64K options</td>
<td><img src="image5" alt="Diagram" /></td>
<td><img src="image6" alt="Diagram" /></td>
</tr>
<tr>
<td>1 - 64/256K option with 64K installed</td>
<td><img src="image7" alt="Diagram" /></td>
<td><img src="image8" alt="Diagram" /></td>
</tr>
<tr>
<td>2 - 32K options</td>
<td><img src="image9" alt="Diagram" /></td>
<td><img src="image10" alt="Diagram" /></td>
</tr>
<tr>
<td>1 - 64K option</td>
<td><img src="image11" alt="Diagram" /></td>
<td><img src="image12" alt="Diagram" /></td>
</tr>
<tr>
<td>2 - 32K options</td>
<td><img src="image13" alt="Diagram" /></td>
<td><img src="image14" alt="Diagram" /></td>
</tr>
<tr>
<td>1 - 64/256K option with 128K installed</td>
<td><img src="image15" alt="Diagram" /></td>
<td><img src="image16" alt="Diagram" /></td>
</tr>
</tbody>
</table>
416K Total Memory
160K + (256K on System Board)

<table>
<thead>
<tr>
<th>System Board Switches</th>
<th>Switch Block 1</th>
<th>Switch Block 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>64/256K Option</strong> Card Switches</td>
<td><strong>64K Option</strong> Card Switches</td>
<td><strong>32K Option</strong> Card Switches</td>
</tr>
<tr>
<td>1 - 64/256K option with 64K installed</td>
<td><img src="#" alt="Switch Setting 1" /></td>
<td><img src="#" alt="Switch Setting 2" /></td>
</tr>
<tr>
<td>1 - 64K option</td>
<td><img src="#" alt="Switch Setting 3" /></td>
<td><img src="#" alt="Switch Setting 4" /></td>
</tr>
<tr>
<td>1 - 32K option</td>
<td><img src="#" alt="Switch Setting 5" /></td>
<td><img src="#" alt="Switch Setting 6" /></td>
</tr>
<tr>
<td>2 - 64K options</td>
<td><img src="#" alt="Switch Setting 7" /></td>
<td><img src="#" alt="Switch Setting 8" /></td>
</tr>
<tr>
<td>1 - 32K option</td>
<td><img src="#" alt="Switch Setting 9" /></td>
<td><img src="#" alt="Switch Setting 10" /></td>
</tr>
<tr>
<td>1 - 64/256K option with 128K installed</td>
<td><img src="#" alt="Switch Setting 11" /></td>
<td><img src="#" alt="Switch Setting 12" /></td>
</tr>
<tr>
<td>1 - 32K option</td>
<td><img src="#" alt="Switch Setting 13" /></td>
<td><img src="#" alt="Switch Setting 14" /></td>
</tr>
</tbody>
</table>
### 448K Total Memory

192K + (256K on System Board)

<table>
<thead>
<tr>
<th>System Board Switches</th>
<th>Switch Block 1</th>
<th>Switch Block 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>64/256K Option</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Card Switches</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 - 64/256K option with 192K installed</td>
<td><img src="image1" alt="Switch Setup" /></td>
<td><img src="image2" alt="Switch Setup" /></td>
</tr>
<tr>
<td>1 - 64/256K option with 128K installed</td>
<td><img src="image3" alt="Switch Setup" /></td>
<td><img src="image4" alt="Switch Setup" /></td>
</tr>
<tr>
<td>1 - 64K option</td>
<td><img src="image5" alt="Switch Setup" /></td>
<td><img src="image6" alt="Switch Setup" /></td>
</tr>
<tr>
<td>1 - 64K option</td>
<td><img src="image7" alt="Switch Setup" /></td>
<td><img src="image8" alt="Switch Setup" /></td>
</tr>
<tr>
<td>2 - 64K options</td>
<td><img src="image9" alt="Switch Setup" /></td>
<td><img src="image10" alt="Switch Setup" /></td>
</tr>
<tr>
<td>3 - 64K options</td>
<td><img src="image11" alt="Switch Setup" /></td>
<td><img src="image12" alt="Switch Setup" /></td>
</tr>
<tr>
<td>1 - 64/256K option with 128 installed</td>
<td><img src="image13" alt="Switch Setup" /></td>
<td><img src="image14" alt="Switch Setup" /></td>
</tr>
<tr>
<td>2 - 32K options</td>
<td><img src="image15" alt="Switch Setup" /></td>
<td><img src="image16" alt="Switch Setup" /></td>
</tr>
</tbody>
</table>
### System Board Switches

- **Switch Block 1**
  - 64/256K Option Card Switches
  - 64K Option Card Switches
  - 32K Option Card Switches
- **Switch Block 2**

<table>
<thead>
<tr>
<th>Switch Setting</th>
<th>System Board Switches</th>
<th>Switch Block 1</th>
<th>Switch Block 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - 64/256K option with 192K installed</td>
<td>1 - 64/256K option with 192K installed</td>
<td>1 - 64/256K option with 192K installed</td>
<td>1 - 64/256K option with 192K installed</td>
</tr>
<tr>
<td>1 - 32K option</td>
<td>1 - 32K option</td>
<td>1 - 32K option</td>
<td>1 - 32K option</td>
</tr>
<tr>
<td>1 - 64/256K option with 128K installed</td>
<td>1 - 64/256K option with 128K installed</td>
<td>1 - 64/256K option with 128K installed</td>
<td>1 - 64/256K option with 128K installed</td>
</tr>
<tr>
<td>1 - 64K option</td>
<td>1 - 64K option</td>
<td>1 - 64K option</td>
<td>1 - 64K option</td>
</tr>
<tr>
<td>1 - 32K option</td>
<td>1 - 32K option</td>
<td>1 - 32K option</td>
<td>1 - 32K option</td>
</tr>
</tbody>
</table>
### 512K Total Memory
**256K + (256K on System Board)**

<table>
<thead>
<tr>
<th>System Board Switches</th>
<th>Switch Block 1</th>
<th>Switch Block 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>64/256K Option</strong></td>
<td>1 - 64/256K option with 128K installed</td>
<td>1 - 64/256K option with 256K installed</td>
</tr>
<tr>
<td><strong>Card Switches</strong></td>
<td>2 - 64K options</td>
<td></td>
</tr>
<tr>
<td><strong>32K Option</strong></td>
<td>1 - 64/256K option with 192K installed</td>
<td>1 - 64/256K option with 192K installed</td>
</tr>
<tr>
<td><strong>Card Switches</strong></td>
<td>1 - 64K option</td>
<td>2 - 32K options</td>
</tr>
<tr>
<td><strong>Card Switches</strong></td>
<td>1 - 64/256K option with 192K installed</td>
<td></td>
</tr>
<tr>
<td><strong>Card Switches</strong></td>
<td>2 - 32K options</td>
<td></td>
</tr>
</tbody>
</table>

Note: Switch settings are indicated by the positions of the switches.
### System Board Switches

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Switch Block 1</th>
<th>Switch Block 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>64/256K option with 192K installed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 - 64K option</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 - 32K option</td>
<td></td>
<td></td>
</tr>
<tr>
<td>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>

### Switch Settings

- **64/256K Option**
- **64K Option**
- **32K Option**

- **Card Switches**
- **Card Switches**
- **Card Switches**

---

**Appendix G**

**Switch Settings**

G-43
### 576K Total Memory
320K + (256K on System Board)

<table>
<thead>
<tr>
<th>System Board Switches</th>
<th>Switch Block 1</th>
<th>Switch Block 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - 64/256K option with 192K installed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 - 64K options</td>
<td></td>
<td></td>
</tr>
<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 - 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 - 64/256K option with 256K installed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 - 32K options</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><img src="image1" alt="Switch Block 1" /></td>
<td><img src="image2" alt="Switch Block 2" /></td>
<td><img src="image3" alt="Switch Block 3" /></td>
</tr>
<tr>
<td>System Board Switches</td>
<td>Switch Block 1</td>
<td>Switch Block 2</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------------</td>
<td>----------------</td>
</tr>
<tr>
<td>64/256K Option Card Switches</td>
<td>64K Option Card Switches</td>
<td>32K Option Card Switches</td>
</tr>
<tr>
<td>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>
<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>
### System Board Switches

<table>
<thead>
<tr>
<th>Setting</th>
<th>Switch Block 1</th>
<th>Switch Block 2</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 - 64K option</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 64/256K Option Card Switches

- Switch 1:
  - Positions 1, 2, 3, 4, 5, 6, 7, 8 set

- Switch 2:
  - Positions 1, 2, 3, 4, 5, 6, 7, 8 set

### 64K Option Card Switches

- Switch 1:
  - Positions 1, 2, 3, 4, 5, 6, 7, 8 set

- Switch 2:
  - Positions 1, 2, 3, 4, 5, 6, 7, 8 set

### 32K Option Card Switches

- Switch 1:
  - Positions 1, 2, 3, 4, 5, 6, 7, 8 set

- Switch 2:
  - Positions 1, 2, 3, 4, 5, 6, 7, 8 set

### Memory Configuration

- **640K Total Memory**
- **384K + (256K on System Board)**
## Extender Card Switch Settings

<table>
<thead>
<tr>
<th>System Memory</th>
<th>Extender Card Switch Block</th>
<th>Memory Segment</th>
</tr>
</thead>
<tbody>
<tr>
<td>16K to 64K</td>
<td><img src="image" alt="Switch Block 1" /></td>
<td>1</td>
</tr>
<tr>
<td>96K to 128K</td>
<td><img src="image" alt="Switch Block 2" /></td>
<td>2</td>
</tr>
<tr>
<td>160K to 192K</td>
<td><img src="image" alt="Switch Block 3" /></td>
<td>3</td>
</tr>
<tr>
<td>224K to 256K</td>
<td><img src="image" alt="Switch Block 4" /></td>
<td>4</td>
</tr>
<tr>
<td>288K to 320K</td>
<td><img src="image" alt="Switch Block 5" /></td>
<td>5</td>
</tr>
<tr>
<td>352K to 384K</td>
<td><img src="image" alt="Switch Block 6" /></td>
<td>6</td>
</tr>
<tr>
<td>416K to 448K</td>
<td><img src="image" alt="Switch Block 7" /></td>
<td>7</td>
</tr>
<tr>
<td>480K to 512K</td>
<td><img src="image" alt="Switch Block 8" /></td>
<td>8</td>
</tr>
<tr>
<td>544K to 576K</td>
<td><img src="image" alt="Switch Block 9" /></td>
<td>9</td>
</tr>
<tr>
<td>608K to 640K</td>
<td><img src="image" alt="Switch Block A" /></td>
<td>A</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 (\(\mu s\)): One-millionth of a second.

milli (m): One thousand or one thousandth.

milliampere (mA): One thousandth of an ampere.
millisecond (ms): One thousandth of a second.

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

mode: (1) A method of operation; for example, the binary mode, the interpretive mode, the alphanumeric mode. (2) The most frequency 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.

National Semiconductor Corporation. INS 8250 Asynchronous Communications Element. This book documents physical and operating characteristics of the INS 8250.
INDEX

A

A/N mode (alphanumeric mode) 1-131
A0-A19 (Address Bits 0 to 19), I/O channel 1-18
adapter card with ROM 2-10
adapter,
   asynchronous communication 1-223
   binary synchronous communication 1-251
   color/graphics monitor 1-131
   diskette drive 1-159
   fixed disk drive 1-187
   game control 1-211
   monochrome display and printer 1-129
   printer 1-117
   synchronous data link control 1-271
Address Bits 0 to 19 (A0-A19), I/O channel 1-131
Address Bits (asynchronous communication) 1-225
Address Enable (AEN), I/O channel 1-22
Address Latch Enable (ALE), I/O channel 1-20
address map, I/O 1-10
AEN (Address Enable), I/O channel 1-22
ALE (Address Latch Enable), I/O channel 1-20
all points addressable mode 1-129, 1-132
alphanumeric mode, 1-136
   high resolution 1-137
   low resolution 1-137
alt (keyboard extended code) 2-15
APA mode (all points addressable mode) 1-131, 1-132
asynchronous communications adapter, 1-223
   adapter address jumper module 1-249
   address bits 1-225
   block diagram 1-224
   connector specifications 1-250
   current loop interface 1-227
   divisor latch least significant bit 1-237
   divisor latch most significant bit 1-238
   I/O decode 1-225
INS8250 functional pin description 1-229
INS8250 input signals 1-229
INS8250 input/output signals 1-233
INS8250 output signals 1-232
interface descriptions 1-226
interface format jumper module 1-249
interrupt control functions 1-237
interrupt enable register 1-243
interrupt identification register 1-240
interrupts 1-226
line control register 1-235
line status register 1-239
modem control register 1-244
modem status register 1-246
modes of operation 1-224
programmable baud rate generator 1-237
programming considerations 1-234
receiver buffer register 1-247
reset functions 1-230
transmitter holding register 1-248
voltage interchange information 1-228
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 1-237
bell (printer) 1-102
bibliography 1-1
binary synchronous communications adapter, 1-251
  8251A programming procedures 1-262
  8251A universal synchronous/asynchronous
    receiver/transmitter 1-252
  8253-5 programmable interval timer 1-257
  8255A-5 programmable peripheral interface 1-256
block diagram 1-252
command instruction format 1-264
connector information 1-269
data bus buffer 1-253
interface signal information 1-266
interrupt information 1-268
mode instruction definition 1-263
read/write control logic 1-253
receive 1-258
receiver buffer 1-255
receiver control 1-255
status read definition 1-265
transmit 1-265
transmitter buffer 1-254
transmitter control 1-255
typical programming sequence 1-259

BIOS,
cassette logic (see cassette logic BIOS)
fixed disk ROM A-87
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)

block diagram

8251A universal synchronous/asynchronous receiver/transmitter 1-252
8273 SDLC protocol controller 1-272
asynchronous communications adapter 1-224
cassette circuits 1-29
color/graphics monitor adapter 1-134
coprocessor 1-37
diskette drive adapter 1-160
expansion board 1-80
extender card 1-87
fixed disk drive adapter 1-188
game control adapter 1-211
keyboard interface 1-75
monochrome display adapter 1-124
printer adapter 1-118
prototype card 1-218
receiver card 1-89
speaker drive system 1-24
synchronous data link control adapter 1-271
system 1-2
break (keyboard extended code) 2-17

BSC adapter
(see binary synchronous communications)
cable  
    communications adapter  1-301
    expansion unit  1-79
    printer  1-91
cancel (printer)  1-103
cancel ignore paper end (printer)  1-103
cancel skip perforation (printer)  1-109
caps lock (keyboard extended code)  2-16
card dimensions and specifications  E-4
card,  
    dimensions and specifications  E-4
    extender  1-85
    prototype  1-217
    receiver  1-88
carriage return (printer)  1-102
cassette circuit block diagram  
    motor control  1-30
    read hardware  1-29
    write hardware  1-29
cassette interface,  1-28  
    connector specifications  1-31
cassette logic,  
    BIOS  2-21
    cassette read  2-23
    cassette write  2-22
    data record architecture  2-24
    data record components  2-24
    error recovery  2-24
    interrupt  2-21
    software algorithms  2-28
cassette read  2-23
cassette ROM BIOS  2-21
cassette write  2-22
CCITT,  F-1  
    standards  F-1
character attributes  
    color/graphics monitor adapter  1-140
    monochrome display adapter  1-140
character codes  
    keyboard  2-11
character set,
  graphics printer (set 1)  1-113
  graphics printer (set 2)  1-115
  matrix printer  1-111
  quick reference  C-12
clear printer buffer (printer)  1-110
CLK (system clock), I/O channel  1-19
color display  1-157
  operating characteristics  1-157
  specifications  E-2
color select register  1-149
color/graphics monitor adapter  1-131
  6845 register description  1-148
  alphanumeric mode  1-136
  alphanumeric mode (high-resolution)  1-144
  alphanumeric mode (low-resolution)  1-142
  block diagram  1-134
  character attributes  1-140
  color-select register  1-149
  composite connector specifications  1-155
  connector specifications  1-156
  direct-drive connector specifications  1-155
  display buffer basic operation  1-145
  graphics mode  1-141
  graphics mode (high resolution)  1-144
  graphics mode (low resolution)  1-142
  graphic mode (medium resolution)  1-142
  light pen connector specifications  1-156
  major components  1-135
  memory requirements  1-154
  mode control and status register  1-149
  mode register summary  1-152
  mode select register  1-151
  programming considerations  1-147
  RF modulator connector specifications  1-156
  sequence of events  1-153
  status register  1-153
  summary of available colors  1-146
  colors, summary of available  1-146
command status register 0  1-172
command status register 1  1-173
command status register 2  1-174
command status register 3  1-175
command summary,
  diskette drive adapter  1-166
  fixed disk drive adapter  1-195
communications adapter cable  1-301
  connector specifications  1-302
communications  F-1
  establishing a link  F-3
component diagram,
  system board  1-7
compressed (printer)  1-103
compressed off (printer)  1-103
connector specifications,
  asynchronous communications adapter  1-250
  binary synchronous communications  1-269
  cassette interface  1-31
  color/graphics monitor adapter  1-155
  communications adapter cable  1-302
  diskette drive adapter (external)  1-182
  diskette drive adapter (internal)  1-181
  game control adapter  1-216
  keyboard interface  1-78
  monochrome display adapter  1-128
  printer adapter  1-122
  synchronous data link control adapter  1-299
connectors,
  power supply (system unit)  1-26
  power supply (expansion unit)  1-83
considerations, programming
  (see programming considerations)
control byte, fixed disk drive adapter  1-194
control codes, printer  1-101
control/read/write logic  1-274
coprocessor,
  (see math coprocessor)
ctrl (keyboard extended code)  2-15
current loop interface  1-227

D
D0-D7 (data bits 0 to 7), I/O channel  1-18
DACK0-DACK3 (DMA Acknowledge 0 to 3), I/O channel  1-21
Data Bits 0 to 7 (D0-D7), I/O channel  1-18
  data flow,
    system board  1-8
data record architecture, cassette 2-24
data record components, cassette 2-24
data register 1-193
data transfer mode register 1-288
DEF SEG (default segment workspace) 2-8
default workspace segment (DEF SEG) 2-8
diagram, block (see block diagram)
digital output register 1-161
diskette drive adapter 1-159
  adapter input 1-179
  adapter output 1-178
block diagram 1-160
command status register 0 1-172
command status register 1 1-173
command status register 2 1-174
command status register 3 1-175
command summary 1-166
connector specifications (external) 1-182
connector specifications (internal) 1-181
digital-output register 1-161
DPC registers 1-174
drive A and B interface 1-178
drive constants 1-176
FDC constants 1-176
floppy disk controller 1-162
functional description 1-161
programming considerations 1-164
programming summary 1-175
symbol descriptions 1-164
system I/O channel interface 1-176
diskette drive, 1-183
  electrical specifications 1-184
  mechanical specifications 1-184
  switch settings G-1
diskettes 1-185
display adapter type switch settings G-1
display,
  color 1-157
  monochrome 1-123
divisor latch,
  least significant bit 1-237
  most significant bit 1-238
DMA Acknowledge 0 to 3 (DACK0-DACK3),
  I/O channel 1-21
DMA Request 1 to 3 (DRQ1-DRQ3), I/O channel 1-21
DOS reserved interrupts 2-9
DOS,
  keyboard functions 2-21
  reserved interrupts 2-9
double strike (printer) 1-106
double strike off (printer) 1-107
double width (printer) 1-99, 1-103
double width off (printer) 1-103
DPC registers 1-175
DRQ1-DRQ3 (DMA Request 1 to 3), I/O channel 1-21

E

EIA, F-1
  standards F-1
  emphasized (printer) 1-106
  emphasized off (printer) 1-106
  error recovery, cassette 2-24
  escape (printer) 1-104
  establishing a communications link F-3
expansion board, 1-79
  block diagram 1-80
expansion channel 1-81
expansion unit, 1-79
  cable 1-79
  expansion board 1-79
  expansion channel 1-81
extender card 1-85
  interface information 1-90
  power supply 1-83
  power supply connectors 1-83
  receiver card 1-88
  specifications E-2
extender card, 1-85
  block diagram 1-87
  programming considerations 1-86
  switch settings G-1
F

FABS 1-44
FADD 1-43
FBLD 1-45
FBSTP 1-46
FCHS 1-46
FCLEX/FNCLEX 1-46
FCOM 1-47
FCOMP 1-47
FCOMPP 1-48
FDECSTP 1-48
FDISI/FNDISI 1-48
FDIV 1-49
FDIVR 1-50
FENI/FNENI 1-51
FFREE 1-51
FICOM 1-51
FICOMP 1-52
FILD 1-52
FINCSTP 1-52
FINIT/FNINIT 1-53
FIST 1-54
FISTP 1-54
fixed disk controller 1-185
fixed disk drive 1-201
fixed disk drive adapter 1-185
   block diagram 1-186
   command summary 1-193
   control byte 1-192
   data register 1-191
   fixed disk controller 1-185
   interface specifications 1-200
   programming considerations 1-187
   programming summary 1-197
   ROM BIOS listing A-87
   sense bytes 1-187
   status register 1-187
   system I/O channel interface 1-198
fixed disk drive, 1-201
   electrical specifications 1-202
   mechanical specifications 1-202
fixed disk ROM BIOS  A-87
FLD  1-55
FLDCW  1-55
FLDENV  1-56
FLDLG2  1-56
FLDLN2  1-56
FLDL2E  1-57
FLDL2T  1-57
FLDPI  1-57
FLDZ  1-58
FLDI  1-58
floppy disk controller  1-160
FMUL  1-59
FNOP  1-60
FPATAN  1-60
FPREM  1-60
FPTAN  1-61
FRNDINT  1-61
FRSTOR  1-61
form feed (printer)  1-102
FSAVE/FNSAVE  1-62
FSCALE  1-62
FSQRT  1-62
FST  1-63
FSTCW/FNSTCW  1-63
FSTENV/FNSTENV  1-64
FSTP  1-64
FSTSW/FNSTSW  1-65
FSUB  1-65
FSUBR  1-66
FTST  1-67
FWAIT  1-68
FXAM  1-68
FXCH  1-69
FXTRACT  1-70
FYL2X  1-70
FYL2XP1  1-71
F2XM1  1-71

G
game control adapter,  1-211
   block diagram  1-211
   connector specifications  1-216

J-10  Index
functional description 1-212
I/O channel description 1-213
interface description 1-214
joy stick schematic diagram 1-215
glossary, H-1
graphics mode, 1-141
  high resolution 1-144
  low resolution 1-142
  medium resolution 1-142

H

hardware interrupt listing 1-11
home head (printer) 1-105
horizontal tab (printer) 1-102

I

I/O address map 1-10
I/O bit map, 8255A 1-12
I/O CH CK (I/O Channel Check), I/O channel 1-20
I/O CH RDY (I/O Channel Ready), I/O channel 1-20
I/O Channel Check (I/O CH CK), I/O channel 1-20
I/O channel interface,
  diskette drive adapter 1-176
  fixed disk drive adapter 1-187
  prototype card 1-217
I/O Channel Ready (I/O CH RDY), I/O channel 1-20
I/O channel, 1-17
  -I/O Channel Check (I/O CH CK) 1-20
  -I/O Read Command (IOR) 1-20
  -I/O Write Command (IOW) 1-21
Address Bits 0 to 19 (A0-A19) 1-20
Address Enable (AEN) 1-21
Address Latch Enable (ALE) 1-20
Data Bits 0 to 7 (D0-D7) 1-20
description 1-20
diagram 1-18
DMA Request 1 to 3 (DRQ1-DRQ3) 1-21
I/O Channel Ready (I/O CH RDY) 1-20
Interrupt Request 2 to 7 (IRQ2-IRQ7) 1-20
Memory Read Command (MEMR) 1-21
Memory Write Command (MEMW) 1-21
Oscillator (OSC) 1-19
Reset Drive (RESET DRV) 1-20
System Clock (CLK) 1-20
Terminal Count (T/C) 1-22
I/O Read Command (IOR), I/O channel 1-21
I/O Write Command (IOW), I/O channel 1-21
IBM 10MB Fixed Disk Drive 1-201
IBM 5-1/4" Diskette Drive 1-183
IBM 5-1/4" Diskette Drive Adapter 1-159
IBM 80 CPS Graphics Printer 1-91
IBM 80 CPS Matrix Printer 1-91
IBM 80 CPS Printers 1-91
IBM Asynchronous Communications Adapter 1-223
IBM Binary Synchronous Communications Adapter 1-251
IBM Color Display 1-157
IBM Color/Graphics Monitor Adapter 1-131
IBM Communicatons Adapter Cable 1-301
IBM Fixed Disk Drive Adapter 1-187
IBM Game Control Adapter 1-211
IBM Memory Expansion Options 1-205
IBM Monochrome Display and Printer Adapter 1-223
IBM Monochrome Display 1-129
IBM Personal Computer Math Coprocessor 1-33
IBM Printer Adapter 1-117
IBM Prototype Card 1-215
IBM Synchronous Data Link Controller Adapter 1-271
ignore paper end (printer) 1-104
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-15
  processor control B-15
  register model B-2
second instruction byte summary  B-3
segment override prefix   B-4
software interrupt listing  2-4
string manipulation  B-10
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  1-12
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  1-273
interrupt enable register  1-243
interrupt identification register  1-243
interrupt listing,
  8088 hardware  1-11
  8088 software  2-4
Interrupt Request 1 to 7 (IRQ2-IRQ7), I/O channel  1-20
interrupts,
  8088 hardware  1-11
  8088 software  2-4
  asynchronous communications adapter  1-223
  BASIC reserved  2-7
  DOS reserved  2-21
  special  2-7
IOR (I/O Read Command), I/O channel  1-20
IOW (I/O Write Command), I/O channel  1-21
IRQ2-IRQ7 (Interrupt Request 2 to 7), I/O channel  1-20

J

joy stick,
  positions  1-221
  schematic diagram  1-215
jumper module, asynchronous communications adapter  1-249
K

keyboard extended codes,
  alt  2-15
  break 2-16
  caps lock 2-16
  ctrl 2-15
  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 1-73
  BASIC screen editor special functions 2-20
  character codes 2-11
  commonly used functions 2-18
  diagram 1-76
  DOS special functions 2-20
  encoding 2-11
  extended functions 2-14
  interface block diagram 1-75
  interface connector specifications 1-78
  scan codes 1-77
  specifications E-1

L

light pen connector specifications 1-156
line control register 1-235
line feed (printer) 1-102
line status register 1-239
logic diagrams D-1

M

math coprocessor 1-33
  block diagram 1-37
  control unit 1-37
  control word 1-40
data types  1-34
exception pointers  1-41
FABS  1-44
FADD  1-44
FBLD  1-45
FBSTP  1-46
FCHS  1-46
FCLEX/FNCLEX  1-46
FCOM  1-47
FCOMP  1-47
FCOMPP  1-48
FDECSTP  1-48
FDISI/FNDISI  1-48
FDIV  1-49
FDIVR  1-50
FENI/FNENI  1-51
FFREE  1-51
FICOM  1-51
FICOMP  1-52
FILD  1-52
FINCSTP  1-52
FINIT/FNINIT  1-53
FIST  1-54
FISTP  1-54
FLD  1-55
FLDCW  1-55
FLDENV  1-56
FLDLG2  1-56
FLDLN2  1-56
FLDL2E  1-57
FLDL2T  1-57
FLDPI  1-57
FLDZ  1-58
FLD1  1-58
FMUL  1-59
FNOP  1-60
FPATAN  1-60
FPREM  1-60
FPTAN  1-61
FRNDINT  1-61
FRSTOR  1-61
FSAVE/FNSAVE 1-62
FSCALE 1-62
FSQRT 1-62
FST 1-63
FSTCW/FNSTCW 1-63
FSTENV/FNSTENV 1-64
FSTP 1-64
FSTSW/FNSTSW 1-65
FSUB 1-65
FSUBR 1-66
FTST 1-67
FWAIT 1-68
FXAM 1-68
FXCH 1-69
FXTRACT 1-70
FYL2X 1-70
FYL2XP1 1-71
F2XM1 1-71
hardware interface 1-35
instruction set 1-43
interconnection 1-36
number system 1-42
programming interface 1-34
register stack 1-38
status word 1-39
tag word 1-41
memory expansion options, 1-205
   DIP module start address 1-208
   memory module description 1-206
   memory module pin configuration 1-207
   memory option switch settings G-1
   R/W memory operating characteristics 1-206
   switch-configurable start address 1-208
memory locations,
   reserved 2-8
memory map,
   BIOS 2-9
   system 1-13
Memory Read Command (MEMR), I/O channel 1-21
memory switch settings, G-1
   extender card G-1
   memory options G-1
   system board G-1

J-16 Index
Memory Write Command (MEMW), I/O channel 1-21
(MEMR) Memory Read Command, I/O channel 1-21
(MEMW) Memory Write Command, I/O channel 1-21
microprocessor (see Intel 8088 microprocessor)
mode control and status register 1-149
mode select register 1-151
modem control register 1-244
modem status register 1-246
monochrome display 1-129
monochrome display and printer adapter 1-123
monochrome display adapter 1-123
   6845 CRT control port 1-127
   6845 CRT status port 1-127
   block diagram 1-124
   character attributes 1-138
   connector specifications 1-128
   I/O address and bit map 1-127
   programming considerations 1-125
monochrome display, 1-129
   operating characteristics 1-129
   specifications E-3
Motorola 6845 CRT Controller,
   (see color/graphics monitor adapter)
   (see monochrome display adapter)

National Semiconductor INS8250 Asynchronous
   (see asynchronous communications adapter)
   functional pin description 1-229
   input signals 1-229
   input/output signals 1-233
   output signals 1-232
null (printer) 1-102

one bit delay mode register 1-289
operating mode register 1-289
OSC (oscillator) 1-19
Oscillator (OSC), I/O channel 1-19
over-voltage/over-current (expansion unit) 1-84
over-voltage/over-current (system unit) 1-27
parameter passing (ROM BIOS) 2-3
pause (keyboard extended code) 2-17
power good signal (expansion unit) 1-84
power good signal (system unit) 1-27
power supply (expansion unit) 1-82
    connectors 1-83
    input requirements 1-82
    over-voltage/current protection 1-84
    pin assignments 1-83
    power good signal 1-83
    Vac output 1-82
    Vdc output 1-82
power supply (system unit) 1-23
    connectors and pin assignments 1-26
    input requirements 1-24
    over-voltage/current protection 1-27
    pin assignments 1-26
    power good signal 1-27
    Vac output 1-25
    Vdc output 1-25
print screen (keyboard extended code) 2-17
printer adapter, 1-117
    block diagram 1-118
    connector specifications 1-122
    programming considerations 1-119
printer control codes, 1-101
    1/8-inch line feeding 1-104
    1920 bit-image graphics mode 1-110
    480 bit-image graphics mode 1-107
    7/72-inch line feeding 1-104
    960 bit-image graphics mode 1-109
    960 bit-image graphics mode normal speed 1-110
    bell 1-102
    cancel 1-103
    cancel ignore paper end 1-105
    cancel skip perforation 1-109
    carriage return 1-102
    clear printer buffer 1-110
    compressed 1-103
    compressed off 1-103
    double strike 1-106
    double strike off 1-107
    double width 1-103, 1-110

J-18 Index
double width off 1-103
emphasized 1-106
emphasized off 1-106
escape 1-103
form feed 1-102
home head 1-105
horizontal tab 1-102
ignore paper end 1-104
line feed 1-102
null 1-102
printer deselected 1-103
printer selected 1-103
select character set 1 1-104
select character set 2 1-104
set horizontal tab stops 1-106
set lines per page 1-106
set skip perforation 1-109
set variable line feeding 1-105, 1-107
set vertical tabs 1-105
starts variable line feeding 1-105
subscript/superscript 1-109
subscript/superscript off 1-109
underline 1-104
unidirectional printing 1-109
vertical tab 1-102
printer deselected (printer) 1-103
printer selected (printer) 1-103
printer, 1-91
  additional specifications 1-93
cable 1-91
  connector pin assignment 1-97
  control codes 1-101
  graphic character set 1 1-113
  graphic character set 2 1-115
  interface signal descriptions 1-96
  matrix character set 1-111
  modes 1-100
  parallel interface 1-96
  parallel interface timing diagram 1-96
  specifications 1-92, E-3
  switch locations 1-94
  switch settings 1-94
processor (see Intel 8088 microprocessor)
programmable baud rate generator 1-237
programming considerations,
  asynchronous communications adapter  1-234
  binary synchronous communications adapter  1-259
  color/graphics monitor adapter  1-131
  diskette drive adapter  1-159
  extender card  1-86
  fixed disk drive adapter  1-187
  monochrome display adapter  1-123
  printer adapter  1-123
  receiver card  1-88
  SDLC adapter  1-281
  prototype card,  1-217
    block diagram  1-218
    external interface  1-222
    I/O channel interface  1-219
    layout  1-219
  system loading and power limitations  1-221

Q
  quick reference, character set  C-12

R
  receiver buffer register  1-247
  receiver card,  1-88
    block diagram  1-89
    programming considerations  1-86
  register,
    6845 description (color/graphic adapter)  1-146
    color select (color/graphic adapter)  1-147
    command status 0 (diskette drive adapter)  1-172
    command status 1 (diskette drive adapter)  1-173
    command status 2 (diskette drive adapter)  1-174
    command status 3 (diskette drive adapter)  1-175
    data (fixed disk drive adapter)  1-192
    data transfer mode (SDLC)  1-288
    digital output (diskette drive adapter)  1-161
    DPC (diskette drive adapter)  1-175
  interrupt enable (asynchronous communications)  1-243
  interrupt identification (asynchronous communications)  1-241
  line control (asynchronous communications)  1-234
  line status (asynchronous communications)  1-239
  mode control and status (color/graphics)  1-149
mode select (color/graphics) 1-151
modem control (asynchronous communications) 1-244
modem status (asynchronous communications) 1-246
one-bit delay mode (SDLC) 1-289
operating mode (SDLC) 1-286
receiver buffer (asynchronous communications) 1-247
serial I/O mode (SDLC) 1-288
status (color/graphics) 1-153
status (fixed disk drive adapter) 1-187
transmitter holding (asynchronous communications) 1-248

reserved interrupts,
  BASIC and DOS 2-7
reserved memory locations 2-7
Reset Drive (RESET DRV), I/O channel 1-19
RESET DRV (Reset Drive), I/O channel 1-19
RF modulator connector specifications 1-156
ROM BIOS, 2-2
  Cassette A-74
  Fixed Disk A-87
  System A-2
ROM, adapter cards with 2-10
RS-232C,
  interface standards F-2

S

scan codes,
  keyboard 1-77
scroll lock (keyboard extended code) 2-16
SDLC (see synchronous data link control)
select character set 1 (printer) 1-104
select character set 2 (printer) 1-104
sense bytes, fixed disk drive adapter 1-189
serial I/O mode register 1-288
set horizontal tab stops (printer) 1-106
set lines per page (printer) 1-106
set skip perforation (printer) 1-109
set variable line feeding (printer) 1-105, 1-107
set vertical tabs (printer) 1-105
shift (keyboard extended code) 2-15
shift key priorities (keyboard code) 2-16
shift states (keyboard extended code) 2-15
software interrupt listing 2-4
speaker connector 1-23
speaker drive system 1-23
speaker interface 1-23
specifications,
  80 CPS printers  E-3
  color display    E-2
  expansion unit  E-2
  keyboard        E-1
  monochrome display E-3
  printer 1-92
  printer (additional)  1-93
  system unit E-1
stack area 2-7
starts variable line feeding (printer) 1-104
status register,
  color/graphics monitor adapter 1-154
  fixed disk drive adapter 1-189
  synchronous data link control adapter 1-282
subscript/superscript (printer) 1-109
subscript/superscript off (printer) 1-109
switch settings, G-1
  diskette drive G-1
  display adapter type G-1
  extender card G-1
  memory options G-1
  printer 1-91
  system board G-1
  system board memory G-1
synchronous data link control communications adapter, 1-271
  8253-5 interval timer control word  1-285
  8253-5 programmable interval timer 1-281
  8255A-5 port A assignments 1-280
  8255A-5 port B assignments 1-280
  8255A-5 port C assignments 1-281
  8255A-5 programmable peripheral interface 1-280
  8273 command phase flow chart 1-292
  8273 commands 1-291
  8273 control/read/write registers 1-275
  8273 data interfaces 1-276
  8273 elements of data transfer interface 1-276
  8273 mode register commands 1-288
  8273 modem control block 1-277
  8273 modem control port A 1-277
  8273 modem control port B 1-278
  8273 modem interface 1-277
  8273 protocol controller operations 1-271
  8273 protocol controller structure 1-273
  8273 register selection 1-274
  8273 SDLC protocol controller block diagram 1-273

J-22  Index
8273 transmit/receiver timing 1-279
block diagram 1-271
command phase 1-290
connector specifications 1-299
control/read/write logic 1-274
data transfer mode register 1-288
device addresses 1-297
execution phase 1-293
general receive 1-294
initialization/configuration commands 1-286
initializing the SDLC adapter 1-283
interface information 1-298
interrupt information 1-297
one bit delay code register 1-289
operating mode register 1-286
partial byte received codes 1-296
processor interface 1-274
programming considerations 1-281
protocol control module features 1-272
protocol controller operations 1-272
result code summary 1-296
result phase 1-291
selective receive 1-295
serial data timing block 1-279
serial I/O mode register 1-288
status register format 1-282
transmit 1-294
system block diagram 1-2
system board, 1-3
component diagram 1-7
data flow 1-8
R/W memory operating characteristics 1-202
switch settings 1-19
System Clock (CLK), I/O channel 1-19
system memory map 1-13
system reset (keyboard extended code) 2-16
system ROM BIOS 1-27
system unit, 1-3
cassette interface 1-31
I/O channel 1-17
I/O channel diagram 1-18
keyboard interface 1-78
power supply 1-23
speaker interface 1-22
specifications E-1
system board 1-3
T

T/C (Terminal Count), I/O channel 1-22
transmitter holding register 1-248

U

underline (printer) 1-104
unidirectional printer (printer) 1-109

V

Vac output,
   expansion unit 1-82
   system unit 1-25
Vdc output,
   expansion unit 1-82
   system unit 1-25
vectors with special meanings 2-5
vertical tab (printer) 1-102
voltage interchange,
   asynchronous communications adapter 1-228

Numerics

1/8 inch line feeding (printer) 1-104
1920 bit-image graphics mode (printer) 1-110
480 bit-image graphics mode (printer) 1-107
6845,
   (see color/graphics monitor adapter)
   (see monochrome display adapter)
7/72 inch line feeding (printer) 1-104
8088,
   (see Intel 8088 microprocessor)
8250,
   (see asynchronous communications adapter)
8253-5,
   (see synchronous data link control adapter)
8255A 1-12
8255A-5,
   (see synchronous data link control adapter)
8273,
   (see synchronous data link control adapter)
960 bit-image graphics mode (printer) 1-109
960 bit-image graphics mode normal speed (printer) 1-110
Product Comment Form

TECHNICAL REFERENCE

Your comments assist us in improving our products. 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.

For prompt resolution to questions regarding set up, operation, program support, and new program literature, contact the Authorized IBM Personal Computer Dealer in your area.

Comments:

If you wish a reply, provide your name and address in this space.

Name ____________________________________________
Address ____________________________________________
City ___________ State ______________________________
Zip Code __________
Your comments assist us in improving our products. 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.

For prompt resolution to questions regarding set up, operation, program support, and new program literature, contact the Authorized IBM Personal Computer Dealer in your area.

Comments:

If you wish a reply, provide your name and address in this space.

Name __________________________________________
Address _______________________________________
City ___________________ State ___________________
Zip Code _______________
IBM PERSONAL COMPUTER
SALES & SERVICE
P.O. BOX 1328-C
BOCA RATON, FLORIDA 33432
Product Comment Form

TECHNICAL REFERENCE

Your comments assist us in improving our products. 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.

For prompt resolution to questions regarding set up, operation, program support, and new program literature, contact the Authorized IBM Personal Computer Dealer in your area.

Comments:

If you wish a reply, provide your name and address in this space.

Name ____________________________________________
Address ____________________________________________
City ______ State ____________________________
Zip Code __________