April 2011

IMPORTANT

1. The motherboard described in this document is the first motherboard, the 16KB-64KB one.

2. If the BIOS on the 16KB-64KB motherboard has been upgraded to the third revision one, then the switch settings shown for SW2 ("switch 2") on the motherboard no longer apply.

The BIOS revision can be determined by examination of the seven digit number on motherboard chip U33.

First revision: 5700051
Second revision: 5700671
Third revision: 1501476
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.
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.

First Edition (August 1981)

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 Product Comment Form is provided at the back of this publication. If this form has been removed, address comment to: IBM Corp., Personal Computer, P.O. Box 1328, Boca Raton, Florida 33432. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligations whatever.

© Copyright International Business Machines Corporation 1981
PREFACE

The IBM Personal Computer Technical Reference Manual is designed to provide hardware design and interface information. This publication also provides Basic Input Output System (BIOS) information as well as programming support matter.

This manual is intended for programmers, engineers involved in hardware and software design, designers, and interested persons who have a need to know how the IBM Personal Computer is designed and works.

This manual has three sections:

Section - 1

"HARDWARE OVERVIEW," features an overview of the system as a whole calling out specific items such as the System Unit, Keyboard, IBM Monochrome Display and the 80 CPS Matrix Printer.

Section - 2

"HARDWARE," contains a description for each functional part of the system. This section also contains specifications for power, timing, and interface. Programming considerations are supported by coding tables, command codes and registers.

Section - 3

"ROM and SYSTEM USAGE," describes BIOS as well as how to use BIOS, interrupt vector listings, memory map, vectors with special meanings, a cassette section, a keyboard encoding section, and a set of Low Memory Maps.

"APPENDICES," to address the ROM BIOS listing, an instruction set, logic diagrams, and expanded charts used to support specific hardware descriptions.
CONTENTS

SECTION 1. HARDWARE OVERVIEW .......... 1-1
System Block Diagram ......................... 1-4

SECTION 2. HARDWARE ......................... 2-1
System Board .................................. 2-3
System Board Data Flow ........................ 2-6
I/O Channel .................................... 2-8
I/O Channel Diagram .......................... 2-9
System Board I/O Channel Description ........ 2-10
System Board Component Diagram ............. 2-13
Keyboard ....................................... 2-14
Keyboard Interface Block Diagram ............. 2-15
Keyboard Diagram ................................ 2-16
Keyboard Scan Codes ........................... 2-17
Keyboard Interface Connector Specifications . 2-18
Cassette User Interface ....................... 2-19
  Cassette Jumpers ............................. 2-19
  Circuit Block Diagrams ...................... 2-19
Cassette Interface Connector Specifications . 2-21
Speaker Interface .............................. 2-22
  Speaker Drive System Block Diagram ....... 2-22
I/O Address Map ................................ 2-23
System Memory Map ............................ 2-25
System Board And Memory Expansion Switch Settings . 2-28
5 1/4" Diskette Drives Switch Settings ........ 2-29
Monitor Type Switch Settings .................. 2-29
System Board Memory Switch Settings ........ 2-30
32/64 KB Memory Expansion Option Switch Settings . 2-31

Power Supply .................................. 2-33
Power Supply Location ......................... 2-34
Input Requirements ............................ 2-34
DC Output ..................................... 2-34
AC Output ..................................... 2-34
Power Supply Connectors And Pin Assignments ... 2-35
Important Operating Characteristics .......... 2-36
  Over Voltage/Current Protection .......... 2-36
  Signal Requirements ......................... 2-36
Programming Considerations ........................................ 2-55
Programming The 6845 Controller .................................. 2-55
6845 Register Description ........................................... 2-56
Programming the Mode Control and Status Register ........... 2-57
Color Select Register .............................................. 2-57
Mode Select Register .............................................. 2-58
Mode Register Summary ............................................. 2-58
Status Register ..................................................... 2-59
Sequence of Events .................................................. 2-59
Memory Requirements ............................................... 2-60
Interrupt Level ...................................................... 2-60
I/O Address and Bit Map ............................................ 2-61
Color/Graphics Monitor Adapter Direct Drive, and
Composite Interface Pin Assignment ............................. 2-62
Color/Graphics Monitor Adapter Auxiliary
Video Connectors ..................................................... 2-63

Parallel Printer Adapter ............................................. 2-65
Parallel Printer Block Diagram .................................... 2-66
Programming Considerations ....................................... 2-67
Parallel Printer Adapter Interface Connector
Specifications .......................................................... 2-69
IBM 80 CPS Matrix Printer .......................................... 2-70
Printer Specifications ............................................... 2-71
Setting The DIP Switches ............................................ 2-72
Functions and Conditions of DIP Switch 1 ....................... 2-72
Functions and Conditions of DIP Switch 2 ....................... 2-73
Parallel Interface Description ...................................... 2-73
Connector Pin Assignment and Descriptions of
Interface Signals ..................................................... 2-74
Parallel Interface Timing Diagram ................................ 2-77
ASCII Coding Table .................................................. 2-78
ASCII Control Codes ................................................ 2-79

5 1/4” Diskette Drive Adapter ...................................... 2-89
5 1/4” Diskette Drive Adapter Block Diagram .................. 2-90
Functional Description .............................................. 2-91
Digital Output Register ............................................. 2-91
Floppy Disk Controller .............................................. 2-91
Programming Considerations ...................................... 2-94
Symbol Descriptions ................................................ 2-94
Command Summary ................................................... 2-96
Command Status Registers .......................................... 2-100
FIGURE LISTING

1. System Block Diagram ........................ 1-4
2. System Board Data Flow ...................... 2-6, 7
3. I/O Channel Diagram .......................... 2-9
4. System Board Component Diagram .............. 2-13
5. Keyboard Interface Block Diagram ............. 2-15
6. Keyboard Diagram .............................. 2-16
7. Cassette Interface Read Hardware .......... 2-19
8. Cassette Interface Write Hardware .......... 2-20
9. Cassette Motor Control ....................... 2-20
10. Speaker Drive System Block Diagram ....... 2-22
11. System Memory Map ......................... 2-25
12. System Memory Map (Increments of 16KB) .... 2-26
13. Power Supply and Connectors ............... 2-35
14. IBM Monochrome Display Adapter Block Diagram 2-38
15. Color/Graphics Monitor Adapter Block Diagram 2-47
16. Parallel Printer Adapter Block Diagram ..... 2-66
17. Location of (Printer) DIP Switches ..... 2-72
18. Parallel Interface Timing .................... 2-77
19. 5 1/4” Diskette Drive Adapter Block Diagram 2-90
20. Game Control Adapter Block Diagram ........ 2-117
21. Joystick Schematic ......................... 2-121
22. Asynchronous Communications Adapter Block Diagram 2-124
23. Current Loop Interface ...................... 2-127
24. Selecting The Interface Format .......... 2-146
25. BIOS Memory Map ........................... 3-7
**TABLE LISTING**

1. Keyboard Scan Codes .................................. 2-17
2. 6845 Initialization Parameters .......................... 2-41
3. Monochrome Vs Color/Graphics Attributes ............... 2-51
4. Color/Graphics Modes .................................... 2-51
5. Summary of Available Colors ............................ 2-55
6. 6845 Register Description ............................... 2-56
7. Printer Specifications ................................... 2-71
8. Functions and Conditions of DIP Switch 1 ............... 2-72
9. Functions and Conditions of DIP Switch 2 ............... 2-73
10. Connector Pin Assignment and Description of Interface Signals .................................. 2-74
11. ASCII Coding Table ..................................... 2-78
12. DC1/DC3 and Data Entry .................................. 2-82
13. Symbol Description ..................................... 2-94
14. Status Register 0 ....................................... 2-100
15. Status Register 1 ....................................... 2-101
16. Status Register 2 ....................................... 2-102
17. Status Register 3 ....................................... 2-103
18. Mechanical and Electrical Specifications ............... 2-112
19. Memory Module Pin Configuration ........................ 2-114
20. DIP Module Start Address ............................... 2-115
21. I/O Decodes (3F8 - 3FF) ................................ 2-125
22. Asynchronous Communications Reset Functions .......... 2-133
23. BAUD Rate at 1.843 Mhz ................................ 2-137
24. Interrupt Control Functions (Asynchronous) .......... 2-140
25. Character Codes ....................................... 3-11
26. Keyboard Extended Functions .......................... 3-14
27. Keyboard - Commonly Used Functions .................. 3-17
28. Basic Screen Editor Special Functions .................. 3-19
29. DOS Special Functions .................................. 3-19
30. 0-7F Interrupt Vectors .................................. 3-21
31. Basic & DOS Reserved Interrupts (80-3FF) ............... 3-22
32. Reserved Memory Locations (400-5FF) .................. 3-22
SECTION I.
HARDWARE OVERVIEW

The IBM Personal Computer has two major elements; a System Unit and a keyboard. In addition, a variety of options are offered including one or two 5-1/4” Diskette Drives with adapter which can be housed inside the System Unit, an IBM Monochrome Display, an 80 CPS Matrix Printer, two display adapters, storage increments to 256 KB, an Asynchronous Communications Adapter, Printer Adapter and a Game Control Adapter.

The System Unit is the heart of your IBM Personal Computer system. The System Unit houses the microprocessor, Read-Only Memory (ROM), Read/Write Memory, Power Supply, and System Expansion Slots for the attachment of up to five options. One or two 5-1/4” Diskette Drives can also be mounted in the system Unit providing 160KB of storage each.

The System Board is a large board which fits horizontally in the base of the System Unit and includes the microprocessor, 40KB ROM and 16KB memory. The memory can be expanded in 16KB increments to 64KB. The System Board also includes an enhanced version of the Microsoft BASIC-80 Interpreter without diskette functions. The BASIC Interpreter is included in the ROM. The System Board also permits the attachment of an audio cassette recorder for loading or saving programs and data.

The System Unit power system is a 63.5 watt, 4 level DC and 120 AC unit. It is a switching regulator design, allowing for light weight and high efficiency. The DC power capacity is designed to support an expanded system.

The 5-1/4” Diskette Drive Adapter fits into one of the five System Expansion Slots. This attachment supports two internal drives. The 5-1/4” Diskette Drive Adapter uses write precompensation and a phase lock loop for clock and data recovery.

The 5-1/4” Diskette Drive permits the IBM Personal Computer to read, write and store data on 5-1/4” diskettes. Each diskette stores approximately 160KB of data. Two of these drives may be installed internally in the System Unit.

The keyboard is attached to the System Unit with a light-weight, coiled cable. The keyboard features 83 keys, and offers commonly used data and word processing functions in a design combining the familiar typewriter and calculator pad layouts.
A base system requires one of two different display adapters, either a Color/Graphics medium resolution Monitor adapter or a high resolution monochrome alphanumeric adapter with a parallel printer adapter.

The Color/Graphics adapter operates at standard television frequencies (15,750Hz), allowing attachment to a variety of industry standard monitors, including home TVs with a user supplied RF modulator.

The Color/Graphic Monitor adapter supports a variety of modes selected by program control. The adapter supports color or black and white alphanumeric modes with line width of 40 or 80 characters and 25 lines. In the alphanumeric mode there are 256 characters.

This adapter provides both a standard composite video and direct drive outputs. In addition, a light pen feature input port is provided.

The IBM Monochrome Display is a high resolution green phosphor display offering the personal computer user quality usually found on larger computer systems. The display features an 11-1/2” screen with an anti-glare surface and a variety of highlighting choices. The screen displays 25 lines of 80 characters. It supports 256 different letters, numbers and special characters that are formed in a nine by 14 dot matrix.

The IBM Monochrome Display requires the Monochrome Display and Printer Adapter Option. This option installs in one of the System Unit’s five System Expansion Slots. The display is powered from the System Unit.

The 80 CPS Matrix Printer is a versatile, low cost, quality printer for the IBM Personal Computer. It prints in both directions at a nominal horizontal speed of 80 characters per second on continuous-feed, single or multi-part paper. The printer features four character sizes (40, 66, 80 or 132 characters per line), Power-on Self-test and simple paper loading and ribbon cartridge uppercase and lower case ASCII character set and 64 special graphic characters.

The 80 CPS Matrix Printer requires either the Monochrome Display and Printer Adapter or the Printer Adapter. These options install in one of the Systems Unit’s five System Expansion Slots. The Printer requires standard 120 volt, 60 Hz power through its own power cord. The printer requires the Printer Cable Option for attachment to the System Unit.
The 16KB Memory Expansion Kits allow you to increase the memory size of your IBM Personal Computer. The base system comes standard with 16KB of memory. Up to three 16KB Memory Expansion Kits may be installed to increase the memory size to 64KB. Memory can be further increased to 256KB with additional memory options once these three Expansion Kits are installed.

The Expansion Kits plug into the System Board and must be installed sequentially. They do not occupy any of the five System Expansion Slots.

The 32KB and 64KB Memory Expansion Options permit you to increase memory capacity beyond 64KB. Multiple 32KB and 64KB Memory Expansion Options may be installed as long as System Expansion Slots are available. A maximum of three 64KB memory options may be installed for a total of 256KB of memory.

The 32KB and 64KB Memory Expansion Options require a System Expansion Slot in the System Unit. The first 64KB on the System Board is required before 32KB and 64KB Memory Expansion Options can be installed.

The Asynchronous Communications Adapter provides a channel to data processing or input/output devices outside of your immediate system. These can be connected by telephone using a plug-in modem, or directly by cable when the device is nearby.

This option utilizes the RS232C asynchronous (start-stop) interface permitting attachment to a variety of devices including a large "host" computer or another IBM Personal Computer.

This option supports 50 to 9600 BPS transmission speeds. One 25 pin "D" shell, male type connector is provided to attach various peripheral devices. A "current loop" interface is located in the same connector, and a jumper block is provided to manually select either the voltage or the current loop interface.

The Asynchronous Communications Adapter requires a System Expansion Slot in the System Unit. An external line modem is required for telephone line transmission.

The Game Control Adapter permits the attachment of user-supplied joysticks or paddles. Two joysticks and up to four paddles may be attached. IBM does not manufacture either the joysticks or the paddles. This option provides connectors for joysticks or paddles and requires a System Expansion Slot in the System Unit.

A block diagram of the system is on the following page (1-4).
Figure 1. SYSTEM BLOCK DIAGRAM
SECTION 2. HARDWARE

Contents:

System Board ................................................. 2-3
Power Supply .................................................. 2-33
IBM Monochrome Display And Printer Adapter ........................................... 2-37
IBM Monochrome Display ........................................... 2-43
Color/Graphics Monitor Adapter ........................................... 2-45
Printer Adapter and Printer ................................. 2-65
5-1/4" Diskette Drive Adapter ........................................... 2-89
5-1/4" Diskette Drive ........................................... 2-110
Memory Expansion Options 32KB and 64KB ........................................... 2-113
Game Control Adapter ........................................... 2-117
Asynchronous Communications Adapter ........................................... 2-123
SYSTEM BOARD

The System Board fits horizontally in the base of the System Unit and has dimensions of approximately 8-1/2 inches by 11 inches. The System Board is a multilayer single land-per-channel design, with ground and power 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 the speaker. Five 62-pin card edge sockets are also mounted on the System Board. The system I/O channel is bussed across these five I/O slots.

There are 16 (13 used) Dual In-line Package (DIP) switches mounted on the card which can be read under program control. These switches are used to indicate to the system software what options are installed. They are used to indicate amounts of installed storage, both on the System Board and in the System Expansion slots, type of display adapter installed, and desired operation modes upon power-up; ie, color or black and white and 80- or 40 character lines. Switches are also used to identify when the operating system is to be loaded from diskette, and how many diskette drives are attached.

The major elements of the System Board are divided into five major functional areas. They are, 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 functions are described in detail in this section, except for the I/O channel, which has its own section. Figure 2.0 “System Board Data Flow” page 2-6, illustrates these functional areas.

The heart of the System Board is the Intel 8088 microprocessor. This processor is an 8-bit bus version of the 16-bit 8086 processor by Intel. It is software compatible with the 8086 and, thus, supports 16-bit operations including multiply and divide. The processor supports 20 bits of addressing (1 megabyte of storage). The processor is implemented in maximum mode so a co-processor can be added as a feature. The processor is operated at 4.77 Mhz. This frequency is derived from a 14.31818 Mhz crystal which is divided by three for the processor clock and by four 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 microsec (m sec).
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 are provided to 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 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 ns if the processor ready line is not deactivated. Refresh DMA cycles take four clocks or 840 ns.

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

Of the eight prioritized levels of interrupt, six are bussed to the I/O slots for use by feature cards. Two levels are used on the System Board. Level 0, the highest priority, is attached to Channel 1 of the timer counter and provides a periodic interrupt. 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 is designed to support both ROM and Read/Write Memory. The System Board contains space for 48K x 8 of ROM or EPROM. Six module sockets are provided, each capable of accepting an 8K x 8 device. Five of the sockets are populated with 40 KB 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 System Board also contains from 16K x 9 to 64K x 9 of Read/Write Memory. A minimum system would have 16 KB of memory with module sockets for an additional 48 KB. In a cassette version of the system, approximately 4 KB is used by the system leaving approximately 12 KB of user’s space for BASIC programs. Additional memory beyond the System Board’s maximum of 64 KB, is obtained by adding memory cards in the System Expansion slots.
The memory is dynamic 16K x 1 chips with an access time of 250 ns and a cycle time of 410 ns. All R/W memory is parity checked.

The System Board contains circuits for attaching an audio cassette, the serial keyboard, and the speaker. The cassette adapter allows the attachment of any good quality audio cassette via either the microphone or auxiliary inputs. The 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 since a different bit-cell time is used for 0’s and 1’s. For diagnostic purposes, the tape interface can loop read to write to test the board’s circuits. The system software blocks cassette data, generates and checks data with a Cyclic Redundancy Check (CRC).

The processor also contains the adapter circuits for attaching the serial interface from the keyboard. This generates an interrupt to the processor when a complete scan code is received. This interface can request execution of a diagnostic in the keyboard.

Both the keyboard and cassette interfaces are provided via 5-pin DIN connectors, which are right angle mounts on the System Board and extend through the rear panel of the System Unit.

The system is provided with a 2-1/4-inch audio speaker mounted inside the System Unit. The System Board contains the control circuits and driver for the speaker. The speaker connects through a 2-wire interface which attaches to a 4-pin header on the System Board.

The speaker drive circuit is capable of approximately a 1/2 watt of power. The control circuits allow the speaker to be driven several different ways. First, a direct program control register bit may be toggled to generate a pulse train; second, the output of Channel 2 of the timer counter may be programmed to generate a waveform to the speaker. Third, the clock input to the timer/counter can be modulated with a program controlled I/O Register Bit. All three forms of control may be performed simultaneously.
System Board Data Flow

Figure 2. SYSTEM BOARD DATA FLOW (SHEET 1 OF 2)
Figure 2. SYSTEM BOARD DATA FLOW (SHEET 2)
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 card +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 210 ns clocks or 1.05 m sec/byte. All DMA transfers require five clocks for a cycle time of 1.05 m sec/byte. Refresh cycles are present once every 72 clocks or approximately 15 m sec and require five 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 NMI to the 8088 processor. Memory Expansion Options use this line to report parity errors.

The I/O channel is repowered so there is sufficient drive to power all five System Expansion Slots, assuming two loads per slot. The IBM Option I/O adapters typically use only one load. A graphic illustration of the System I/O Channel and its descriptions are on the following pages.
NOTE:
A description of each signal is on the following pages.

Figure 3. I/O CHANNEL DIAGRAM
**System Board I/O Channel Description**

The following is a description of the IBM Personal Computer System Board I/O Channel. All signal 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: This signal is a high speed clock with a 70 nsec. period (14.31818 MHz.) It has a 50% duty cycle.</td>
</tr>
<tr>
<td>CLK</td>
<td>O</td>
<td>Clock: This is the system clock. It is a divide-by-three of the oscillator and has a period of 210 nsec. (4.77 MHz.) The clock has a 33% duty cycle.</td>
</tr>
<tr>
<td>RESET DRV</td>
<td>O</td>
<td>Reset Driver: 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) while A19 is the Most Significant Bit (MSB). These lines are generated by either the processor or the DMA Controller. They are active HIGH.</td>
</tr>
<tr>
<td>D0-D7</td>
<td>I/O</td>
<td>Data Bits 0 to 7: These lines provide data bus bits 0 to 7 for the processor, memory, and I/O Devices. D0 is the Least Significant Bit (LSB) and D7 is the Most Significant Bit (MSB). These lines are active HIGH.</td>
</tr>
<tr>
<td>ALE</td>
<td>O</td>
<td>Address Latch Enable: This 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 in conjunction with AEN). Processor addresses are latched with the falling edge of ALE.</td>
</tr>
<tr>
<td>I/O CHK</td>
<td>I</td>
<td>I/O Channel Check: This line provides the CPU 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>
</tbody>
</table>
### I/O CH RDY

**I/O Channel Ready:** This line (normally high or "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 for any period in excess of 10 clock cycles (2.1 usec.) Machine cycles (I/O or memory) are extended by an integral number of CLK cycles (210 ns).

### IRQ2-IRQ7

**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).

### IOR

**-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.

### IOW

**-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.

### MEMR

**-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.

### MEMW

**-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.
DRQ1-DRQ3  I  DMA Request 1 to 3: These lines are asynchronous channel requests used by peripheral devices to gain DMA service. They are prioritized with DRQ1 having highest priority and DRQ3 the lowest. 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.

DACK0-DACK3  O  -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.

AEN  O  Address Enable: This line is used to degate the processor and other devices from the I/O Channel to allow Direct Memory Access (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).

T/C  O  Terminal Count: This line provides a pulse when the terminal count for any DMA channel is reached. This signal is active HIGH.

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.
Figure 4. SYSTEM BOARD COMPONENT DIAGRAM
Keyboard

The Keyboard is a device separate from the System Unit. It is attached via a serial interface cable approximately 6 feet in length which plugs into the rear of the System Unit. The attaching cable is coiled, like that of a telephone headset, and is a shielded four-wire wire connection. The interface contains power (+5 Vdc), ground and two bidirectional signal lines. The cable is permanently attached at the keyboard end and plugs into the System Unit via a DIN connector.

The keyboard uses a capacitive technology with a microcomputer (Intel 8048) performing the keyboard scan function. The keyboard is packaged in a low-profile enclosure with a tilt adjustment for 5 degree or 15 degree orientations.

The keyboard contains 83 keys laid out in three major groupings. The central portion of the keyboard contains a standard typewriter keyboard layout. On the left side, arranged as a 2x5 block, are 10 function keys. These keys are user-defined by software. On the right is a 16-key, key pad area. This area is, defined by the software but contains legends for the functions of numeric entry, cursor control calculator pad screen edit.

The keyboard interface is defined so system software has the maximum flexibility in defining keyboard operations such as shift states of keys, make/break keys, and typematic operation. This is accomplished by having the keyboard return scan codes rather than American National Standard Control Characters (ASCII) codes. In addition, all keys except control keys are typematic and generates both a make and a break-scan code. For example, key 1 produces scan code 01 on make, and code 81 on break. Break codes are formed by adding X ‘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 and when requested by the System Unit. This diagnostic CRC checks the microcomputer ROM, tests memory and checks for stuck keys. Additional functions are: keyboard scanning, key debounce, buffering of up to 20 key scan codes, maintaining bidirectional serial communications with the System Unit, and executing the hand shake protocol required by each scan code transfer. A keyboard diagram and table of scan codes are on the following pages. Figure (5) is a block diagram of the keyboard interface on the System Board.
Figure 5. KEYBOARD INTERFACE BLOCK DIAGRAM
NOTE
1 NOMENCLATURE IS ON BOTH TOP AND FRONT FACE OF KEY BUTTON AS SHOWN.
THE NUMBER TO THE UPPER LEFT DESIGNATES THE BUTTON POSITION.

Figure 6. KEYBOARD DIAGRAM
Table 1. Keyboard Scan Codes

<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 Interface Connector Specifications

<table>
<thead>
<tr>
<th>PIN</th>
<th>SIGNAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>+ Keyboard Clock</td>
</tr>
<tr>
<td>2</td>
<td>+ Keyboard Data</td>
</tr>
<tr>
<td>3</td>
<td>– Keyboard Reset (Not used by keyboard)</td>
</tr>
<tr>
<td>4</td>
<td>Ground</td>
</tr>
<tr>
<td>5</td>
<td>+5 Volts</td>
</tr>
</tbody>
</table>
Cassette User Interface

The cassette interface control is implemented in software. (See Firmware Section). An 8253 timer output is used to control the data to the cassette recorder. This output exits the System Board, at the rear, through pin 5 of a DIN connector. The cassette input data is read by an 8255A-5 Programmable Peripheral Interface (PPI) input port bit. This signal 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 & 3 of the cassette connector. The motor on/off is controlled by an 8255A-PPI output port bit (Port ‘61H’, bit 3). The 8255A address and bit assignments are defined in the I/O Address Map page. On the following pages are read, write, and motor control block diagrams.

Cassette Jumpers

A 2x2 Berg Pin and Jumper are used on the cassette Data Out line. The jumper will allow the Data Out line to be used as a microphone input (75 mv.) when the jumper is placed across M and C pins. An auxiliary input is available when the jumper is placed across the A and C pins. The auxiliary input provides a .68 volt input to the recorder. Refer to System Board Component Diagram page (2-13) for cassette jumper location.

Circuit Block Diagrams

![Circuit Block Diagram](image)

Figure 7. CASSETTE INTERFACE READ HARDWARE
Figure 8. CASSETTE INTERFACE WRITE HARDWARE

Figure 9. CASSETTE MOTOR CONTROL
Cassette Interface Connector Specifications

### Pin Signals and Electrical Characteristics

<table>
<thead>
<tr>
<th>PIN</th>
<th>SIGNAL</th>
<th>ELECTRICAL CHARACTERISTICS*</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Motor Control</td>
<td>Common from Relay</td>
</tr>
<tr>
<td>2</td>
<td>Ground</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Motor Control</td>
<td>6 VDC; 1A (Relay N.O.)</td>
</tr>
<tr>
<td>4</td>
<td>Data In</td>
<td>500nA at ±13V - at 1,000 - 2,000 Baud</td>
</tr>
</tbody>
</table>
| 5   | Data Out (Mic or Aux)   | 250 μA at \(
|     |                         | \begin{align*} \text{.68V} \\
|     |                         | \text{or} \\
|     |                         | \text{75mV} \end{align*} |

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

**Data out can be chosen using a jumper located on planar.
(AUX → .68V or MIC → 75 mV).

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

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

1. An 8255A-5 PPI output bit. The address and bit are defined in the I/O Address Map pages 2-23 and 2-24.
2. A timer Channel Clock out where the output is programmable within the functions of the 8253-5 timer with a 1.19 Mhz clock input. The timer gate is also controlled by an 8255A-5 PPI output port bit. Address and bit assignment are in the I/O Address Map pages 2-23 and 2-24.

![Figure 10. SPEAKER DRIVE SYSTEM BLOCK DIAGRAM](image)

Channel 2 (Tone generation for Speaker)
- Gate 2 — Controlled by 8255A-5 PPI Bit (See I/O Map)
- CLK IN 2 — 1.19318 MHz OSC
- CLK OUT 2 — Used to drive Speaker
- Used to write data on the Audio Cassette

Speaker Connection - 4 Pin Berg Connector, Refer to System Board Diagram page 2-13 for speaker connection.

<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>
# I/O Address Map

<table>
<thead>
<tr>
<th>HEX RANGE</th>
<th>9 8 7 6 5 4</th>
<th>3 2 1 0</th>
<th>DEVICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>00–0F</td>
<td>0 0 0 0 0 Z</td>
<td>A3 A2 A1 A0</td>
<td>DMA CHIP 8237–2</td>
</tr>
<tr>
<td>20–21</td>
<td>0 0 0 0 1 Z</td>
<td>Z Z Z A0</td>
<td>INTERRUPT 8259A</td>
</tr>
<tr>
<td>40–43</td>
<td>0 0 0 1 0 Z</td>
<td>Z Z A1 A0</td>
<td>TIMER 8253–5</td>
</tr>
<tr>
<td>60–63</td>
<td>0 0 0 1 1 Z</td>
<td>Z Z A1 A0</td>
<td>PPI 8255A–5</td>
</tr>
<tr>
<td>80–83</td>
<td>0 0 1 0 0 Z</td>
<td>Z Z A1 A0</td>
<td>DMA PAGE REGS</td>
</tr>
<tr>
<td>* AX</td>
<td>0 0 1 0 1</td>
<td>NMI MASK REG</td>
<td></td>
</tr>
<tr>
<td>CX</td>
<td>0 0 1 1 0</td>
<td>RESERVED</td>
<td></td>
</tr>
<tr>
<td>EX</td>
<td>0 0 1 1 1</td>
<td>RESERVED</td>
<td></td>
</tr>
<tr>
<td>3F8–3FF</td>
<td>1 1 1 1 1 1</td>
<td>A2 A1 A0</td>
<td>TP RS–232–C CD</td>
</tr>
<tr>
<td>3FO–3F7</td>
<td>1 1 1 1 1 0</td>
<td>A2 A1 A0</td>
<td>5 1/4” DRV ADAPTER</td>
</tr>
<tr>
<td>2F8–2FF</td>
<td>1 0 1 1 1 1</td>
<td>A2 A1 A0</td>
<td>RESERVED</td>
</tr>
<tr>
<td>378–37F</td>
<td>1 1 0 1 1 1</td>
<td>Z A1 A0</td>
<td>PARALLEL PRTR PRT</td>
</tr>
<tr>
<td>3D0–3DF</td>
<td>1 1 1 0 1 0</td>
<td>A3 A2 A1 A0</td>
<td>COLOR/GRAPHICS ADAPTER</td>
</tr>
<tr>
<td>278–27F</td>
<td>1 0 0 1 1 1</td>
<td>Z A1 A0</td>
<td>RESERVED</td>
</tr>
<tr>
<td>200–20F</td>
<td>1 0 0 0 0 0</td>
<td>A3 A2 A1 A0</td>
<td>GAME I/O ADAPTER</td>
</tr>
<tr>
<td>3B0–3BF</td>
<td>1 1 0 1 1 1</td>
<td>A3 A2 A1 A0</td>
<td>IBM MONOCHROME DISPLAY PARALLEL PRINTER ADAPTER</td>
</tr>
</tbody>
</table>

Z = Don’t Care, i.e., Not in Decode

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

Set mask: write X’80’ to I/O Address X’A0’ (enable NMI)

Clear mask: write X’00’ to I/O Address X’A0’ (disable NMI)
# I/O Address Map

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA0</td>
<td>*KBD SCAN CODE</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>N</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>P</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>U</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>T</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PB0</td>
<td>*SPEAKER DATA</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>U</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>T</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>P</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>U</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>T</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>(ENABLE KBD) &amp; + (CLR KBD &amp; ENABLE SENSE SW'S)</td>
<td>7</td>
<td>7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC0</td>
<td>I/O READ/WRITE MEMORY (SW2-1)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>N</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>P</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>U</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>T</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>+I/O CHANNEL CHECK</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>+READ/WRITE MEMORY PCK</td>
<td>7</td>
<td>7</td>
</tr>
</tbody>
</table>

## Mode Reg/Mode Register

<table>
<thead>
<tr>
<th>Mode Reg Value</th>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001 0110101</td>
<td>10</td>
<td>1010</td>
</tr>
</tbody>
</table>

## Notes

- **PA bit=0** implies switch "ON".
- **PA bit=1** implies switch "OFF".

NOTE: PA bit=0 implies switch "ON". 
PA bit=1 implies switch "OFF".

2-24
System Memory Map

Figure 11. SYSTEM MEMORY MAP
## System Memory Map (Increments of 16KB)

<table>
<thead>
<tr>
<th>START ADDRESS: DECIMAL</th>
<th>HEX</th>
<th>FUNCTION:</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00000</td>
<td>16–64 KB READ/WRITE MEMORY ON SYSTEM BOARD</td>
</tr>
<tr>
<td>16K</td>
<td>04000</td>
<td></td>
</tr>
<tr>
<td>32K</td>
<td>08000</td>
<td></td>
</tr>
<tr>
<td>48K</td>
<td>0C000</td>
<td></td>
</tr>
<tr>
<td>64K</td>
<td>10000</td>
<td></td>
</tr>
<tr>
<td>80K</td>
<td>14000</td>
<td></td>
</tr>
<tr>
<td>96K</td>
<td>18000</td>
<td></td>
</tr>
<tr>
<td>112K</td>
<td>1C000</td>
<td></td>
</tr>
<tr>
<td>128K</td>
<td>20000</td>
<td>UP TO 192 KB MEMORY IN I/O CHANNEL</td>
</tr>
<tr>
<td>144K</td>
<td>24000</td>
<td></td>
</tr>
<tr>
<td>160K</td>
<td>28000</td>
<td></td>
</tr>
<tr>
<td>178K</td>
<td>2C000</td>
<td></td>
</tr>
<tr>
<td>192K</td>
<td>30000</td>
<td></td>
</tr>
<tr>
<td>208K</td>
<td>34000</td>
<td></td>
</tr>
<tr>
<td>224K</td>
<td>38000</td>
<td></td>
</tr>
<tr>
<td>240K</td>
<td>3C000</td>
<td></td>
</tr>
<tr>
<td>256K</td>
<td>40000</td>
<td></td>
</tr>
<tr>
<td>272K</td>
<td>44000</td>
<td></td>
</tr>
<tr>
<td>288K</td>
<td>48000</td>
<td></td>
</tr>
<tr>
<td>304K</td>
<td>4C000</td>
<td></td>
</tr>
<tr>
<td>320K</td>
<td>50000</td>
<td></td>
</tr>
<tr>
<td>336K</td>
<td>54000</td>
<td></td>
</tr>
<tr>
<td>352K</td>
<td>58000</td>
<td></td>
</tr>
<tr>
<td>368K</td>
<td>5C000</td>
<td></td>
</tr>
<tr>
<td>384K</td>
<td>60000</td>
<td>384 KB FUTURE R/W MEMORY EXPANSION IN I/O CHANNEL</td>
</tr>
<tr>
<td>400K</td>
<td>64000</td>
<td></td>
</tr>
<tr>
<td>416K</td>
<td>68000</td>
<td></td>
</tr>
<tr>
<td>432K</td>
<td>6C000</td>
<td></td>
</tr>
<tr>
<td>448K</td>
<td>70000</td>
<td></td>
</tr>
<tr>
<td>464K</td>
<td>74000</td>
<td></td>
</tr>
<tr>
<td>480K</td>
<td>78000</td>
<td></td>
</tr>
<tr>
<td>496K</td>
<td>7C000</td>
<td></td>
</tr>
<tr>
<td>512K</td>
<td>80000</td>
<td></td>
</tr>
<tr>
<td>528K</td>
<td>84000</td>
<td></td>
</tr>
<tr>
<td>544K</td>
<td>88000</td>
<td></td>
</tr>
<tr>
<td>560K</td>
<td>8C000</td>
<td></td>
</tr>
<tr>
<td>576K</td>
<td>90000</td>
<td></td>
</tr>
<tr>
<td>592K</td>
<td>94000</td>
<td></td>
</tr>
<tr>
<td>608K</td>
<td>98000</td>
<td></td>
</tr>
<tr>
<td>624K</td>
<td>9C000</td>
<td></td>
</tr>
</tbody>
</table>

Figure 12. SYSTEM MEMORY MAP (INCREMENTS OF 16KB) (SHEET 1 OF 2)
## System Memory Map Cont.

<table>
<thead>
<tr>
<th>START ADDRESS: DECIMAL</th>
<th>HEX</th>
<th>FUNCTION:</th>
</tr>
</thead>
<tbody>
<tr>
<td>640K</td>
<td>A0000</td>
<td>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>VIDEO BUFFER</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></td>
</tr>
<tr>
<td>816K</td>
<td>CC000</td>
<td></td>
</tr>
<tr>
<td>832K</td>
<td>D0000</td>
<td>192 KB MEMORY EXPANSION AREA</td>
</tr>
<tr>
<td>848K</td>
<td>D4000</td>
<td></td>
</tr>
<tr>
<td>864K</td>
<td>D8000</td>
<td></td>
</tr>
<tr>
<td>880K</td>
<td>DC000</td>
<td></td>
</tr>
<tr>
<td>896K</td>
<td>E0000</td>
<td></td>
</tr>
<tr>
<td>912K</td>
<td>E4000</td>
<td></td>
</tr>
<tr>
<td>928K</td>
<td>E8000</td>
<td></td>
</tr>
<tr>
<td>944K</td>
<td>EC000</td>
<td></td>
</tr>
<tr>
<td>960K</td>
<td>F0000</td>
<td>RESERVED</td>
</tr>
<tr>
<td>976K</td>
<td>F4000</td>
<td>48 KB BASE SYSTEM ROM</td>
</tr>
<tr>
<td>992K</td>
<td>F8000</td>
<td></td>
</tr>
<tr>
<td>1.008M</td>
<td>FC000</td>
<td></td>
</tr>
</tbody>
</table>

Figure 12. SYSTEM MEMORY MAP (16KB) (SHEET 2)
System Board and Memory
Expansion Switch Settings

On the following four pages are graphic illustrations of switch settings. These are necessary for the system to address components attached, and to specify the amount of memory installed both on the System Board and in the System Expansion Slots. Refer to the System Board Component Diagram (page 13) for DIP switch locations.

### SWITCH 1

<table>
<thead>
<tr>
<th>POSITION</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-7-8</td>
<td>NUMBER OF 5¼&quot; DISKETTE DRIVES INSTALLED; PAGE 2-29</td>
</tr>
<tr>
<td>2</td>
<td>UNUSED-MUST BE ON (RESERVED FOR CO-PROCESSOR)</td>
</tr>
<tr>
<td>3-4</td>
<td>AMOUNT OF MEMORY ON SYSTEM BOARD; PAGE 2-30</td>
</tr>
<tr>
<td>5-6</td>
<td>TYPE OF MONITOR YOU ARE USING; PAGE 2-29</td>
</tr>
</tbody>
</table>

### SWITCH 2

<table>
<thead>
<tr>
<th>POSITION</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-2-3-4</td>
<td>AMOUNT OF MEMORY OPTIONS INSTALLED; PAGE 2-30</td>
</tr>
<tr>
<td>5-6-7-8</td>
<td>ALWAYS IN THE OFF POSITION</td>
</tr>
</tbody>
</table>

2-28
5-1/4" Diskette Drives Switch Settings

Monitor Type Switch Settings

NOTE: SOME TELEVISIONS AND MONITORS OPERATED IN (80 x 25) MODE MAY HAVE CHARACTER LOSS.
System Board Memory Switch Settings

<table>
<thead>
<tr>
<th>Memory Option</th>
<th>Switch 1</th>
<th>Switch 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>16KB</td>
<td><img src="image1" alt="16KB Switch 1" /></td>
<td><img src="image2" alt="16KB Switch 2" /></td>
</tr>
<tr>
<td>32KB</td>
<td><img src="image3" alt="32KB Switch 1" /></td>
<td><img src="image4" alt="32KB Switch 2" /></td>
</tr>
<tr>
<td>48KB</td>
<td><img src="image5" alt="48KB Switch 1" /></td>
<td><img src="image6" alt="48KB Switch 2" /></td>
</tr>
<tr>
<td>64KB</td>
<td><img src="image7" alt="64KB Switch 1" /></td>
<td><img src="image8" alt="64KB Switch 2" /></td>
</tr>
<tr>
<td>96KB</td>
<td><img src="image9" alt="96KB Switch 1" /></td>
<td><img src="image10" alt="96KB Switch 2" /></td>
</tr>
<tr>
<td>128KB</td>
<td><img src="image11" alt="128KB Switch 1" /></td>
<td><img src="image12" alt="128KB Switch 2" /></td>
</tr>
<tr>
<td>160KB</td>
<td><img src="image13" alt="160KB Switch 1" /></td>
<td><img src="image14" alt="160KB Switch 2" /></td>
</tr>
<tr>
<td>192KB</td>
<td><img src="image15" alt="192KB Switch 1" /></td>
<td><img src="image16" alt="192KB Switch 2" /></td>
</tr>
<tr>
<td>224KB</td>
<td><img src="image17" alt="224KB Switch 1" /></td>
<td><img src="image18" alt="224KB Switch 2" /></td>
</tr>
<tr>
<td>256KB</td>
<td><img src="image19" alt="256KB Switch 1" /></td>
<td><img src="image20" alt="256KB Switch 2" /></td>
</tr>
</tbody>
</table>
32/64KB Memory Expansion Option
Switch Settings

Note: Positions 6-7-8 must always be ON. The sequence shown below must be followed to allow the system to address the memory properly.
Power Supply

The system DC power supply is a 63.5 watt, 4 voltage level switching regulator. It is integrated into the System Unit and supplies power for the System Unit, its options, and the keyboard. The supply provides 7 amps of +5 Vdc, ±5% 2 amps of +12 Vdc, ±5% 300 ma of -5 Vdc, ±10% and 250 ma of -12 Vdc, ±10%. All power levels are regulated with overvoltage and over current protection. The input is 120 Vac and fused. DC over-load or over-voltage conditions exist, the supply will automatically shut down until the condition is corrected. The supply is designed for continuous operation at 63.5 watts.

The System Board takes approximately 3 amps of +5 Vdc thus allowing approximately 4 amps of 5 Vdc for the adapters in the System Expansion Slots. The +12 Vdc power level is designed to power the two internal 5-1/4" Diskette Drives and the system’s dynamic memory. It is assumed that only one drive motor is active at a time. The -5 Vdc level is used for memory bias voltage and analog circuits in the diskette adapter phase lock loop. The +12 Vdc and -12 Vdc are used for powering the serial interface card EIA drivers and receivers for the Asynchronous Communications Adapter. All four power levels are bussed across the five System Expansion Slots and available for option adapter.

The IBM Monochrome Display is self-powered. However, the high resolution display receives its AC power from the System Unit power system. It is switched on and off with the power switch, which saves a wall outlet. The AC output for the display is a nonstandard connector, so only the AC high resolution Display can use this AC port.
Power Supply Location

The Power Supply is located at the right rear area of the System Unit. It supplies operating voltages to the System Board, IBM Monochrome Display, and provides two separate connections for power to the 5-1/4" Diskette Drives (if installed). The nominal power requirements and output voltages are listed on the following tables:

Input Requirements

Voltage

<table>
<thead>
<tr>
<th>VOLTAGE @ 60 Hz</th>
<th>NOMINAL</th>
<th>MINIMUM</th>
<th>MAXIMUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vac</td>
<td>120</td>
<td>104</td>
<td>127</td>
</tr>
</tbody>
</table>

Frequency

60 Hz +/- .5 Hz

Current

2.5 AMPS MAX @ LOW LINE INPUT
VOLTAGE OF 120 VAC 60 HZ

DC Output

<table>
<thead>
<tr>
<th>VOLTAGE</th>
<th>CURRENT</th>
<th>REGULATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vac</td>
<td>AMPS</td>
<td>TOLERANCE</td>
</tr>
<tr>
<td>NOMINAL</td>
<td>MIN</td>
<td>MAX</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>0.0</td>
<td>2.0</td>
</tr>
<tr>
<td>-12.0</td>
<td>0.0</td>
<td>0.25</td>
</tr>
</tbody>
</table>

AC Output

<table>
<thead>
<tr>
<th>VOLTAGE</th>
<th>CURRENT</th>
<th>VOLTAGE LIMITS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vac</td>
<td>AMPS</td>
<td>MIN</td>
</tr>
<tr>
<td>NOMINAL</td>
<td>MIN</td>
<td>MAX</td>
</tr>
<tr>
<td>120.0</td>
<td>0.0</td>
<td>.75</td>
</tr>
</tbody>
</table>
PIN 1, +12 VDC
PIN 2, +12 VDC RTN
PIN 3, +5 VDC RTN
PIN 4, +5 VDC
PIN 5, +5 VDC
PIN 6, +5 VDC
PIN 1, GND
PIN 2, GND
PIN 3, 5 VDC
PIN 4, 5 VDC
PIN 5, 5 VDC
PIN 6, 5 VDC
PIN 1, PWR GOOD
PIN 2, KEY
PIN 3, +12 VDC
PIN 4, +12 VDC
PIN 5, +12 VDC RTN
PIN 6, +5 VDC RTN
PIN 7, +5 VDC
PIN 8, +5 VDC
PIN 9, +5 VDC
PIN 10, +5 VDC
PIN 11, +5 VDC
PIN 12, +5 VDC
PIN 13, +5 VDC
PIN 14, +5 VDC
PIN 15, +5 VDC
PIN 16, +5 VDC

Figure 13. POWER SUPPLY AND CONNECTORS

The power connector on the System Board is a 12 pin male connector which plugs into the power supply connectors. The pin configurations and location are shown below.
Important Operating Characteristics

Over Voltage/Current Protection

<table>
<thead>
<tr>
<th>VOLTAGE NOMINAL VAC</th>
<th>TYPE PROTECTION</th>
<th>RATING AMPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>120</td>
<td>60 Hz FUSE TYPE 2 SOC SD4</td>
<td>2 AMPS</td>
</tr>
</tbody>
</table>

Power On/Off Cycle: When the supply is turned off for a maximum of 5 seconds, and then turned on, the power good signal will be regenerated.

Signal Requirements

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

The Power Supply. Provides a power good signal out, to indicate the presence of the +/-5V and +/-12V outputs are above the sense level defined in the chart below, the power good signal is an up level (2.4V to 5.5V), TTL compatible and capable of sourcing 60 UA. When any of the four sensed output voltages is below its sense level voltage as defined in the chart below, the power good signal is down level (0V to 0.4V), TTL compatible and capable of sinking 500 UA. The power good signal (after all levels of the output voltage are good) has a turn on delay of 100 MS, but no greater than 500 MS.

The sense levels of the +/-5V and +/-12V outputs are:

<table>
<thead>
<tr>
<th>OUTPUT</th>
<th>MIN</th>
<th>SENSE VOLTAGE NOMINAL</th>
<th>MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>+5V</td>
<td>+3.7</td>
<td>+4.0</td>
<td>+4.3</td>
</tr>
<tr>
<td>-5V</td>
<td>-3.7</td>
<td>-4.0</td>
<td>-4.3</td>
</tr>
<tr>
<td>+12V</td>
<td>+8.5</td>
<td>+9.6</td>
<td>+10.5</td>
</tr>
<tr>
<td>-12V</td>
<td>-8.5</td>
<td>-9.6</td>
<td>-10.5</td>
</tr>
</tbody>
</table>
IBM Monochrome Display and Parallel Printer Adapter

This adapter has dual functions. The first is to provide the interface to the IBM Monochrome Display. The second function is a parallel interface for the IBM 80 CPS Matrix Printer.

The monitor interface is designed around the Motorola 6845 CRT Controller module. There are 4K bytes of static memory on the card which are used for the display buffer. The memory is dual ported and may be accessed directly by the CPU. No parity is provided on the display buffer. A block diagram of the Monochrome Display function in on page 2-38.

The characteristics of the design are listed below:

- 80x25 Screen
- Direct Drive Output
- 9x14 Character Box
- 7x9 Character
- 18 KHz Monitor
- Character Attributes

The adapter supports 256 character codes. An 8K byte character generator contains the fonts for the character codes. The characters, values, keystrokes and screen characteristics are tabled in Appendix C. Of Characters, Keystrokes and Color.

Note: This Adapter when used with a display containing P39 Phosphor, will not support a light pen!

Parallel Interface Description

This topic is discussed in full on pages 2-65 through page 2-69.
Figure 14. IBM MONOCHROME DISPLAY ADAPTER BLOCK DIAGRAM
System Channel Interface

Lines Used
This card uses the address and data bus, memory and I/O read/write signals, reset, I/O Ready, I/O Clock, and IRQ7.

 Loads
Where possible, only one “LS” load is on the signals present at the I/O slot. Some of the address bus lines have two “LS” loads. No signal has more than two “LS” loads.

Special Timing
At least one wait state will be inserted on all memory and I/O accesses from the CPU. The duration of the wait-state will vary because the CPU/monitor access is synchronized with the character clock on this adapter.

To insure proper initialization of the attachment, the first instruction issued to the card must be to set the high resolution bit of the monitor output Port 1. (OUT PORT 3B8 = 01H). A CPU access to this adapter must never occur if the high resolution bit is not set.

System configurations which have two display adapter cards must insure that both adapters are properly initialized after a power on reset. Damage to either display may occur if not properly initialized.

Data Rates
For the IBM Monochrome Display Adapter, two bytes are fetched from the display buffer in 553 ns providing a data rate of 1.8M bytes/second.

Interrupt and DMA Response Requirements
• The display buffer can be written into, or read from using DMA.
• The parallel interface uses the +IRQ7 line. Interrupt becomes active when the acknowledge input is low, and interrupts are enabled via the control port.
Modes of Operation

The IBM Monochrome Display and Printer Adapter supports 256 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</th>
<th>FOREGROUND</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>R G B</td>
<td>R G B</td>
<td>NON DISPLAY</td>
</tr>
<tr>
<td>0 0 0</td>
<td>0 0 0</td>
<td>UNDERLINE</td>
</tr>
<tr>
<td>0 0 0</td>
<td>0 0 1</td>
<td>WHITE CHARACTER/</td>
</tr>
<tr>
<td>0 0 0</td>
<td>1 1 1</td>
<td>BLACK BACKGROUND</td>
</tr>
<tr>
<td>1 1 1</td>
<td>0 0 0</td>
<td>REVERSE VIDEO</td>
</tr>
</tbody>
</table>
Programming Considerations

Programming the 6845 CRT Controller

The following table summarizes the 6845 Internal Data Registers and their functions and parameters. For the IBM Monochrome Display, the values in the table must be programmed into the 6845 to insure proper initialization of the device.

### Table 2. 6845 INITIALIZATION PARAMETERS

<table>
<thead>
<tr>
<th>REGISTER #</th>
<th>REGISTER FILE</th>
<th>PROGRAM UNIT</th>
<th>80x25 MONOCHROME</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0</td>
<td>HORIZONTAL TOTAL</td>
<td>CHARACTERS</td>
<td>61H</td>
</tr>
<tr>
<td>R1</td>
<td>HORIZONTAL DISPLAYED</td>
<td>CHARACTERS</td>
<td>50H</td>
</tr>
<tr>
<td>R2</td>
<td>HSYNC POSITION</td>
<td>CHARACTERS</td>
<td>52H</td>
</tr>
<tr>
<td>R3</td>
<td>HSYNC WIDTH</td>
<td>CHARACTERS</td>
<td>FH</td>
</tr>
<tr>
<td>R4</td>
<td>VERTICAL TOTAL</td>
<td>CHAR ROWS</td>
<td>19H</td>
</tr>
<tr>
<td>R5</td>
<td>VTOTAL ADJUST</td>
<td>SCAN LINE</td>
<td>6H</td>
</tr>
<tr>
<td>R6</td>
<td>VERTICAL DISPLAYED</td>
<td>CHAR ROW</td>
<td>19H</td>
</tr>
<tr>
<td>R7</td>
<td>VSYNC POSITION</td>
<td>CHAR ROW</td>
<td>19H</td>
</tr>
<tr>
<td>R8</td>
<td>INTERLACE MODE</td>
<td>---</td>
<td>02</td>
</tr>
<tr>
<td>R9</td>
<td>MAX SCAN LINE ADDRESS</td>
<td>SCAN LINE</td>
<td>DH</td>
</tr>
<tr>
<td>R10</td>
<td>CURSOR START</td>
<td>SCAN LINE</td>
<td>BH</td>
</tr>
<tr>
<td>R11</td>
<td>CURSOR END</td>
<td>SCAN LINE</td>
<td>CH</td>
</tr>
<tr>
<td>R12</td>
<td>START ADDRESS (H)</td>
<td>---</td>
<td>00H</td>
</tr>
<tr>
<td>R13</td>
<td>START ADDRESS (L)</td>
<td>---</td>
<td>00H</td>
</tr>
<tr>
<td>R14</td>
<td>CURSOR (H)</td>
<td>---</td>
<td>00H</td>
</tr>
<tr>
<td>R15</td>
<td>CURSOR (L)</td>
<td>---</td>
<td>00H</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>

Sequence of Events

The first command issued to this attachment must be to output to PORT 3B8, hex 01, to set high resolution mode. If the high resolution mode is not set, an infinite CPU wait-state will occur!

Memory Requirements

The attachment has 4K bytes of memory which is used for the display buffer. The memory supports one screen of 25 rows of 80 characters, plus a character attribute for each display character. No parity is provided on the memory. No system Read/Write memory is required for the monochrome adapter portion. The display buffer starts at address ‘B0000’.
DMA Channels

The display buffer will support a DMA operation, however CPU wait-states will be inserted during DMA.

Interrupt Levels

Interrupt Level 7 is used on the parallel interface. Interrupts can be enabled or disabled via the Printer Control Port. The interrupt is a high level active signal.

I/O Address and Bit Map

The table below breaks down the functions of the I/O Address decode for the card. The I/O address decode is from ‘3B0’ through ‘3BF’. The bit assignment for each I/O address follows:

<table>
<thead>
<tr>
<th>I/O 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 to the high resolution Monochrome Display.

- CRT Output Port 1 (I/O Address ‘3B8’)

<table>
<thead>
<tr>
<th>Bit #</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>
- CRT Status Port (I/O Address '3BA')

<table>
<thead>
<tr>
<th>Bit</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>+B/W Video</td>
</tr>
</tbody>
</table>

**IBM Monochrome Display**

The high resolution IBM Monochrome Display unit attaches to the System Unit via two cables of approximately 3’ (914 mm) in length. One cable is a signal cable which contains 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 ON/OFF switch to also control the display unit. An additional benefit is a reduction in the requirements for wall outlets to power the system. The monitor contains an 12” (305 mm) 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 table top or desk. The unit has both brightness and contrast adjustment controls on the front available to the operator.

**Operating Characteristics**

**Screen**

High persistance green phosphor (P 39) with an etched surface to reduce glare. Unit displays an 80 character by 25 line screen with a 9 dot wide by 14 dot tall character box.

**Video Signal**

Maximum video bandwidth of 16.27 Mhz.

**Vertical Drive**

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

**Horizontal Drive**

Positive level TTL compatible frequency, 18.432 Khz.
At Standard TTL Levels

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

IBM Monochrome Display and Parallel Printer Adapter

NOTE: Signal voltages are 0 - 0.6 Vdc at down level +5 Vdc at high level
The Color/Graphics Monitor Adapter is designed to attach a wide variety of TV frequency monitors and TV sets (user-supplied RF modulator required for TVs). It is capable of operating in black and white or color, and provides three video interfaces; a composite video port, a direct drive port, and 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 A/N and APA modes. In A/N mode, the display can be operated in a 40x25 mode for low resolution monitors and TVs or 80x25 mode for high resolution monitors. In both modes, characters are defined in an 8x8 box and are 5x7 with one line of descender for lowercase (both uppercase and lowercase characters are supported in all modes). In black and white mode, the character attributes of Reverse Video, Blinking and Highlighting are available. In color mode, there are 16 foreground colors and 8 background colors available per character. In addition, blinking on a per character basis is available.

The adapter card contains 16KB of storage; thus, for a 40x25 screen, 1000 bytes are used to store character information and 1000 bytes are used for attribute/color information. This means that up to 8 pages of screens can be stored in the adapter memory. Similarly, in an 80x25 mode, 4 pages of display screen may be stored in the adapter. The full 16KB storage on the display adapter is directly addressable by the processor allowing maximum software flexibility in managing the screen. In A/N color modes, it is also possible to select the screen border color. One of 16 colors may be selected.
In APA mode, there are two resolutions available; 320x200 and 640x200. In the 320x200, 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 3 colors come from one of the two software selectable palettes. One palette contains red/green/brown, the other contains cyan/magenta/white.

The 640x200 mode is only available in black and white since the full 16KB of storage is used to define the on or off state of the pel.

The adapter operates in noninterlace mode at either 7 or 14 megahertz (Mhz) video bandwidth depending on the mode of operation selected.

In A/N mode, characters are formed from a ROM character generator. The character generator contains dot patterns for 256 characters. The character set contains the following major grouping of characters. Sixteen special characters for game support, 15 characters for support of word processing editing functions, the standard 96 ASCII graphic set, 48 characters to support foreign languages, 48 characters for business block graphics allowing drawing of charts, boxes and tables using single and double lines, 16 of the most often used Greek characters, and 15 of the most often used scientific notation characters.

The Color/Graphics Monitor Adapter function is packaged on a single card which fits into one of the five System Expansions Slots on the System Board. The direct drive and composite video ports are right-angle mounted connectors at the rear of the adapter and extend through the rear panel of the System Unit.

The display adapter is implemented using a Motorola 6845 CRT controller device. This adapter is highly programmable with respect to raster and character parameters. Thus, many additional modes are possible with clever programming of the adapter. A block diagram of the Color/Graphics Adapter is on the following page.
Color/Graphics Monitor Adapter Block Diagram

Figure 15. COLOR/GRAPHICS MONITOR ADAPTER BLOCK DIAGRAM
Major Components Definitions

Motorola 6845 CRT Controller
This device provides the necessary interface to drive a raster scan CRT.

Mode Set And Status Registers
This is a general purpose programmable I/O register. It has I/O points which may be individually programmed. Its function in this attachment is to provide mode selection (page 2-49 and 2-50) and color selection in the medium resolution color graphics mode (page 2-51.)

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

Character Generator
This attachment utilizes a ROM character generator. It consists of 8K bytes of storage which cannot be read/written under software control. This is a general purpose ROM character generator with three different character fonts. Two character fonts are used on this card (a 7x7 double dot and 5x7 single dot), selected by a card jumper. No jumper gives a 7x7 double dot, with a jumper a single dot font is selected.

Timing Generator
This block generates the timing signals used by the 6845 CRT controller and by the dynamic memory. It also resolves the CPU/graphic controller contentions for accessing the Display Buffer.

Composite Color Generator
The logic in this block generates base band video color information.
Modes of Operation

There are two basic modes of operation, ‘Alphanumeric’ and ‘Graphics’. Each of these modes provide further options in both color and black-and-white. The following text describes each mode of operation.

Alphanumeric Mode

Alphanumeric Display Architecture

Every display character position is defined by two bytes in the regen buffer (part of display adapter, not system memory). Both the color and the black and white display adapter use this 2 byte character/attribute format.

<table>
<thead>
<tr>
<th>DISPLAY CHAR 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>

Attribute Byte Definition

<table>
<thead>
<tr>
<th>ATTRIBUTE FUNCTION</th>
<th>FG</th>
<th>BACKGROUND</th>
<th>FOREGROUND</th>
</tr>
</thead>
<tbody>
<tr>
<td>NORMAL</td>
<td>B</td>
<td>0 0 0</td>
<td>I 1 1 1</td>
</tr>
<tr>
<td>REVERSE VIDEO</td>
<td>B</td>
<td>1 1 1</td>
<td>I 0 0 0</td>
</tr>
<tr>
<td>NON DISPLAY (BLK)</td>
<td>B</td>
<td>0 0 0</td>
<td>I 0 0 0</td>
</tr>
<tr>
<td>NON DISPLAY (WHITE)</td>
<td>B</td>
<td>1 1 1</td>
<td>I 1 1 1</td>
</tr>
</tbody>
</table>

I = HIGH LIGHT FOREGROUND (CHAR)
B = BLINK FOREGROUND (CHAR)

Color TV

- Display up to 25 rows of 40 characters each
- Maximum of 256 characters
- Requires 2000 bytes of Read/Write Memory (on the adapter)
- 8x8 character box
- 7x7 double dotted characters (one descender)
- Character attributes (one for each character)
**Character Code Attribute Code**

<table>
<thead>
<tr>
<th>Character Code</th>
<th>Attribute Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>EVEN ADDRESS (M)</td>
<td>ODD ADDRESS (M+1)</td>
</tr>
</tbody>
</table>

**Attribute Byte Definitions**

- **R**: Red
- **G**: Green
- **B**: Blue
- **I**: Intensity

![Attribute Byte Diagram]

Foreground Color
Background Color
Blinking

**Note:** The starting address of the display buffer must be an even location.

**Color Monitor (with Direct Drive input capability)**

- Display up to 25 rows of 80 characters each
- Requires 4000 bytes of Read/Write Memory (on the adapter)
- Maximum of 256 character set
- 8x8 character box
- 7x7 character with one descender
- Same format for attributes as for color TV

**Note:** The starting address of the display buffer must be an even location.
IBM Monochrome Display Adapter Vs. Color/Graphics Adapter Attribute Relationship

Table 3. Monochrome Vs Color/Graphics Attributes

<table>
<thead>
<tr>
<th></th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>R</td>
<td>G</td>
<td>B</td>
<td>I</td>
<td></td>
<td>R</td>
<td>G</td>
<td>B</td>
<td>CH.</td>
<td>BKGD.</td>
</tr>
<tr>
<td>FG</td>
<td>BACKGROUND</td>
<td>FOREGROUND</td>
<td>CH. COLOR</td>
<td>BKGD. COLOR</td>
<td>CH. COLOR</td>
<td>BKGD. COLOR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NORMAL</td>
<td>B</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>I</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>WHITE</td>
<td>BLACK</td>
</tr>
<tr>
<td>RVV</td>
<td>B</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>I</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>BLACK</td>
<td>WHITE</td>
</tr>
<tr>
<td>NON DISP (BLK)</td>
<td>B</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>I</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>BLACK</td>
<td>BLACK</td>
</tr>
<tr>
<td>NON DISP (WHT)</td>
<td>B</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>I</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>WHITE</td>
<td>WHITE</td>
</tr>
</tbody>
</table>

ALL OTHER CODES
DEFINE FOREGROUND
BACKGROUND COLOR
COMBINATIONS

Note: Not all Monitors Recognize the (1) Bit

Table 4. Color/Graphics Modes

<table>
<thead>
<tr>
<th>Horizontal</th>
<th>Vertical</th>
<th>No of Colors (Incl. Background Color)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOW RES</td>
<td>160</td>
<td>100</td>
</tr>
<tr>
<td>MED RES</td>
<td>320</td>
<td>200</td>
</tr>
<tr>
<td>HIGH RES</td>
<td>640</td>
<td>200</td>
</tr>
</tbody>
</table>

IBM Monochrome Display Adapter

NORMAL: All Other Codes Define Foreground, Background Color Combinations

RVV: White on Black Background

Non Disp (Blk): All Other Codes Change Foreground, Background Color to Selected Value

Non Disp (Wht): All Other Codes Change Foreground, Background Color to Selected Value

Table 4. Color/Graphics Modes

<table>
<thead>
<tr>
<th>HORIZONTAL</th>
<th>VERTICAL</th>
<th>NO OF COLORS (INCL. BACKGROUND COLOR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOW RES</td>
<td>160</td>
<td>100</td>
</tr>
<tr>
<td>MED RES</td>
<td>320</td>
<td>200</td>
</tr>
<tr>
<td>HIGH RES</td>
<td>640</td>
<td>200</td>
</tr>
</tbody>
</table>

Note: Not all Monitors Recognize the (1) Bit
1. Low resolution color graphics (TV or monitor). (Note: This mode is not supported in ROM).
   - Up to 100 rows of 160 pels each (2x2)
   - 1 of 16 colors each pel specified by I, R, G, and B
   - Requires 8000 byte of Read/Write Memory (on the adapter)
   - Memory mapped graphics (requires special memory map and set up to be defined later)

2. Medium resolution color graphics (TV or monitor)
   - Up to 200 rows of 320 pels each (1x1)
   - 1 out of 4 preselected colors in each box
   - Requires 16000 bytes of Read/Write Memory (on the adapter)
   - Memory mapped graphics
     4 pels/byte

   FORMAT: 7 6 5 4 3 2 1 0
   
   C1 C0 C1 C0 C1 C0 C1 C0

   First display pel

   - Graphics storage is organized in two banks of 8000 bytes each.

**Graphics Storage Map**

<table>
<thead>
<tr>
<th>Memory Address</th>
<th>Memory Address</th>
<th>Memory Address</th>
<th>Memory Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>#0000</td>
<td>even scans</td>
<td>odd scans</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(0, 2, 4, ...</td>
<td>(1, 3, 5, ...</td>
<td></td>
</tr>
<tr>
<td></td>
<td>198)</td>
<td>199)</td>
<td></td>
</tr>
</tbody>
</table>

(8000 bytes)

Address #0000 contains pel information for upper left corner of display area.
Color selection is determined by the following logic:
C1 and C0 will select 4 of 16 preselected colors.

This color selection (palette) is preloaded in an I/O port.

### Color Selection Logic

- **C1 C0** CODE SELECT COLOR FOR DISPLAY POSITION
- **0 0** DOT TAKES ON COLOR OF 1 OF 16 PRESELECTED BACKGROUND COLORS.
- **0 1** SELECT 1ST COLOR OF PRESELECT COLOR SET “1” OR “2”
- **1 0** SELECT 2ND COLOR OF PRESELECT COLOR SET “1” OR “2”
- **1 1** SELECT 3RD COLOR OF PRESELECT COLOR SET “1” OR “2”

The two color sets are:

**SET ONE**
- COLOR 1 - CYAN
- COLOR 2 - MAGENTA
- COLOR 3 - WHITE

**SET TWO**
- COLOR 1 - GREEN
- COLOR 2 - RED
- COLOR 3 - BROWN

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

### Black and White High Resolution Graphics (Monitor)

- Up to 200 rows of 640 pels each (1x1)
- Black and white only
- Requires 16000 bytes of Read/Write Memory (on the adapter)
- Addressing and mapping is the same as for 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.
- 8 pels/byte
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 position in the buffer.

The CPU and the display control unit have equal access to the display buffer during all the operating modes except high resolution alphanumeric. During this mode, the CPU should access the display buffer during the vertical retrace time (if not, then the display will be affected with random patterns as the CPU is using the display buffer). The characters are displayed from a prestored “character generator” which contains the dot patterns of all the displayable characters.

In the graphics mode the displayed dots and colors are also fetched from the display buffer (up to 16K bytes). In the Color/Graphics Mode Section, the bit configuration for each graphics mode is explained.
Table 5. Summary of Available Colors

<table>
<thead>
<tr>
<th>R GB</th>
<th>COLOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td>Black</td>
</tr>
<tr>
<td>0 0 1</td>
<td>Blue</td>
</tr>
<tr>
<td>0 1 0</td>
<td>Red</td>
</tr>
<tr>
<td>0 1 1</td>
<td>Magenta</td>
</tr>
<tr>
<td>0 1 1</td>
<td>Brown</td>
</tr>
<tr>
<td>0 1 1</td>
<td>Light Gray</td>
</tr>
<tr>
<td>1 0 0</td>
<td>Dark Gray</td>
</tr>
<tr>
<td>1 0 1</td>
<td>Light Blue</td>
</tr>
<tr>
<td>1 1 0</td>
<td>Light Red</td>
</tr>
<tr>
<td>1 1 1</td>
<td>Yellow</td>
</tr>
<tr>
<td>1 1 1</td>
<td>White</td>
</tr>
</tbody>
</table>

Note: “I” provides extra luminance (brightness) to each shade available. Resulting in the light colors listed above, except where the “I” bit is not recognized by some monitors.

Programming Considerations

Programming the 6845 CRT Controller

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

In order to load any of the other 18 registers, the Address Register is first loaded with the necessary pointer and then the CPU may output a value to I/O address 3D5 in order to load the information in the preselected register.

The following table defines the values which must be loaded in 6845 Registers in order to control the different modes of operation supported by the attachment.
**Table 6. 6845 Register Description**

<table>
<thead>
<tr>
<th>ADDR REG.</th>
<th>REG. #</th>
<th>REGISTER TYPE</th>
<th>UNITS</th>
<th>I/O</th>
<th>40x25 ALPHA</th>
<th>80x25 ALPHA</th>
<th>GRAPHIC MODES</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>R0</td>
<td>Horizontal Total</td>
<td>Char.</td>
<td>Write</td>
<td>38</td>
<td>71</td>
<td>38</td>
</tr>
<tr>
<td>1</td>
<td>R1</td>
<td>Horizontal Displayed</td>
<td>Char.</td>
<td>Write</td>
<td>28</td>
<td>50</td>
<td>28</td>
</tr>
<tr>
<td>2</td>
<td>R2</td>
<td>Horiz. Sync Position</td>
<td>Char.</td>
<td>Write</td>
<td>2D</td>
<td>5A</td>
<td>2D</td>
</tr>
<tr>
<td>3</td>
<td>R3</td>
<td>Horiz. Sync Width</td>
<td>Char.</td>
<td>Write</td>
<td>0A</td>
<td>0A</td>
<td>0A</td>
</tr>
<tr>
<td>4</td>
<td>R4</td>
<td>Vertical Total</td>
<td>Char.</td>
<td>Write</td>
<td>1F</td>
<td>1F</td>
<td>7F</td>
</tr>
<tr>
<td>5</td>
<td>R5</td>
<td>Vertical Total Adj.</td>
<td>Scan</td>
<td>Write</td>
<td>06</td>
<td>06</td>
<td>06</td>
</tr>
<tr>
<td>6</td>
<td>R6</td>
<td>Vertical Displayed</td>
<td>Char.</td>
<td>Write</td>
<td>19</td>
<td>19</td>
<td>64</td>
</tr>
<tr>
<td>7</td>
<td>R7</td>
<td>Vert. Sync Position</td>
<td>Char.</td>
<td>Write</td>
<td>1C</td>
<td>1C</td>
<td>70</td>
</tr>
<tr>
<td>8</td>
<td>R8</td>
<td>Interlace Mode</td>
<td>–</td>
<td>Write</td>
<td>02</td>
<td>02</td>
<td>02</td>
</tr>
<tr>
<td>9</td>
<td>R9</td>
<td>Max Scan Line Addr.</td>
<td>Scan</td>
<td>Write</td>
<td>07</td>
<td>07</td>
<td>01</td>
</tr>
<tr>
<td>A</td>
<td>R10</td>
<td>Cursor Start</td>
<td>Scan</td>
<td>Write</td>
<td>06</td>
<td>06</td>
<td>06</td>
</tr>
<tr>
<td>B</td>
<td>R11</td>
<td>Cursor End</td>
<td>Scan</td>
<td>Write</td>
<td>07</td>
<td>07</td>
<td>07</td>
</tr>
<tr>
<td>C</td>
<td>R12</td>
<td>Start Addr. (H)</td>
<td>–</td>
<td>Write</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>D</td>
<td>R13</td>
<td>Start Addr. (L)</td>
<td>–</td>
<td>Write</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>E</td>
<td>R14</td>
<td>Cursor Addr. (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 Addr. (L)</td>
<td>–</td>
<td>Read/Write</td>
<td>XX</td>
<td>XX</td>
<td>XX</td>
</tr>
<tr>
<td>10</td>
<td>R16</td>
<td>Light Pen (H)</td>
<td>–</td>
<td>Read Only</td>
<td>XX</td>
<td>XX</td>
<td>XX</td>
</tr>
<tr>
<td>11</td>
<td>R17</td>
<td>Light Pen (L)</td>
<td>–</td>
<td>Read Only</td>
<td>XX</td>
<td>XX</td>
<td>XX</td>
</tr>
</tbody>
</table>

**Note:** All register values are given in hexadecimal.
Programming the Mode Control and Status Register

The following I/O devices are defined on the Color/Graphics Adapter.

<table>
<thead>
<tr>
<th>HEX ADDR.</th>
<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>FUNCTION OF REGISTER</th>
</tr>
</thead>
<tbody>
<tr>
<td>X'308'</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>DO REG (MODE CONTROL)</td>
</tr>
<tr>
<td>X'309'</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>DO REG (COLOR SELECT)</td>
</tr>
<tr>
<td>X'30A'</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>CLEAR LIGHT PEN LATCH</td>
</tr>
<tr>
<td>X'30B'</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>PRE SET LIGHT PEN LATCH</td>
</tr>
<tr>
<td>X'3D8'</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Z</td>
<td>Z</td>
<td>0</td>
<td>0</td>
<td>6845 REGISTERS</td>
</tr>
<tr>
<td>X'3D9'</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Z</td>
<td>Z</td>
<td>0</td>
<td>1</td>
<td>6845 REGISTERS</td>
</tr>
<tr>
<td>X'DA'</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Z</td>
<td>Z</td>
<td>1</td>
<td>0</td>
<td>6845 REGISTERS</td>
</tr>
<tr>
<td>X'DB'</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Z</td>
<td>Z</td>
<td>0</td>
<td>1</td>
<td>6845 REGISTERS</td>
</tr>
<tr>
<td>X'DC'</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Z</td>
<td>Z</td>
<td>1</td>
<td>0</td>
<td>6845 REGISTERS</td>
</tr>
<tr>
<td>Z = don’t care condition</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>

Color Select Register

This is a 6 bit output only, register, it can not be read, its address is X'3D9' and can be written using the 8088 I/O OUT command.

The following is a description of the Register functions.

| Bit 0 | B (BLUE) Border Color Select ALPHA/BACKGROUND |
| Bit 1 | G (GREEN) Border Color Select ALPHA/BACKGROUND |
| Bit 2 | R (RED) Border Color Select ALPHA/BACKGROUND |
| Bit 3 | Intensifies Border Color Select ALPHA/BACKGROUND IN 320 x 200 |
| Bit 4 | Select Alt Back Color Set For Alpha Color Modes |
| Bit 5 | 320 x 200 Color Set Select |
| Bit 6 | Not Used |
| Bit 7 | Not Used |

Bits 0, 1, 2, 3. Select the screens border color in 40x25 alpha mode. In graphics mode (medium resolution) 320 x 200 color, the screen background color (C0-C1) is selected by these bit settings.

Bit 4. This bit when set will select on alternate, intensified, set of background colors in the alpha mode.

Bit 5 is only used in the medium resolution color mode (320 x 200). It is used to select the active set of screen colors for the display.
When bit 5 is set to a “1” colors are determined as follows.

The C1 C0 Set selected are:
0 0  Background as defined by Bit 0-3 of Port ‘3D9’
0 1  Cyan
1 0  Magenta
1 1  White

When bit 5 is set to a “0” Colors are determined as follows.

The C0 C1 Set selected are:
0 0  Background as defined by Bit 0-3 of Port ‘3D9’
0 1  Green
1 0  Red
1 0  Yellow

Mode Select Register

This is a 6 bit output only register, it can not be read. Its address is X’3D8’. It can be written using the 8088 I/O OUT command.

The following is a description of the registers functions.

Bit 0

Bit 0  80 x 25 mode
Bit 1  Graphic Select
Bit 2  B & W Select
Bit 3  Enable Video Signal
Bit 4  High Res 640 x 200 B & W Mode
Bit 5  Change BACKGROUND INTENSITY to Blink Bit
Bit 6  Not Used
Bit 7  Not Used

Bit 0  Selects between 40 x 25 and 80 x 25 alpha mode, a “1” sets it to 80 x 25 mode.

Bit 1  Selects between ALPHA mode and 320 x 200 graphics mode, a “1” select 320 x 200 graphics mode.

Bit 2  Selects color or B & W mode, a “1” selects B & W.

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

Bit 4  When on, this bit selects the 640 x 200 B & W graphics mode. One color of 8 can be selected on direct drive sets in this mode by using register 3D9.
Bit 5  When on, this bit will change the character background intensity to the blinking attribute function for ALPHA modes. When the high order attribute bit is not selected, 16 background colors (or intensified colors) are available. For normal operation, this bit should be set to "1" to allow the blinking function.

Mode Register Summary

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

40 x 25 ALPHA B & W
40 x 25 ALPHA COLOR
80 x 25 ALPHA B & W
80 x 25 ALPHA COLOR
320 x 200 B & W GRAPHICS
320 x 200 COLOR GRAPHICS
640 x 200 B & W GRAPHICS

* THE LOW RESOLUTION 160 x 100 MODE REQUIRES SPECIAL PROGRAMMING AND IS SET UP AS ALPHA MODE 40 x 25

Status Register

The status register is a 4 bit read only register. Its address is X‘3DA’. It can be read using the 8088 I/O IN instruction.
The following is a description of the register functions.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Display Enable</td>
</tr>
<tr>
<td>1</td>
<td>Light Pen Trigger Set</td>
</tr>
<tr>
<td>2</td>
<td>Light Pen SW Made</td>
</tr>
<tr>
<td>3</td>
<td>Alpha Dots</td>
</tr>
<tr>
<td>4</td>
<td>Not Used</td>
</tr>
<tr>
<td>5</td>
<td>Not Used</td>
</tr>
<tr>
<td>6</td>
<td>Not Used</td>
</tr>
<tr>
<td>7</td>
<td>Not Used</td>
</tr>
</tbody>
</table>

Bit 0  This input 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 input has set the light pen trigger. This trigger is reset on power on and may also be cleared by doing an I/O OUT command to address X'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 the switch is on.

Bit 3  The ALPHA video output signal is readable in this status bit. Its purpose is to verify that video information is being generated for RAS purposes.

Sequence of Events
1. Determine mode of operation
2. Reset Video Enable bit
3. Program 6845 to select mode
4. Program mode/color select registers

Memory Requirements
The memory used by this adapter is self-contained. It consists of 16k bytes of memory without parity. This memory is used as both a display buffer for alphanumeric data and as a bit map for graphics data. The Regen Buffers address starts at X'B8000'.

Interrupt Level (Vertical Retrace)
Level 2
I/O Address and Bit Map
Read/Write Memory Address Space

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>01000</td>
<td>System Read/Write Memory</td>
</tr>
<tr>
<td>B8000</td>
<td>Display Buffer (16K Bytes)</td>
</tr>
<tr>
<td>BBFFFF</td>
<td>128K RESERVED REGEN AREA</td>
</tr>
<tr>
<td>C8FFFF</td>
<td>Display Buffer (16K Bytes)</td>
</tr>
</tbody>
</table>
Color/Graphics Monitor Adapter Direct Drive, and Composite Interface Pin Assignment

**Rear Panel**

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

**Composite Video Signal**

Composite Video Signal of approximately 1.5 Volts Peak to Peak Amplitude

<table>
<thead>
<tr>
<th>Pin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Chassis Ground</td>
</tr>
</tbody>
</table>

**Composite Phono Jack**

Hook-up to Monitors

<table>
<thead>
<tr>
<th>Pin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Color/Graphics Composite Jack</td>
</tr>
</tbody>
</table>
**Color/Graphics Monitor Adapter**

**Auxiliary Video Connectors**

- **PI-4 PIN BERG STRIP FOR RF MODULATOR**
- **P2-6 PIN BERG STRIP FOR LIGHT PEN CONNECTOR**

---

**RF Modulator Interface**

<table>
<thead>
<tr>
<th>Pin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>+12 Volts</td>
</tr>
<tr>
<td>2</td>
<td>(key) Not Used</td>
</tr>
<tr>
<td>3</td>
<td>Composite Video Output</td>
</tr>
<tr>
<td>4</td>
<td>Logic Ground</td>
</tr>
</tbody>
</table>

**Light Pen Interface**

<table>
<thead>
<tr>
<th>Pin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Light Pen Input</td>
</tr>
<tr>
<td>2</td>
<td>(key) Not Used</td>
</tr>
<tr>
<td>3</td>
<td>Light Pen Switch</td>
</tr>
<tr>
<td>4</td>
<td>Chassis Ground</td>
</tr>
<tr>
<td>5</td>
<td>+ 5 Volts</td>
</tr>
<tr>
<td>6</td>
<td>+ 12 Volts</td>
</tr>
</tbody>
</table>

---

**Note:** Key indicates not used.
Parallel 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 which 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 instructions. 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.

This function is packaged on an adapter which fits into any of the five System Expansion slots on the System Board. The input/output signals are made available at the back of the adapter via a right angle PCB mounted 25 PIN “D” type connector. This connector protrudes through the rear panel of the System Unit where a cable and shield 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 fault isolation determination 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 following page.
Parallel Printer Adapter Block Diagram

Figure 16. PARALLEL PRINTER ADAPTER BLOCK DIAGRAM
Programming Considerations

The Printer Adapter responds to 5 I/O instructions - 2 output and 3 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 CPU to read back the contents of the two latches. The third allows the CPU 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>Parallel Printer Adapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output to address 3BCH</td>
<td>Output to address 378H</td>
</tr>
<tr>
<td>Bit 7</td>
<td>Bit 3</td>
</tr>
<tr>
<td>Pin 9</td>
<td>Pin 5</td>
</tr>
<tr>
<td>Bit 6</td>
<td>Bit 2</td>
</tr>
<tr>
<td>Pin 8</td>
<td>Pin 4</td>
</tr>
<tr>
<td>Bit 5</td>
<td>Bit 1</td>
</tr>
<tr>
<td>Pin 7</td>
<td>Pin 3</td>
</tr>
<tr>
<td>Bit 4</td>
<td>Bit 0</td>
</tr>
<tr>
<td>Pin 6</td>
<td>Pin 2</td>
</tr>
</tbody>
</table>

This 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>Parallel Printer Adapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output to address 3BEH</td>
<td>Output to address 37AH</td>
</tr>
<tr>
<td>Bit 4</td>
<td>Bit 3</td>
</tr>
<tr>
<td>IRQ Enable</td>
<td>Bit 1</td>
</tr>
<tr>
<td>Enable</td>
<td>Bit 0</td>
</tr>
<tr>
<td></td>
<td>Pin 17</td>
</tr>
<tr>
<td></td>
<td>Pin 16</td>
</tr>
<tr>
<td></td>
<td>Pin 14</td>
</tr>
</tbody>
</table>

This instruction causes this latch to capture the five least significant bits of 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 1, the card will interrupt the CPU on the condition that Pin 10 transitions high to low.

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

Note: For pin references, see Parallel Interface Connector Specifications, page 2-69.
This command presents the CPU with data present on the pins associated with the out to x’ ‘3BC’. This should normally reflect the exact value that was last written to x ‘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 ‘or’ ed with the latch contents.

<table>
<thead>
<tr>
<th>IBM Monochrome Display &amp; Printer Adapter</th>
<th>Parallel Printer Adapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input from address x’ ‘3BC’</td>
<td>Input from address 378H</td>
</tr>
</tbody>
</table>

This command presents real time status to the CPU from the pins as follows.

| Bit 7 Bit 6 Bit 5 Bit 4 Bit 3 Bit 2 Bit 1 Bit 0 |
|--------|--------|--------|--------|--------|--------|--------|--------|
| Pin 11*| Pin 10 | Pin 12 | Pin 13 | Pin 15 | -      | -      | -      |

This instruction causes the data present on pins 1, 14, 16, 17 and IRQ bit to be read by the CPU. In the absence of external drive applied to these pins, data read by the CPU will exactly match data last written to x’ ‘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 ‘or’ ed with data applied to the pins by the x’ ‘3BE’ latch.

| Bit 7 Bit 6 Bit 5 Bit 4 Bit 3 Bit 2 Bit 1 Bit 0 |
|--------|--------|--------|--------|--------|--------|--------|--------|
| IRQ    | Pin 17 | Pin 16 | Pin 14 | Pin 1  |

Enable
Por=0
Por=1
Por=0
Por=1
Por=1
Por=1

These pins assume the states shown after a reset from the CPU.

Note: For pin references see Parallel Printer Adapter Interface Connector Specifications page 2-69.
Parallel Printer Adapter  
Interface Connector Specifications

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>AMP Pin No.</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>
IBM 80 CPS Matrix Printer

The printer is a self powered, standalone table top unit. It attaches to the System Unit via a parallel signal cable which is 6 feet in length. The unit obtains its AC power from a standard wall outlet (120 Vac). The printer is an 80 Character Per Second (CPS) bidirectional wire matrix device. It has a 9 wire head, allowing it to print characters in a 9x9 dot matrix. It can print in compressed mode 132 characters per line and in standard font, 80 characters per line. A large font also prints in 66 characters per line mode. The printer can print double size characters and double dotted characters. The printer prints the standard ASCII 96 character uppercase and lowercase character sets. In addition, a set of 64 special block graphic characters are available.

The printer can also accept commands setting the feed control desired for the application. Setting of 1 to 66 lines per page can be programmed and the lines per inch may be set to 5, 8, or 10. This printer attaches to the System Unit via the Parallel Printer Adapter or the combination Monochrome Display Adapter and Parallel Printer Adapter. The cable is a 25 lead shielded cable with a 25 pin “D” type connector at the System Unit end, and a 36 pin connector on the printer end.

Note: You may lose data anytime you are running a program with the printer off and attached to the System Unit.
Table 7. Printer Specifications

<table>
<thead>
<tr>
<th>(1) PRINT METHOD:</th>
<th>Serial impact dot matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>(2) PRINT SPEED:</td>
<td>80 CPS</td>
</tr>
<tr>
<td>(3) PRINT DIRECTION:</td>
<td>Bidirectional with logical seeking</td>
</tr>
<tr>
<td>(4) NUMBER OF PINS IN HEAD:</td>
<td>9</td>
</tr>
<tr>
<td>(5) LINE SPACING:</td>
<td>4.23 mm (1/6&quot;) or programmable</td>
</tr>
<tr>
<td>(6) PRINTING CHARACTERISTICS</td>
<td>9 x 9</td>
</tr>
<tr>
<td>Matrix:</td>
<td>Full 96-character ASCII with decoders, plus 9 international characters/symbols</td>
</tr>
<tr>
<td>Character Set:</td>
<td>64 block characters</td>
</tr>
<tr>
<td>Graphic Character:</td>
<td></td>
</tr>
<tr>
<td>(7) PRINTING SIZES</td>
<td></td>
</tr>
<tr>
<td>Normal:</td>
<td>Maximum</td>
</tr>
<tr>
<td></td>
<td>Characters</td>
</tr>
<tr>
<td></td>
<td>per inch</td>
</tr>
<tr>
<td></td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>16.5</td>
</tr>
<tr>
<td></td>
<td>8.25</td>
</tr>
<tr>
<td>Enlarged:</td>
<td></td>
</tr>
<tr>
<td>Condensed:</td>
<td></td>
</tr>
<tr>
<td>Condensed Enlarged:</td>
<td></td>
</tr>
<tr>
<td>(8) MEDIA HANDLING</td>
<td>Adjustable sprocket pin feed</td>
</tr>
<tr>
<td>Paper Feed:</td>
<td></td>
</tr>
<tr>
<td>Paper Width Range:</td>
<td>101.6 mm (4&quot;) to 254 mm (10&quot;)</td>
</tr>
<tr>
<td>Copies:</td>
<td>One original plus two carbon copies (total thickness not to exceed 0.3 mm (0.012&quot;)</td>
</tr>
<tr>
<td>Paper Path:</td>
<td>Rear</td>
</tr>
<tr>
<td>(9) INTERFACES</td>
<td></td>
</tr>
<tr>
<td>Standard:</td>
<td>Parallel 8-bit</td>
</tr>
<tr>
<td></td>
<td>Data &amp; Control Lines</td>
</tr>
<tr>
<td>(10) INKED RIBBON</td>
<td></td>
</tr>
<tr>
<td>Color:</td>
<td>Black</td>
</tr>
<tr>
<td>Type:</td>
<td>Cartridge</td>
</tr>
<tr>
<td>Life Expectancy:</td>
<td>3 million characters</td>
</tr>
<tr>
<td>(11) ENVIRONMENTAL CONDITIONS</td>
<td></td>
</tr>
<tr>
<td>Operating Temperature Range:</td>
<td>5 to 35°C (41 to 95°F)</td>
</tr>
<tr>
<td>Operating Humidity:</td>
<td>10 to 80% non-condensing</td>
</tr>
<tr>
<td>(12) POWER REQUIREMENT</td>
<td></td>
</tr>
<tr>
<td>Voltage:</td>
<td>120VAC, 60 Hz</td>
</tr>
<tr>
<td>Current:</td>
<td>1 Amp maximum</td>
</tr>
<tr>
<td>Power Consumption:</td>
<td>100 VA maximum</td>
</tr>
<tr>
<td>(13) PHYSICAL CHARACTERISTICS</td>
<td></td>
</tr>
<tr>
<td>Height:</td>
<td>107 mm (4.2&quot;)</td>
</tr>
<tr>
<td>Width:</td>
<td>374 mm (14.7&quot;)</td>
</tr>
<tr>
<td>Depth:</td>
<td>305 mm (12.0&quot;)</td>
</tr>
<tr>
<td>Weight:</td>
<td>5.5 kg (12 lbs.)</td>
</tr>
</tbody>
</table>
Setting The DIP Switches

There are two DIP switches on the control circuit board. In order to suit 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 Table 8 (DIP Switch 1) and Table 9 (DIP Switch 2).

Table 8. Functions and Conditions of DIP Switch 1

<table>
<thead>
<tr>
<th>Pin No.</th>
<th>Function</th>
<th>ON</th>
<th>OFF</th>
<th>Factory-set Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Not applicable</td>
<td></td>
<td></td>
<td>ON</td>
</tr>
<tr>
<td>2</td>
<td>CR</td>
<td>Print only</td>
<td>Print &amp; line feed</td>
<td>ON</td>
</tr>
<tr>
<td>3</td>
<td>Buffer full</td>
<td>Print only</td>
<td>Print &amp; line feed</td>
<td>ON</td>
</tr>
<tr>
<td>4</td>
<td>Cancel code</td>
<td>Invalid</td>
<td>Valid</td>
<td>OFF</td>
</tr>
<tr>
<td>5</td>
<td>Delete code</td>
<td>Invalid</td>
<td>Valid</td>
<td>ON</td>
</tr>
<tr>
<td>6</td>
<td>Error Buzzer</td>
<td>Sounds</td>
<td>Does not sound</td>
<td>ON</td>
</tr>
<tr>
<td>7</td>
<td>Character generator</td>
<td>N.A.</td>
<td>Graphic patterns select</td>
<td>OFF</td>
</tr>
<tr>
<td></td>
<td>(Graphic pattern select)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>SLCT IN signal</td>
<td>Fixed</td>
<td>Not fixed</td>
<td>ON</td>
</tr>
<tr>
<td></td>
<td>Fixed internally</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Not fixed internally</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 9. Functions and Conditions of DIP Switch 2

<table>
<thead>
<tr>
<th>Pin No.</th>
<th>Function</th>
<th>ON</th>
<th>OFF</th>
<th>Factory-set Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Not applicable</td>
<td>-</td>
<td>-</td>
<td>ON</td>
</tr>
<tr>
<td>2</td>
<td>AUTO FEED</td>
<td>-</td>
<td>-</td>
<td>ON</td>
</tr>
<tr>
<td></td>
<td>{ Fixed internally</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>{ Not fixed internally</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>XT signal</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fixed internally</td>
<td>Fixed</td>
<td>Not fixed</td>
<td>OFF</td>
</tr>
<tr>
<td>4</td>
<td>Coding table select</td>
<td>N.A.</td>
<td>Standard</td>
<td>OFF</td>
</tr>
</tbody>
</table>

Parallel Interface Description

(1) Specifications
(a) Data transfer rate: 1000 CPS (max.)
(b) Synchronization: By externally supplied STROBE pulses.
(c) Handshaking: ACKNLG or BUSY signals.
(d) Logic level: Input data and all interface control signals are compatible with the TTL level.

(2) Connector
Plug: 57-30360 (AMPHENOL)

(3) Connector pin assignment and descriptions of signals. Connector pin assignment and descriptions of respective interface signals are provided in Table (10) page 2-74.
<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 0.5μs at receiving terminal. The signal level is normally &quot;HIGH&quot;; read-in of data is performed at the &quot;LOW&quot; level of this signal.</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>DATA 1</td>
<td>In</td>
<td>These signals represent information of the 1st to 8th bits of parallel data respectively. Each signal is at &quot;HIGH&quot; level when data is logical &quot;1&quot; and &quot;LOW&quot; when logical &quot;0&quot;.</td>
</tr>
<tr>
<td>3</td>
<td>21</td>
<td>DATA 2</td>
<td>In</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>22</td>
<td>DATA 3</td>
<td>In</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>23</td>
<td>DATA 4</td>
<td>In</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>24</td>
<td>DATA 5</td>
<td>In</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>25</td>
<td>DATA 6</td>
<td>In</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>26</td>
<td>DATA 7</td>
<td>In</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>27</td>
<td>DATA 8</td>
<td>In</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>28</td>
<td>ACKNLG</td>
<td>Out</td>
<td>Approx. 5μs pulse. &quot;LOW&quot; indicates that data has been received and that the printer is ready to accept other data.</td>
</tr>
<tr>
<td>11</td>
<td>29</td>
<td>BUSY</td>
<td>Out</td>
<td>A &quot;HIGH&quot; signal indicates that the printer cannot receive data. The signal becomes &quot;High&quot; in the following cases: 1. During data entry 2. During printing operation 3. In OFF-LINE state 4. During printer error status.</td>
</tr>
</tbody>
</table>
### Table 10. Connector Pin Assignment and Descriptions of Interface Signals (cont.)

<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. Not used.</td>
</tr>
<tr>
<td>18</td>
<td>-</td>
<td>NC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19–30</td>
<td>-</td>
<td>GND</td>
<td></td>
<td>TWISTED-PAIR RETURN signal GND level.</td>
</tr>
<tr>
<td>31</td>
<td>-</td>
<td>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 50µs at the receiving terminal.</td>
</tr>
<tr>
<td>Signal Pin No.</td>
<td>Return Pin No.</td>
<td>Signal</td>
<td>Direction</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>----------------</td>
<td>--------</td>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>32</td>
<td></td>
<td>ERROR</td>
<td>Out</td>
<td>The level of this signal becomes “LOW” when the printer is in— 1. PAPER END state 2. OFF-LINE state 3. Error state</td>
</tr>
<tr>
<td>33</td>
<td></td>
<td>GND</td>
<td>—</td>
<td>Same as with Pin No. 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 +5V through 4.7KΩ resistance.</td>
</tr>
<tr>
<td>36</td>
<td></td>
<td>SLCT IN</td>
<td>In</td>
<td>Data entry to the printer is possible only when the level of this signal is “LOW”. (Internal fixing can be carried out with DIP SW 1-8. The condition at the time of shipment is set “LOW” for this signal.)</td>
</tr>
</tbody>
</table>

NOTES 1: “Direction” refers to the direction of signal flow as viewed from the printer.
2: “Return” denotes “TWISTED PAIR RETURN” and is to be connected at signal ground level.
As to the wiring for 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 the printer, respectively.
3: All interface conditions are based on TTL level. Both the rise and fall times of each signal must be less than 0.2μs.
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”.)
(4) Data transfer sequence

Fig. 17 shows the sequence for data transmission.

![Diagram showing data transfer sequence with BUSY, ACKNLG, APPROX. 5μs, DATA, STROBE signals and timing details.

**Figure 18. PARALLEL INTERFACE TIMING DIAGRAM**
ASCII Coding Table

Table 11 shows all available codes when the Printer is set for operation with standard coding by setting the DIP switch pin 2-4 to the OFF position. This DIP switch pin is factory-set to the OFF position.

<table>
<thead>
<tr>
<th>ASCII Coding Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>A</td>
</tr>
<tr>
<td>B</td>
</tr>
<tr>
<td>C</td>
</tr>
<tr>
<td>D</td>
</tr>
<tr>
<td>E</td>
</tr>
<tr>
<td>F</td>
</tr>
</tbody>
</table>

2-78
ASCII Control Codes

Control Codes

Various kinds of control codes are contained in Table 11. These control codes are recognized by the printer and perform specified functions upon receipt of these codes. The following are descriptions of respective control codes.

(1) CR (Carriage Return)
When the CR code is transmitted to the print buffer, all data stored in the print buffer is printed.
(When AUTO FEED XT (Pin No. 14) is at “LOW” level or DIP switch pin 2-2 is ON, the paper is advanced one line automatically after printing.)

Note: When 80 columns of print data (including spaces) are continuously received and the following data is valid and printable, the Printer automatically begins to print the data stored in the print buffer. In this case, if AUTO FEED XT is at “LOW” level or DIP switch pin 2-3 is ON, the paper is advanced one line after printing.

(2) LF (Line Feed)
When the LF code is input, all data in the print buffer is printed and the paper is advanced one line.

Note: If no data precedes the LF code, or if all preceding data is “SPACE”, only paper feeding is performed. For example, if the data is transferred in the order of DATA→CR→LF, DATA will be printed by the CR code, and when the Printer receives the LF code, it only carries out one line feed.

(3) VT (Vertical Tab)
When the VT code is input, all data preceding this code is printed. And the paper is advanced to the line position set by “ESC B” (described later). If no vertical tab position is set by ESC B, the VT code behaves like the LF code. Therefore, the paper is advanced one line after printing.

(4) FF (Form Feed)
The FF code carries out the printing of all data stored in the print buffer and advances the paper to the next predetermined Top of Form position. The Top of Form is determined when the POWR switch is turned on or the INIT signal is applied. If the form length per page is not set by “ESC C+n”, it is regarded as 66 or 72 lines.
Note: The form length of 72 lines per page is applicable to only the version marked with identifier code “M72” on the rear side of the lower case of the Printer. This code always initializes the printing of the data stored in the print buffer.

(5) SO (Shift Out)
When the SO code is input, all data that follows it in the same line will be printed out in enlarged (double width) characters. This code is cancelled by the printing operation or the input of “DC 4” code and can be input at any column position on a line. Therefore, normal size and enlarged characters can be mixed on the same line.

1. [DATA] ABC SO DEF DC4 GHI CR LF
[PRINT] ABCDEFGHI

2. [DATA] ABCD SO EF GH CR LF IJLK SO MNOP CR LF
[PRINT] ABCDEFGH IJKLMNOP

(6) SI (Shift In)
When the SI code is input, all data that follows it will be printed out in condensed characters. This code is cancelled by the input of “DC 2” code. The SI code can be input at any column position on a line, but all characters/symbols on the line containing SI code are printed out in condensed characters. When printing condensed characters, the data capacity of the print buffer will become 132 columns per line.
When the SO code is received after the input of the SI code, condensed enlarged characters (double width of condensed characters) can be printed. This condition is cancelled by “DC 4” code, and the character size returns to “condensed”.

1. [DATA] SI ABCDEFGHIJKLMNOP CR LF
[PRINT] ABCDEFGHIJKLMNOP

2. [DATA] ABC SI DEF SO GHI KL CR LF
[PRINT] ABCDEFGHIJKLMNOP

(7) DC 4 (Device Control 4)
The DC 4 code cancels the SO mode.

[DATA] SI ABCDEFGHIJKLMNOP
[PRINT] ABCDEFGHIJKLMNOP
(8) DC 2 (Device Control 2)
The DC 2 code cancels the SI mode.

<table>
<thead>
<tr>
<th>DATA</th>
<th>SI</th>
<th>ABCDEF</th>
<th>SO</th>
<th>GHI</th>
<th>LF</th>
<th>DC 2</th>
<th>JKLMN</th>
<th>CR</th>
<th>LF</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRINT</td>
<td>ABCDEFGHI</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>JKLNM</td>
<td></td>
</tr>
</tbody>
</table>

(9) HT (Horizontal Tab)
The HT code carries out the horizontal tabulation. If there is no tab position set, this code is ignored. The tab stop positions are set by “ESC D+n” (described later).

(10) CAN (Cancel)
Upon the input of the CAN code, all data previously stored in the print buffer is cancelled. Therefore, this code is regarded as the print buffer clear command. This code clears the print buffer, but control codes (Excluding the SO code) are still valid even if the CAN code is transferred. The validity or invalidity of the CAN code is selectable by the DIP switch pin 1-4 on the control circuit board.

(11) DEL (Delete)
This code functions the same as the CAN code. The validity or invalidity of the DEL code is selectable by the DIP switch pin 1-5 on the control circuit board.

(12) DC 1 (Device Control 1)
The DC 1 code places the Printer in the Selected state. With the Printer in the Selected state, if the DC 1 code is input during data transfer, all data stored before the DC 1 code is ignored.

(13) DC 3 (Device Control 3)
The DC 3 code places the Printer in the Deselected state. In other words, it disables the Printer to receive data. Once the Printer is put in the Deselected state by the DC 3 code, the Printer will not revert to the Selected state unless the DC 1 code is input again.

Note: When the DC 1 and DC 3 codes are used, DIP switch pin 1-8 should be in the “OFF” position.

<table>
<thead>
<tr>
<th>DATA</th>
<th>DC 1</th>
<th>AAAAA</th>
<th>DC 3</th>
<th>BBBBB</th>
<th>DC 1</th>
<th>CCCCC</th>
<th>CR</th>
<th>LF</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRINT</td>
<td>AAAAAACCCCC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>JKLNM</td>
<td></td>
</tr>
</tbody>
</table>

2. [DATA] AAAAAA DC 1 BBBBBB DC 3 CCCCC C DC 1 CR LF

[PRINT] BBBBBB
Relations among the ON LINE switch, SLCT IN signal, DC1/DC3 code and interface signals are shown in Table 12 below.

Table 12. DC1/DC3 And Data Entry

<table>
<thead>
<tr>
<th>ON LINE Switch</th>
<th>SLCT IN</th>
<th>DC1/DC3</th>
<th>ERROR</th>
<th>BUSY</th>
<th>ACKNLG</th>
<th>SLCT</th>
<th>DATA ENTRY</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFF-LINE</td>
<td>HIGH/LOW</td>
<td>DC1/DC3</td>
<td>LOW</td>
<td>HIGH</td>
<td>Not</td>
<td>LOW</td>
<td>Impossible</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Generated</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ON-LINE</td>
<td>HIGH</td>
<td>DC1</td>
<td>HIGH</td>
<td>LOW/</td>
<td>Generated</td>
<td>HIGH</td>
<td>Possible (Normal entry)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DC3</td>
<td>HIGH</td>
<td>LOW/</td>
<td>Generated</td>
<td>LOW</td>
<td>Possible (See Note 1.)</td>
</tr>
<tr>
<td></td>
<td>LOW</td>
<td>DC1/DC3</td>
<td>HIGH</td>
<td>LOW/</td>
<td>Generated</td>
<td>HIGH</td>
<td>Possible (Normal entry)</td>
</tr>
</tbody>
</table>

NOTES 1: In Table 12, it is assumed that as soon as the Printer receives data, it sends back the ACKNLG signal, though this data is not stored in the print buffer. In this status, the Printer is waiting for the DC1 code for normal entry.

2: The DC1/DC3 code is valid under the condition that the DIP switch pin 1-8 is OFF, namely, the level of SLCT IN at the pin No. 36 of the interface connector is “HIGH”. When SLCT IN is “LOW”, the DC1/DC3 code is not valid.

(14) NUL (Null)
The NUL code is regarded as the termination for tabulation setting sequence (described in detail later).

(15) BEL (Bell)
When the BEL code is input, the buzzer sounds for about 3 seconds.

(16) Escape (ESC) control
(a) Escape numerical control
Input of an “ESC” code followed by an ASCII numeric code permits each of the following functions to be performed.

1) ESC 0 (Escape 0)
Receipt of an “ESC” followed by ASCII code “0” causes the line spacing to be set at 1/8 inch. Input of the ESC 2 code or INIT signal to the interface connector or turning the power off and on again causes the line spacing to return to 1/6 inch.

2) ESC 1 (Escape 1)
Receipt of an “ESC” followed by ASCII code “1” causes the line spacing to be set at 7/72 inch. Input of the ESC 2 code or INIT signal to the interface connector or turning the power off and on again causes the line spacing to return to 1/6 inch.
3) ESC 2 (Escape 2)
Receipt of an “ESC” followed by ASCII code “2” causes the line spacing to be set at 1/6 inch. When the POWER switch is turned on, the line spacing is set at initial 1/6 inch. The ESC 2 code is also a command to execute “ESC A+n” modes (described later).

4) ESC 8 (Escape 8)
The ESC 8 code makes it possible to transmit data even if there is no paper in the Printer. This code should be transmitted before the Printer runs out of paper. After transmitting this code, when the Printer runs out of paper, the PE signal of the interface connector turns to High level; the ERROR signal remains at High level.

5) ESC 9 (Escape 9)
This code cancels the ESC 8 condition. When the power is turned on, the Printer is initialized into ESC 9 status. Therefore, the Printer cannot receive data when there is no paper.

6) ESC SI
This code functions the same as “SI”.

7) ESC SO
This code functions the same as “SO”.

(b) ESC alphabetic control
Receipt of an “ESC” code followed by ASCII code “X”(alphabetic code) permits each of the following functions to be performed.

Note: “n” represents a 7-bit binary number, and the most significant bit is not treated as data. “+” is inserted for the purpose of legibility only, and should not be input in actual operation.

1) ESC A+n
This code specifies the amount of line spacing in the Line Feed 1≤<n>10≤85 (Decimal): “n” is a binary number. “n”=1 is equivalent to 1/72 inch paper advancement. Since the distance between any two dot wires of the print head is 1/72 inch, any line spacing in increments proportional to the distance between the dot wires can be established.
The ESC A code is the command only to store spacing data into the memory. In other words, even if spacing data was transferred into the memory, the Printer does not actually carry out the line spacing in accordance with the spacing data. To execute the line spacing in accordance with the stored data, the ESC 2 code should be followed. Namely, the ESC 2 code is considered as the execution command for the line spacing.

<table>
<thead>
<tr>
<th>DATA</th>
<th>AAAAAAA</th>
<th>CR</th>
<th>LF</th>
<th>BBBBBBBB</th>
<th>CR</th>
<th>LF</th>
<th>ESC A+24</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CCCCCC</td>
<td>CR</td>
<td>LF</td>
<td>DDDDDDD</td>
<td>ESC 2</td>
<td>CR</td>
<td>LF</td>
</tr>
<tr>
<td></td>
<td>EEEEEEE</td>
<td>CR</td>
<td>LF</td>
<td>FFFFFFFF</td>
<td>CR</td>
<td>LF</td>
<td></td>
</tr>
</tbody>
</table>

Note: <How to input “n”>
When “n” is actually transferred to the Printer as data, it is transferred in the form of a 7-bit binary number.
In case of “ESC A+24”, actual output to the Printer is performed as <1B>H<41>H<18>H in hexadecimal code.

2) ESC B+n1+n2+nk+NUL
(1≤<n>≤66, 1≤k≤64, nk≤nk+1)
This code specifies the vertical tab stop positions. The first 64 valid tab stops per page are recognized in the Printer; subsequent tab stops are ignored.
A tab stop set at a line exceeding the form length is ignored. Tab stop numbers must be received in incremental numerical order. To execute predetermined tab stop positions, the VT code should be input. Once vertical tab stops are established, the data will be valid until new tab stops are specified. If no tab stop is set, the VT code
behaves like the LF code. Therefore, the paper is advanced one line after printing.

Receipt of “ESC B” code causes the Printer to accept the following codes as tab stop line numbers until the NUL code is input. The lack of the NUL code will cause incorrect data printout.

The form length must be set by “ESC C+n” code prior to setting tab stops.

Input of “ESC B” code followed by only the NUL code cancels predetermined tab stops.

3) ESC C+n (1≤<n>10≤66)
   This code specifies the form length per page. The form length is determined by the number of lines (=“n”). The amount of a line spacing at this point is a predetermined numerical value by “ESC A+n”. When the form length is not programmed, one page is assumed at 66 or 72 lines. Prior to setting the vertical tab position, the form length should be set.

4) ESC D+n1+n2+...+nk+NUL
   (1≤<n>10≤127,k≤112)
   This code specifies the horizontal tab stop positions. The first 112 tab stops per line are recognized in the Printer, and subsequent tab stops are ignored. Tab stop numbers must be received in incremental numerical order.

If a tab stop position of higher value than 80 is received in normal character printing mode, all horizontal tab functions after 80 columns are ignored.

To execute tab stop positions, the HT code should be input. The HT code is ignored when the horizontal tab position has not been programmed.
The NUL code should be input as the command for the termination of the tab set sequence, and the lack of this code will cause incorrect data printout.

1. In case of 5th, 10th and 21st columns.
   [DATA] ESC D <5> H <A> H <15> H NUL ABC HT DEF HT GHI HT JKL
   [PRINT] ABC DEF GHI JKL

2. In case of lack of stop position.
   [DATA] ESC D <5> H <A> H NUL ABC HT DEF HT GHI HT JKL CR LF
   [PRINT] ABCDEF GHI JKL

3. In case of character data transferring over next tab stop.
   [DATA] ESC D <5> H <A> H <15> H NUL ABCDEF HT GHI HT JKL CR LF
   [PRINT] ABCDEF GHI JKL

4. In case of transferring two HT codes at a time.
   [DATA] ESC D <5> H <A> H <15> H NUL ABCD HT SPACE HT EFGH CR LF
   [PRINT] ABCD EFGH

5) ESC E
   The ESC E code causes the Printer to print emphasized characters. Emphasized printing gives the character a stronger impression on the paper.
   
   This code can be input in any column position on a line.
   
   The speed of the head carriage reduces to 40 CPS while printing emphasized characters.

   1. [DATA] ESC E ABCDEFGHI CR LF
      [PRINT] ABCDEFGHI
   2. [DATA] SO ESC E ABCDEFGHI CR LF
      [PRINT] ABCDEFGHI

6) ESC F
   The ESC F code cancels the emphasized printing mode.

7) ESC G
   The ESC G code causes the Printer to perform the double printing. Double printing is carried out in the following manner:
   a) A character is printed.
   b) The paper is advanced by 1/216 inch.
   c) The print head prints the same character again.

   In this way, the character becomes bold.

2-86
8) **ESC H**

The ESC H code cancels the double printing mode.
5 1/4-Inch Diskette Drive Adapter

The System Unit has space and power for one or two 5-1/4" Diskette Drives. The drives are soft sectored, single sided, with 40 tracks. They are Modified Frequency Modulation (MFM) coded in 512 byte sectors, giving a formatted capacity of 163,840 bytes per drive. They have a track to track access time of 8 ms and a motor start time of 500 ms.

The 5-1/4" Diskette Drive Adapter fits in one of the System Board's five System Expansion Slots. It attaches to the two drives via an internal daisy chained flat cable which connects to one end of the drive adapter. The adapter has a second connector on the other end which extends through the rear panel of the System Unit. This connector contains the signals for two additional external drives, thus the 5-1/4" Diskette Drive Adapter is capable of attaching four 5-1/4" drives, two internal, and two external.

The adapter is designed for double density MFM coded drives and uses write precompensation with an analog phase locked loop for clock and data recovery. The adapter is a general purpose device using the NEC µPD765 compatible controller. Thus the drive parameters are programmable. In addition, the attachment supports the drive's write protect feature.

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 is also used to indicate operation complete and status condition requiring processor attention.

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

Figure 19. 5 1/4" 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 a NEC μPD765 or equivalent Floppy Disk Controller (FDC).

In the following description, drives numbers 0-3 are equivalent to drives A-D respectively.

**Digital Output Register (DOR)**

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

**Bits 0 and 1**

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

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

- Drive 0: A
- Drive 1: B
- Drive 2: C
- Drive 3: D

**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 (FDC)**

The following is a brief summary of the registers and commands implemented by the FDC.

The FDC contains two registers which 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 FDD status information. Data bytes are read out of, or written into, the Data
Register in order to program or obtain the 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 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 FDD</td>
<td>FDD A Busy</td>
<td>DAB</td>
<td>FDD number 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/</td>
<td>DIO</td>
<td>Indicates direction of data transfer between FDC and Processor. If DIO = “1”, then transfer is from FDC Data Register to the Processor. If DIO = “0”, then transfer is from the Processor to FDC Data Register. Indicates Data Register is ready to send or receive data to or from the Processor. Both bits DIO and RQM should be used to perform the handshaking functions of “ready” and “direction” to the processor.</td>
</tr>
</tbody>
</table>
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 are made available to the processor.
### Programming Considerations

Table 13. Symbol Descriptions

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>AO 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 which 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 which users are going to read out or write into 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, and 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>
<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 Head.</td>
</tr>
<tr>
<td>SYMBOL</td>
<td>NAME</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</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 which 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 register (DOR) p 2-91</td>
</tr>
</tbody>
</table>
Command Summary

0 indicates 'logical 0' for that bit, 1 means 'logical 1',
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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Command Codes</strong></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>USI</td>
<td>USO</td>
<td><strong>Sector ID information prior to Command execution</strong></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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Data-transfer between the FDD and main-system</strong></td>
</tr>
<tr>
<td>Result</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Status information after Command execution</strong></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>ST0</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>ST1</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>ST2</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><strong>Sector ID information after Command execution</strong></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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Command Codes</strong></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>USI</td>
<td>USO</td>
<td><strong>Sector ID information prior to Command execution</strong></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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Data-transfer between the FDD and main-system</strong></td>
</tr>
<tr>
<td>Result</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Status information after Command execution</strong></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>ST0</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>ST1</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>ST2</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><strong>Sector ID information after Command execution</strong></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>

| Command  | W   |    |    |    |    |    |    |    |    | **Command Codes**                            |
|          | W   | X  | X  | X  | X  | X  | HD | USI| USO| **Sector ID information to command execution** |
|          | W   | C  |    |    |    |    |    |    |    |                                              |
|          | W   | H  |    |    |    |    |    |    |    |                                              |
|          | W   | R  |    |    |    |    |    |    |    |                                              |
|          | W   | N  |    |    |    |    |    |    |    |                                              |
|          | W   | EOT|    |    |    |    |    |    |    |                                              |
|          | W   | GPL|    |    |    |    |    |    |    |                                              |
|          | W   | DTL|    |    |    |    |    |    |    |                                              |
| Execution|     |    |    |    |    |    |    |    |    | **Data-transfer between the main-system and FDD** |
| Result   | R   |    |    |    |    |    |    |    |    | **Status information after command execution** |
|          | R   | ST0|    |    |    |    |    |    |    |                                              |
|          | R   | ST1|    |    |    |    |    |    |    |                                              |
|          | R   | ST2|    |    |    |    |    |    |    |                                              |
|          | R   | C  |    |    |    |    |    |    |    | **Sector ID information after command execution** |
|          | R   | H  |    |    |    |    |    |    |    |                                              |
|          | R   | R  |    |    |    |    |    |    |    |                                              |
|          | R   | N  |    |    |    |    |    |    |    |                                              |
### Command Summary (continued)

<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><strong>Command</strong></td>
<td>W</td>
<td>MT MF 0 0 1 0 0 1</td>
<td><strong>WRITE DELETED DATA</strong></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>X X X X X HD US1 USO</td>
<td>Command Codes</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>C</td>
<td>Sector ID information prior to command execution</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>EOT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>GPL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>DTL</td>
<td></td>
</tr>
<tr>
<td><strong>Execution</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Result</strong></td>
<td>R</td>
<td>ST 0</td>
<td>Data-transfer between FDD and main-system</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>ST 1</td>
<td>Status ID information after common execution</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>ST 2</td>
<td>Sector ID information after command execution</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td><strong>Command</strong></td>
<td>W</td>
<td>0 MF 0 0 1 0 1 0</td>
<td><strong>READ A TRACK</strong></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>X X X X X HD US1 USO</td>
<td>Command Codes</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>C</td>
<td>Sector ID information prior to command execution</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>EOT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>GPL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>DTL</td>
<td></td>
</tr>
<tr>
<td><strong>Execution</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Result</strong></td>
<td>R</td>
<td>ST 0</td>
<td>Data-transfer between the FDD and main-system. FDC reads all of cylinders contents from index hole to EOT.</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>ST 1</td>
<td>Status information after command execution</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>ST 2</td>
<td>Sector ID information after command execution</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td><strong>Command</strong></td>
<td>W</td>
<td>0 MF 0 0 1 0 1 0</td>
<td><strong>READ ID</strong></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>X X X X X HD US1 USO</td>
<td>Command Codes</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td></td>
<td>The first correct ID information on the cylinder is stored in data register.</td>
</tr>
<tr>
<td><strong>Execution</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Result</strong></td>
<td>R</td>
<td>ST 0</td>
<td>Status information after command execution</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>ST 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>ST 2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>PHASE</td>
<td>R/W</td>
<td>D7 D6 D5 D4 D3 D2 D1 D0</td>
<td>DATA BUS</td>
</tr>
<tr>
<td>-------</td>
<td>-----</td>
<td>------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>Command</td>
<td>W</td>
<td>0 MF 0 0 1 1 0 0</td>
<td>FORMAT A TRACK</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>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>SC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>GPL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>Execution</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Result</td>
<td>R</td>
<td>ST 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>ST 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>ST 2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>Command</td>
<td>W</td>
<td>MT MF SK 1 0 0 1</td>
<td>SCAN EQUAL</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>Execution</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Result</td>
<td>R</td>
<td>ST 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>ST 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>ST 2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>Command</td>
<td>W</td>
<td>MT MF SK 1 1 0 0 1</td>
<td>SCAN LOW OR EQUAL</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>
</tbody>
</table>

2-98
## Command Summary (continued)

<table>
<thead>
<tr>
<th>PHASE</th>
<th>R/W</th>
<th>DATA BUS</th>
<th>REMARKS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SCAN HIGH OR EQUAL</strong></td>
<td></td>
<td></td>
<td><strong>Remarks</strong></td>
</tr>
<tr>
<td>Command</td>
<td>W</td>
<td>MT MF SK</td>
<td>W X X X X HD US1 US0</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>C</td>
<td>W H R N</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>EOT</td>
<td>W GPL STP</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td></td>
<td><strong>Remarks</strong></td>
</tr>
<tr>
<td><strong>DATA BUS</strong></td>
<td></td>
<td></td>
<td><strong>Remarks</strong></td>
</tr>
<tr>
<td><strong>SCAN HIGH OR EQUAL</strong></td>
<td></td>
<td></td>
<td><strong>Remarks</strong></td>
</tr>
<tr>
<td>Command</td>
<td>W</td>
<td>MT MF SK</td>
<td>W X X X X HD US1 US0</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>C</td>
<td>W H R N</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>EOT</td>
<td>W GPL STP</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td></td>
<td><strong>Remarks</strong></td>
</tr>
<tr>
<td><strong>RESULTS</strong></td>
<td></td>
<td></td>
<td><strong>Remarks</strong></td>
</tr>
<tr>
<td><strong>SCAN HIGH OR EQUAL</strong></td>
<td></td>
<td></td>
<td><strong>Remarks</strong></td>
</tr>
<tr>
<td>Command</td>
<td>W</td>
<td>MT MF SK</td>
<td>W X X X X HD US1 US0</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>C</td>
<td>W H R N</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>EOT</td>
<td>W GPL STP</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td></td>
<td><strong>Remarks</strong></td>
</tr>
<tr>
<td><strong>EXECUTION</strong></td>
<td></td>
<td></td>
<td><strong>Remarks</strong></td>
</tr>
<tr>
<td><strong>SCAN HIGH OR EQUAL</strong></td>
<td></td>
<td></td>
<td><strong>Remarks</strong></td>
</tr>
<tr>
<td>Command</td>
<td>W</td>
<td>MT MF SK</td>
<td>W X X X X HD US1 US0</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>C</td>
<td>W H R N</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>EOT</td>
<td>W GPL STP</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td></td>
<td><strong>Remarks</strong></td>
</tr>
<tr>
<td><strong>RESULT</strong></td>
<td></td>
<td></td>
<td><strong>Remarks</strong></td>
</tr>
<tr>
<td><strong>SCAN HIGH OR EQUAL</strong></td>
<td></td>
<td></td>
<td><strong>Remarks</strong></td>
</tr>
<tr>
<td>Command</td>
<td>W</td>
<td>MT MF SK</td>
<td>W X X X X HD US1 US0</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>C</td>
<td>W H R N</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>EOT</td>
<td>W GPL STP</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td></td>
<td><strong>Remarks</strong></td>
</tr>
<tr>
<td><strong>RECALIBRATE</strong></td>
<td></td>
<td></td>
<td><strong>Remarks</strong></td>
</tr>
<tr>
<td>Command</td>
<td>W</td>
<td>0 0 0 0 0 1 1 1</td>
<td>Command Codes</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>X X X X X 0</td>
<td>US1 US0</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td></td>
<td>Head retracted to track 0</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td></td>
<td><strong>Remarks</strong></td>
</tr>
<tr>
<td><strong>SENSE INTERRUPT STATUS</strong></td>
<td></td>
<td></td>
<td><strong>Remarks</strong></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></td>
<td>R</td>
<td></td>
<td>ST 0</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td></td>
<td>ST 1</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td></td>
<td>ST 2</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td></td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td></td>
<td>H</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td></td>
<td>R</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td></td>
<td>N</td>
</tr>
<tr>
<td><strong>SPECIFY</strong></td>
<td></td>
<td></td>
<td><strong>Remarks</strong></td>
</tr>
<tr>
<td>Command</td>
<td>W</td>
<td>0 0 0 0 0 0 0 1</td>
<td>Command Codes</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>SRT</td>
<td>HUT</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>HLT</td>
<td>ND</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td></td>
<td><strong>Remarks</strong></td>
</tr>
<tr>
<td><strong>SENSE DRIVE STATUS</strong></td>
<td></td>
<td></td>
<td><strong>Remarks</strong></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></td>
<td>W</td>
<td>X X X X X HD US1 US0</td>
<td>Status information about FDD</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td></td>
<td>ST 3</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td></td>
<td><strong>Remarks</strong></td>
</tr>
<tr>
<td><strong>SEEK</strong></td>
<td></td>
<td></td>
<td><strong>Remarks</strong></td>
</tr>
<tr>
<td>Command</td>
<td>W</td>
<td>0 0 0 0 0 1 1 1</td>
<td>Command Codes</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>X X X X X HD US1 US0</td>
<td>Head is positioned over proper cylinder on diskette</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td>NCN</td>
<td><strong>Remarks</strong></td>
</tr>
<tr>
<td><strong>INVALID</strong></td>
<td></td>
<td></td>
<td><strong>Remarks</strong></td>
</tr>
<tr>
<td>Command</td>
<td>W</td>
<td></td>
<td>Invalid Codes</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td></td>
<td>ST 0=80</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td></td>
<td><strong>Remarks</strong></td>
</tr>
</tbody>
</table>
# Command Status Registers

## Table 14. Status Register 0

<table>
<thead>
<tr>
<th>NO.</th>
<th>NAME</th>
<th>SYMBOL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
</table>
| D7  | Interrupt Code        | IC     | D7 = 0 and D6 = 0
Normal termination of command, (NT), Command was completed and properly executed. |
| D6  |                       |        | 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 which was issued was never started. |
|     |                       |        | D7 = 1 and D6 = 1
Abnormal termination because during command execution the ready signal from FDD changed state. |
<p>| 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   |                                                                               |</p>
<table>
<thead>
<tr>
<th>BIT</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 systems 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>NO</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, this flag is set. During the execution of the Read-a-Cylinder command, if the starting sector cannot be found, 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, 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 Data Field) of Status Register 2 is set.</td>
</tr>
<tr>
<td>NO.</td>
<td>NAME</td>
<td>SYMBOL</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>-----</td>
<td>------</td>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>D7</td>
<td>-</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 which 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 with 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 which 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 with the ND bit, and when the contents of C on the medium are different from that stored in the ID Register, and the content 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>
Table 17. Status Register 3

<table>
<thead>
<tr>
<th>NO.</th>
<th>NAME</th>
<th>SYMBOL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>07</td>
<td>Fault</td>
<td>FT</td>
<td>This bit is used to indicate the status of the Fault signal from the FDD.</td>
</tr>
<tr>
<td>06</td>
<td>Write Protected</td>
<td>WP</td>
<td>This bit is used to indicate the status of the Write Protected signal from the FDD.</td>
</tr>
<tr>
<td>05</td>
<td>Ready</td>
<td>RY</td>
<td>This bit is used to indicate the status of the Ready signal from the FDD.</td>
</tr>
<tr>
<td>04</td>
<td>Track 0</td>
<td>TO</td>
<td>This bit is used to indicate the status of the Track 0 signal from the FDD.</td>
</tr>
<tr>
<td>03</td>
<td>Two Side</td>
<td>TS</td>
<td>This bit is used to indicate the status of the Two Side signal from the FDD.</td>
</tr>
<tr>
<td>02</td>
<td>Head Address</td>
<td>HD</td>
<td>This bit is used to indicate the status of the Head Address signal to the FDD.</td>
</tr>
<tr>
<td>01</td>
<td>Unit Select 1</td>
<td>US 1</td>
<td>This bit is used to indicate the status of the Unit Select 1 signal to the FDD.</td>
</tr>
<tr>
<td>00</td>
<td>Unit Select 0</td>
<td>US 0</td>
<td>This bit is used to indicate the status of the Unit Select 0 signal to the FDD.</td>
</tr>
</tbody>
</table>

**Programming Summary**

**DPC Registers (Ports)**

- **FDC Data Reg**
  - I/O Address 3F5
  - **Bit 0** Drive
    - 00: DR #A
    - 10: DR #C
  - 1 Select
    - 01: DR #B
    - 11: DR #D

- **FDC Main Status Reg**
  - I/O Address 3F4

- **Digital Output Reg**
  - I/O Address 3F2

- 2 Not FDC Reset
- 3 Enable INT & DMA Requests
- 4 Drive A Motor Enable
- 5 Drive B Motor Enable
- 6 Drive C Motor Enable
- 7 Drive D Motor Enable

All bits cleared with channel reset.
Interrupt  6
DMA    2

100 Disk Format
1 Head, 45 cylinders, 8 sectors/TRK, 512 bytes/sector, MFM.

FDC Constants
N: H'02', SC: 08, HUT: F, SRT: C, GPL FORMAT: H'05',
GPL RD/WR: 2A, HLT: 01, (8ms track-track)

Drive Constants
HD Load    35 ms
HD Settle  25 ms
Motor Start 500 ms

Comments
1.  Head loads with drive select, wait HD Load time before RD/WR.
2.  Following access, wait HD Settle time before RD/WR.
3.  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 RD/WR.
4.  Motor must be on for drive to be selected.
5.  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:
MPUL 5.5 Vdc
LPUL 2.7 Vdc
MPDL 0.5 Vdc
LPDL -0.5 Vdc

The following lines are used by this adapter.
+D0-7  (Bidirectional, Load: 174LS; 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 via 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 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 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 com-
pleted 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:

MPUL 5.5 Vdc  
LPUL  2.4 Vdc  
MPDL 0.4 Vdc  
LPDL -0.5 Vdc

All adapter outputs are driven by open-collector gates. The drive(s) must provide termination networks to Vcc (except Motor Enable 1 which has a two kohm resistor to Vcc).

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

Adapter Outputs

-Drive Select A&B (Driver: 7438)
  These two lines are used by drives A&B to degate all drivers to the adapter and receivers from the attachment (except Motor Enable) when the line associated with a drive is not active.

-Motor Enable A&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 not active.

-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 not-active.

-Write Data (Driver: 7438)
  For each not-active to active transition of this line while Write Enable is active, the selected drive causes a flux change to be stored on the disk.

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

<table>
<thead>
<tr>
<th>Input</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-Index</td>
<td>The selected drive supplies one pulse per disk revolution on this line.</td>
</tr>
<tr>
<td>-Write Protect</td>
<td>The selected drive makes this line active if a write protected diskette is mounted in the drive.</td>
</tr>
<tr>
<td>-Track 0</td>
<td>The selected drive makes this line active if the read/write head is over track 0.</td>
</tr>
<tr>
<td>-Read Data</td>
<td>The selected drive supplies a pulse on this line for each flux change encountered on the disk.</td>
</tr>
</tbody>
</table>
5-1/4” Diskette Drive Adapter
Internal Interface Specifications

34 PIN KEYED
EDGE CONNECTOR

NOTE: LANDS 1–33 ARE ON THE BACKSIDE
OF THE BOARD, LANDS 2–34 ARE ON THE
FRONT, OR COMPONENT SIDE.

<table>
<thead>
<tr>
<th>AT STANDARD TTL LEVELS</th>
<th>Land No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ground-Odd Numbers</td>
<td>1-33</td>
</tr>
<tr>
<td>Unused</td>
<td>2,4,6</td>
</tr>
<tr>
<td>Index</td>
<td>8</td>
</tr>
<tr>
<td>Motor Enable A</td>
<td>10</td>
</tr>
<tr>
<td>Drive Select B</td>
<td>12</td>
</tr>
<tr>
<td>Drive Select A</td>
<td>14</td>
</tr>
<tr>
<td>Motor Enable B</td>
<td>16</td>
</tr>
<tr>
<td>Direction (Stepper Motor)</td>
<td>18</td>
</tr>
<tr>
<td>Step Pulse</td>
<td>20</td>
</tr>
<tr>
<td>Write Data</td>
<td>22</td>
</tr>
<tr>
<td>Write Enable</td>
<td>24</td>
</tr>
<tr>
<td>Track 0</td>
<td>26</td>
</tr>
<tr>
<td>Write Protect</td>
<td>28</td>
</tr>
<tr>
<td>Read Data</td>
<td>30</td>
</tr>
<tr>
<td>Select Head 1</td>
<td>32</td>
</tr>
<tr>
<td>Unused</td>
<td>34</td>
</tr>
</tbody>
</table>

IBM 5 1/4” Diskette Drives

5 1/4” Diskette Drive Adapter
5-1/4" Diskette Drive Adapter
External Interface Specifications

**External Interface Specifications**

![Diagram of 37 pin 'D' shell connector](image)

<table>
<thead>
<tr>
<th>AT STANDARD TTL LEVELS</th>
<th>Pin no.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unused</td>
<td>1-5</td>
</tr>
<tr>
<td>index</td>
<td>6</td>
</tr>
<tr>
<td>Motor Enable C</td>
<td>7</td>
</tr>
<tr>
<td>Drive Select D</td>
<td>8</td>
</tr>
<tr>
<td>Drive Select C</td>
<td>9</td>
</tr>
<tr>
<td>Motor Enable D</td>
<td>10</td>
</tr>
<tr>
<td>Direction (Stepper Motor)</td>
<td>11</td>
</tr>
<tr>
<td>Step Pulse</td>
<td>12</td>
</tr>
<tr>
<td>Select Head 1</td>
<td>13</td>
</tr>
<tr>
<td>Write Enable</td>
<td>14</td>
</tr>
<tr>
<td>Track 0</td>
<td>15</td>
</tr>
<tr>
<td>Write Protect</td>
<td>16</td>
</tr>
<tr>
<td>Read Data</td>
<td>17</td>
</tr>
<tr>
<td>Write Data</td>
<td>18</td>
</tr>
<tr>
<td>Ground</td>
<td>20-37</td>
</tr>
</tbody>
</table>

---

**Diagram of 37 pin 'D' shell connector**

---

**5-1/4" Diskette Drive Adapter**
5-1/4" Diskette Drive

The IBM 5-1/4" Diskette Drive is a single sided, double density, 40 track unit. The Diskette Drive has a formatted capacity of 163,840 bytes, and is capable of reading and recording digital data using Modified Frequency Modulation (MFM) methods. User access for diskette loading is provided by way of a slot located at the front of the unit.

The Diskette Drive is fully self-contained and requires no operator intervention during normal operation. The Drive consists of a spindle drive system, a head positioning system, and read/write/erase system.

When the front latch is opened, access is provided for the insertion of a diskette. The diskette is positioned in place by plastic guides, and the front latch. In/out location is ensured when the diskette is inserted until a back stop is encountered.

Closing the front latch activates the cone/clamp system resulting in centering of the diskette and clamping of the diskette to the drive hub. The drive hub is driven at a constant speed of 300 rpm by a servo controlled DC motor. In operation, the magnetic head is loaded into contact with the recording medium whenever the front latch is closed.

The magnetic head is positioned over the desired track by means of a 4-phase stepper motor/band assembly and its associated electronics. This positioner employs a one-step rotation to cause a 1-track linear movement. When a write-protected diskette is inserted into the Drive, the write-protect sensor disables the write electronics of the Drive and an appropriate signal is applied to the interface.

When performing a write operation, a 0.33 mm (0.013-in.) data track is recorded. This track is then tunnel erased to 0.30 mm (0.012 in.).

Data recovery electronics include a low-level read amplifier, differentiator, zero-crossing detector, and digitizing circuits. All data decoding is provided by the adapter card.

The Drive is also supplied with the following sensor systems:

(1) A track 00 switch which senses when the Head/Carriage assembly is positioned at Track 00.

(2) The index sensor, which consists of a LED light source and phototransistor, is positioned such that when an index hole is detected, a signal signal is generated.
The write-protect sensor disables the Diskette Drive electronics whenever a write-protect tab is applied to the diskette. For Interface Information, refer to the Diskette Drive Adapter section.

Diskettes

The IBM 5-1/4" Diskette Drive uses a standard 133.4 mm (5.25 in.) diskette. For programming considerations, single sided, double density soft sectored diskettes are used. 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.
<table>
<thead>
<tr>
<th>Media</th>
<th>Industry-compatible 5¼-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>85.85 mm (3.38 inches)</td>
</tr>
<tr>
<td>Width</td>
<td>149.10 mm (5.87 inches)</td>
</tr>
<tr>
<td>Depth</td>
<td>203.2 mm (8.0 inches)</td>
</tr>
<tr>
<td>Weight</td>
<td>2.04 Kg (4.5 lbs.)</td>
</tr>
<tr>
<td>Temperature</td>
<td></td>
</tr>
<tr>
<td>(Exclusive of Media)</td>
<td></td>
</tr>
<tr>
<td>Operating</td>
<td>10°C to 44°C (50°F to 112°F)</td>
</tr>
<tr>
<td>Non-operating</td>
<td>-40°C to 60°C (-40°F to 140°F)</td>
</tr>
<tr>
<td>Relative Humidity</td>
<td></td>
</tr>
<tr>
<td>(Exclusive of Media)</td>
<td></td>
</tr>
<tr>
<td>Operating</td>
<td>20% to 80% (Non-condensing)</td>
</tr>
<tr>
<td>Non-operating</td>
<td>5% to 95% (Non-condensing)</td>
</tr>
<tr>
<td>Seek Time</td>
<td>8 msec track to track</td>
</tr>
<tr>
<td>Head Setting Time</td>
<td>25 msec (last track addressed)</td>
</tr>
<tr>
<td>Error Rate</td>
<td>1 per 10^9 (recoverable)</td>
</tr>
<tr>
<td></td>
<td>1 per 10^12 (non-recoverable)</td>
</tr>
<tr>
<td></td>
<td>1 per 10^6 (seeks)</td>
</tr>
<tr>
<td>Head Life</td>
<td>20,000 hours (normal use)</td>
</tr>
<tr>
<td>Media Life</td>
<td>3.0 x 10^6 passes per track</td>
</tr>
<tr>
<td>Disk Speed</td>
<td>300 rpm ± 1.5% (long term)</td>
</tr>
<tr>
<td>Instantaneous Speed Variation</td>
<td>± 3.0%</td>
</tr>
<tr>
<td>Start/Stop Time</td>
<td>500 msec (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 dc ± 0.6v 900 ma AVE.</td>
</tr>
<tr>
<td></td>
<td>+5v dc ± 0.25 v, 600 ma AVE.</td>
</tr>
</tbody>
</table>
Memory Expansion Options

Two Memory Expansion Options offered for the IBM Personal Computer are the 32K x 9 and the 64K x 9 Memory Expansion Options. These options plug into any of the five System Expansion slots on the System Board. These options are used to extend system memory beyond 64KB. A maximum of 64KB of memory may be installed on the System Board as modules without using any System Expansion Slots or Expansion Options.

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

The expansion options are designed with 250 ns access 16K x 1 dynamic memory chips. On the 32KB card, 16-pin industry standard parts are used. On the 64KB card, stacked modules are used resulting in a 32K x 1 18-pin module. This allows the 32KB and 64KB to have approximately the same packaging densities.

Both expansion options are parity checked and 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 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 which are not performed on the expansion options but are done once on the System Board and made available in the I/O channel for all devices.

To allow the System to address 32KB and 64KB Memory Expansion Options, refer to the system configuration switch settings page 2-28.

Operating Characteristics

The System Board operates at a frequency of 4.77 Mhz, which results in a clock frequency of 210 ns.

Normally, four clock cycles are required for a bus cycle so that an 840 nsec 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 both cards are:

Access - 250 ns
Cycle - 410 ns

Memory Module Description

Each option contains 18 dynamic memory modules. The 32KB Memory Expansion Option utilizes 16K x 1 bit modules and the 64KB Memory Expansion Options utilizes 32K x 1 bit modules.

Both memory modules require three voltage levels (+5Vdc, -5Vdc, +12Vdc) and 128 refresh cycles every 2 msec. Absolute maximum access times are:

From RAS: 250 ns
From CAS: 165 ns

Table 19. Memory Module Pin Configuration

<table>
<thead>
<tr>
<th>PIN NO.</th>
<th>16K X 1 BIT MODULE (Used on 32KB Card)</th>
<th>32K X 1 BIT MODULE (Used on 64KB Card)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-5V</td>
<td>-5V</td>
</tr>
<tr>
<td>2</td>
<td>Data In **</td>
<td>Data In **</td>
</tr>
<tr>
<td>3</td>
<td>- Write</td>
<td>- Write</td>
</tr>
<tr>
<td>4</td>
<td>- RAS</td>
<td>- RAS 0</td>
</tr>
<tr>
<td>5</td>
<td>A0</td>
<td>- RAS 1</td>
</tr>
<tr>
<td>6</td>
<td>A2</td>
<td>A0</td>
</tr>
<tr>
<td>7</td>
<td>A1</td>
<td>A2</td>
</tr>
<tr>
<td>8</td>
<td>+12V</td>
<td>A1</td>
</tr>
<tr>
<td>9</td>
<td>+5V</td>
<td>+12V</td>
</tr>
<tr>
<td>10</td>
<td>A5</td>
<td>+5V</td>
</tr>
<tr>
<td>11</td>
<td>A4</td>
<td>A5</td>
</tr>
<tr>
<td>12</td>
<td>A3</td>
<td>A4</td>
</tr>
<tr>
<td>13</td>
<td>A6</td>
<td>A3</td>
</tr>
<tr>
<td>14</td>
<td>Data Out **</td>
<td>A6</td>
</tr>
<tr>
<td>15</td>
<td>- CAS</td>
<td>Data Out **</td>
</tr>
<tr>
<td>16</td>
<td>GND</td>
<td>- CAS 1</td>
</tr>
<tr>
<td>17</td>
<td>- *</td>
<td>- CAS 0</td>
</tr>
<tr>
<td>18</td>
<td>- *</td>
<td>GND</td>
</tr>
</tbody>
</table>

* 16K X 1 bit module has only 16 pins.
** Data In and Data Out are tied together (three state bus).
Switch - Configurable Start Address

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

Table 20. DIP Module Start Address

<table>
<thead>
<tr>
<th>NO.</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ON: A19=0; OFF: A19=1</td>
</tr>
<tr>
<td>2</td>
<td>ON: A18=0; OFF: A18=1</td>
</tr>
<tr>
<td>3</td>
<td>ON: A17=0; OFF: A17=1</td>
</tr>
<tr>
<td>4</td>
<td>ON: A16=0; OFF: A16=1</td>
</tr>
<tr>
<td>5</td>
<td>ON: A15=0; OFF: A15=1 *</td>
</tr>
<tr>
<td>6</td>
<td>Not Used</td>
</tr>
<tr>
<td>7</td>
<td>Not Used</td>
</tr>
<tr>
<td>8</td>
<td>Used Only In 64KB RAM Card *</td>
</tr>
</tbody>
</table>

* Switch No. 8 may be set on the 64KB Memory Expansion Option to use only half the memory on the card (i.e., 32KB). If Switch No. 8 is ON, all 64KB is accessible. If Switch No. 8 is OFF, address bit A15 (as set by Switch No. 5) is used to determine which 32KB are accessible and the 64KB option behaves exactly like a 32KB option.
Game Control Adapter

The Game Control Adapter allows the system to attach paddles and joysticks. Up to four paddles or two joysticks may be attached. In addition, four input for switches are provided. Paddle and joystick positions are determined by changing resistive values sent to the adapter. The adapter plus system software converts the present resistive value to a relative paddle or joystick position. On receipt of an output signal, four timing circuits are started. By determining the time required for the circuit to time out (a function of the resistance), the paddle position can be determined. This card could be used as a general purpose I/O card with four analog (resistive) inputs plus four digital input points. This card fits into any of the five System Board I/O slots. The game control interface cable attaches to the rear of the card which protrudes through the rear panel of the System Unit.

Game Control Adapter Block Diagram

![Game Control Adapter Block Diagram](image)

Figure 20. GAME CONTROL ADAPTER BLOCK DIAGRAM

2-117
Functional Description

Address Decode
The select on the Game Control Adapter is generated by two 74LS138's as an address decoder. AEN must be inactive while the address is 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 X'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 via an IN from address X'201'. A trigger button is on each joystick/paddle. These values are seen on data bits 7 through 4 (see Software Interface sub-section). These buttons default to an open state and are read as “1”. When a button is depressed, it is read as “0”. Software should be aware that these buttons are NOT debounced in hardware.

Joystick Positions
The joystick position is indicated by a potentiometer for each coordinate. Each potentiometer has a range from 0 to 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 X'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 via an IN from address X'201' and are seen on data bits 3 through 0.
I/O Channel Description

A9-A0: Address lines 9 through 0 are used to address the Game Control Adapter.

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

IOR, IOW: I/O Read and I/O Write are used when reading from or writing to an adapter (IN, OUT).

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

+5V: Power for the Game Control Adapter.

GND: Common ground.

A19-A10: Unused

MEMR, MEMW: Unused

DACK0-DACK3: Unused

IRQ7-IRQ2: Unused

DRQ3-DRQ1: Unused

ALE, T/C: Unused

CLK, OSC: Unused

I/O CHCK: Unused

I/O CH RDY: Unused

HRQ I/O CH: Unused

RESET DRV: Unused

-5v, +12v, -12v: Unused

Interface Description

The Game Control Adapter has 8 input lines, 4 of which are digital inputs and 4 of which are resistive inputs. The inputs are read with one IN from address x‘201’.

The 4 digital inputs each have a 1K ohm pullup resistor to +5V. With no drive on these inputs, a ‘1’ is read. For a ‘0’ reading, the inputs must be pulled to ground.

The 4 resistive inputs, measured to +5V, will be converted to a digital pulse with a duration proportional to the resistive load, according to the following equation:

\[ \text{Time} = 24.2 \mu\text{sec} + 0.011 \times (r) \mu\text{sec} \]
The user must first begin the conversion by an OUT to address x'201'. An IN from address x'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.

Input from address x'201'

<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>
</table>

Digital Inputs

Resistive Inputs

The typical input to the Game Control Adapter is a set of joysticks or game paddles.

The joysticks will typically have a set of two joysticks (A&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>
</tbody>
</table>
NOTE: POTENTIOMETER FOR X & Y COORDINATES HAS A RANGE OF 0 TO 100KΩ.
BUTTON IS NORMALLY OPEN; CLOSED WHEN DEPRESSED.

Figure 21. JOYSTICK SCHEMATIC
Game Controller Adapter (Analog Input)
Connector Specifications

REAR PANEL

15 PIN “D” SHELL CONNECTOR

AT STANDARD TTL LEVELS

<table>
<thead>
<tr>
<th>Voltage</th>
<th>AMP Pin No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>+5 Volts</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 Volts</td>
<td>8</td>
</tr>
<tr>
<td>+5 Volts</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 Volts</td>
<td>15</td>
</tr>
</tbody>
</table>
Asynchronous Communications Adapter

The Asynchronous Communications Adapter is a 4”H x 5”W card that plugs into a System Expansion Slot. All system control signals and voltage requirements are provided through a 2 x 31 position card edge tab. A jumper module is provided to select either RS-232-C or current loop operation.

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.

Figure (22) is a block diagram of the Asynchronous Communications Adapter.

The heart of the adapter is a INS8250 LSI chip or functional equivalent. The following is a summary of the 8250's key features:

- Adds or Delete Standard Asynchronous Communication Bits (Start, Stop, and Parity) to or from Serial Data Stream.
- Full Double Buffering Eliminates Need for Precise Synchronization.
- Independently Controlled Transmit, Receive, Line Status, and Data Set Interrupts.
- Programmable Baud Rate Generator Allows Division of Any Input Clock by 1 to \(2^{16}-1\) and Generates the Internal 16x Clock.
- 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.
- Fully Programmable Serial-Interface Characteristics
  - 5-, 6-, 7-, or 8-Bit Characters
  - Even, Odd, or No-Parity Bit Generation and Detection
  - 1-, 1 1/2-, or 2-Stop Bit Generation
  - Baud Rate Generation (DC to 9600 Baud)
• False Start Bit Detection.
• Complete Status Reporting Capabilities.
• Line Break Generation and Detection.
• Internal Diagnostic Capabilities.
  - Loopback Controls for Communications Link Fault Isolation.
  - Break, Parity, Overrun, Framing Error Simulation.
• Full Prioritized Interrupt System Controls.

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.

Asynchronous Communications Block Diagram

![Asynchronous Communications Block Diagram](image)

Figure 22. 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 (3F8 to 3FF) and writing data out to the card. Address bit A0, A1 and A2 select the different registers which define the modes of operation. Also, the Divisor Latch Access Bit (Bit 7) of the line control register is used to select certain registers.

I/O Decode for Communications Adapter

Table 21. I/O Decodes (3F8 to 3FF)

<table>
<thead>
<tr>
<th>I/O DECODE</th>
<th>REGISTER SELECTED</th>
<th>DLAB STATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>3F8</td>
<td>TX BUFFER</td>
<td>DLAB=0 (WRITE)</td>
</tr>
<tr>
<td>3F8</td>
<td>RX BUFFER</td>
<td>DLAB=0 (READ)</td>
</tr>
<tr>
<td>3F8</td>
<td>DIVISOR LATCH LSB</td>
<td>DLAB=1</td>
</tr>
<tr>
<td>3F9</td>
<td>DIVISOR LATCH MSB</td>
<td>DLAB=1</td>
</tr>
<tr>
<td>3F9</td>
<td>INTERRUPT ENABLE REGISTER</td>
<td>DLAB=0</td>
</tr>
<tr>
<td>3FA</td>
<td>INTERRUPT IDENTIFICATION REGISTERS</td>
<td></td>
</tr>
<tr>
<td>3FB</td>
<td>LINE CONTROL REGISTER</td>
<td></td>
</tr>
<tr>
<td>3FC</td>
<td>MODEM CONTROL REGISTER</td>
<td></td>
</tr>
<tr>
<td>3FD</td>
<td>LINE STATUS REGISTER</td>
<td></td>
</tr>
<tr>
<td>3FE</td>
<td>MODEM STATUS REGISTER</td>
<td></td>
</tr>
</tbody>
</table>

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

One interrupt line is provided to the system. This interrupt is IRQ4 and will be positive active. To allow the communications card to send interrupts to the system, Bit 3 of the Modem Control Register must be set = 0 (low). At this point, any interrupts allowed by the Interrupt Enable Register will cause an interrupt.

The data format will be as follows:

**TRANSMITTER OUTPUT AND RECEIVER INPUT**

<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></td>
<td></td>
<td></td>
<td></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-232-C 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 Corporation that use this particular type of interface.

Pin 18 + receive current loop data (20Ma)
Pin 25 - receive current loop return (20Ma)
Pin 9 + transmit current loop return (20Ma)
Pin 11 - transmit current loop data (20Ma)
The voltage interface is a serial interface. It supports certain data and control signals as listed below.

- Pin 2: Transmit Data
- Pin 3: Receive 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 Indicate

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

+15V — — — — — — — — — — — — — — — — — — — —
On Function

+3V — — — — — — — — — — — — — — — — — — — —
0V
-3V — — — — — — — — — — — — — — — — — — — —
Off Function

-15V — — — — — — — — — — — — — — — — — — — —
Invalid Levels

The signal will be considered in the “marking” condition when the voltage on the interchange circuit, measured at the interface point, is more negative than minus three volts with respect to signal ground. The signal will be considered in the “spacing” condition when the voltage is more positive than plus three volts with respect to signal ground. The region between plus three volts and minus three volts is defined as the transition region, will be considered in invalid levels. The voltage which is more negative than -15V or more positive than +15V will be considered in invalid levels.

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

For interface control circuits, the function is “on” when the voltage is more positive than +3V with respect to signal ground and is “off” when the voltage is more negative than -3V 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 logic 0 (0 volt nominal) and a high represents a logic 1 (+2.4 volts 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 communication between the INS8250 and the CPU.

Data Input Strobe (DISTR, DISTR) Pins 22 and 21: When DISTR is high or DISTR is low while the chip is selected, allows the CPU 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 CPU 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 DPSTR 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 (SOC, 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, 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 into as indicated in the table below. Note that the state of the Divisor Latch Access Bit (DLAB), which is the most significant bit of the Line Control Register, affects the selection of certain INS8250 registers. The DLAB must be set high by the system software to access the Baud Generator Divisor Latches.

<table>
<thead>
<tr>
<th>DLAB</th>
<th>A2</th>
<th>A1</th>
<th>A0</th>
<th>Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Receiver Buffer (read), Transmitter Holding Register (write)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Interrupt Enable</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Interrupt Identification (read only)</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Line Control</td>
</tr>
<tr>
<td>X</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>MODEM Control</td>
</tr>
<tr>
<td>X</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Line Status</td>
</tr>
<tr>
<td>X</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>MODEM Status</td>
</tr>
<tr>
<td>X</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>None</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Divisor Latch (least significant byte)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Divisor Latch (most significant byte)</td>
</tr>
</tbody>
</table>

Master Reset (MR), Pin 35: When high, clears all the registers (except the Receiver Buffer, Transmitter Holding, and Divisor Latches), and the control logic of the INS8250. Also, the state of various output signals (SOUT, INTRPT, OUT 1, OUT 2, RTS, DTR) are affected by an active MR input. (Refer to Table 1.)

Receiver Clock (RCLK), Pin 9: This input is the 16x 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 CPU 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 CPU 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 state 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 has been detected by the MODEM or data set. The RLSD signal is a MODEM-Control function input whose condition can be tested by the CPU 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 CPU 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 a 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 Interrupt is enabled.

VCC, Pin 40: +5 volt supply.

VSS, Pin 20: Ground (0-volt) reference.
Output Signals

Data Terminal Ready (DTR), Pin 33: When low, informs the MODEM or data set that the INS8250 is ready to communicate. The DTR output signal can be set to an active low by programming Bit 0 (DTR) of the MODEM Control Register to a high level. The DTR signal is set high upon a Master Reset operation.

Request to Send (RTS), Pin 32: When low, informs the MODEM or data set that the INS8250 is ready to transmit data. The RTS output signal can be set to an active low by programming Bit 1 (RTS) of the MODEM Control Register. The RTS signal is set high upon a Master Reset operation.

Output 1 (OUT 1), Pin 34: User-designated output that can be set to an active low by programming Bit 2 (OUT 1) of the MODEM Control Register to a high level. The OUT 1 signal is set high upon a Master Reset operation.

Output 2 (OUT 2), Pin 31: User-designated output that can be set to an active low by programming Bit 3 (OUT 2) of the MODEM Control Register to a high level. The OUT 2 signal is set high upon a Master Reset operation.

Chip Select Out (CSOUT), Pin 24: When high, indicates that the chip has been selected by active CS0, CS1, and CS2 inputs. No data transfer can be initiated until the CSOUT signal is a logic 1.

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

Baud Out (BAUDOUT), Pin 15: 16x 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 via the IER: Receiver Error Flag; Received Data Available; Transmitter Holding Register Empty; and 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 (Logic 1) state upon a Master Reset operation.

2-132
Input/Output Signals

Data (D7-D0) Bus, Pins 1-8: This bus comprises eight TRI-STATE input/output lines. The bus provides bidirectional communications between the INS8250 and the CPU. Data, control words, and status information are transferred via 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

Table 22. Asynchronous Communications Reset Functions

<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 &amp; 6 are High</td>
</tr>
<tr>
<td>MODEM Status Register</td>
<td>Master Reset</td>
<td>Bits 0–3 Low, Bits 4–7 High</td>
</tr>
<tr>
<td>SOUT</td>
<td>Master Reset</td>
<td>High</td>
</tr>
<tr>
<td>INTRPT (RCVR Errs)</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>
INS8250 Accessible Registers

The system programmer may access or control any of the INS8250 registers via the CPU. These registers are used to control INS8250 operations and to transmit and receive data.

INS8250 Line Control Register

The system programmer specifies the format of the asynchronous data communications exchange via 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)

3FB

<table>
<thead>
<tr>
<th>BIT 7 6 5 4 3 2 1 0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td>I' I</td>
</tr>
<tr>
<td>Word Length SoI,01 Bit D</td>
</tr>
<tr>
<td>IW</td>
</tr>
<tr>
<td>LSD</td>
</tr>
<tr>
<td>Number of Stop Bits (STB)</td>
</tr>
<tr>
<td>Parity Enable (PEN)</td>
</tr>
<tr>
<td>Even Parity Select (EPS)</td>
</tr>
<tr>
<td>Stick Parity</td>
</tr>
<tr>
<td>Set Break</td>
</tr>
<tr>
<td>Divisor Latch Access Bit (DLAB)</td>
</tr>
</tbody>
</table>

Bit 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 logic 0, 1 Stop bit is generated or checked in the transmit or receive data, respectively. If bit 2 is logic 1 when a 5-bit word length is selected via bits 0 and 1, 1-1/2 Stop bits are generated or checked. If bit 2 is logic 1 when either a 6-, 7-, or 8-bit word length is selected, 2 Stop bits are generated or checked.
Bit 3: This bit is the Parity Enable bit. When bit 3 is a logic 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 logic 1 and bit 4 is a logic 0, an odd number of logic 1’s is transmitted or checked in the data word bits and Parity bit. When bit 3 is a logic 1 and bit 4 is a logic 1, an even number of bits is transmitted or checked.

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

Bit 6: This bit is the Set Break Control bit. When bit 6 is a logic 1, the serial output (SOUT) is forced to the Spacing (logic 0) state and remains there regardless of other transmitter activity. The set break is disabled by setting bit 6 to a logic 0. This feature enables the CPU 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 (logic 1) to access the Divisor Latches of the Baud Rate Generator during a Read or Write operation. It must be set low (logic 0) to access the Receiver Buffer, the Transmitter Holding Register, or the Interrupt Enable Register.

**INS8250 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 16x the Baud rate \([\text{divisor} = \frac{\text{frequency input}}{\text{baud rate} \times 16}]\). Two 8-bit latches store the divisor in a 16-bit binary format. These Divisor Latches must be loaded during initialization in order to ensure desired operation of the Baud Rate Generator. Upon loading either of the Divisor Latches, a 16-bit Baud counter is immediately loaded. This prevents long counts on initial load.
Table 23 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 23. BAUD RATE AT 1.843 Mhz

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

Line Status Register

This 8-bit register provides status information to the CPU concerning the data transfer. The contents of the Line Status Register are indicated and described below.

Line Status Register (LSR)

3FD

<table>
<thead>
<tr>
<th>BIT</th>
<th>DATA READY (DR)</th>
<th>OVERRUN ERROR (OR)</th>
<th>PARITY ERROR (PE)</th>
<th>FRAMING ERROR (FE)</th>
<th>BREAK INTERRUPT (BI)</th>
<th>TRANSMITTER HOLDING REGISTER EMPTY (THRE)</th>
<th>TX SHIFT REGISTER EMPTY (TSRE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>= 0</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Bit 0: This bit is the receiver Data Ready (DR) indicator. Bit 0 is set to a logic 1 whenever a complete incoming character has been received and transferred into the Receiver Buffer Register. Bit 0 may be reset to a logic 0 either by the CPU reading the data in the Receiver Buffer Register or by writing a logic 0 into it from the CPU.

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 CPU before the next character was transferred into the Receiver Buffer Register, thereby destroying the previous character. The OE indicator is reset whenever the CPU 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 logic 1 upon detection of a parity error and is reset to a logic 0 whenever the CPU 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 logic 1 whenever the Stop bit following the last data bit or parity bit is detected as a zero bit (Spacing level).

Bit 4: This bit is the Break Interrupt (BI) indicator. Bit 4 is set to a logic 1 whenever the received data input is held in the Spacing (logic 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 CPU when the Transmit Holding Register Empty Interrupt enable is set high. The THRE bit is set to a logic 1 when a character is transferred from the Transmitter Holding Register into the Transmitter Shift Register. The bit is reset to logic 0 concurrently with the loading of the Transmitter Holding Register by the CPU.

Bit 6: This bit is the Transmitter Shift Register Empty (TSRE) indicator. Bit 6 is set to a logic 1 whenever the Transmitter Shift Register is idle. It is reset to logic 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 logic 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. The four levels of interrupt conditions are as follows: 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 that interrupt are stored in the Interrupt Identification Register (refer to Table 5). 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 CPU. The contents of the IIR are indicated and described below.

Interrupt Identification Register (IIR)

<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>0 IF INTERRUPT PENDING</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INTERRUPT ID BIT (0)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INTERRUPT ID BIT (1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>= 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>= 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>= 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>= 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>= 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bit 0: This bit can be used in either a hardwired prioritized or polled environment to indicate whether an interrupt is pending. When bit 0 is a logic 0, 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 logic 1, no interrupt is pending and polling (if used) continued.

Bits 1 and 2: These two bits of the IIR are used to identify the highest priority interrupt pending as indicated in Table 5.

Bits 3 through 7: These five bits of the IIR are always logic 0.
Table 24. Interrupt Control Functions

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

This 8-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 logic 1, selected interrupts can be enabled. Disabling the interrupt system inhibits the Interrupt Identification Register and the active (high) INTRPT output from the chip. All other system functions operate in their normal manner, including the setting of the Line Status and MODEM Status Registers. The contents of the Interrupt Enable Register are indicated and described below.

Interrupt Enable Register (IER)

3F9  DLAB=0

<table>
<thead>
<tr>
<th>BIT</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1= ENABLE DATA AVAILABLE INTERRUPT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1= ENABLE TX HOLDING REG EMPTY INTERRUPT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1= ENABLE RECEIVE LINE STATUS INTERRUPT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1= ENABLE MODEM STATUS INTERRUPT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>= 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>= 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>= 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>= 0</td>
</tr>
</tbody>
</table>

Bit 0: This bit enables the Received Data Available Interrupt when set to logic 1.

Bit 1: This bit enables the Transmitter Holding Register Empty Interrupt when set to logic 1.

Bit 2: This bit enables the Receiver Line Status Interrupt when set to logic 1.

Bit 3: This bit enables the MODEM Status Interrupt when set to logic 1.

Bits 4 through 7: These four bits are alway logic 0.
MODEM Control Register

This 8-bit register controls the interface with the MODEM or data set (or a peripheral device emulating a MODEM). The contents of the MODEM Control Register are indicated and described below.

MODEM Control Register (MCR)

<table>
<thead>
<tr>
<th>BIT</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **DATA TERMINAL READY (DTR)**
- **REQUEST TO SEND (RTS)**
- **OUT 1**
- **OUT 2**
- **LOOP**

**Bit 0:** This bit controls the Data Terminal Ready (DTR) output. When bit 0 is set to a logic 1, the DTR output is forced to a logic 0. When bit 0 is reset to a logic 0, the DTR output is forced to a logic 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 logic 1, the following occur: the transmitter Serial Output (SOUT) is set to the Marking (logic 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, DSR, 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.

2-142
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 logic 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 logic 0.

**Bits 5 through 7:** These bits are permanently set to logic 0.

**MODEM Status Register**

This 8-bit register provides the current state of the control lines from the MODEM (or peripheral device) to the CPU. In addition to this current-state information, four bits of the MODEM Status Register provide change information. These bits are set to a logic 1 whenever a control input from the MODEM changes state. They are reset to logic 0 whenever the CPU reads the MODEM Status Register.

The content of the MODEM Status Register are indicated and described below.

**MODEM Status Register (MSR)**

<table>
<thead>
<tr>
<th>BIT</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DELTA CLEAR TO SEND (DCTS)</td>
<td>DELTA DATA SET READY (DDSR)</td>
<td>TRAILING EDGE RING INDICATOR (TERI)</td>
<td>DELTA RX LINE SIGNAL DETECT (DRLSD)</td>
<td>CLEAR TO SEND (CTS)</td>
<td>DATA SET READY (DSR)</td>
<td>RING INDICATOR (RI)</td>
<td>RECEIVE LINE SIGNAL DETECT (RLSD)</td>
</tr>
</tbody>
</table>

3FE
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 CPU.

Bit 1: This bit is the Delta Data Set Ready (DDSR) indicator. Bit 1 indicates that the DSR input to the chip has changed state since the last time it was read by the CPU.

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 (logic 1) to an Off (logic 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 logic 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 1, this bit 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 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 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 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.

**Receiver Buffer Register (RBR)**

3F8 DLAB=0 READ ONLY

![Diagram of Receiver Buffer Register]

- 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:

Transmitter Holding Register (THR)

3F8  DLAB=0  WRITE ONLY

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

The Voltage or Current loop interface is selected by plugging the programmed shunt module, with the locator dot up or down. See the figure below for the two configurations.

![Diagram of selecting the interface format](image)

Figure 23. SELECTING THE INTERFACE FORMAT
Asynchronous Communications Adapter Connector Interface Specifications

### AT STANDARD TTL LEVELS

<table>
<thead>
<tr>
<th>Description</th>
<th>Pin</th>
</tr>
</thead>
<tbody>
<tr>
<td>NC</td>
<td>1</td>
</tr>
<tr>
<td>Transmit Data</td>
<td>2</td>
</tr>
<tr>
<td>Receive 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>Carrier detect</td>
<td>8</td>
</tr>
<tr>
<td>+Transmit current loop return (20 ma)</td>
<td>9</td>
</tr>
<tr>
<td>NC</td>
<td>10</td>
</tr>
<tr>
<td>-Transmit current loop data (20 ma)</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 (20 ma)</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 Indicate</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 (20 ma)</td>
<td>25</td>
</tr>
</tbody>
</table>

**NOTE:** To avoid inducing voltage surges on interchange circuits, signals from interchange circuits shall not be used to drive inductive devices, such as relay coils.
SECTION 3. ROM and SYSTEM USAGE

Contents:

ROM BIOS ......................................................... 3-2
BIOS Cassette Logic ................................. 3-8
Keyboard Encoding and Usage .................. 3-11
Low Memory Maps ................................. 3-21
ROM BIOS

The ROM resident Basic I/O System (BIOS) provides the device level control of the major I/O devices in the System Unit. The BIOS routines allow the assembly language programmer to perform block (diskette and cassette) or character (Video, communications, keyboard and printer) level I/O operations without any concern for device address and operating characteristics. Additionally, system services such as time of day and memory size determination are provided. The goal is to provide an operational interface to the system and relieve the programmer from concern over hardware device characteristics. Finally the BIOS interface insulates the user from the hardware allowing new devices to be added to the System Unit, yet retaining the BIOS level interface to the device. In this manner, user programs become transparent to hardware modifications and enhancements. A complete listing of the BIOS is provided in Appendix “A”.

Use of BIOS

Access to the BIOS function is through the 8088 software interrupts. Each BIOS entry point is available through its own interrupt, which can be found in the interrupt vector listing. The software interrupts 10H through 1AH each access a different BIOS routine. For example, to determine the amount of memory available in the system,

```
INT 12H
```

will invoke the memory size determination routine in BIOS and return the value to the caller.

Parameter Passing

All parameters passed to and from the BIOS routines go through the 8088 registers. The prologue of each BIOS function indicate the registers used on the call and the return. For the memory size example above, no parameters are passed, and the result, memory size in 1K Byte increments is returned in the AX register.

Where a BIOS function has several possible operations, the AH register is used on input to indicate the desired operation. For example, to set the time of day, the following code is required.
MOV AH, 1 ;function is to set time of day.
MOV CX, HIGH_COUNT ;establish the current time.
MOV DX, Low_COUNT 
INT 1AH ;Set the time.

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

As a general rule, the BIOS routines preserve all registers except for AX and the flags. Other registers are modified on return only if they are returning a value to the caller. The exact register usage can be seen in the prologue of each BIOS function.

Interrupt Vector Listing

<table>
<thead>
<tr>
<th>Interrupt Number</th>
<th>Name</th>
<th>BIOS Initialization</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Divide by Zero</td>
<td>None</td>
</tr>
<tr>
<td>1</td>
<td>Single Step</td>
<td>None</td>
</tr>
<tr>
<td>2</td>
<td>Non Maskable</td>
<td>NMI_INT (F000:E2C3)</td>
</tr>
<tr>
<td>3</td>
<td>Breakpoint</td>
<td>None</td>
</tr>
<tr>
<td>4</td>
<td>Overflow</td>
<td>None</td>
</tr>
<tr>
<td>5</td>
<td>Print Screen</td>
<td>PRINT_SCREEN (F000:FF54)</td>
</tr>
<tr>
<td>6</td>
<td>Unused</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Time of Day</td>
<td>TIMER_INT (F000:FEA5)</td>
</tr>
<tr>
<td>8</td>
<td>Keyboard</td>
<td>KB_INT (F000:E987)</td>
</tr>
<tr>
<td>9</td>
<td>Unused</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>Unused (Reserved Communications)</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>Diskette</td>
<td>Disk_INT (F000:EF57)</td>
</tr>
<tr>
<td>C</td>
<td>Unused</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>Unused</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>Diskette</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>Unused (Reserved Printer)</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Video</td>
<td>VIDEO_I O (F000:F065)</td>
</tr>
<tr>
<td>11</td>
<td>Equipment Check</td>
<td>EQUIPMENT (F000:F840)</td>
</tr>
<tr>
<td>12</td>
<td>Memory</td>
<td>MEMORY_SIZE DETERMINE (F000:F841)</td>
</tr>
<tr>
<td>13</td>
<td>Diskette</td>
<td>DISKETTE_I O (F000:EC59)</td>
</tr>
<tr>
<td>14</td>
<td>Communications</td>
<td>RS232_I O (F000:E739)</td>
</tr>
<tr>
<td>15</td>
<td>Cassette</td>
<td>CASSETTE_I O (F000:F859)</td>
</tr>
<tr>
<td>16</td>
<td>Keyboard</td>
<td>KEYBOARD_I O (F000:E82E)</td>
</tr>
<tr>
<td>17</td>
<td>Printer</td>
<td>PRINTER_I O (F000:EF02)</td>
</tr>
<tr>
<td>18</td>
<td>Cassette BASIC</td>
<td>F600-00000</td>
</tr>
<tr>
<td>19</td>
<td>Keyboard</td>
<td>BOOT Strap (F000:E6F2)</td>
</tr>
<tr>
<td>1A</td>
<td>Time of Day</td>
<td>TIME_OF_DAY (F000:FE6E)</td>
</tr>
<tr>
<td>1B</td>
<td>User Supplied</td>
<td>DUMMY_RETURN (F000:FF53)</td>
</tr>
<tr>
<td>1C</td>
<td>Routines</td>
<td>DUMMY_RETURN (F000:FF53)</td>
</tr>
<tr>
<td>1D</td>
<td>BIOS Parameters</td>
<td>VIDEO_PARMS (F000:F04A4)</td>
</tr>
<tr>
<td>1E</td>
<td>Parameters</td>
<td>DISK_BASE (F000:EF07)</td>
</tr>
<tr>
<td>1F</td>
<td>Video Initialization</td>
<td>None</td>
</tr>
</tbody>
</table>
Vectors With Special Meanings

Interrupt 1BH - Keyboard Break Address
This vector points to the code to be exercised when the CTRL BREAK keys are depressed on the keyboard. The vector is invoked while responding to the keyboard interrupt, and control should be returned via an IRET instruction. The power on routines initialize this vector to point to an IRET instruction, so that nothing happens when CTRL BREAK keys are depressed 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 set to the 8259 controller. Also, all I/O devices should be reset in case an operation was underway at that time.

Interrupt 1CH - Timer Tick
This vector points to the code to be executed on every tick of the system clock. This vector is invoked while responding to the timer interrupt, and control should be returned via an IRET instruction. The power on routines initialize this vector to point to an IRET instruction, so that nothing happens unless the application modifies the pointer. It is the responsibility of the application to save and restore all registers that will be modified.

Interrupt 1DH - 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 routine.

Interrupt 1EH - 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 to reflect the specifications of the other drives attached may be necessary.
Interrupt 1FH - Graphics Character Extensions

When operating in the graphics modes of the Color/Graphics Monitor Adapter (320 x 200 or 640 x 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 other 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 8 bytes of graphic information. At power on this vector is initialized to 0:0, and it is the responsibility of the user to change this vector if the additional code points are required.

Other Read/Write Memory Usage

The IBM ROM BIOS routines use 256 bytes of memory starting at absolute 400 to 4FF. Locations 400-407 contain the base addresses of any RS232 cards attached to the system, 0's if none attached. These locations, in order, represent the 0 to 3 values used as the parameter to the RS232 BIOS routine. Locations 408-40F provide the same function, but for the PRINTER.

Memory locations 300-3FF are used as a stack area during the power on initialization, and the bootstrap, when control passed to it from power on. If the user desires the stack in a different area, it must be set by the application.

Note: Use the Interrupt Vector Listing as an aid to locate these topics in the ROM BIOS listing, Appendix "A".

BIOS Programming Tip

When programming with BIOS you should keep in mind that if an error is reported by the diskette code, to reset the diskette adapter and retry the operation. A specified number of retries should be required on reads to ensure the problem is not due to motor start-up.
### BIOS Memory Map

<table>
<thead>
<tr>
<th>Starting Address Hex</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>F4000</td>
<td>User Read Only Memory</td>
</tr>
<tr>
<td>F6000</td>
<td>Cassette BASIC Interpreter</td>
</tr>
<tr>
<td>FE000</td>
<td>BIOS Program Area</td>
</tr>
</tbody>
</table>

**Figure 24. BIOS MEMORY MAP**
BIOS Cassette Logic
Software Algorithms

Interrupt 15

The cassette routine will be called with the request type in AH and the address of the bytes to be read or written will be specified by (ES):(BX) and the number of bytes to read/write will be specified by (CX). The actual number of bytes read will be returned in (DX). Read block and write block will automatically turn the motor on at the start and off at the end. The requests are as follows:

- (AH) = 0: Turn the cassette motor on.
- (AH) = 1: Turn the cassette motor off.
- (AH) = 2: (Read Block) Read (CX) bytes into memory beginning at address (ES):(BX) and return actual number of bytes read in (DX). Return the cassette status in (AH).
- (AH) = 3: (Write Block) Write (CX) bytes onto the cassette beginning at address (DS):(BX). Return the cassette status in (AH).

STATUS:

- AH = 00: No errors
- AH = 01: CRC-Error (Read Block)
- AH = 02: No data transitions
- AH = 04: No leader
- AH = 80: Invalid command

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

Cassette Write

The WRITE BLOCK routine writes a tape block on the cassette. The tape block is described in Data Record Architecture page (3-10).

The WRITE BLOCK routine turns on the cassette motor and a synchronization bit (0) and then writes 256 bytes of all ones, the leader, to the tape. Next, one or more data blocks are written (depends on number in CX). After each data block of 256 bytes, a two byte CRC is written. The data bytes are taken from the memory location pointed at by ES.

The WRITE BYTE routine disassembles the byte and writes it 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 one bit and 0.5 millisecond for a zero bit.
The timer is set mode 3 which means it will output a square wave with period given by its count register. The timer’s period is changed on the fly for each data bit to be written to the cassette. If the number of data bytes to be written is not an integral multiple off 256, then after the last desired data byte from memory has been written, the data block will be extended to 256 bytes by writing multiples of the last data byte. The last block will be closed with two CRC bytes as usual. After the last data block, a trailer consisting of four bytes of all one bits will be written. Finally, the motor will be turned off. There are no errors reported by this routine.

---

**Cassette Read**

The READ BLOCK routine turns on the cassette motor and then delays for approximately 0.5 secs for it to come up to speed.

The READ BLOCK routine then searches for leader and must detect all one bits for approximately 1/4 of leader length before it can look for the sync byte. If a correct sync byte (X ‘16’) is not found, the routine goes back and searches for leader again.

The data is read a bit a time and assembled into bytes. After each byte is assembled it is written into memory at location ES:BX and then BX is incremented by one.

After each multiple of 256 data bytes are 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 status (AH) - 01 for CRC error. DX will contain the number of bytes written into memory.

**Note:** The Time of Day Interrupt (IRQ0) is disabled during the cassette read operation.
Data Record Architecture

1. Leader 256 bytes (of ones)
2. Sync byte ASCII Sync Char (X‘16’)
3. Sync byte (X ‘16’)
4. Data Blocks 256 bytes
5. CRC — 2 bytes — for each data block

Error Recovery

Error recovery is handled by software. A cyclic redundancy check (CRC) is used to detect errors. The polynomial used is:

\[ G(X) = X^{16} \ll X^{12} \ll X^5 \ll 1 \]

Which is the polynomial used by the SDLC interface. Essentially, as bits are written/read from 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 tape. On 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 one, the processor’s carry flag is set and status (AH) is set to X’01’ to indicate a CRC error has occurred. Also, the routine is exited on CRC error.
Keyboard Encoding and Usage

Encoding

The keyboard routine provided by IBM in 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 0-255, an extended code for certain extended keyboard functions and functions that are 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 exact codes. Use keyboard Scan Code diagram for reference page 2-17.

Table 25. Character Codes

<table>
<thead>
<tr>
<th>KEY #</th>
<th>BASE CASE</th>
<th>UPPER CASE</th>
<th>CTRL</th>
<th>ALT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ESC</td>
<td>ESC</td>
<td>ESC</td>
<td>-1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>÷</td>
<td></td>
<td>-1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>@</td>
<td>NUL (000) Note 1</td>
<td>Note 1</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>#</td>
<td></td>
<td>-1</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>$</td>
<td></td>
<td>-1</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>%</td>
<td></td>
<td>-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>Backspace (008)</td>
<td>Backspace (008)</td>
<td>DEL(127)</td>
<td>-1</td>
</tr>
<tr>
<td>14</td>
<td>Backspace (009)</td>
<td>Backspace (009)</td>
<td></td>
<td>-1</td>
</tr>
<tr>
<td>15</td>
<td>q</td>
<td>Q</td>
<td>DC1 (017)</td>
<td>Note 1</td>
</tr>
<tr>
<td>16</td>
<td>w</td>
<td>W</td>
<td>ETB (023)</td>
<td>Note 1</td>
</tr>
<tr>
<td>17</td>
<td>e</td>
<td>E</td>
<td>ENQ (005)</td>
<td>Note 1</td>
</tr>
<tr>
<td>18</td>
<td>r</td>
<td>R</td>
<td>DC2 (018)</td>
<td>Note 1</td>
</tr>
<tr>
<td>19</td>
<td>t</td>
<td>T</td>
<td>DC4 (020)</td>
<td>Note 1</td>
</tr>
<tr>
<td>20</td>
<td>y</td>
<td>Y</td>
<td>EM (025)</td>
<td>Note 1</td>
</tr>
<tr>
<td>21</td>
<td>u</td>
<td>U</td>
<td>NAK (021)</td>
<td>Note 1</td>
</tr>
<tr>
<td>22</td>
<td>i</td>
<td>I</td>
<td>HT (009)</td>
<td>Note 1</td>
</tr>
<tr>
<td>23</td>
<td>o</td>
<td>O</td>
<td>SI (015)</td>
<td>Note 1</td>
</tr>
<tr>
<td>24</td>
<td>p</td>
<td>P</td>
<td>DLE (016)</td>
<td>Note 1</td>
</tr>
<tr>
<td>25</td>
<td>[</td>
<td>]</td>
<td>ESC (027)</td>
<td>-1</td>
</tr>
<tr>
<td>26</td>
<td>]</td>
<td>[</td>
<td>GS (029)</td>
<td>-1</td>
</tr>
<tr>
<td>27</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 25. Character Codes (cont.)

<table>
<thead>
<tr>
<th>KEY #</th>
<th>BASE CASE</th>
<th>UPPER CASE</th>
<th>CTRL</th>
<th>ALT</th>
</tr>
</thead>
<tbody>
<tr>
<td>28</td>
<td>CR</td>
<td>CR</td>
<td>LF (010)</td>
<td>-1</td>
</tr>
<tr>
<td>29</td>
<td>CTRL</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>I</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</td>
<td>SHIFT</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</td>
<td>SHIFT</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</td>
<td>ALT</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>57</td>
<td>SP</td>
<td>SP</td>
<td>SP</td>
<td>SP</td>
</tr>
<tr>
<td>58</td>
<td>CAPS</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>LOCK</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>59</td>
<td>NUL (Note 1)</td>
<td>NUL (Note 1)</td>
<td>NUL (Note 1)</td>
<td>NUL (Note 1)</td>
</tr>
<tr>
<td>60</td>
<td>NUL (Note 1)</td>
<td>NUL (Note 1)</td>
<td>NUL (Note 1)</td>
<td>NUL (Note 1)</td>
</tr>
<tr>
<td>61</td>
<td>NUL (Note 1)</td>
<td>NUL (Note 1)</td>
<td>NUL (Note 1)</td>
<td>NUL (Note 1)</td>
</tr>
<tr>
<td>62</td>
<td>NUL (Note 1)</td>
<td>NUL (Note 1)</td>
<td>NUL (Note 1)</td>
<td>NUL (Note 1)</td>
</tr>
<tr>
<td>63</td>
<td>NUL (Note 1)</td>
<td>NUL (Note 1)</td>
<td>NUL (Note 1)</td>
<td>NUL (Note 1)</td>
</tr>
<tr>
<td>64</td>
<td>NUL (Note 1)</td>
<td>NUL (Note 1)</td>
<td>NUL (Note 1)</td>
<td>NUL (Note 1)</td>
</tr>
<tr>
<td>65</td>
<td>NUL (Note 1)</td>
<td>NUL (Note 1)</td>
<td>NUL (Note 1)</td>
<td>NUL (Note 1)</td>
</tr>
<tr>
<td>66</td>
<td>NUL (Note 1)</td>
<td>NUL (Note 1)</td>
<td>NUL (Note 1)</td>
<td>NUL (Note 1)</td>
</tr>
<tr>
<td>67</td>
<td>NUL (Note 1)</td>
<td>NUL (Note 1)</td>
<td>NUL (Note 1)</td>
<td>NUL (Note 1)</td>
</tr>
<tr>
<td>68</td>
<td>NUL (Note 1)</td>
<td>NUL (Note 1)</td>
<td>NUL (Note 1)</td>
<td>NUL (Note 1)</td>
</tr>
<tr>
<td>69</td>
<td>NUM</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>LOCK</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>SCROLL</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>LOCK</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note 1: Refer to Extended Codes Page (3-13).

Note 2: Refer to Special Handling Page (3-15).
Keys 71-83 have meaning only in base case, in NUMLOCK (or shifted) states, or in CTRL state. It should be noted that the shift key temporarily reverses the current NUMLOCK state.

<table>
<thead>
<tr>
<th>KEY #</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>Note 1</td>
<td>Clear Screen</td>
</tr>
<tr>
<td>72</td>
<td>8</td>
<td>(Note 1)</td>
<td>Note 1</td>
<td>-1</td>
</tr>
<tr>
<td>73</td>
<td>9</td>
<td>PageUp (Note 1)</td>
<td>Note 1</td>
<td>Top of Text &amp; Home</td>
</tr>
<tr>
<td>74</td>
<td>-</td>
<td>-</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>75</td>
<td>4</td>
<td>(Note 1)</td>
<td>Note 1</td>
<td>Reverse Word (Note 1)</td>
</tr>
<tr>
<td>76</td>
<td>5</td>
<td>-1</td>
<td>Note 1</td>
<td>-1</td>
</tr>
<tr>
<td>77</td>
<td>6</td>
<td>(Note 1)</td>
<td>Note 1</td>
<td>Adv 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>Note 1</td>
<td>Erase to EOL (Note 1)</td>
</tr>
<tr>
<td>80</td>
<td>2</td>
<td>(Note 1)</td>
<td>Note 1</td>
<td>-1</td>
</tr>
<tr>
<td>81</td>
<td>3</td>
<td>PageDown (Note 1)</td>
<td>Note 1</td>
<td>Erase to EOS (Note 1)</td>
</tr>
<tr>
<td>82</td>
<td>0</td>
<td>INS (Note 1,2)</td>
<td>Note 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>

Note 1: Refer to Extended Codes Page (3-13).
Note 2: Refer to Special Handling Page (3-15).

Extended Codes
A. Extended Functions

For certain functions that can not 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 26. Keyboard Extended Functions

<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-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 &amp; Home Cursor</td>
</tr>
<tr>
<td>75</td>
<td>←</td>
</tr>
<tr>
<td>77</td>
<td>→</td>
</tr>
<tr>
<td>79</td>
<td>End</td>
</tr>
<tr>
<td>80</td>
<td>↓</td>
</tr>
<tr>
<td>81</td>
<td>Page Down &amp; Home Cursor</td>
</tr>
<tr>
<td>82</td>
<td>INS</td>
</tr>
<tr>
<td>83</td>
<td>DEL</td>
</tr>
<tr>
<td>84-93</td>
<td>F11-F20 (Upper Case F1-F10)</td>
</tr>
<tr>
<td>94-103</td>
<td>F21-F30 (CTRL F1-F10)</td>
</tr>
<tr>
<td>104-113</td>
<td>F31-F40 (ALT F1-F10)</td>
</tr>
<tr>
<td>114</td>
<td>CTRL PRTSC (Start/Stop Echo to Printer) Key 55</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 EOL</td>
</tr>
<tr>
<td>118</td>
<td>CTRL PG DN Erase 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 PG UP TOP 25 Lines of Text &amp; Home Cursor</td>
</tr>
</tbody>
</table>

B. Shift States

Most shift states are handled within the keyboard routine transparently 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:


CTRL - Temporarily shifts keys 3, 7, 12, 14, 16-28, 30-38, 43-50, 55, 59-71, 73, 75, 77, 79, 81 to CTRL state. Used with ALT and DEL to cause “system reset” function described in Section I.3. Used with SCROLL LOCK to cause “break” function described in Section I.3. Used with NUMLOCK to cause “pause” function described in Section I.3.
ALT - Temporarily shifts keys 2-13, 16-25, 30-38, 44-50, and 59-68 to ALT state. Used with CTRL and DEL to cause system reset function described in Section 1.3.

ALT has a special use to allow the user to enter any character code (0-255) into the system from the keyboard. The user holds down the ALT key and types the decimal value of characters using the numeric keyboard (keys 71-73, 75-77, 79-82). The ALT key is then released. If more than three digits are typed, a modulo 256 result is created. These three keys are interpreted as a character code (000-255) and are transmitted through the keyboard routine to the system or application program. ALT is handled internal to keyboard routine.

CAPS LOCK - Shifts keys 16-25, 30-38, 44-50 to upper case. A second depression of CAPS LOCK reverses the action. Handled internal to keyboard routine.

NUM LOCK - Shifts keys 71-73, 75-77, 79-83 to numeric state. A second depression of NUM LOCK reverses the action. Handled internal to keyboard routine.

SCROLL LOCK - Interpreted by appropriate application programs as indicating that the use of the cursor control keys should cause windowing over the text rather than cursor movement. A second depression of SCROLL LOCK reverses the action. The keyboard routine simply records the current shift state of SCROLL LOCK. It is up to the system or application program to perform the function.

C. Shift Key Priorities and Combinations

If combinations of ALT, CTRL and SHIFT are pressed and only one is valid, the precedence is as follows: Highest is ALT, then CTRL, then SHIFT. The only valid combination is ALT CTRL, which is used in system reset.

Special Handling

A. System Reset

The combination of ALT CTRL DEL (Key 83) will result in the keyboard routine initiating the equivalent of a system reset/reboot. Handled internal to keyboard routine.
B. Break
The combination CTRL BREAK will result in the keyboard routine signaling interrupt -1A. Also, the extended characters (AL = 00H, AH = 00H) will be returned.

Power up initialization, this interrupt is set up to cause the break sequence to be ignored. It is up to the system or application initialization code to change the interrupt vector in order to support an actual “break” function.

C. Pause
The combination CTRL NUM-LOCK will cause the keyboard interrupt routine to loop, waiting for any key except NUM-LOCK to be pressed. This provides a system/application transparent method of suspending list/print/etc. temporarily, and then resuming. The “Unpause” key is thrown away. Handled internal to keyboard routine.

D. The following keys will have their typematic action suppressed by the keyboard routine: CTRL, SHIFT, ALT, NUM-LOCK, SCROLL-LOCK, CAPS LOCK, INS.

E. Print Screen
The combination SHIFT-PRINT SCREEN (Key 55) will result in an interrupt invoking the print screen routine. This routine works in alpha/graphics mode, with unrecognizable characters printing as blanks.

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

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

Table 27. Keyboard - Commonly Used Functions

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>KEY(S)</th>
<th>COMMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Home Cursor</td>
<td>HOME</td>
<td>Editors; word processors</td>
</tr>
<tr>
<td>Return to outermost menu</td>
<td>HOME</td>
<td>Menu driven applications</td>
</tr>
<tr>
<td>Move cursor up</td>
<td>↑</td>
<td>Full screen editor, word processor</td>
</tr>
<tr>
<td>Page up, scroll backwards</td>
<td>PG UP</td>
<td>Editors; word processors</td>
</tr>
<tr>
<td>Place cursor at end of line</td>
<td>END</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>END</td>
<td>Editors; word processors</td>
</tr>
<tr>
<td>Move cursor down</td>
<td>↓</td>
<td>Full screen editor, word processor</td>
</tr>
<tr>
<td>Page down, scroll forwards</td>
<td>PG DN</td>
<td>Editors; word processors</td>
</tr>
<tr>
<td>Start/Stop insert text at</td>
<td>INS</td>
<td>Text, command entry</td>
</tr>
<tr>
<td>cursor, shift text right in buffer</td>
<td></td>
<td></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>→</td>
<td>Text entry</td>
</tr>
<tr>
<td>Tab reverse</td>
<td>←</td>
<td>Text entry</td>
</tr>
<tr>
<td>Clear screen and home</td>
<td>CTRL HOME</td>
<td>Command entry</td>
</tr>
<tr>
<td>Scroll up</td>
<td>↑</td>
<td>In scroll lock mode</td>
</tr>
<tr>
<td>Scroll down</td>
<td>↓</td>
<td>In scroll lock mode</td>
</tr>
<tr>
<td>Scroll left</td>
<td>←</td>
<td>In scroll lock mode</td>
</tr>
<tr>
<td>Scroll right</td>
<td>→</td>
<td>In scroll lock mode</td>
</tr>
<tr>
<td>Delete from cursor to EOL</td>
<td>CTRL END</td>
<td>Text, command entry</td>
</tr>
<tr>
<td>Exit/Escape</td>
<td>ESC</td>
<td>Editor, 1 level of menu, etc</td>
</tr>
<tr>
<td>Start/Stop Echo screen to printer</td>
<td>PRTSC, CTRL K55</td>
<td>Any time</td>
</tr>
<tr>
<td>Delete from cursor to EOS</td>
<td>CTRL PG DN</td>
<td>Text, command entry</td>
</tr>
<tr>
<td>FUNCTION</td>
<td>KEY(S)</td>
<td>COMMENT</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>----------------</td>
<td>---------------------------------------------------</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>
<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 + NUMLOCK</td>
<td>Stop list, stop program, etc. 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 PG UP</td>
<td>Editors, word processors</td>
</tr>
<tr>
<td>Standard Function Keys</td>
<td>F1–F10</td>
<td>Primary function keys</td>
</tr>
<tr>
<td>Secondary function keys</td>
<td>SHIFT F1–F10</td>
<td>Extra function keys if 10 are not sufficient</td>
</tr>
<tr>
<td></td>
<td>CTRL F1–F10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ALT F1–F10</td>
<td></td>
</tr>
<tr>
<td>Extra function keys</td>
<td>ALT Keys 2–13</td>
<td>Used when stickers are put along top of keyboard</td>
</tr>
<tr>
<td></td>
<td>(1–9,0,–,=)</td>
<td></td>
</tr>
<tr>
<td>Extra function keys</td>
<td>ALT A–Z</td>
<td>Used when function starts with same letter as one of the alpha keys</td>
</tr>
</tbody>
</table>
### Table 28. BASIC Screen Editor Special Functions

<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 NUMLOCK</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>
## Low Memory Maps (0-‘0600’x)

### Table 30. Interrupt Vectors (0-7F)

<table>
<thead>
<tr>
<th>ADDRESS HEX</th>
<th>INTERRUPT HEX</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-3</td>
<td>0</td>
<td>Divide by Zero</td>
</tr>
<tr>
<td>4-7</td>
<td>1</td>
<td>Single step</td>
</tr>
<tr>
<td>8-B</td>
<td>2</td>
<td>Non-Maskable Interrupt (NMI)</td>
</tr>
<tr>
<td>C-F</td>
<td>3</td>
<td>Break Point Instruction ('CC'x)</td>
</tr>
<tr>
<td>10-13</td>
<td>4</td>
<td>Overflow</td>
</tr>
<tr>
<td>14-17</td>
<td>5</td>
<td>Print Screen</td>
</tr>
<tr>
<td>18-1F</td>
<td>6,7</td>
<td>Reserved</td>
</tr>
<tr>
<td>20-23</td>
<td>8</td>
<td>Timer (18.2 per second)</td>
</tr>
<tr>
<td>24-27</td>
<td>9</td>
<td>Keyboard Interrupt</td>
</tr>
<tr>
<td>28-37</td>
<td>A,B,C,D</td>
<td>Reserved</td>
</tr>
<tr>
<td>38-3B</td>
<td>E</td>
<td>Diskette Interrupt</td>
</tr>
<tr>
<td>3C-3F</td>
<td>F</td>
<td>Reserved</td>
</tr>
<tr>
<td>40-43</td>
<td>10</td>
<td>Video I/O Call</td>
</tr>
<tr>
<td>44-47</td>
<td>11</td>
<td>Equipment Check Call</td>
</tr>
<tr>
<td>48-4B</td>
<td>12</td>
<td>Memory Check Call</td>
</tr>
<tr>
<td>4C-4F</td>
<td>13</td>
<td>Diskette I/O Call</td>
</tr>
<tr>
<td>50-53</td>
<td>14</td>
<td>RS232 I/O Call</td>
</tr>
<tr>
<td>54-57</td>
<td>15</td>
<td>Cassette I/O Call</td>
</tr>
<tr>
<td>58-5B</td>
<td>16</td>
<td>Keyboard I/O Call</td>
</tr>
<tr>
<td>5C-5F</td>
<td>17</td>
<td>Printer I/O Call</td>
</tr>
<tr>
<td>60-63</td>
<td>18</td>
<td>ROM Basic Entry Code</td>
</tr>
<tr>
<td>64-67</td>
<td>19</td>
<td>Boot Strap Loader</td>
</tr>
<tr>
<td>68-6B</td>
<td>1A</td>
<td>Time of Day Call</td>
</tr>
<tr>
<td>6C-6F</td>
<td>1B</td>
<td>Get Control on Keyboard Break: Note 1</td>
</tr>
<tr>
<td>70-73</td>
<td>1C</td>
<td>Get Control on timer interrupt: Note 1</td>
</tr>
<tr>
<td>74-77</td>
<td>1D</td>
<td>Pointer to video initialization table: Note 2</td>
</tr>
<tr>
<td>78-7B</td>
<td>1E</td>
<td>Pointer to diskette parameter table: Note 2</td>
</tr>
<tr>
<td>7C-7F</td>
<td>1F</td>
<td>Pointer to table (1KB) for graphics character Generator for ASCII 128–255. Defaults to 0:0</td>
</tr>
</tbody>
</table>

Notes:  
1. Initialized at power up to point to an IRET Instruction.  
2. Initialized at power up to point to tables in ROM.
Table 31. BASIC and DOS Reserved Interrupts (80-3FF)

<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-BRK 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>100-1FF</td>
<td>40-7F</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>

Table 32. Reserved Memory Locations (400-5FF)

<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>490-4CF</td>
<td>DOS</td>
<td>Used by DOS Mode Command</td>
</tr>
<tr>
<td>4D0-4EF</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>4F0-4FF</td>
<td></td>
<td>Reserved as Intra-Application Communication area for any application.</td>
</tr>
<tr>
<td>500-5FF</td>
<td>DOS</td>
<td>Reserved for DOS and BASIC</td>
</tr>
<tr>
<td>500</td>
<td>DOS</td>
<td>Print Screen status flag store.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0—Print screen not active or successful 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>
**BASIC Workspace Variables**

If you do DEF SEG (Default workspace segment)

<table>
<thead>
<tr>
<th>Offset into segment of start of program text</th>
<th>Offset into start of variables</th>
<th>(end of program text 1-1)</th>
<th>Keyboard buffer contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>X '2E'</td>
<td>X '30'</td>
<td>X '6A'</td>
<td></td>
</tr>
<tr>
<td>X '347'</td>
<td>X '358'</td>
<td></td>
<td>if 0—no characters in buffer</td>
</tr>
<tr>
<td>X '30'</td>
<td></td>
<td></td>
<td>if 1—characters in buffer</td>
</tr>
<tr>
<td>X '358'</td>
<td></td>
<td></td>
<td>if you POKE &amp; H6A, 0 you</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>flush any characters in buffer</td>
</tr>
</tbody>
</table>

**Example:**

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

<table>
<thead>
<tr>
<th>L</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>X '64'</td>
<td>X '00'</td>
</tr>
</tbody>
</table>
APPENDICES

Contents:

Appendix A: ROM BIOS Listing

Appendix B: Assembly Instruction Set Reference

Appendix C: Of Characters, Keystrokes and Color

Appendix D: Logic Diagrams

Appendix E: Unit Specifications
# APPENDIX A

## ROM BIOS LISTINGS

## CONTENTS

<table>
<thead>
<tr>
<th>ITEM</th>
<th>LINE NUMBER</th>
<th>ADDRESS</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equates and Data Areas</td>
<td>1</td>
<td>---</td>
<td>A-2</td>
</tr>
<tr>
<td>Power-on Self-test</td>
<td>198</td>
<td>E016</td>
<td>A-4</td>
</tr>
<tr>
<td>Boot Strap Loader</td>
<td>1355</td>
<td>E6F2</td>
<td>A-20</td>
</tr>
</tbody>
</table>

### I/O Support

- Asynchronous Communications
  - (RS 232) I/O: 1410, E729, A-20
  - Keyboard I/O: 1640, E82E, A-23
  - Diskette I/O: 2255, EC59, A-32
  - Printer I/O: 3007, EFD2, A-42
  - Display (VIDEO) I/O: 3119, F045, A-43
  - Cassette I/O: 4977, F859, A-68

### System Configuration Analysis

- Memory-Size-Determination: 4903, F841, A-67
- Equipment-Determination (Options): 4933, F84D, A-67

### Graphics-Character Generator

- 5503, FA6E, A-75
- Time-of-day: 5642, FE6E, A-77
- Print Screen: 5817, FF53, A-79

Notes for the BIOS Listing
<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>LINE</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0060</td>
<td></td>
<td>5</td>
<td>PORT_A EQU 60H ; 8255 PORT A ADDR</td>
</tr>
<tr>
<td>0061</td>
<td></td>
<td>6</td>
<td>PORT_B EQU 61H ; 8255 PORT B ADDR</td>
</tr>
<tr>
<td>0062</td>
<td></td>
<td>7</td>
<td>PORT_C EQU 62H ; 8255 PORT C ADDR</td>
</tr>
<tr>
<td>0063</td>
<td></td>
<td>8</td>
<td>CND_PORT EQU 63H</td>
</tr>
<tr>
<td>0064</td>
<td></td>
<td>9</td>
<td>INTA00 EQU 20H ; 8259 PORT</td>
</tr>
<tr>
<td>0065</td>
<td></td>
<td>10</td>
<td>INTA01 EQU 21H ; 8259 PORT</td>
</tr>
<tr>
<td>0066</td>
<td></td>
<td>11</td>
<td>EOI EQU 20H</td>
</tr>
<tr>
<td>0067</td>
<td></td>
<td>12</td>
<td>EOI EQU 20H</td>
</tr>
<tr>
<td>0068</td>
<td></td>
<td>13</td>
<td>TIM_CTL EQU 43H ; 8255 TIMER CONTROL PORT ADDR</td>
</tr>
<tr>
<td>0069</td>
<td></td>
<td>14</td>
<td>TIMER EQU 40H ; 8253 TIMER/COUNTER 0 PORT ADDR</td>
</tr>
<tr>
<td>0070</td>
<td></td>
<td>15</td>
<td>THINT EQU 01 ; TIMER 0 INTR NOV MASK</td>
</tr>
<tr>
<td>0071</td>
<td></td>
<td>16</td>
<td>DMA00 EQU 08 ; DMA STATUS REG PORT ADDR</td>
</tr>
<tr>
<td>0072</td>
<td></td>
<td>17</td>
<td>DMA01 EQU 00 ; DMA CHANNEL 0 ADDRESS REG PORT ADDR</td>
</tr>
<tr>
<td>0073</td>
<td></td>
<td>18</td>
<td>MAX_PERIOD EQU 540H</td>
</tr>
<tr>
<td>0074</td>
<td></td>
<td>19</td>
<td>MIN_PERIOD EQU 410H</td>
</tr>
<tr>
<td>0075</td>
<td></td>
<td>20</td>
<td>KBD_IN EQU 60H ; KEYBOARD DATA IN ADDR PORT</td>
</tr>
<tr>
<td>0076</td>
<td></td>
<td>21</td>
<td>KBCTL EQU 61H ; CONTROL BITS FOR KEYBOARD SENSE DATA</td>
</tr>
<tr>
<td>0077</td>
<td></td>
<td>22</td>
<td>KB_DATA EQU 60H ; KEYBOARD SCAN CODE PORT</td>
</tr>
<tr>
<td>0078</td>
<td></td>
<td>23</td>
<td>KB_CTL EQU 61H ; KEYBOARD INTR MASK</td>
</tr>
<tr>
<td>0079</td>
<td></td>
<td>24</td>
<td>ABOS SEGMENT AT 0</td>
</tr>
<tr>
<td>0080</td>
<td></td>
<td>25</td>
<td>STG_LOC0 LABEL BYTE</td>
</tr>
<tr>
<td>0081</td>
<td></td>
<td>26</td>
<td>ORG 244</td>
</tr>
<tr>
<td>0082</td>
<td></td>
<td>27</td>
<td>INTS_PTR LABEL WORD</td>
</tr>
<tr>
<td>0083</td>
<td></td>
<td>28</td>
<td>ORG 544</td>
</tr>
<tr>
<td>0084</td>
<td></td>
<td>29</td>
<td>INTS_PTR LABEL WORD</td>
</tr>
<tr>
<td>0085</td>
<td></td>
<td>30</td>
<td>ORG 844</td>
</tr>
<tr>
<td>0086</td>
<td></td>
<td>31</td>
<td>INTS_PTR LABEL WORD</td>
</tr>
<tr>
<td>0087</td>
<td></td>
<td>32</td>
<td>ORG 1044</td>
</tr>
<tr>
<td>0088</td>
<td></td>
<td>33</td>
<td>INTS_PTR LABEL WORD</td>
</tr>
<tr>
<td>0089</td>
<td></td>
<td>34</td>
<td>ORG 11044</td>
</tr>
<tr>
<td>0090</td>
<td></td>
<td>35</td>
<td>INTS_PTR LABEL WORD</td>
</tr>
<tr>
<td>0091</td>
<td></td>
<td>36</td>
<td>ORG 11044</td>
</tr>
<tr>
<td>0092</td>
<td></td>
<td>37</td>
<td>INTS_PTR LABEL WORD</td>
</tr>
<tr>
<td>0093</td>
<td></td>
<td>38</td>
<td>ORG 11044</td>
</tr>
<tr>
<td>0094</td>
<td></td>
<td>39</td>
<td>INTS_PTR LABEL WORD</td>
</tr>
<tr>
<td>0095</td>
<td></td>
<td>40</td>
<td>ORG 11044</td>
</tr>
<tr>
<td>0096</td>
<td></td>
<td>41</td>
<td>INTS_PTR LABEL WORD</td>
</tr>
<tr>
<td>0097</td>
<td></td>
<td>42</td>
<td>ORG 11044</td>
</tr>
<tr>
<td>0098</td>
<td></td>
<td>43</td>
<td>EXT_PTR LABEL DWORD ; LOCATION OF POINTER</td>
</tr>
<tr>
<td>0099</td>
<td></td>
<td>44</td>
<td>ORG 11044</td>
</tr>
<tr>
<td>0100</td>
<td></td>
<td>45</td>
<td>EXT_PTR LABEL DWORD ; LOCATION OF EXTENSION</td>
</tr>
<tr>
<td>0101</td>
<td></td>
<td>46</td>
<td>ORG 11044</td>
</tr>
<tr>
<td>0102</td>
<td></td>
<td>47</td>
<td>ORG 11044</td>
</tr>
<tr>
<td>0103</td>
<td></td>
<td>48</td>
<td>ORG 11044</td>
</tr>
<tr>
<td>0104</td>
<td></td>
<td>49</td>
<td>ORG 11044</td>
</tr>
<tr>
<td>0105</td>
<td></td>
<td>50</td>
<td>ORG 11044</td>
</tr>
<tr>
<td>0106</td>
<td></td>
<td>51</td>
<td>ORG 11044</td>
</tr>
<tr>
<td>0107</td>
<td></td>
<td>52</td>
<td>ORG 11044</td>
</tr>
<tr>
<td>0108</td>
<td></td>
<td>53</td>
<td>ORG 11044</td>
</tr>
<tr>
<td>0109</td>
<td></td>
<td>54</td>
<td>ORG 11044</td>
</tr>
<tr>
<td>0110</td>
<td></td>
<td>55</td>
<td>ORG 11044</td>
</tr>
<tr>
<td>0111</td>
<td></td>
<td>56</td>
<td>ORG 11044</td>
</tr>
<tr>
<td>0112</td>
<td></td>
<td>57</td>
<td>ORG 11044</td>
</tr>
<tr>
<td>0113</td>
<td></td>
<td>58</td>
<td>ORG 11044</td>
</tr>
<tr>
<td>0114</td>
<td></td>
<td>59</td>
<td>DATA SEGMENT AT 40H</td>
</tr>
<tr>
<td>0115</td>
<td></td>
<td>60</td>
<td>RS232_BASE DW 4 DUP (?) ; ADDRESSES OF RS232 ADAPTERS</td>
</tr>
<tr>
<td>0116</td>
<td></td>
<td>61</td>
<td>PRINTER_BASE DW 4 DUP (?) ; ADDRESSES OF PRINTERS</td>
</tr>
<tr>
<td>0117</td>
<td></td>
<td>62</td>
<td>EQUIP_FLAG DW ? ; INSTALLED HARDWARE</td>
</tr>
<tr>
<td>0118</td>
<td></td>
<td>63</td>
<td>MFG_TST DB ? ; INITIALIZATION FLAG</td>
</tr>
<tr>
<td>0119</td>
<td></td>
<td>64</td>
<td>MEMORY_SIZE DW ? ; MEMORY SIZE IN K BYTES</td>
</tr>
<tr>
<td>0120</td>
<td></td>
<td>65</td>
<td>IO_RAM_SIZE DW ? ; MEMORY IN I/O CHANNEL</td>
</tr>
<tr>
<td>0121</td>
<td></td>
<td>66</td>
<td></td>
</tr>
<tr>
<td>0122</td>
<td></td>
<td>67</td>
<td></td>
</tr>
<tr>
<td>0123</td>
<td></td>
<td>68</td>
<td></td>
</tr>
<tr>
<td>0124</td>
<td></td>
<td>69</td>
<td>KB_Flag DB ?</td>
</tr>
<tr>
<td>0125</td>
<td></td>
<td>70</td>
<td></td>
</tr>
<tr>
<td>0126</td>
<td></td>
<td>71</td>
<td></td>
</tr>
<tr>
<td>0127</td>
<td></td>
<td>72</td>
<td></td>
</tr>
<tr>
<td>0128</td>
<td></td>
<td>73</td>
<td>INS_STATE EQU 60H ; INSERT STATE IS ACTIVE</td>
</tr>
<tr>
<td>0129</td>
<td></td>
<td>74</td>
<td>CAPS_STATE EQU 40H ; CAPS LOCK STATE HAS BEEN Toggled</td>
</tr>
<tr>
<td>0130</td>
<td></td>
<td>75</td>
<td>NUM_STATE EQU 20H ; NUM LOCK STATE HAS BEEN Toggled</td>
</tr>
<tr>
<td>0131</td>
<td></td>
<td>76</td>
<td>SCROLL_STATE EQU 10H ; SCROLL LOCK STATE HAS BEEN Toggled</td>
</tr>
<tr>
<td>0132</td>
<td></td>
<td>77</td>
<td>ALF_SHIFT EQU 08H ; ALTERNATE SHIFT KEY DEPRESSED</td>
</tr>
</tbody>
</table>

A-2
LOC OBJ     LINE     SOURCE

0004       70  CTL_SHIFT EQU 04H ; CONTROL SHIFT KEY DEPRESSED
0002       79  LEFT_SHIFT EQU 02H ; LEFT SHIFT KEY DEPRESSED
0001       80  RIGHT_SHIFT EQU 01H ; RIGHT SHIFT KEY DEPRESSED

0018 ??    82  KB_FLAG_1 DB ? ; SECOND BYTE OF KEYBOARD STATUS

0000       84  INS_SHIFT EQU 06H ; INSERT KEY IS DEPRESSED
0040       85  CAPS_SHIFT EQU 04H ; CAPS LOCK KEY IS DEPRESSED
0006       86  NUM_SHIFT EQU 02H ; NUM LOCK KEY IS DEPRESSED
0010       87  SCROLL_SHIFT EQU 01H ; SCROLL LOCK KEY IS DEPRESSED
0008       88  HOLD_STATE EQU 00H ; SUSPEND KEY HAS BEEN TOGGLED

0019 ??    90  ALT_INPUT DB ? ; STORAGE FOR ALTERNATE KEYPAD ENTRY
001A  ??    91  BUFFER_HEAD DW ? ; POINTER TO HEAD OF KEYBOARD BUFFER
001C ??    92  BUFFER_TAIL DW ? ; POINTER TO TAIL OF KEYBOARD BUFFER
001E ??    93  KB_BUFFER DW 16 DUP(?); ROOM FOR 16 ENTRIES
003E       94  KB_BUFFER_END LABEL WORD

96  ------- HEAD = TAIL INDICATES THAT THE BUFFER IS EMPTY

97  -------------- ---------------------------

0045       98  NUM_KEY EQU 69 ; SCAN CODE FOR NUMBER LOCK
0046       99  SCROLL_KEY EQU 70 ; SCROLL LOCK KEY
0030       100 ALT_KEY EQU 56 ; ALTERNATE SHIFT KEY SCAN CODE
0010       101 CTL_KEY EQU 29 ; SCAN CODE FOR CONTROL KEY
003A       102 CAPS_KEY EQU 55 ; SCAN CODE FOR SHIFT LOCK
002A       103 LEFT_KEY EQU 42 ; SCAN CODE FOR LEFT SHIFT
0036       104 RIGHT_KEY EQU 54 ; SCAN CODE FOR RIGHT SHIFT
0002       105 INS_KEY EQU 82 ; SCAN CODE FOR INSERT KEY
0003       106 DEL_KEY EQU 83 ; SCAN CODE FOR DELETE KEY

107  -------------- ---------------------------

108  ; DISKETTE DATA AREAS

109  -------------- ---------------------------

005F ??    111 SEEK_STATUS DB ? ; DRIVE RECALIBRATION STATUS
112  ; BIT 3-0 = DRIVE 3-0 NEEDS RECAL BEFORE
113  ; NEXT SEEK IF BIT IS = 0

0060       114 INT_FLAG EQU 050H ; INTERRUPT OCCURRENCE FLAG

003F ??    115 MOTOR_STATUS DB ? ; MOTOR STATUS
116  ; BIT 3-0 = DRIVE 3-0 IS CURRENTLY RUNNING
117  ; BIT 7 = CURRENT OPERATION IS A WRITE, REQUIRES DELAY

0040 ??    118 MOTOR_COUNTER DB ? ; TIME COUNTER FOR DRIVE GOVERN
0025       119 MOTOR_WAIT EQU 37 ; TWO SECONDS OF COUNTS FOR MOTOR GOVERN

120  -------------- ---------------------------

121  ; CASSETTE DATA AREA

122  -------------- ---------------------------

0000       123 TIME_OUT EQU 08H ; ATTACHMENT FAILED TO RESPOND
0040       124 BAD_SEEK EQU 40H ; SEEK OPERATION FAILED
0020       125 BAD_NEC EQU 20H ; NEC CONTROLLER HAS FAILED
0010       126 BAD_CRC EQU 10H ; BAD CRC ON DISKETTE READ
0009       127 DMA_BOUNDARY EQU 09H ; ATTEMPT TO DMA ACROSS 64K BOUNDARY
0018       128 BAD_DMA EQU 08H ; DMA OVERFLOW ON OPERATION
0004       129 RECORD_NOT_FOUND EQU 04H ; REQUESTED SECTOR NOT FOUND
0003       130 WRITE_PROTECT EQU 03H ; WRITE ATTEMPTED ON WRITE PROTECT DISK
0002       131 BAD_ADDR_MARK EQU 02H ; ADDRESS MARK NOT FOUND
0001       132 BAD_CMD EQU 01H ; BAD COMMAND PASSED TO DISKETTE I/O

0042 (??) 133 NEC_STATUS DB 7 DUP(?); STATUS BYTES FROM NEC

134  -------------- ---------------------------

135  ; VIDEO DISPLAY DATA AREA

136  -------------- ---------------------------

0049 ??    139 CRT_MODE DB ? ; CURRENT CRT MODE
0044 ??    140 CRT_COLS DW ? ; NUMBER OF COLUMNS ON SCREEN
004C ??    141 CRT_LEN DW ? ; LENGTH OF REGEN IN BYTES

004E ??    142 CRT_START DW ? ; STARTING ADDRESS IN REGEN BUFFER
0050 ??    143 CURSOR_POSN DW 8 DUP (?) ; CURSOR FOR EACH OF UP TO 8 PAGES
0060 ??    144 CURSOR_MODE DW ? ; CURRENT CURSOR MODE SETTING
0062 ??    145 ACTIVE_PAGE DB ? ; CURRENT PAGE BEING DISPLAYED
0063 ??    146 ADDR_6045 DW ? ; BASE ADDRESS FOR ACTIVE DISPLAY CARD
0065 ??    147 CRT_MODE_SET DB ? ; CURRENT SETTING OF THE 3X8 REGISTER
0066 ??    148 CRT_PALLETTE DB ? ; CURRENT PALLETTE SETTING COLOR CARD

149  -------------- ---------------------------

150  ; CASSETTE DATA AREA

151  -------------- ---------------------------

007F ??    153 EDGE_CNT DW ? ; TIME COUNT AT DATA EDGE
0069 ??    154 CRC_REG DW ? ; CRC REGISTER
LOC OBJ
LINE SOURCE

0068 ?? 155 LAST_VAL DB ? ; LAST INPUT VALUE
156
157 
158 1 ------------------------------- -------------------------------
159 1 ; TIMER DATA AREA
160 
161 160 ; TIMER_LOW DW ? ; LOW WORD OF TIMER COUNT
162 161 ; TIMER_HIGH DW ? ; HIGH WORD OF TIMER COUNT
163 162 ; TIMER_DFPL DB ? ; TIMER HAS ROLLED OVER SINCE LAST READ
164 163 ; COUNTS_SEC EQU 18
165 164 ; COUNTS_MIN EQU 102
166 165 ; COUNTS_HOUR EQU 65543
167 166 ; COUNTS_DAY EQU 1573040 = 1000004
168 167
169 1 ------------------------------- -------------------------------
170 1 ; SYSTEM DATA AREA
171 
172 0071 ?? 171 BIOS_BREAK DB ? ; BIT 7 = 1 IF BREAK KEY HAS BEEN DEPRESSED
173 172
174 173 ; COUNTS_SEC EQU 18
175 174 ; COUNTS_MIN EQU 102
176 175
177 0079 ? ? 176 TIMER_OFL DB ? ; TIMER HAS ROLLED OVER SINCE LAST READ
178 177
179 178
180 180
181 181
182 182
183 0080 ??? 183 : VIDEO DISPLAY BUFFER
184 
185 0080 184 VIDEO_RAM SEGMENT AT 0000H
186 185
187 0080 186 REGEN LABEL BYTE
188 186
189 0080 187 REGEN LABEL WORD
190 187
191 0080 (16304 ??) 188 DB 16304 DUP(?)
192 188
193 189
194 0080 189 VIDEO_RAM ENDS
195 190
196 
197 
198 0080 191 ; ROM RESIDENT CODE
199 191
200 0080 192 ; -------------------------- --------------------------
201 192
202 0080 193 CODE SEGMENT AT 01000H
203 193
204 0080 (57344 ??) 194 DB 57344 DUP(?) ; FILL LOWEST 56K
205 194
206 0080 (3573030303531 195 DB '5700051 CDRH. IBM 1981' ; COPYRIGHT NOTICE
207 196 204
208 0080 205 CI DW C11 ; RETURN ADDRESS
209 205
210 0080 206 C2 DW C24 ; RETURN ADDRESS FOR DUMMY STACK
211 206
212 0080 207 -------------------------- --------------------------
213 207
214 0080 208 THIS SUBROUTINE PERFORMS A READ/WRITE STORAGE TEST ON A 16K BLOCK
215 208
216 0080 209 ; OF STORAGE.
217 209
218 0080 210 ENTRY REQUIREMENTS:
219 210
220 0080 211 ES = ADDRESS OF STORAGE SEGMENT BEING TESTED
221 211
222 0080 212 DS = ADDRESS OF STORAGE SEGMENT BEING TESTED
223 212
224 0080 213 WHEN ENTERING AT STGTST_CNT, CX MUST BE LOADED WITH THE BYTE COUNT.
225 213
226 0080 214 EXIT PARAMETERS:
227 214
228 0080 215 ZERO FLAG = 0 IF STORAGE ERROR (DATA COMPARE OR PARITY CHECK. AL = 0
229 215
230 0080 216 AL DENOTES A PARITY CHECK. ELSE AL = XOR OF THE EXPECTED DATA
231 216
232 0080 217 CX, DX, Cx, Cx, 01, AND DI ARE ALL DESTROYED.
233 217
234 0080 218
235 0080 219
236 0080 220
237 0080 221 STGTST PROC NEAR
238 221
239 0080 222 STGTST_DBX CODE
240 222
241 0080 223 CI DW C11 ; RETURN ADDRESS
242 223
LOC OBJ LINE SOURCE

E02A 229 C3: DEC DI ;FIND TO LAST BYTE JUST WRITTEN
E02A 4F 230 STD ;SET DIR FLAG TO GO BACKWARDS
E02C B0FF 231 C4: MOV SI,DI ;SET DIR FLAG TO GO BACKWARDS
E02E 86CB 232 MOV CX,BS ;READ CHAR FROM STORAGE
E030 3AC 233 C5: LODSB ;READ CHAR FROM STORAGE
E031 32CA 234 XOR AL,AM ;READ CHAR FROM STORAGE
E033 7925 235 JNE C7 ;READ CHAR FROM STORAGE
E035 E62 236 IN AL,PORT_C ;READ CHAR FROM STORAGE
E037 24C0 237 AND AL,OCOH ;READ CHAR FROM STORAGE
E039 B000 238 MOV AL,0 ;READ CHAR FROM STORAGE
E03B 7810 239 JNZ C7 ;READ CHAR FROM STORAGE
E03D 80FC00 240 CMP AH,0 ;READ CHAR FROM STORAGE
E042 8AC2 241 JNE C7 ;READ CHAR FROM STORAGE
E044 A6 242 JS C7 ;READ CHAR FROM STORAGE
E046 E02E609 243 STOSB ;READ CHAR FROM STORAGE
E048 7403 244 JZ C4 ;READ CHAR FROM STORAGE
E04A 88tB 245 MOV CX,BX ;READ CHAR FROM STORAGE
E04C 7BC7 246 XOR AL,AM ;READ CHAR FROM STORAGE
E04E 231 247 MOV AL,0 ;READ CHAR FROM STORAGE
E050 7403 248 JE C6 ;READ CHAR FROM STORAGE
E052 7BC7 249 MOV AL,0 ;READ CHAR FROM STORAGE
E054 7403 250 JE C6 ;READ CHAR FROM STORAGE
E056 7BC7 251 MOV AL,0 ;READ CHAR FROM STORAGE
E058 7403 252 JE C6 ;READ CHAR FROM STORAGE
E05A 7BC7 253 MOV AL,0 ;READ CHAR FROM STORAGE
E05C 7BC7 254 MOV AL,0 ;READ CHAR FROM STORAGE
E05E 7BC7 255 MOV AL,0 ;READ CHAR FROM STORAGE
E060 7403 256 MOV AL,0 ;READ CHAR FROM STORAGE
E062 7BC7 257 MOV AL,0 ;READ CHAR FROM STORAGE
E064 7403 258 MOV AL,0 ;READ CHAR FROM STORAGE
E066 7BC7 259 MOV AL,0 ;READ CHAR FROM STORAGE
E068 7403 260 MOV AL,0 ;READ CHAR FROM STORAGE
E06A 7BC7 261 MOV AL,0 ;READ CHAR FROM STORAGE
E06C 7403 262 MOV AL,0 ;READ CHAR FROM STORAGE
E06E 7BC7 263 MOV AL,0 ;READ CHAR FROM STORAGE
E070 7403 264 MOV AL,0 ;READ CHAR FROM STORAGE
E072 7BC7 265 MOV AL,0 ;READ CHAR FROM STORAGE
E074 7403 266 MOV AL,0 ;READ CHAR FROM STORAGE
E076 7BC7 267 MOV AL,0 ;READ CHAR FROM STORAGE
E078 7403 268 MOV AL,0 ;READ CHAR FROM STORAGE
E07A 7BC7 269 MOV AL,0 ;READ CHAR FROM STORAGE
E07C 7403 270 MOV AL,0 ;READ CHAR FROM STORAGE
E07E 7BC7 271 MOV AL,0 ;READ CHAR FROM STORAGE
E080 7403 272 MOV AL,0 ;READ CHAR FROM STORAGE
E082 7BC7 273 MOV AL,0 ;READ CHAR FROM STORAGE
E084 7403 274 MOV AL,0 ;READ CHAR FROM STORAGE
E086 7BC7 275 MOV AL,0 ;READ CHAR FROM STORAGE
E088 7403 276 MOV AL,0 ;READ CHAR FROM STORAGE
E08A 7BC7 277 MOV AL,0 ;READ CHAR FROM STORAGE
E08C 7403 278 MOV AL,0 ;READ CHAR FROM STORAGE
E08E 7BC7 279 MOV AL,0 ;READ CHAR FROM STORAGE
E090 7403 280 MOV AL,0 ;READ CHAR FROM STORAGE
E092 7BC7 281 MOV AL,0 ;READ CHAR FROM STORAGE
E094 7403 282 MOV AL,0 ;READ CHAR FROM STORAGE
E096 7BC7 283 MOV AL,0 ;READ CHAR FROM STORAGE
E098 7403 284 MOV AL,0 ;READ CHAR FROM STORAGE
E09A 7BC7 285 MOV AL,0 ;READ CHAR FROM STORAGE
E09C 7403 286 MOV AL,0 ;READ CHAR FROM STORAGE
E09E 7BC7 287 MOV AL,0 ;READ CHAR FROM STORAGE
E0A0 7403 288 MOV AL,0 ;READ CHAR FROM STORAGE
E0A2 7BC7 289 MOV AL,0 ;READ CHAR FROM STORAGE
E0A4 7403 290 MOV AL,0 ;READ CHAR FROM STORAGE
E0A6 7BC7 291 MOV AL,0 ;READ CHAR FROM STORAGE
E0A8 7403 292 MOV AL,0 ;READ CHAR FROM STORAGE
E0AA 7BC7 293 MOV AL,0 ;READ CHAR FROM STORAGE
E0AC 7403 294 MOV AL,0 ;READ CHAR FROM STORAGE
E0A8 7BC7 295 MOV AL,0 ;READ CHAR FROM STORAGE
E0B0 7403 296 MOV AL,0 ;READ CHAR FROM STORAGE
E0B2 7BC7 297 MOV AL,0 ;READ CHAR FROM STORAGE
E0B4 7403 298 MOV AL,0 ;READ CHAR FROM STORAGE
E0B6 7BC7 299 MOV AL,0 ;READ CHAR FROM STORAGE
E0B8 7403 300 MOV AL,0 ;READ CHAR FROM STORAGE
E0BA 7BC7 301 MOV AL,0 ;READ CHAR FROM STORAGE
E0BB 7403 302 MOV AL,0 ;READ CHAR FROM STORAGE
E0BC 7BC7 303 MOV AL,0 ;READ CHAR FROM STORAGE
E0BD 7403 304 MOV AL,0 ;READ CHAR FROM STORAGE
E0BF 7BC7 305 MOV AL,0 ;READ CHAR FROM STORAGE
E0C1 7403 306 MOV AL,0 ;READ CHAR FROM STORAGE

APPENDIX A
LOC OBJ | LINE | SOURCE

E0F0 80F6 | 307 | MOV AL,50H
E0F0 80FX | 308 | MOV DL,51H
E0F2 7907 | 309 | JNC C9
E0F4 35C7 | 310 | XOR AX,DI
E0F6 7507 | 311 | JNZ ERR01
E0FA 73DC | 312 | CLC
E0FB 7263 | 313 | JNC CB
E0FA 80C7 | 314 | C9:
E0FA 7401 | 315 | OR AX,DI
E0F9 F4 | 316 | JZ C10
E317 | ERRO1: HLT
E318 | HALT SYSTEM

E319 | TEST:02
E320 | ; ROS CHECKSUM TEST I
E321 | ; DESCRIPTION
E322 | ; A CHECKSUM IS DONE FOR THE 8K ROS MODULE CONTAINING POD AND BIOS.
E323 | ;
E324 | C10:
E325 | MOV AL,P
E326 | OUT BAH,AL
E327 | OUT BMH,AL
E328 | MOV AL,99H
E329 | OUT CMD_PORT,AL
E330 | MOV AL,ATCH
E331 | OUT PORT_H,AL
E332 | SUB AL,AL
E333 | MOV DX,300H
E334 | OUT DX,AL
E335 | JNC AL
E336 | MOV DX,300H
E337 | OUT DX,AL
E338 | MOV AX,0000H
E339 | ADD AX,0000H
E340 | MOV DX,AX
E341 | OUT DX,AX
E342 | CMP DX,0000H
E343 | JNZ C11
E344 | C11:
E345 | JIE ERR01
E346 | HALT SYSTEM IF ERROR
E347 | ;
E348 | ;
E349 | ; 8237 DMA INITIALIZATION CHANNEL REGISTER TEST
E350 | ;
E351 | ; DESCRIPTION
E352 | ; DISABLE THE 8237 DMA CONTROLLER; VERIFY THAT TIMER 1 FUNCTIONS OK.
E353 | ;
E354 | ;
E355 | ; VERIFY THAT TIMER 1 FUNCTIONS OK
E356 | ;
E357 | ;
E358 | ;
E359 | MOV AL,54H
E360 | OUT TIMER+3,AL
E361 | SUB CX,CX
E362 | MOV BL,CL
E363 | MOV AL,CL
E364 | OUT TIMER+1,AL
E365 | C12:
E366 | MOV AL,40H
E367 | JLTCH TIMER 1 COUNT
E368 | OUT TIMER+3,AL
E369 | IN AL,TIMER+1
E370 | OR BL,AL
E371 | CMP AL,TIMER+1
E372 | JE C13
E373 | LOOP C12
E374 | JMP SHORT ERR01
E375 | TIMER 1 FAILURE, HALT SYS
E376 | C13:
E377 | MOV AL,DL
E378 | SUB CX,CX
E379 | OUT TIMER+1,AL
E380 | TIMER_LOOP
E381 | MOV AL,40H
E382 | JLTCH TIMER 1 COUNT
E383 | OUT TIMER+3,AL
E384 | IN AL,TIMER+1
E385 | LOOP C14
E386 | TIMER LOOP
LaC OBJ LINE SOURCE

E100 EBA0 365 JMP SHORT ERR01
366
367  ; INITIALIZE TIMER 1 TO REFRESH MEMORY
368
E10F 399 C15: MOV AL,54H ; WRAP DMA REG
E10F B094 390 OUT TIMER+3,AL ; SEL TIM 1, LSB, MODE 2
E111 E643 391 MOV AL,10 ; WRITE TIMER MODE REG
E113 B012 392 OUT TIMER+1,AL ; SET UP DIVIDOR FOR REFRESH
E115 E641 393 OUT DMA+O1H,AL ; WRITE TIMER 1 CNT REG
E117 E600 394 OUT DMA+O0H,AL ; SEND MASTER CLEAR TO DMA
395  ; WRAP DMA CHANNELS ADDRESS AND COUNT REGISTERS
396
E119 BOFF 398 MOV AL,OFFH ; WRITE PATTERN FFH TO ALL REGS
E119 8ADD 399 MOV BL,AL ; SAVE PATTERN FOR COMPARE
E11D 8ADD 400 MOV BH,AL
E11F 8ADD 401 MOV CX,0B
E122 BADD 402 MOV DX,OMA ; SETUP I/O ADDR OF REG
E125 EE 403 MOV BX,DX ; READ 16-BIT REG, LSB
E127 EADD 404 OUT BX,AL ; READ DMA ADDR, MSB
E12A EC 405 MOV AX,BX ; AX TO ANOTHER PRT BEFORE RO
E12B EE 406 MOV BX,A0H ; AX TO ANOTHER PRT BEFORE RO
E12C EADD 407 MOV AL,OFFH ; WRITE PATTERN FFH TO ALL REGS
E12D EADD 408 MOV AH,AL ; WRITE PATTERN TO REG, LSB
E130 7ADD 409 MOV AX,AL ; SAVE PATTERN AS WRITTEN?
E132 E97AFF 410 JMP ERR01 ; YES - CHECK NEXT REG
E135 411 ; NEXT_DMA_CHAN
E135 412 MOV AX,0 ; CLEAR REGS
E137 EADD 413 MOV BL,AX ; CLEAR REGS
E138 EADD 414 MOV CL,AX ; CLEAR REGS
E13A 7ADD 415 MOV AL,OFFH ; CLEAR REGS
E13B EADD 416 MOV AH,AL ; CLEAR REGS
E13C EADD 417 MOV AX,0 ; CLEAR REGS
E13F EADD 418 MOV AL,0 ; CLEAR REGS
E141 EADD 419 MOV AH,0 ; CLEAR REGS
E143 EADD 420 MOV AL,0 ; CLEAR REGS
E144 EADD 421 MOV AH,0 ; CLEAR REGS
E146 EADD 422 MOV AH,0 ; CLEAR REGS
E148 EADD 423 MOV AH,0 ; CLEAR REGS
E14A EADD 424 MOV AH,0 ; CLEAR REGS
E14C EADD 425 MOV AH,0 ; CLEAR REGS
E14D EADD 426 MOV AH,0 ; CLEAR REGS
E14F EADD 427 MOV AH,0 ; CLEAR REGS
E151 EADD 428 MOV AH,0 ; CLEAR REGS
E153 EADD 429 MOV AH,0 ; CLEAR REGS
E155 EADD 430 MOV AH,0 ; CLEAR REGS
E157 EADD 431 MOV AH,0 ; CLEAR REGS
E159 EADD 432 MOV AH,0 ; CLEAR REGS
E15B EADD 433 MOV AH,0 ; CLEAR REGS
E15D EADD 434 MOV AH,0 ; CLEAR REGS
E15F EADD 435 MOV AH,0 ; CLEAR REGS
E161 EADD 436 MOV AH,0 ; CLEAR REGS
E163 EADD 437 MOV AH,0 ; CLEAR REGS
E165 EADD 438 MOV AH,0 ; CLEAR REGS
E167 EADD 439 MOV AH,0 ; CLEAR REGS
E169 EADD 440 MOV AH,0 ; CLEAR REGS
E16B EADD 441 MOV AH,0 ; CLEAR REGS
E16D EADD 442 MOV AH,0 ; CLEAR REGS
E16F EADD 443 MOV AH,0 ; CLEAR REGS
E171 E3EO 444 MOV AH,0 ; CLEAR REGS
E173 E3EO 445 MOV AH,0 ; CLEAR REGS
E175 A8EO 446 MOV AH,0 ; CLEAR REGS
E177 FC 447 MOV AX,0 ; CLEAR REGS
E179 AA 448 MOV AX,0 ; CLEAR REGS

;-----------------------------------------------
;TEST.04
; BASE 16K READ/WRITE STORAGE TEST
;DESCRIPTION
; WRITE/READ/VERIFY DATA PATTERNS FF,55,AA,01, AND 00 TO 1ST 16K OF
; STORAGE. VERIFY STORAGE ADDRESSECITY.
; INITIALIZE THE 8259 INTERRUPT CONTROLLER CHIP FOR CHECKING
; MANUFACTURING TEST 2 MODE.
; DETERMINE MEMORY SIZE AND FILL MEMORY WITH DATA

E150 BADD 449 MOV AX,DATA ;POINT DS TO DATA SEG
E154 8ADD 450 MOV DS,AX
E155 BADD 451 MOV ES,AX ; SETUP ES SEGMENT REG
E157 BADD 452 MOV DS,AX
E161 EADD 453 MOV DX,PORT_A ; DETERMINE BASE RAM SIZE
E163 EADD 454 MOV AX,AL ; ISOlate RAM SIZE SWs
E165 EADD 455 MOV CL,12 ; CALCULATE MEMORY SIZE
E167 EADD 456 MOV DL,4
E169 EADD 457 MOV AH,0 ; SET DS, ES FLAG TO INCR
E16B EADD 458 MOV AX,CL
E16D EADD 459 MOV AX,AL
E16F EADD 460 MOV CX,AX
E171 EADD 461 MOV AH,AL
E179 E2FD | LOOP C19 | LOUP TIL ALL ZERO
E17B E462 | IN AL, PORT_C
E17D 240F | AND AL, OFM
E17F 7418 | JE C21
E181 B401D | MOV DX, 1000H | SEGMENT FOR I/O RAM
E184 8E0 | MOV AH, AL
E186 8D0 | MOV AL, 0
E188 472 | LOOP C20 | FILL_ID:
E18A 8EC2 | MOV ES, DX
E18A B90080 | MOV CX, 8000H | FILL 32K BYTES
E18D ZBF | SUB DI, DI
E18F F3 | REP STOSB
E190 AA
E191 B1C0080 | ADD DX, 8000H | NEXT SEGMENT VALUE
E195 FECC | DEC AH
E197 75EF | JNZ C20 | FILL_ID
E199 | -------------------------------------------
E199 B015 | DETERMINE 10 CHANNEL RAM SIZE
E19B 8E20 | MOV AL, 13H | ICW1 - EDGE, SNGL, ICW4
E19D B008 | MOV AL, 0 | SETUP ICW2 - INT TYPE 0 (6-F?)
E19F 8E21 | MOV AL, 1A | SETUP ICW4 - BUFFER, 8086 MODE
E1A1 B009 | MOV AL, 19 | SETUP ICW4 - 8086 MODE
E1A3 8E21 | MOV AL, 1A | SETUP ICW4 - 8086 MODE
E1A5 8EC0 | SUB AX, AX | POINT DS AND ES TO BEGIN
E1A7 8EC0 | MOV ES, AX | OF R/W STORAGE
E1A9 8E00 | MOV SI, DATA | POINT AS TO DATA SEG
E1AB 8EC0 | MOV AX, SI | POINT AS TO DATA SEG
E1AC 891E7200 | CMP RESET_FLAG, 1234H | RESET_FLAG SET?
E1AE 891E7200 | CMP RESET_FLAG, 1234H | RESET_FLAG SET?
E1AF 7430 | JE C25 | YES - SKIP STG TEST
E1B0 8E08 | MOV DX, AX | FILL 1ST 64K OF STG
E1B1 8E08 | MOV DS, AX | POINT DS TO 1ST 16K OF STG
E1B2 BC03F | INITIALIZE THE 8259 INTERRUPT CONTROLLER CHIP
E1B6 8E0D0 | MOV SP, 3FF0H | ESTABLISH TEMPORARY STACK
E1B8 8E0D0 | MOV SP, 3FF0H |
E1C1 8F60 | MOV SI, AX |
E1C3 B6400 | MOV DI, AX |
E1C5 C70062 | MOV WORD PTR [BX].OFFSET D11 | SET UP KB INTERRUPT
E1CA 643 | INC BX |
E1CB 643 | INC BX |
E1CC 8C0F | MOV [BX].CS |
E1CE E90740 | CALL KBD_RESET | READ IN KB RESET CODE TO BL
E1D1 80756 | CMP BL, 065H | IS THIS MANUFACTURING TEST 2?
E1D4 756E | JNZ C33 | JUMP IF NOT MAN. TEST
E1D6 82FF | MOV DL, 055 | READ IN TEST PROGRAM
E1D8 80A04 | CALL SP_TEST |
E1DA 8AC3 | MOV AL, BL |
E1DE AA | STOSB |
E1E0 FECA | DEC DL |
E1E2 75F6 | JNZ C22 | JUMP IF NOT DONE YET
E1E4 C3E4 | MOV AX, 0 | GET STACK VALUE
E1E5 17 | POP SS |
E1E6 F1 | CLI |
E1E7 B00E0 | MOV SP, OFFSET C2 | SETUP RETURN ADDRESS
E1EA E92DFE | JMP STGST | GO TO RD/WR STG SUBROUTINE
E1ED 7404 | JE C25 | GO TO NEXT TEST IF OK
E1EF E90DFE | JMP ERROR |
E1F2 83000 | MOV AX, STACK |
E1F3 8E00 | MOV SS, AX |
E1F7 BC0001 | MOV SP, OFFSET TOS | STACK IS READY TO GO
E1F9 FECA | DEC SP |
E1FA FECA | DEC SP |
E1FB 83000 | MOV AX, STACK |
E1FC 8E00 | MOV SS, AX |
E1FD BC0001 | MOV SP, OFFSET TOS |
E1FE FECA | DEC SP |
E1FF FECA | DEC SP |
E1A 26C7060800C3EZ R 537 MOV ES:NMI_PTR,OFFSET NMI_INT
E20 26C7060A0000F0 R 538 MOV ES:NMI_PTR+2,CODE
E200 92400 539 JMP TST6 540
E20B 100200 541 RO6_CHEC6USM PROC NEAR NEXT_ROS_MODULE
E20E 32C0 542 MOV CX,8192 543
E210 544 C26: 545 ADD AL,CS:[BX] 546
E213 45 547 INC BX 548 ;POINT TO NEXT BYTE
E214 26FA 549 LOOP C26 550 ;ADD ALL BYTES IN ROS MODULE
E216 0AC0 551 ;SUM = 0?
E218 5C 552 ROS_CHECKSUM ENDP
553
E219 504155249565929 554 ; INITIAL RELIABILITY TEST -- PHASE 2
000E 555 ; ASSUME CS:CODE,ES:4850
556 D1 DB 'PARITY CHECK 2'
557 D2 DB 'PARITY CHECK 1'
558 DIL EQU 9-D1
559 D2L EQU 9-D2
560 ITEST,66
561 8259 INTERRUPT CONTROLLER TEST
562 ; DESCRIPTION
563 ; READ/WRITE THE INTERRUPT MASK REGISTER (IMR) WITH ALL ONES AND ZEROS.
564 ; ENABLE SYSTEM INTERRUPTS, MASK DEVICE INTERRUPTS OFF. CHECK FOR
565 ; HOT INTERRUPTS (UNEXPECTED).
566 ; -------------------------------
E235 567 TST6:
568 SUB AX,AX ; SET UP ES REG
E237 0EC0 569 MOV ES,AX
570 E23E 28CO 571
572 --- SET UP THE INTERRUPT 5 POINTER TO A DUMMY
573 E39 26C706140054FAF R 574 MOV ES:INT5_PTR,OFFSET PRINT_SCREEN ;PRINT SCREEN
E240 26C706160000F0 R 575 MOV ES:INT5_PTR+2,CODE ;
E247 576 ; TEST THE IMR REGISTER
577 E247 FA 578 CLI ;DISABLE INTERRUPTS
E248 B000 579 MOV AL,0 ;SET IMR TO ZERO
E24A E621 580 OUT INTA01,AL 
E24C E421 581 IN AL,INTA01 ;READ IMR
E24E 0AC0 582 OR AL,AL ;IMR = 0?
E250 752B 583 JNZ D6 ;GO TO ERR ROUTINE IF NOT 0
E252 20FF 584 MOV AL,OFFH ;DISABLE DEVICE INTERRUPTS
E254 E621 585 OUT INTOA1,AL ;WRITE TO IMR:
E256 42E1 586 IN AL,INTA01 ;READ IMR
E25E 0401 587 ADD AL,1 ;ALL IMR BIT ON?
E25A T821 588 JNZ D6 ;NO - GO TO ERR ROUTINE
589 E267 FA 590 CLI ;CHECK FOR HOT INTERRUPTS
591 E2CB FC 592 CLD ;SET DIR FLAG TO GO FORWARD
E2DC B08080 593 MOV CX,8 ;SETUP TEMP INT RYN IN PRT TBL
E2D0 BF2001 594 MOV DI,OFFSET INT_PTR ;GET ADDRESS OF INT PROC TABLE
E2D3 595 MOV AX,OFFSET D11 ;VECTBL:
E266 AB 596 STOSW ;MOVE ADDR OF INTR PROC TO TBL
E267 B000F0 R 597 MOV AX,CODE ;GET ADDR OF INTR PROC SEG
E26A AB 598 STOSW
E26B 50C304 599 ADD BX,4 ;SET BX TO POINT TO NEXT VAL
E26E EF3 600 LOOP D3 ;VECTBL
E26F 601 602 603 604 ; INTERRUPTS ARE MASKED OFF. CHECK THAT NO INTERRUPTS OCCUR.
E270 32E4 605 JOR AH,AH ;CLEAR AN REG
E272 FB 606 MOV AH,AN ;CLEAR ANY REG
E273 28C9 607 SUB CX,CX ;EXTERNAL INTERRUPTS
E278 2F8E 608 HAXT 1 SEC FOR ANY INTRS THAT
E277 2F8E 609 D4: LOOP D4 ;MIGHT OCCUR
E279 0AE4 610 D5: LOOP D5 ;IDID ANY INTERRUPTS OCCUR?
E270 B400 625 MOV AH, 0 ;RESET TIMER INTR RECVD FLAG
E272 32ED 626 XOR CH,CH ;CLEAR THE CH REG
E273 B0F 627 MOV AL, 0FEH ;MASK ALL INTRs EXCEPT LV0 0
E273 B0E 628 OUT INTA01,AL ;WRITE THE B259 INR
E273 E63 629 MOV AL,00010000B ;ISEL TIM 0, L50, MODE 0, BINARY
E273 F63 630 OUT TIM.CL,AL ;SET TIMER CONTROL MODE REG
E274 8116 631 MOV CL,16H ;SET PGH LOOP CHK
E274 8AC1 632 MOV AL,CL ;SET TIMER 0 CHK REG
E275 EE40 633 OUT TIMERO,AL ;WRITE TIMER 0 CHK REG
E276 E6D 634 DSI: TEST AH,OFFH ;DID TIMER 0 INTERRUPT OCCUR?
E277 7506 635 JZ 01 2 ; YES - CHECK TIMER OF FOR SLOW TIME
E278 E6F9 636 LOOP D8 ;WAIT FOR INTR FOR SPECIFIED TIME
E279 E8ED 637 JMP D6 ;TIMER 0 INTERRUPT OCCURRENCE - ERR
E280 E812 638 MOV CL,18H ;SET PGH LOOP CHK
E281 E640 639 MOV AL,OFFH ;WRITE TIMER 0 CHK REG
E282 E641 640 OUT TIMERO,AL ;WRITE TIMER 0 CHK REG
E283 E642 641 MOV AH, 0 ;RESET INTR RECEIVED FLAG
E284 E643 642 MOV AL,OFFH ;REENABLE TIMER 0 INTRUPTS
E285 A621 643 OUT INTA01,AL
E286 F640 644 D10: TEST AH,OFFH ;DID TIMER 0 INTERRUPT OCCUR?
E287 75CD 645 JZ 01 ;YES - TIMER CHING TOO FAST, ERR
E288 E2F9 646 LOOP D10 ;WAIT FOR INTR FOR SPECIFIED TIME
E289 E93600 647 JMP TST6 ;GO TO NEXT TEST ROUTINE
E28A 6D11 648 PROC NEAR
E28B 8401 651 d11 PROC NEAR
E28C 50 652 MOV AH, 1 ;SAVE REG AX CONTENTS
E28E B0F 653 PUSH AX ;MASK ALL INTERRUPTS OFF
E28F E621 654 MOV AL,OFFH
E290 E640 655 OUT INTA01,AL
E291 9020 656 MOV AL,EOI
E292 E620 657 OUT INTA00,AL
E293 5B 658 POP AX ;RESTORE REG AX CONTENTS
E294 CF 659 IRET
E295 60 660 D11 ENDPROC
E296 6D11 661 called
E297 50 662 NT Int PROC NEAR
E298 55 663 PUSH AX ;SAVE ORIG CONTENTS OF AX
E299 E642 664 IN AL,PORT.C
E29A A640 665 TEST AL,40H ;IO CH PARITY CHECK?
E29B 7400 666 JZ D12 ;YES - FLAG IS SET TO 0
E29C 6E1FEE R 667 MOV SI,OFFSET D1 ;ADDR OF ERROR MSG
E29D B90000 668 MOV CX,DIL
E29E EB0A 669 JMP SHORT D13 ;DISPLAY ERROR MSG
E29F 670 D12: 670 TEST AL,B0H ;PLANAR RAM P-CHECK?
E29F 7410 671 JZ D14 ;IND - AUX INT
E2A0 6E2E7E R 672 MOV SI,OFFSET D2 ;ADDR OF ERROR MSG
E2A1 B90000 673 MOV CX,DIL
E2A2 E6D D13: 674 MOV AX, 0
E2A3 C0D 675 INT 10H ;INIT AND SET MODE FOR VIDEO
E2A4 E8603 676 CALL P_MSG ;CALL VIDEO IO PROCEDURE
E2A5 FA 677 CLI
E2A6 F4 678 HLT
E2A7 E61 D14: 679 RET
E2A8 5B 680 POP AX ;RESTORE ORIG CONTENTS OF AX
E2A9 CF 681 IRE1
E2A9 D14: 682 INT 10H
E2AB INT 10H
E2AC INT 10H
E2AD INT 10H
E2AE INT 10H
E2AF INT 10H
E2B0 INT 10H
E2B1 INT 10H
E2B2 INT 10H
E2B3 INT 10H
E2B4 INT 10H
E2B5 INT 10H
E2B6 INT 10H
E2B7 INT 10H
E2B8 INT 10H
E2B9 INT 10H
E2BA INT 10H
E2BB INT 10H
E2BC INT 10H
E2BD INT 10H
E2BE INT 10H
E2BF INT 10H
E2C0 INT 10H
E2C1 INT 10H
E2C2 INT 10H
E2C3 INT 10H
E2C4 INT 10H
E2C5 INT 10H
E2C6 INT 10H
E2C7 INT 10H
E2C8 INT 10H
E2C9 INT 10H
E2CA INT 10H
E2CB INT 10H
E2CC INT 10H
E2CD INT 10H
E2CE INT 10H
E2CF INT 10H
E2D0 INT 10H
E2D1 INT 10H
E2D2 INT 10H
E2D3 INT 10H
E2D4 INT 10H
E2D5 INT 10H
E2D6 INT 10H
E2D7 INT 10H
E2D8 INT 10H
E2D9 INT 10H
E2DA INT 10H
E2DB INT 10H
E2DC INT 10H
E2DD INT 10H
E2DE INT 10H
E2DF INT 10H
E2E0 INT 10H
E2E1 INT 10H
E2E2 INT 10H
E2E3 INT 10H
E2E4 INT 10H
E2E5 INT 10H
E2E6 INT 10H
E2E7 INT 10H
E2E8 INT 10H
E2E9 INT 10H
E2EA INT 10H
E2EB INT 10H
E2EC INT 10H
E2ED INT 10H
E2EE INT 10H
E2EF INT 10H
E2F0 INT 10H
E2F1 INT 10H
E2F2 INT 10H
E2F3 INT 10H
E2F4 INT 10H
E2F5 INT 10H
E2F6 INT 10H
E2F7 INT 10H
E2F8 INT 10H
E2F9 INT 10H
E2FA INT 10H
E2FB INT 10H
E2FC INT 10H
E2FD INT 10H
E2FE INT 10H
E2FF INT 10H
A-10
685 ; INITIAL RELIABILITY TEST -- PHASE 3
686 ; -----------------------------------------
687 ; ASSUME CS:CODE,DS:DATA
688 ; --------------------------

690  E20B 20323031 0004  DB  ' 201'  ' 201'
691  E20E E000  EQU  $-E0
692  E645 i ESTABLISH BIOS SUBROUTINE CALL INTERRUPT VECTORS

695 i TSTB:
696  E2EC FC  CLD  ; SET DIR FLAG TO GO FORWARD
697  E2ED B4000  MOV  DI,OFFSET VIDEO_INT  ; SETUP ADDR TO INTR AREA
698  E300 0E00AA  PUSH  CS
699  E2F1 1F  POP  DS  ; SETUP ADDR OF VECTOR TABLE
700  E2F2 BE13FF  MOV  DI,OFFSET13F ; OFFSET VECTOR_TABLE+32
701  E2F5 B92000  MOV  CX,20H
702  E2F6 F3  REP  MOVSW  ; MOVE VECTOR TABLE TO RAM
703  E2F9 A5

704 i SETUP TIMER 0 TO MODE 3
705
706  E2FA B0FF  MOV  AL,OFFH  ; DISABLE ALL DEVICE INTERRUPTS
707  E2FC E621  OUT  INTAD1,AL
708  E2FE B056  MOV  AL,36H  ; SET TIM 0-LSB,MSB,MODE 3
709  E300 E643  OUT  TIMER+3,AL  ; WRITE TIMER MODE REG
710  E302 B000  MOV  AL,0
711  E304 E640  OUT  TIMER,AL  ; WRITE LSB TO TIMER 0 REG
712  E306 E640  OUT  TIMER,AL  ; WRITE MSB TO TIMER 0 REG

713 i SETUP TIMER 0 TO BLINK LED IF MANUFACTURING TEST MODE
714
715  E308 B04000  R  MOV  AX,DATA  ; POINT DS TO DATA SEG
716  E308 008B  MOV  DS,AX
717  E30B 0083  CALL  KBD_RESET  ; SOFTWARE KEYBOARD INIT
718  E310 000AA  CMP  BL,0AAH  ; SCAN CODE AA' RETURNED?
719  E315 7426  JE  E3  ; YES - CONTINUE (NON MFG MODE)
720  E315 7421  MOV  AL,0FH  ; ENABLE INTERRUPT
721  E315 7422  OUT  PORT_B,AL
722  E317 E641  MOV  AX,DATA  ; Validator TSR
723  E319 90  MOV  DS,AX
724  E31A 90  MOV  DS,AX
725  E31B E640  MOV  DI,PORT_A  ; WAS A BIT CLOCKED IN?
726  E31D 24FF  IN  AL,PORT_A
727  E31F 7516  JNZ  E2  ; YES - CONTINUE (NON MFG MODE)
728  E321 FE061200  R  INC  MFG_TST  ; ELSE SET SW FOR MFG TEST MODE
729  E325 26C706200000265  R  MOV  ES:INT_ADDR,OFFSET BLINK_INT  ; SETUP TIMER INTR TO BLINK LED
730  E32C 26C70620000082E  R  MOV  ES:INT_ADDR+2,CODE  ; ENABLE INTERRUPT
731  E333 80FE  MOV  ES:INT_ADDR+2,AL  ; SETUP TIMER INTO BLINK LED
732  E335 E621  OUT  INTAD1,AL
733  E337 7426  JE  E3  ; YES - CONTINUE
734  E337 7422  MOV  AL,0CH  ; RESET THE KEYBOARD
735  E339 E641  MOV  AX,DATA  ; POINT DS TO DATA SEG
736  E339 7421  MOV  DS,AX
737  E33A 7420  MOV  DS,AX

738 i TSTF.OS
739  E33B B204  MOV  DL,4  ; NO. OF ROS MODULES TO CHECK
740  E33B E00060  MOV  BX,6000H  ; SETUP STARTING ROS ADDR
741  E340 E4  CALL  ROS_CHECKSUM  ; CHECK/ros
742  E345 7507  JNE  E3  ; A CHECKSUM IS DONE FOR THE 4 ROS MODULES CONTAINING BASIC CODE

743 i ES:
744  E33B B204  MOV  DL,4  ; NO. OF ROS MODULES TO CHECK
745  E33D E00060  MOV  BX,6000H  ; SETUP STARTING ROS ADDR
746  E340 E4  CALL  ROS_CHECKSUM  ; CHECK/ros
747  E345 7507  JNE  E3  ; A CHECKSUM IS DONE FOR THE 4 ROS MODULES CONTAINING BASIC CODE

748  E345 7507  JNE  E3  ; A CHECKSUM IS DONE FOR THE 4 ROS MODULES CONTAINING BASIC CODE
749  E346 FECA
750  E347 75F7  JNZ  E4  ; ANY MORE TO DO?
751  E349 EB0790  JNP  E6  ; YES - CONTINUE
752  E34C 7522  JNZ  ERR_BEEP  ; GO TO NEXT TEST
753  E34C B40101  MOV  DX,10H  ; ROS_ERROR:
754  E34F E0EC02  CALL  ERR_BEEP  ; BEEP SPEAKER
E532 764 E532 E460 765 IN AL,PORT_A ; READ SENSE SWITCHES
E534 B400 766 MOV AX,0
E535 A31000 R 767 MOV EQUF_FLAG,AX ; STORE SENSE SW INFO
E539 2630 768 AND AL,3OH ; ISOLATE VIDEO SMS
E53B 7503 769 JNZ E7 ; Video SMS SET TO 0?
E53D E99000 770 JNP E19 ; SKIP VIDEO TESTS FOR BURN-IN
E540 771 E7: XCHG AH,AL ; TEST VIDEO:
E546 8E0 772 MOV AX,0 ; B/W CARD ATTACHED?
E546 800C30 773 CMP AH,3OH ; YES - SET MODE FOR B/W CARD
E546 7409 774 JE E8 ; SET COLOR MODE FOR COLOR CD
E546 FECC0 775 MOV AX,AL ; B/W VIDEO CARD ATTACHED?
E546 7502 776 JNE E8 ; YES - GO TEST VIDEO STG
E546 E8003 777 MOV AX,AL+3 ; SET MODE FOR 4X25
E570 779 E9: MOV AX,0 ; YES - SET MODE FOR COLOR CD
E570 80B000 780 MOV BX,0B000H ; BEG VIDEO RAM ADDR B/W CD
E57A BABB03 781 MOV DX,380H ; MODE REG FOR B/W
E57B 800010 782 MOV CX,4096 ; RAM BYTE CNT FOR B/W CD
E57C 00FC30 783 MOV DX,DX ; MODE SELECTED?
E585 7408 784 MOV AX,E9 ; B/W VIDEO CARD ATTACHED?
E586 8000B0 785 JE E7 ; YES - GO TEST VIDEO STG
E586 840003 786 MOV CX,4096 ; MODE REG FOR COLOR CD
E590 FED4 787 MOV AX,10 ; B/W MODE FOR B/W CARD
E593 004000 788 MOV CX,4000H ; RAM BYTE CNT FOR COLOR CD
E594 50 789 DEC AX ; SET MODE TO 0 FOR COLOR CD
E595 09 790 MOV AX,0 ; TEST_VIDEO_STG:
E59E EE 791 OUT DX,AL ; RESTORE VIDEO SENSE SMS IN AH
E59E 8000 792 MOV AX,700H ; WRT BLANKS IN VIDEO
E59F A8000 793 SUB DI,DI ; SETUP STARTING LOC
E5A0 826000 794 MOV CX,4000H ; NO. OF BLANKS TO DISPLAY
E5A3 0F01 795 REP MOV BX,1234H ; FOO INITIATED BY KBD RESET?
E5A4 7400 796 JE E10 ; YES - SKIP VIDEO RAM TEST
E5A6 E8DB 797 MOV DS,DX ; POINT DS TO VIDEO RAM STG
E5A7 EE7FC 798 CALL STATST_CNT ; POINT DS TO VIDEO RAM STG
E5A8 7406 799 JE E10 ; IDG TEST VIDEO RAM STG
E5A9 8400 800 MOV CX,120H ; 1STG OK - CONTINUE TESTING
E5A9 8400 801 CALL ERR_BEEP ; SETUP B OF BEEPS
E5AC E08102 802 CALL ERR_BEEP ; BEEPER SPEAKER

007 E10: POP AX ; SET VIDEO SENSE SMS (AN)
010 PUSH AX ; SAVE IT
011 MOV AX,0 ; ENABLE VIDEO AND SET MODE
012 MOV AX,10H ; VIDEO
013 MOV AX,7020H ; NOT BLANKS IN REVERSE VIDEO
014 SUB DI,DI ; SETUP STARTING LOC
015 MOV CX,40 ; NO. OF BLANKS TO DISPLAY
016 MOV CX,40 ; WRITE VIDEO STORAGE

018
LOC  OBJ  line  source

824 | ------------------------------------- -------------------
825 | GET VIDEO SENSE SW INFO
826 | I:TEST.10
827 |-:DESCRIPTION
828 | I: SENSE ON/OFF TRANSITION OF THE VIDEO ENABLE AND HORIZONTAL
829 | I: SYNC LINES.
830 | I:------------------------

831 | CRTC INTERFACE LINES TEST
832 | DESCRIPTION
833 | SENSE ON/OFF TRANSITION OF THE VIDEO ENABLE AT HORIZONTAL SYNCHRONIZATION LINES.
834 | ;----------------------------------------------------------
835 | GET VIDEO SENSE SW INFO
836 | ;SAVE IT
837 | ;OFLOOP_CNT:
838 | GET VIDEO SENSE SW INFO
839 | ;SAVE IT
840 | ;OFLOOP_CNT:

841 | ;---------
842 | GET VIDEO SENSE SW INFO
843 | ;SAVE IT
844 | ;OFLOOP_CNT:

845 | ;---------
846 | GET VIDEO SENSE SW INFO
847 | ;SAVE IT
848 | ;OFLOOP_CNT:

849 | ;---------
850 | GET VIDEO SENSE SW INFO
851 | ;SAVE IT
852 | ;OFLOOP_CNT:

853 | ;---------
854 | GET VIDEO SENSE SW INFO
855 | ;SAVE IT
856 | ;OFLOOP_CNT:

857 | ;---------
858 | GET VIDEO SENSE SW INFO
859 | ;SAVE IT
860 | ;OFLOOP_CNT:

861 | ;---------
862 | GET VIDEO SENSE SW INFO
863 | ;SAVE IT
864 | ;OFLOOP_CNT:

865 | ;---------
866 | GET VIDEO SENSE SW INFO
867 | ;SAVE IT
868 | ;OFLOOP_CNT:

869 | ;---------
870 | GET VIDEO SENSE SW INFO
871 | ;SAVE IT
872 | ;OFLOOP_CNT:

873 | ;---------
874 | GET VIDEO SENSE SW INFO
875 | ;SAVE IT
876 | ;OFLOOP_CNT:

877 | ;---------
878 | GET VIDEO SENSE SW INFO
879 | ;SAVE IT
880 | ;OFLOOP_CNT:

881 | ;---------
882 | GET VIDEO SENSE SW INFO
883 | ;SAVE IT
884 | ;OFLOOP_CNT:

885 | ;---------
886 | GET VIDEO SENSE SW INFO
887 | ;SAVE IT
888 | ;OFLOOP_CNT:

889 | ;---------
890 | GET VIDEO SENSE SW INFO
891 | ;SAVE IT
892 | ;OFLOOP_CNT:

893 | ;---------
894 | GET VIDEO SENSE SW INFO
895 | ;SAVE IT
896 | ;OFLOOP_CNT:

897 | ;---------
898 | GET VIDEO SENSE SW INFO
899 | ;SAVE IT
900 | ;OFLOOP_CNT:

901 | ;---------
902 | GET VIDEO SENSE SW INFO
903 | ;SAVE IT
904 | ;OFLOOP_CNT:

905 | ;---------
906 | GET VIDEO SENSE SW INFO
907 | ;SAVE IT
908 | ;OFLOOP_CNT:

909 | ;---------
910 | GET VIDEO SENSE SW INFO
911 | ;SAVE IT
912 | ;OFLOOP_CNT:

913 | ;---------
914 | GET VIDEO SENSE SW INFO
915 | ;SAVE IT
916 | ;OFLOOP_CNT:

917 | ;---------
918 | GET VIDEO SENSE SW INFO
919 | ;SAVE IT
920 | ;OFLOOP_CNT:

921 | ;---------
922 | GET VIDEO SENSE SW INFO
923 | ;SAVE IT
924 | ;OFLOOP_CNT:

925 | ;---------
926 | GET VIDEO SENSE SW INFO
927 | ;SAVE IT
928 | ;OFLOOP_CNT:

929 | ;---------
930 | GET VIDEO SENSE SW INFO
931 | ;SAVE IT
932 | ;OFLOOP_CNT:

933 | ;---------
934 | GET VIDEO SENSE SW INFO
935 | ;SAVE IT
936 | ;OFLOOP_CNT:

937 | ;---------
938 | GET VIDEO SENSE SW INFO
939 | ;SAVE IT
940 | ;OFLOOP_CNT:

941 | ;---------
942 | GET VIDEO SENSE SW INFO
943 | ;SAVE IT
944 | ;OFLOOP_CNT:

945 | ;---------
946 | GET VIDEO SENSE SW INFO
947 | ;SAVE IT
948 | ;OFLOOP_CNT:

949 | ;---------
950 | GET VIDEO SENSE SW INFO
951 | ;SAVE IT
952 | ;OFLOOP_CNT:

953 | ;---------
954 | GET VIDEO SENSE SW INFO
955 | ;SAVE IT
956 | ;OFLOOP_CNT:

957 | ;---------
958 | GET VIDEO SENSE SW INFO
959 | ;SAVE IT
960 | ;OFLOOP_CNT:

961 | ;---------
962 | GET VIDEO SENSE SW INFO
963 | ;SAVE IT
964 | ;OFLOOP_CNT:

965 | ;---------
966 | GET VIDEO SENSE SW INFO
967 | ;SAVE IT
968 | ;OFLOOP_CNT:

969 | ;---------
970 | GET VIDEO SENSE SW INFO
971 | ;SAVE IT
972 | ;OFLOOP_CNT:

973 | ;---------
974 | GET VIDEO SENSE SW INFO
975 | ;SAVE IT
976 | ;OFLOOP_CNT:

977 | ;---------
978 | GET VIDEO SENSE SW INFO
979 | ;SAVE IT
980 | ;OFLOOP_CNT:

981 | ;---------
982 | GET VIDEO SENSE SW INFO
983 | ;SAVE IT
984 | ;OFLOOP_CNT:

985 | ;---------
986 | GET VIDEO SENSE SW INFO
987 | ;SAVE IT
988 | ;OFLOOP_CNT:

989 | ;---------
990 | GET VIDEO SENSE SW INFO
991 | ;SAVE IT
992 | ;OFLOOP_CNT:

993 | ;---------
994 | GET VIDEO SENSE SW INFO
995 | ;SAVE IT
996 | ;OFLOOP_CNT:

997 | ;---------
998 | GET VIDEO SENSE SW INFO
999 | ;SAVE IT
1000 | ;OFLOOP_CNT:

...
LOC OBJ  
LINE  
SOURCE  

E42B 7440  
900  
JE  
E22  
;YES - SKIP MEMORY TEST  
901  
902  
903  

E42D BB0004  
904  
MOV  
BX,400H  
905  
906  

E43D BD9100  
907  
908  
909  
910  

E43E B8E4  
911  
ADD  
CX,16  
912  
ADD  
BX,400H  
913  
914  

E442 53  
915  
916  
917  

E443 52  
918  
919  
920  

E445 E00F0B  
921  
922  

E446 5A  
923  
924  

E447 58  
925  
926  

E448 59  
927  
928  

E449 7466  
929  
930  

E44A 80C0  
931  
932  

E44B 860F  
933  
934  

E44C 860F  
935  
936  

E44E E02700  
937  
938  

E44F 8900  
939  
940  

E450 E94A00  
941  
942  

E451 E967  
943  
944  

E452 E7D0  
945  
946  

E453 B04000  
947  
948  

E454 B04000  
949  
950  

E455 B04000  
951  
952  

E456 7F67  
953  
954  

E457 B00010  
955  
956  

E458 E96B  
957  
958  

E459 7440  
959  
960  

E460 E98B  
961  
962  

E461 E99B  
963  
964  

E462 E9A9  
965  
966  

E463 E9B0  
967  
968  

E464 E9B1  
969  
970  

E465 E9C1  
971  
972  

E466 E9C3  
973  
974  

E467 E9D7E4  
975  
976  

E468 E9E7  
977  
978  

E469 E9F0  
979  
980  

E470 E9F7  
981  
982  

E471 E9F8  
983  
984  

E472 E9F9  
985  
986  

E473 E9FA  
987  
988  

E474 E9FB  
989  
990  

E475 E9FC  
991  
992  

A-14
LOC OBJ LINE SOURCE

974 ; INITIAL RELIABILITY TEST -- PHASE 4
976 ;
977 ASSUME CS:CODE,DS:DATA

E4A7 20333031
E4AB 313331
E4AE 363031
E4B1 BC03
E4B3 7003
E4B5 7002
E4B7 30313233343536
990 ASCII_TBL DB '0123456789ABCDEF'

991 |
992 |
993 KEYBOARD TEST
994 |
995 |
996 |
997 |
E4C7 998 |
E4C7 B04000
E4CA 8ED0
E4C7 E03E120001 R
E4D1 7439
E4D3 E0B201
E4D6 E32B
E4D7 E8B201 MOV AX,DATA ;POINT OS TO DATA SEG
E4D8 B04D MOV AL,4DH ; ENABLE KEYBOARD
E4E1 B04C MOV AL,4CH IENABLE KBD,ELK IN NEXT BYTE
E4E3 B04C OUT PORT_B,AL
E4E5 907BAA MOV AL,BLAAM ;SCAN CODE AS EXPECTED?
E4E7 7002 MOV JF F6 ;NO - DISPLAY ERROR MSG
E4E7 907BAA MOV E7,AL .SAVE SCAN CODE
E4F3 8AE8 MOV eH,Al .SAVE SCAN CODE
E4F5 B904 MOV Cl,4
E4F7 E89CFF CALL XLAT_PRINT_CODE ;CONVERT AND PRINT
E4F9 E895FF MOV Al.CH JRECOVER SCAN CODE
E4F9 240F MOV Ax,Al
E4F9 B90400 MOV CX,FIL
E4F9 E08E01 CALL P_MSG ;PRINT MSG ON SCREEN

1032 |
1033 SETUP INTERRUPT VECTOR TABLE
1034 |
1035 |
1036 |
1037 |
1038 |
1039 |
1040 |
1041 |
1042 |
1043 |
1044 |

A-15
1045 ;-----------------------------------------------------------------------
1046 ;TEST.13
1047 ;DESCRIPTION
1048 ;turn cassette motor off, write a bit out to the cassette data bus.
1049 ;verify that cassette data read is within a valid range.
1050 ;-----------------------------------------------------------------------
1051 ;
1052 ...
1053 ; turn the cassette motor off
1054 ...
1055 mov ax, data
1056 mov ds, ax
1057 mov al, 040h
1058 out port_0, al
1059 ;write a bit
1060 ...
1061 mov al, 0ffh
1062 mov al, offh
1063 out int_0, al
1064 mov al, bh
1065 xor cx, cx
1066 call read_half_bit
1067 call read_half_bit
1068 mov bx, max_period
1069 xor bx, bx
1070 mov bx, min_period
1071 ...
E585 1501 1122 MOV CH,1 ;SELECT TRACK 1
E587 0100300 R 1123 MOV SEEK_STATUS,DL ;RECALLIBRATE DISKETTE
E588 00500 1124 CALL SEEK ;GO TO ERR SUBROUTINE IF ERR
E589 7207 1125 JC F13 ;GO TO ERR
E590 B522 1126 MOV CH,34 ;SELECT TRACK 34
E592 00008 1127 CALL SEEK ;SEEK TO TRACK 34
E595 7309 1128 JNC F14 ;OK, TURN MOTOR OFF
E597 00000 R 1129 MOV BX OFFSET F3 ;GET ADDR OF MSG
E599 090300 1131 MOV CX,3F3L ;GET MSG BYTE COUNT
E59D 00201 1132 CALL P.MSG ;GO PRINT ERROR MSG
E59E 1134 l TURN DRIVE 0 MOTOR OFF
E5A0 1136 l POP ES
E5A1 0000300 R 1137 MOV AL,0CH ;TURN DRIVE 0 MOTOR OFF
E5A5 00000 1138 MOV DX,03F2H ;FOC CTL ADDRESS
E5AB 1141 l SETUP PRINTER AND RS232 BASE ADDRESSES IF DEVICE ATTACHED
E5A6 1143 F15: l JMP BOOT:
E5AC 000000 R 1144 MOV BUFFER_HEAD,OFFSET KB_BUFFER ;SETUP KEYBOARD PARAMETERS
E5BD 00000 1145 MOV BUFFER_TAIL,OFFSET KB_BUFFER ;
E5BE 00000 1146 MOV BX OFFSET F4 ;PRT_SRC_TBL
E5B8 1147 MOV SI,0 ;
E5BB 1148 MOV BX OFFSET F3 ;PRT_BASE:
E5BC 000000 1149 MOV DX,CS:[BP] ;GET PRINTER BASE ADDR
E5BC 00000 1150 MOV AL,0AH ;WRITE DATA TO PORT A
E5BD 00000 1151 OUT DX,AL ;READ PORT A
E5BF 00000 1152 SUB AL,AL ;DATA PATTERN SAME
E5C1 EC 1153 MOV AL,0AH ;READ PORT A
E5C2 00000 1154 CMP AL,0AH ;DATA PATTERN SAME
E5C4 00000 1155 JNE F14 ;END - CHECK NEXT PRT CD
E5C6 000000 R 1156 MOV PRINTER_BASE[SI],DX ;YES - STORE PRT BASE ADDR
E5CA 00000 1157 INC SI ;INCREMENT TO NEXT WORD
E5CB 00000 1158 INC SI ;
E5CC 1159 F17: l NO_STORE:
E5CD 00000 1160 INC BP ;POINT TO NEXT BASE ADDR
E5CE 00000 1161 INC BP ;
E5D4 00000 1162 MOV BP,OFFSET F4E ;ALL POSSIBLE ADDRS CHECKED?
E5D5 00000 1163 JNE F16 ;
E5D6 00000 1164 MOV BX,0 ;POINTER TO RS232 TABLE
E5D7 BF403 1165 MOV DX,3FAH ;CHECK IF RS232 CD 1 ATTCH?
E5D9 EC 1166 IN AL,DX ;READ INTR ID REG
E5D9 00000 1167 TEST AL,0FH ;READ INTR ID REG
E5DA 00000 1168 JNE F16 ;
E5E0 00000 1169 MOV RS232_BASE[BX],0FH ;SETUP RS232 CD #1 ADDR
E5E1 00000 1170 MOV BX ;
E5E2 00000 1171 INC BX ;SETUP RS232 CD #2
E5E3 43 1172 INC BX ;
E5E4 43 1173 INC BX ;
E5E7 01002 1174 TEST AL,0FH ;READ INTERRUPT ID REG
E5E8 EC 1175 INC BX ;READ INTERRUPT ID REG
E5E9 00000 1176 MOV RS232_BASE[BX],0FH ;SETUP RS232 CD #2
E5F5 43 1177 INC BX ;
E5F6 43 1178 INC BX ;
E5F7 1179 l----- SET UP EQUIP FLAG TO INDICATE NUMBER OF PRINTERS AND RS232 CARDS
E5F8 1180 l BASE_END:
E5F9 00000 1181 MOV AX,51 ;SI HAS 2 NUMBER OF RS232
E5FA 00103 1182 MOV CL,3 ;SHIFT COUNT
E5FB 01002 1183 MOV AL,CL ;SHIFTED COUNT
E5FC 01002 1184 MOV AL,CL ;ROTATE RIGHT 3 POSITIONS
E5FD 0AC3 1185 OR AL,DL ;OR IN THE PRINTER COUNT
E5FF 0A1100 R 1186 MOV BYTE PTR EQUIP_FLAG+1,AL ;STORE AS SECOND BYTE
E602 01002 1187 MOV DX,201H ;
E603 EC 1188 IN AL,DX ;READ PORT A
E604 AD0F 1189 TEST AL,0FH ;
E605 7506 1190 INC BX ;
E606 7506 1191 JNZ F20 ;NO GAME CARD
E607 000110010 R 1192 OR BYTE PTR EQUIP_FLAG+1,AL ;NO GAME CARD
E60F 1193 F20: l NO GAME CARD:
E610 00000 1194 l ENABLE NMI INTERRUPTS
E615 00000 1195 l ENABLE NMI INTERRUPTS
E619 E640 1196 MOV AL,0DH ;
E61E 1197 OUT DX,AL ;

APPENDIX A
LOC OBJ            LINE       SOURCE
E613 803120001 R  1199  CMP MFG_TST.1          MFG MODE?
E616 7406          1200  JE F21               LOAD_BOOT_Strap
E61A B4D100        1201  MOV DX.1            ;
E61C 0100          1202  CALL ERR_BEEP       ;BEEP 1 SHORT TONE:
E620              1203  F21: JMP BOOT_Strap    ;GO TO THE BOOT LOADER
E629              1204  CMP HFG_Tst.1        ;MANUFACTURING TEST MODE?
E62B              1205  CMP MFG.TST.1         ;MFG MODE?
E62E              1206  CMP MFG.TST.1         ;MANUFACTURING TEST MODE?
E638 7003          1207  JNE F23              ;DO - GO TO BOOT LOADER
E64A E92EFA        1208  JMP START            ;YES - LOOP POWER-ON-DIAGS
E62D              1209  F23:                   ;GO TO BOOT:
E62O              1210  JMP F15              ;
E630              1225  ERR_BEEP PROC NEAR  ;SAVE FLAGS
E633 9C            1226  PUSHF               ;DISABLE SYSTEM INTERRUPTS
E631 FA            1227  CLI                 ;SAVE DS RD CONTENTS
E632 1E            1228  PUSH DS               ;POINT DS TO DATA SEG
E635 804001 R      1229  MOV AX,DATA         ;ASSUME CS:CODE,DS:DATA
E636 84DD          1230  MOV DS,AX            ;SUBROUTINES FOR POWER ON DIAGNOSTICS
E638 046F          1231  OR DH,DH              ;ANY LONG ONES TO BEEP
E63A 7410          1232  JZ G3                ;NO, DO THE SHORT ONES
E63C              1233  G1:                   ;LONG_BEEP:
E63C B306          1234  MOV BL,6             ;COUNTER FOR BEEPS
E63E 802500        1235  CALL BEEP             ;DO THE BEEP
E641 E2F0          1236  G2: LOOP G2          ;DELAY BETWEEN BEEPS
E643 FFC0          1237  DEC DH               ;ANY MORE TO DO
E645 75F5          1238  JNZ G1               ;DO IT
E647 803120001 R   1239  CMP MFG.TST.1        ;MFG TEST MODE?
E64C 7006          1240  JNE G3               ;YES - CONTINUE BEEPING SPEAKER
E64E 80C0          1241  MOV AL,00CH         ;STOP BLINKING LED
E650 E661          1242  OUT PORT_B.AL       ;SHORT_BEEP:
E652 E608          1243  JMP SHORT G1        ;
E656              1244  G3:                   ;
E659 B301          1245  MOV BL,1             ;COUNTER FOR A SHORT BEEP
E666 80D00         1246  CALL BEEP             ;DO THE SOUND
E659 E2F0          1247  G4: LOOP G4          ;DELAY BETWEEN BEEPS
E65B FECA          1248  DEC DL               ;DONE WITH SHORTS
E65D 75F5          1249  JNZ G3               ;DO SOME MORE
E661 E2F0          1250  GS: LOOP G5          ;LONG DELAY BEFORE RETURN
E661 E2F1          1251  Ge: LOOP G6          ;
E663 IF            1252  POP DS               ;RESTORE DS RC DS SEG
E664 9D            1253  POPF                ;RESTORE FLAGS TO ORIG SETTINGS
E665 C3            1254  RET                 ;RETURN TO CALLER
E666              1255  ERR_BEEP ENDP
E666 80B6          1256  BEEP PROC NEAR      ;
E668 E643          1257  MOV AL,10110110B     ;SEL TIM 2, LSB, MFG, BINARY
E66A 803505        1258  OUT TIMER+3,AL      ;WRITE THE TIMER MODE REG
E66A E642          1259  MOV AL,533H         ;DIVISOR FOR 1000 Hz
E66B 8AC0          1260  OUT TIMER+2,AL      ;WRITE TIMER 2 CNT - LSB
E671 E642          1261  MOV AL,1AH          ;WRITE TIMER 2 CNT - MSB
E673 E641          1262  IN AL,PORT_B       ;GET CURRENT SETTING OF PORT
E675 BAE0          1263  MOV AH,AL          ;SAVE THAT SETTINGS
E677 8C01          1264  OR AL,03            ;TURN SPEAKER ON
E679 E661          1265  OUT PORT_B,AL      ;SET CNT TO WAIT 500 MS
E67B DECF          1266  SUB CX,CX            ;
E67D E2F0          1267  LOOP G7             ;DELAY BEFORE TURNING OFF
E67F DECB          1268  DEC BL              ;DELAY CHT EXPRED?
E681 75FA          1269  JNZ G7              ;IND - CONTINUE BEEPING SPK
E683 8AC4          1270  MOV AL,1AH         ;RECOVER VALUE OF PORT
A-18
LOC OBJ | LINE | SOURCE

E685 E661 | 1275 | OUT PORT_B,AL
E667 C3 | 1276 | RET
E683 E661 | 1277 | BEEP ENDP
E685 | |----------------------------------------
E686 | 1279 | ; THIS PROCEDURE WILL SEND A SOFTWARE RESET TO THE KEYBOARD.
E687 C3 | 1280 | ; SCAN CODE AA' SHOULD BE RETURNED TO THE CPU.
E688 | 1281 | ----------------------------------------
E688 BOOC | AL,OCH ;SET KBD CLK LINE LOW
E690 | 1282 | PORT.B,AL ;WRITE 8255 PORT B
E692 E2FE | 1283 | MOY CX, 10582 ;HOLD KBD CLK LOW FOR 20 MS
E695 | 1284 | SCAN CODE AA' SHOULD BE RETURNED TO THE CPU.
E697 Boce | 1285 | G8: LOOP G8 ,LOOP FOR 20 MS
E699 Boce | 1286 | MOV AL,OCCH ,SET CLK, ENABLE LINES HIGH
E69B E661 | 1287 | OUT PORT_B,AL
E69D | 1288 | SP _TEST: I ENTRY FOR MANUFACTURING TEST 2
E69F E2FE | 1289 | MOV AL,OCCH ,SET KBD CLK HIGH, ENABLE LOW
E6A1 E661 | 1290 | OUT PORT_B,AL
E6A3 B04C | 1291 | JNE G12 ;NO - DISPLAY ERROR MSG
E6A5 7502 | 1292 | IN AL,PORT_A ;READ CURRENT VAL OF PORT B
E6A7 E2F9 | 1293 | AND AL,0FH
E6A9 E661 | 1294 | OUT INTAD0.AL ;WRITE 0259 IMR
E6AD E2FE | 1295 | GI}: LOOP G11
E6B0 | 1296 | OR AL,40H iSTOP BLINKING LED
E6B2 28C9 | 1297 | OUT PORT_B,AL
E6B4 | 1298 | POP AX ,RESTORE AX REG
E6B5 58 | 1299 | POP CX !RESTORE CX REG
E6B6 CF | 1300 | IRET
E6B8 FB | 1301 | BLINK LED PROCEDURE FOR MFG BURN-IN AND RUN-IN TESTS
E6B9 S1 | 1302 | (LED WILL BLINK APPROXIMATELY .25 SECOND)
E6BD | 1303 | ----------------------------------------
E6C0 | 1304 | KBD_RESET ENDP
E6C1 2E8A04 | 1305 | iRETURN TO CALLER
E6C3 8E08 | 1306 | i--------------------------
E6C4 | 1307 | THIS SUBROUTINE WILL PRINT A MESSAGE ON THE DISPLAY
E6C5 803£120001 | 1308 | ; ENTRY REQUIREMENTS:
E6C6 | 1309 | SI = OFFSET ADDRESS) OF MESSAGE BUFFER
E6C7 840£ | 1310 | CX = MESSAGE BYTE COUNT
E6C8 | 1311 | MAXIMUM MESSAGE LENGTH IS 36 CHARACTERS
E6CA | 1312 | ----------------------------------------
E6CB B84000 | 1313 | P_MSG PROC NEAR
E6CD 8E6B | 1314 | iPOINT DS TO DATA SEG
E6CF 003E120001 | 1315 | iMSG TEST MSG?
E6D0 7505 | 1316 | JNE G12 ;IND - DISPLAY ERROR MSG
E6D2 B601 | 1317 | MOV DH,1 ;YES - SETUP TO BEEP SPEAKER
E6D4 E95FF | 1318 | JMP ERP_BEEP ;YES - BEEP SPEAKER
E6D6 | 1319 | ----------------------------------------
E6D8 | 1320 | G12: iWRITE MSG:
E6DA 2E8A04 | 1321 | MOV AL,C5:ISI ;PUT CHAR IN AL
E6DB 46 | 1322 | INC SI
E6DF B700 | 1323 | MOV AH,0 ;SET PAGE # TO ZERO
E6E1 B40E | 1324 | MOV AH,16
E6E3 CD10 | 1325 | INT 10H
E6E5 E2F4 | 1326 | LOOP G12
E6E7 B600DE | 1327 | CONTINUE TILL MSG WRITTEN
E6E8 CD10 | 1328 | MOV AX,0000
E6E9 CD10 | 1329 | ISEND CARRIAGE RETURN AND

A-19
1351 MOV AX,0EOAH ;LINE FEED CHAR
1352 INT 10H
1353 RET
1354 P_MSG ENDP
1355 \--- INT 19 ---------------
1356 \---BOOT STRAP LOADER-------
1357 \---IF A 5 1/4" DISKETTE DRIVE IS AVAILABLE
1358 \---ON THE SYSTEM, TRACK 0, SECTOR 1 IS READ INTO THE
1359 \---BOOT LOCATION (SEGMENT 0, OFFSET 7CO0).
1360 \---AND CONTROL IS TRANSFERRED THERE.
1361 \---IF THERE IS NO DISKETTE DRIVE, OR IF THERE IS
1362 \---IS A HARDWARE ERROR CONTROL IS TRANSFERRED
1363 \---TO THE CASSETTE BASIC ENTRY POINT.
1364 \---I
1365 \---I PL ASSUMPTIONS
1366 \---I 0255 PORT 60M BIT 0
1367 \---I = 1 IF IPL FROM DISKETTE
1368 \---I
1369 \---I
1370 ASSUME CS:C6D4,DS:DATA
1371 "BOOT Strap" PROC NEAR
1372 "---STI ENABLE INTERRUPTS
1373 MOV AX,DATA ; ESTABLISH ADDRESSING
1374 MOV DS,AX
1375 "---MOV AX,EQUIP_FLAG ; GET THE EQUIPMENT SWITCHES
1376 MOV AX,EQUIP_FLAG
1377 TEST AL,01 ; ISOLATE IPL SENSE SWITCH
1378 JZ H3 ; GO TO CASSETTE BASIC ENTRY POINT
1379 "---MUST LOAD SYSTEM FROM DISKETTE -- CX HAS RETRY COUNT
1380 L003 MOV CX,4 ; SET RETRY COUNT
1381 MOV AX,SEGMENT IPL SYSTEM
1382 STI ; ENABLE INTERRUPTS
1383 "---PUSH CX ; SAVE RETRY COUNT
1384 MOV AH,0 ; RESET THE DISKETTE SYSTEM
1385 "---INT 13H ; DISKETTE IO
1386 INT 13H
1387 JC H2 ; IF ERROR, TRY AGAIN
1388 "---READ IN THE SINGLE SECTOR
1389 MOV AX,0 ; TO THE BOOT LOCATION
1390 MOV ES,AX
1391 "---MOV BX,0 ; TO THE BOOT LOCATION
1392 MOV BX,0 ; DRIVE 0, HEAD 0
1393 MOV CX,1 ; SECTOR 1, TRACK 0
1394 MOV AL,1 ; READ ONE SECTOR
1395 INT 13H ; DISKETTE IO
1396 HZ: POP CX ; RECOVER RETRY COUNT
1397 JC H4 ; CF SET BY UNSUCCESSFUL READ
1398 "---DO IT FOR RETRY TIMES
1399 LOOP H1
1400 "---UNABLE TO IPL FROM THE DISKETTE
1401 H3:CASSETTE_JUMP:
1402 INT 10H ; USE INTERRUPT VECTOR TO GET TO BASIC
1403 ; I PL WAS SUCCESSFUL
1404 "---IPL
1405 H4:
1406 "---E722 CD16
1407 JMP BOOT_LOCK
1408 "---BOOT STRAP ENDP
1409 "---INT 14 ---
1410 "---RS232 IO
1411 THIS ROUTINE PROVIDES BYTE STREAM I/O TO THE COMMUNICATIONS
1412 PORT ACCORDING TO THE PARAMETERS:
1413 (AH)=0 INITIALIZE THE COMMUNICATIONS PORT
1414 I ALL HAS PARMS FOR INITIALIZATION
1415 | I
1416 | 7 6 5 4 3 2 1 0
1417 | --- BAUD RATE -- --PARITY-- STOPBIT --WORD LENGTH--
1418 | 112;----
1419 | 1420 | 000 - 110 X0 = NONE 0 - 1 10 - 7 BITS
1419 | 001 - 150 01 = C0D 2 - 11 7 - 8 BITS
1420 | 010 - 309 11 = EVEN
1421 | 011 - 600
1422 | 100 - 1200
1423 | 101 - 2400
1424 | 110 - 4800
1425 | 111 - 9600
1426 | A-20
LOC  OBJ  LINE  SOURCE
1428  1  ON RETURN, CONDITIONS SET AS IN CALL TO COMPO STATUS (AH=3)
1429  1  (AH)=1  SEND THE CHARACTER IN (AL) OVER THE COMPO LINE
1430  1  (AH) REGISTER IS PRESERVED
1431  1  ON EXIT, BIT 7 OF AH IS SET IF THE ROUTINE WAS UNABLE TO
1432  1  TRANSMIT THE BYTE OF DATA OVER THE LINE. THE
1433  1  REMAINDER OF AH IS SET AS IN A STATUS REQUEST,
1434  1  REFLECTING THE CURRENT STATUS OF THE LINE.
1435  1  (AH)=2  RECEIVE A CHARACTER IN (AL) FROM COMPO LINE BEFORE
1436  1  RETURNING TO CALLER
1437  1  ON EXIT, AH HAS THE CURRENT LINE STATUS, AS SET BY THE
1438  1  THE STATUS ROUTINE, EXCEPT THAT THE ONLY BITS
1439  1  LEFT ON ARE THE ERROR BITS (7,6,5,4,3,2,1)
1440  1  IN THIS CASE, THE TIME OUT BIT INDICATES DATA SET
1441  1  READY WAS NOT RECEIVED.
1442  1  THUS, AH IS NON ZERO ONLY WHEN AN ERROR OCCURRED.
1443  1  (AH)=3  RETURN THE COMPO PORT STATUS IN (AX)
1444  1  AH CONTAINS THE LINE CONTROL STATUS
1445  1  BIT 7 = TIME OUT
1446  1  BIT 6 = TRANS SHIFT REGISTER EMPTY
1447  1  BIT 5 = TRANS HOLDING REGISTER EMPTY
1448  1  BIT 4 = BREAK DETECT
1449  1  BIT 3 = FRAMING ERROR
1450  1  BIT 2 = PARITY ERROR
1451  1  BIT 1 = OVERRUN ERROR
1452  1  BIT 0 = DATA READY
1453  1  AL CONTAINS THE MODERM STATUS
1454  1  BIT 7 = RECEIVED LINE SIGNAL DETECT
1455  1  BIT 6 = RING INDICATOR
1456  1  BIT 5 = DATA SET READY
1457  1  BIT 4 = CLEAR TO SEND
1458  1  BIT 3 = DELTA RECEIVED LINE SIGNAL DETECT
1459  1  BIT 2 = TRAILING EDGE RING DETECTOR
1460  1  BIT 1 = DELTA DATA SET READY
1461  1  BIT 0 = DELTA CLEAR TO SEND
1462  1  (DX) = PARAMETER INDICATING WHICH RS232 CARD IS ALLOWED)
1463  1  DATA AREA RS232_BASE CONTAINS THE BASE ADDRESS OF THE 8550 ON THE CARD
1464  1  LOCATION 400H CONTAINS UP TO 4 RS232 ADDRESSES POSSIBLE
1465  1  ON EXIT, AX MODIFIED ACCORDING TO FAMHS OF CALL
1466  1  ALL OTHERS UNCHANGED
1467  1  -------------------------------------
1468  1  RS232_IO PORC  PROC  FAR
1469  1  ------ VECTOR TO APPROPRIATE ROUTINE
1470  1  -------------------------------------
1471  1  ASSUME CS:CODE, DS:DATA
1472  E729  1704  E729  1704  DW  1047  ; 110 BAUD ; TABLE OF INIT VALUE
1473  E729  0003  E729  0003  DW  768  ; 150
1474  E729  0001  E729  0001  DW  304  ; 300
1475  E729  C000  E729  C000  DW  192  ; 600
1476  E729  6000  E729  6000  DW  96  ; 1200
1477  E729  3000  E729  3000  DW  48  ; 2400
1478  E729  1000  E729  1000  DW  24  ; 4800
1479  E729  0000  E729  0000  DW  12  ; 9600
1480  E739  81  E739  81  RS232.IO  PROC  FAR
1481  1482  ---- VECTOR TO APPROPRIATE ROUTINE
1483  1484  -------------------------------
1485  E799  FB  E799  FB  STI  ; INTERRUPTS BACK ON
1486  E79A  1E  E79A  1E  PUSH DS  ; SAVE SEGMENT
1487  E73B  52  E73B  52  PUSH DX
1488  E73C  56  E73C  56  PUSH SI
1489  E73D  57  E73D  57  PUSH DI
1490  E73E  51  E73E  51  PUSH CX
1491  E73F  00F8  E73F  00F8  MOV SI,DX  ; RS232 VALUE TO SI
1492  E761  D16E  E761  D16E  SHL SI,1  ; WORD OFFSET
1493  E763  B4A000  E763  B4A000  MOV DX,DATA
1494  E766  6EDA  E766  6EDA  MOV DS,DX  ; SET UP OUR SEGMENT
1495  E768  60040000  E768  60040000  MOV DX,RS232_BASE[SI]  ; GET BASE ADDRESS
1496  E74C  00D2  E74C  00D2  OR DX,DX  ; TEST FOR 0 BASE ADDRESS
1497  E74E  7416  E74E  7416  JZ A3  ; RETURN
1498  E750  0A46  E750  0A46  OR AH,AL  ; TEST FOR (AH)=D
1499  E752  7410  E752  7410  JZ A4  ; COMPLIANT INIT
1500  E754  FECC  E754  FECC  DEC AH  ; TEST FOR (AH)=1
1501  E756  740E  E756  740E  JC A5  ; SEND AL
1502  E758  FECC  E758  FECC  DEC AH  ; TEST FOR (AH)=2
1503  E75A  7503  E75A  7503  JNZ A2  ; RECEIVE INTO AL
1504  E75C  E09900  E75C  E09900  JMP A12  ; RE...
E75F   1505  A2:  DEC AH  ; TEST FOR (AH)=3
E75F  FECC  1506  JNZ A3  ; COMMUNICATION STATUS
E761  7503  1507  JMP A18  ; RETURN FROM RS232
E763  E9B900  1508  ; LINE SOURCE
E766  1509  A3:  POP CX
E766  1510  POP DI
E768  1511  POP DS
E769  5A  1512  POP DX
E76A  17  1513  POP DS
E76B  CF  1514  IRET  ; RETURN TO CALLER, NO ACTION
E76C  1515  A4:  MOV AH,AL  ; SAVE INIT PARMS IN AH
E76E  83C203  1516  ADD DX,3  ; POINT TO 8250 CONTROL REGISTER
E771  B080  1517  MOV DX,AL  ; SET DLAB=1
E774  1518  A5:  MOV DL,AH  ; GET PARMS TO DL
E776  1519  8404  MOV AL,DL  ; GET BAUD RATE TERM TO LOW BITS
E778  00C2  1520  ROL DL  ; *2 FOR WORD TABLE ACCESS
E77A  81E20EOO  1521  MOV OX,00  ; ISOLATE THEM
E780  08940000  1522  MOV OX.RS232_BASE([DI]+1)  ; POINT TO HIGH ORDER OF DIVISOR
E782  42  1523  INC DX
E786  83C9  1524  XOR CX,CX  ; INITIALIZE TIME OUT COUNT
E787  83C202  1525  ADD DX,2  ; MODEM STATUS REGISTER
E788  58  1526  POP AX
E78A  1527  MOV AL,AH  ; GET PARGS BACK
E78B  51F  1528  AH  ; STRIP OFF THE BAUD BITS
E78C  EE  1529  OUT DX,AL  ; LINE CONTROL TO 8 BITS
E78E  E3EA02  1530  ADD DX,DX  ; PUT INTO INDEX REGISTER
E790  80CC50  1531  OR AH,80H  ; INTERRUPT ENABLES ALL OFF
E792  EB79  1532  JMP SHORT A18  ; COMM_STATUS
E794  42  1533  INC DX
E796  1534  8405  MOV AL,DL  ; GET LOW ORDER OF DIVISOR
E798  1535  00C2  ROL DL  ; *2 FOR WORD TABLE ACCESS
E79A  00C2  1536  ROL DL  ; *2 FOR WORD TABLE ACCESS
E79C  83C9  1537  XOR CX,CX  ; INITIALIZE TIME OUT COUNT
E79D  83C202  1538  ADD DX,2  ; MODEM STATUS REGISTER
E79E  58  1539  POP AX
E7A0  1540  MOV AL,0H  ; MODEM STATUS
E7A2  50  1541  PUSH AX  ; SAVE CHAR TO SEND
E7A4  83C0H  1542  MOV AL,3  ; DTR AND RTS
E7A6  EE  1543  OUT DX,AL  ; DATA TERMINAL READY, REQUEST TO SEND
E7A8  33C9  1544  XOR CX,CX  ; INITIALIZE TIME OUT COUNT
E7AA  83C202  1545  ADD DX,2  ; MODEM STATUS REGISTER
E7AC  58  1546  POP AX
E7AEC  1547  SUB CX,CX  ; INITIALIZE WAIT COUNT
E7B0  E8A4H  1548  OR AH,80H  ; INDICATE TIME OUT
E7B2  EB79  1549  JMP A3  ; RETURN
E7B4  1550  A7:  SUB CX,CX  ; TEST_CLEAR_TO_SEND
E7B6  28C9  1551  MOV AH,DX  ; LINE STATUS REGISTER
E7B8  42  1552  INC DX
E7BA  1553  EC  1554  TEST AH,AL  ; DATA TERMINAL READY, REQUEST TO SEND
E7BC  7506  1555  JNZ A7  ; TEST_CLEAR_TO_SEND
E7BD  E2F9  1556  LOOP A6  ; WAIT_DATA_SET_READY
E7BE  5A  1557  POP AX
E7BF  6CC50  1558  OR AH,80H  ; INDICATE TIME OUT
E7C1  EE79  1559  JMP A3  ; RETURN
E7C3  28C9  1560  SUB CX,CX  ; TEST_CLEAR_TO_SEND
E7C5  42  1561  INC DX
E7C7  EC  1562  TEST AH,AL  ; DATA TERMINAL READY, REQUEST TO SEND
E7C9  7506  1563  JNZ A7  ; TEST_CLEAR_TO_SEND
E7CB  E2F9  1564  LOOP A6  ; WAIT_DATA_SET_READY
E7CD  5A  1565  POP AX
E7CF  E0B6  1566  OR AH,80H  ; INDICATE TIME OUT
E7D1  1567  JMP A3  ; RETURN
E7D3  42  1568  INC DX
E7D5  EC  1569  TEST AH,AL  ; DATA TERMINAL READY, REQUEST TO SEND
E7D7  7506  1570  JNZ A7  ; TEST_CLEAR_TO_SEND
E7D9  E2F9  1571  LOOP A6  ; WAIT_DATA_SET_READY
E7DB  5A  1572  POP AX
E7DF  E0B6  1573  OR AH,80H  ; INDICATE TIME OUT
E7D1  1574  JMP A3  ; RETURN
E7D3  42  1575  INC DX
E7D5  EC  1576  TEST AH,AL  ; DATA TERMINAL READY, REQUEST TO SEND
E7D7  7506  1577  JNZ A7  ; TEST_CLEAR_TO_SEND
E7D9  E2F9  1578  LOOP A6  ; WAIT_DATA_SET_READY
E7DB  5A  1579  POP AX
E7DF  E0B6  1580  OR AH,80H  ; INDICATE TIME OUT
E7D1  1581  JMP A3  ; RETURN
LOC 0BJ  
LINE  
SOURCE

E701 EC 1582   IN AL,DX ; GET STATUS
E702 A360 1583   TEST AL,20H ; IS TRANSMITTER READY
E704 7508 1584   JNC A11 ; OUT_CHAR
E706 E2F9 1585   LOOP A10 ; GO BACK FOR MORE, AND TEST FOR TIME OUT
E708 58 1586   POP AX ; RECOVER ORIGINAL INPUT
E709 80C0 1587   OR AH,80H ; SET THE TIME OUT BIT
E70C E800 1588   JMP A3 ; RETURN
E70E 1589   A11:  
E70E 85E05 1590   SUB DX,S ; DATA PORT
E70F 59 1591   POP CX ; RECOVER IN CX TEMORARILY
E712 E0C1 1592   MOV AL,CL ; GET OUT CHAR TO AL FOR OUT, STATUS IN AH
E716 EE 1593   OUT DX,AL ; OUTPUT CHARACTER
E718 EY7EFF 1594   JMP A3 ; RETURN
E71F 1595
E720 ---- RECEIVE CHARACTER FROM COMM LINE
E724 1596
E725 E40571007F R 1599   AND BIOS_BREAK.07FH ; TURN OFF BREAK BIT IN BYTE
E727 E3C204 1600   ADD DX,4 ; MODEM CONTROL REGISTER
E72B E001 1601   MOV AL,1 ; DATA TERMINAL READY
E72F EE 1602   OUT DX,AL
E733 E5C302 1603   ADD DX,2 ; MODEM STATUS REGISTER
E737 E0C9 1604   SUB CX,CX ; ESTABLISH TIME OUT COUNT
E73F E7 1605   A13: ; WAIT_JSR
E73F E70C 1606   ADD DX,AL
E741 E920 1607   TEST AL,20H ; DATA SET READY
E745 7507 1608   JNC A15 ; IS IT READY YET
E747 E2F9 1609   LOOP A13 ; WAIT UNTIL IT IS
E74F E7 1610   A14: ; TIMEOUT_ERR
E757 B400 1611   MOV AH,80H ; SET TIME OUT ERROR
E75F E920 1612   JMP A3 ; RETURN WITH ERROR
E769 E04 1613   A15: ; WAIT_DSP_END
E76C E04 1614   DEC DX ; LINE STATUS REGISTER
E76F EE 1615   MOV AL,1 ; WAIT_RECV
E76F E800 1616   IN AL,DX ; GET STATUS
E774 A021 1617   TEST AL,1 ; RECEIVE BUFFER FULL
E778 750F 1618   JNC A17 ; SET CHAR
E77D E0A 1619   TEST BIOS_BREAK.60H ; TEST FOR BREAK KEY
E780 7F4 1620   JZ A16 ; LOOP IF NOT
E783 E1EC 1621   JMP A14 ; SET TIME OUT ERROR
E786 E805 1622   IN AL,DX
E78B A001 1623   TEST AL,1
E78F 750F 1624   JNC A17 ; GET_CHAR
E793 E41E 1625   AND AL,00011110B ; TEST FOR ERROR CONDITIONS ON RECIV CHAR
E798 E6E0 1626   MOV AH,AL ; SAVE THIS PART OF STATUS FOR LATER OPERATION
E79D E810 1627   IN AL,DX ; GET CHARACTER FROM LINE
E7A2 E947FF 1628   JMP A3 ; RETURN
E7A8 1629
E7A9 ---- COMM PORT STATUS ROUTINE
E7AC 1630
E7B1 E81F 1631
E7B6 E80400000 R 1632   MOV DX,RS232_BASE[S1] ; DATA PORT
E7B9 E82C 1633   ADD DX,5 ; CONTROL PORT
E7BE EC 1634   IN AL,DX ; GET LINE CONTROL STATUS
E7C2 EAE0 1635   MOV AH,AL ; PUT IN AH FOR RETURN
E7C8 42 1636   INC DX ; POINT TO MODEM STATUS REGISTER
E7CD EA 1637   IN AL,DX ; GET MODEM CONTROL STATUS
E7D2 E935FF 1638   JMP A3 ; RETURN
E7D9 RS232_ID 1639
E7DA 1640
E7DE 1641 ; KEYBOARD I/O
E7E2 1642 ; THESE ROUTINES PROVIDE KEYBOARD SUPPORT
E7E8 1643 ; INPUT
E7F4 1644 ; (AX)<>O READ THE NEXT ASCII CHARACTER STRUCK FROM THE KEYBOARD
E7F8 1645 ; RETURN THE RESULT IN (AL), SCAN CODE IN (AX)
E7FC 1646 ; (AX)=1 SET THE Z FLAG TO INDICATE IF AN ASCII CHARACTER IS AVAILABLE
E800 1647 ; TO BE READ.
E804 1648 ; (ZF)=1 -- NO CODE AVAILABLE
E808 1649 ; (ZF)=0 -- CODE IS AVAILABLE
E80C 1650 ; IF ZF = 0, THE NEXT CHARACTER IN THE BUFFER TO BE READ IS
E810 1651 ; IN AX, AND THE ENTRY REMAINS IN THE BUFFER
E814 1652 ; (AH)=3 RETURN THE CURRENT SHIFT STATUS IN AH REGISTER
E818 1653 ; THE BIT SETTINGS FOR THIS CODE ARE INDICATED IN THE
E81C 1654 ; THE EQUATES FOR KB_FLAG
E820 1655 ; OUTPUT
E824 1656 ; AS NOTED ABOVE, ONLY AX AND FLAGS CHANGED
E828 1657 ; ALL REGISTERS RETAINED
E82D 1658
A-23
LaC OBJ
LINE SOURCE
1659 ASUME CS:CODE,DS:DATA
E81E KEYBOARD_IO PROC FAR
E820 FB 1660 STI ; INTERRUPTS BACK ON
E82F 1E 1661 PUSH DS ; SAVE CURRENT DS
E835 53 1662 PUSH BX ; SAVE BX TEMPORARILY
E831 B04000 1663 MOV BX,DATA 1
E836 0E8B 1664 MOV DS,BX ; ESTABLISH POINTER TO DATA REGION
E83A 8A45 1665 OR AH,AL ; AH=0
E83F 7408 1666 JZ K1 ; ASCII_READ
E83A FECC 1667 DEC AH ; AH=1
E83C 7420 1668 JZ K2 ; ASCII_STATUS
E83E FECC 1669 DEC AH ; AH=2
E840 7420 1670 JZ K3 ; SHIFT_STATUS
E844 50 1671 POP BX ; RECOVER REGISTER
E844 IF 1672 POP DS ; RECOVER SEGMENT
E844 CF 1673 IRET ; INVALID COMMAND
1676 1------ READ THE KEY TO FIGURE OUT WHAT TO DO
E845 1677 K1: ; ASCII READ
E845 FB 1678 STI ; INTERRUPTS BACK ON DURING LOOP
E846 90 1679 POP ; ALLOW AN INTERRUPT TO OCCUR
E847 FI 1680 CLI ; INTERRUPTS BACK OFF
E848 0811A00 1681 MOV BX,BUFFER_HEAD 1 GET POINTER TO HEAD OF BUFFER
E848 3B1E00 1682 CMP BX,BUFFER_TAIL ; TEST END OF BUFFER
E850 74F3 1683 JZ K1 ; LOOP UNTIL SOMETHING IN BUFFER
E852 0807 1684 MOV AX,[BX] ; GET SCAN CODE AND ASCII CODE
E855 861E00 1685 CALL K4 ; MOVE POINT TO NEXT POSITION
E857 9811A00 1686 MOV BUFFER_HEAD,BX ; STORE VALUE IN VARIABLE
E858 5B 1687 POP BX ; RECOVER REGISTER
E858 IF 1688 POP DS ; RECOVER SEGMENT
E858 CF 1689 IRET ; RETURN TO CALLER
1692 1------ ASCII STATUS
E85E 1693 K2: ; INTERRUPTS OFF
E85F FA 1694 CLI ; INTERRUPTS BACK OFF
E860 0811A00 1695 MOV BX,BUFFER_HEAD 1 GET HEAD POINTER
E863 3B1E00 1696 CMP BX,BUFFER_TAIL ; IF EQUAL IZ=1) THEN NOTHING THERE
E867 0807 1697 MOV AX,[BX] ;
E869 FB 1698 STI ; INTERRUPTS BACK ON
E86A 5B 1699 POP BX ; RECOVER REGISTER
E86B IF 169A POP DS ; RECOVER SEGMENT
E86C CA0000 169B MOV AX,SUFFER_HEAD ; GET HEAD POINTER
E86C 7503 169C JNE .s ; NO. CONTINUE
E86E 8807 169D MOV ax,OFFSET KB_BUFFER_END ; AT END OF BUFFER?
167E 8778 169E 81FB3E00 1717 CHP BX,OFFSET KB_BUFFER_END ; AT END OF BUFFER?
E87C 7503 1718 JNE KS ; NO, CONTINUE
E87D 881E00 1719 MOV BX,OFFSET KB_BUFFER ; YES, RESET TO BUFFER BEGINNING
E881 1720 KS: ;
E881 C3 1721 HLT ;
E881 IF 1722 IRET ; RETURN TO CALLER
1724 1------ SHIFT_STATUS
E86F 1725 K3: ;
E86F A01700 1726 MOV AL,KB_FLAG ; GET THE SHIFT STATUS FLGS
E870 5B 1727 POP BX ; RECOVER REGISTER
E875 1F 1728 POP DS ; RECOVER SEGMENT
E874 CF 1729 IRET ; RETURN TO CALLER
172E 1------ INCREMENT A BUFFER POINTER
E875 1730 K4 ;
E875 03C302 1731 POP NEAM ; MOVE TO NEXT WORD IN LIST
E878 81FB3E00 1732 CHP BX,OFFSET KB_BUFFER_END ; AT END OF BUFFER?
E87C 7503 1733 JNE KS ; NO, CONTINUE
E87D 881E00 1734 MOV BX,OFFSET KB_BUFFER ; YES, RESET TO BUFFER BEGINNING
E881 1735 KS: ;
E881 C3 1736 HLT ;
E881 IF 1737 IRET ; RETURN TO CALLER
173D 1------ TABLE OF SHIFT KEYS AND MASK VALUES
E882 173E K6 ;
E882 5E 173F DB INS_KEY ; INSERT KEY
E883 544546381D 1740 DB CAPS_KEY,JAM_KEY,SCROLL_KEY,ALT_KEY,CTRL_KEY
E884 2A36 1741 DB LEFT_KEY,RIGHT_KEY
E886 0038 1742 DB L6,EUQ,~K6
E888 1743 DB ;
E88A 80 1744 DB INS_SHIFT ; INSERT MODE SHIFT
1751 1------ SHIFT_MASK TABLE
1753 E886 K7 ;
E88A 1754 DB INS_KEY ; INSERT KEY
175C 1------ T ABLE OF SHIFT KEYS AND MASK VALUES
A-24
<table>
<thead>
<tr>
<th>LOC OBJ</th>
<th>LINE</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>E890</td>
<td>0201</td>
<td>DB</td>
</tr>
<tr>
<td>E898</td>
<td>0100004</td>
<td>CAPS_SHIFT, NUM_SHIFT, SCROLL_SHIFT, ALT_SHIFT, CTRL_SHIFT</td>
</tr>
<tr>
<td>E899</td>
<td>0201</td>
<td>DB</td>
</tr>
<tr>
<td>E896</td>
<td>0100004</td>
<td>LEFT_SHIFT, RIGHT_SHIFT</td>
</tr>
<tr>
<td>E897</td>
<td>0100004</td>
<td>I------- SCAN CODE TABLES</td>
</tr>
<tr>
<td>E898</td>
<td>0100004</td>
<td>KF</td>
</tr>
<tr>
<td>E899</td>
<td>0100004</td>
<td>DB</td>
</tr>
<tr>
<td>E89A</td>
<td>01191509009</td>
<td>27, -1, 1, 0, -1, -1, 30, -1</td>
</tr>
<tr>
<td>E89B</td>
<td>01191509009</td>
<td>K0</td>
</tr>
<tr>
<td>E89C</td>
<td>01191509009</td>
<td>DB</td>
</tr>
<tr>
<td>E89D</td>
<td>01191509009</td>
<td>-1, -1, -1, 31, -1, 127, -1, 17</td>
</tr>
<tr>
<td>E89E</td>
<td>01191509009</td>
<td>23, 5, 10, 20, 25, 21, 9, 5</td>
</tr>
<tr>
<td>E89F</td>
<td>01191509009</td>
<td>K1</td>
</tr>
<tr>
<td>E8A0</td>
<td>01191509009</td>
<td>DB</td>
</tr>
<tr>
<td>E8A1</td>
<td>01191509009</td>
<td>16, 27, 29, 10, -1, 119</td>
</tr>
<tr>
<td>E8A2</td>
<td>01191509009</td>
<td>4, 6, 7, 8, 10, 11, 12, -1, -1</td>
</tr>
<tr>
<td>E8A3</td>
<td>01191509009</td>
<td>K2</td>
</tr>
<tr>
<td>E8A4</td>
<td>01191509009</td>
<td>DB</td>
</tr>
<tr>
<td>E8A5</td>
<td>01191509009</td>
<td>-1, -1, 28, 26, 24, 23, 22</td>
</tr>
<tr>
<td>E8A6</td>
<td>01191509009</td>
<td>K3</td>
</tr>
<tr>
<td>E8A7</td>
<td>01191509009</td>
<td>DB</td>
</tr>
<tr>
<td>E8A8</td>
<td>01191509009</td>
<td>14, 13, -1, -1, -1, 1, -1</td>
</tr>
<tr>
<td>E8A9</td>
<td>01191509009</td>
<td>K4</td>
</tr>
<tr>
<td>E8AA</td>
<td>01191509009</td>
<td>DB</td>
</tr>
<tr>
<td>E8AB</td>
<td>01191509009</td>
<td>'1', -1</td>
</tr>
<tr>
<td>E8AC</td>
<td>01191509009</td>
<td>K5</td>
</tr>
<tr>
<td>E8AD</td>
<td>01191509009</td>
<td>DB</td>
</tr>
<tr>
<td>E8AE</td>
<td>01191509009</td>
<td>94, 95, 96, 97, 98, 99, 100, 101</td>
</tr>
<tr>
<td>E8AF</td>
<td>01191509009</td>
<td>K6</td>
</tr>
<tr>
<td>E8B0</td>
<td>01191509009</td>
<td>DB</td>
</tr>
<tr>
<td>E8B1</td>
<td>01191509009</td>
<td>102, 103, -1, -1, 119, -1, 132, -1</td>
</tr>
<tr>
<td>E8B2</td>
<td>01191509009</td>
<td>K7</td>
</tr>
<tr>
<td>E8B3</td>
<td>01191509009</td>
<td>DB</td>
</tr>
<tr>
<td>E8B4</td>
<td>01191509009</td>
<td>-1</td>
</tr>
<tr>
<td>E8B5</td>
<td>01191509009</td>
<td>K8</td>
</tr>
<tr>
<td>E8B6</td>
<td>01191509009</td>
<td>DB</td>
</tr>
<tr>
<td>E8B7</td>
<td>01191509009</td>
<td>010H, '123', '456', '789', '0'</td>
</tr>
<tr>
<td>E8B8</td>
<td>01191509009</td>
<td>K9</td>
</tr>
<tr>
<td>E8B9</td>
<td>01191509009</td>
<td>DB</td>
</tr>
<tr>
<td>E8BA</td>
<td>01191509009</td>
<td>'qwertuyopil', '00H', -1, 'asdfgjk', '02H</td>
</tr>
<tr>
<td>E8BB</td>
<td>01191509009</td>
<td>K10</td>
</tr>
<tr>
<td>E8BC</td>
<td>01191509009</td>
<td>DB</td>
</tr>
<tr>
<td>E8BD</td>
<td>01191509009</td>
<td>60H, -1, 5CH, 'x', 'v', 'b', 'm', '.', '!', ',', '1', '2', '3', '4', '5', '6', '7', '8', '9', '0', ..., '!', ' ', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!'</td>
</tr>
</tbody>
</table>
E994 8E08 1798 MOV DS,AX ; SET UP ADDRESSING
E996 E460 1799 IN AL,KB_DATA ; READ IN THE CHARACTER
E998 50 1800 PUSH AX ; SAVE IT
E999 E461 1801 IN AL,KB_CTL ; GET THE CONTROL PORT
E99B 6A00 1802 MOV AX,AL ; SAVE VALUE
E99D 0900 1803 OR AL,0FH ; RESET BIT FOR KEYBOARD
E99F E661 1804 OUT KB_CTL,AL
E9A1 8E00 1805 XCHG AH,AL ; GET BACK ORIGINAL CONTROL
E9A3 E661 1806 OUT KB_CTL,AL ; KB HAS BEEN RESET
E9A5 58 1807 POP AX ; RECOVER SCAN CODE
E9A6 8A00 1808 HOV AH,AL ; SAVE SCAN CODE IN AX ALSO

1809 1------ TEST FOR OVERRUN SCAN CODE FROM KEYBOARD
E9A8 3E00 1810 CMP AL,OFFH ; IS THIS AN OVERRUN CHAR
E9B0 7503 1811 JNZ K16 ; NO, TEST FOR SHIFT KEY
E9B2 E97502 1812 JZ K17 ; JUMP IF MATCH FOUND

1813 1------ SHIFT KEY FOUND
E9B4 810F83E0 R 1814 SUB DI,OFFSET K6 ; ADJUST PTR TO SCAN CODE MATCH
E9B6 260A4A00 R 1815 MOV AH,CS:K7 [DI] ; SCAN CODE MATCH
E9B8 754D 1816 JAE K18 IF SCROLL_SHIFT OR ABOVE, TOGGLE KEY
E9BA 7400 1817 JZ K20 ; IF NO MATCH, THEN SHIFT NOT FOUND

1818 1------ PLAIN SHIFT KEY, SET SHIFT ON
E9BD E85800 1819 TEST AL,6OH ; TEST FOR BREAK KEY
E9BF 7403 1820 JNE K19 JUMP IF NOT ALTERNATE SHIFT
E9C0 F606170020 1821 TEST KBJLAG, NUM_STATE ; CHECK FOR BASE STATE
E9C5 7500 1822 JZ K25 ; JUMP IF NUM LOCK IS ON
E9C7 750003 1823 JZ K20 ; JUMP FOR NUMERIC, NOT INSERT

1824 1------ Toggled SHIFT KEY, TEST FOR 1ST MAKE OR NOT
E9CB 84261700 1825 TEST KB_FLAG,AH ; IS KEY ALREADY DEPRESSED
E9CD 7541 1826 JNE K26 JUMP IF NOT ALTERNATE SHIFT
E9CF 7404 1827 JZ K22 ; JUMP IF BASE STATE

1828 1------ NUMERIC ZERO, NOT INSERT KEY
E9D1 E85800 1829 TEST KB_FLAG,AL ; CHECK KB_FLAG, AL
E9D3 7403 1830 JNE K25 ; JUMP IF KB_FLAG, AL
E9D5 84261800 1831 TEST KB_FLAG, 1 ; IS KEY ALREADY DEPRESSED
E9D7 7541 1832 JNE K22 ; JUMP IF KEY ALREADY DEPRESSED

1833 1------ SHAFT KEY TOGGLE KEY HIT, PROCESS IT
E9DB E85800 1834 TEST KB_FLAG,AL ; CHECK KB_FLAG, AL
E9DF 7404 1835 JNZ K25 ; JUMP IF KB_FLAG, AL
E9E1 F606170004 R 1836 TEST KB_FLAG, CTL_SHIFT ; CHECK CTL SHIFT STATE
E9E3 755C 1837 JNZ K25 ; JUMP IF CTL STATE
E9E5 58 1838 POP AX ; PUT OUT AN ASCII ZERO
E9E7 E85800 1839 TEST KB_FLAG, CTL_SHIFT ; CHECK KB_FLAG, CTL_SHIFT
E9F1 7500 1840 JZ K25 ; JUMP IF KB_FLAG, CTL_SHIFT
E9F3 750003 1841 JZ K22 ; JUMP IF KB_FLAG, 1
E9F5 E85800 1842 TEST KB_FLAG, CTL_SHIFT ; CHECK KB_FLAG, CTL_SHIFT
E9F7 7404 1843 JNZ K25 ; JUMP IF KB_FLAG, CTL_SHIFT

1844 1------ CHECK FOR INSERT KEY
E9FC 750003 1845 JZ K25 ; JUMP IF KB_FLAG, INSERT
E9FF 750003 1846 JZ K22 ; JUMP IF KB_FLAG, INSERT

1847 1------ CHECK FOR ALT_SHIFT
E9FE E85800 1848 TEST KB_FLAG, ALT_SHIFT ; CHECK FOR ALT_SHIFT
E9F0 755C 1849 JNZ K25 ; JUMP IF ALT_SHIFT
E9F2 58 1850 POP AX ; PUT OUT AN ASCII ZERO
E9F4 E85800 1851 TEST KB_FLAG, ALT_SHIFT ; CHECK FOR ALT_SHIFT
E9F6 7500 1852 JZ K25 ; JUMP IF KB_FLAG, ALT_SHIFT
E9F8 750003 1853 JZ K22 ; JUMP IF KB_FLAG, ALT_SHIFT

1854 1------ CHECK FOR NUM_LOCK
E9FD E85800 1855 TEST KB_FLAG, NUM_LOCK ; CHECK FOR NUM_LOCK
E9F9 7500 1856 JZ K25 ; JUMP IF NUM_LOCK IS ON
E9FB 750003 1857 JZ K22 ; JUMP IF NUM_LOCK IS ON

1858 1------ CHECK FOR NUMERIC
E9F9 8500 1859 MOV AX, 5230H ; PUT OUT AN ASCII ZERO
EA01 E90001 1860 JMP K57 ; BUFFER_FILL
EA03 E85800 1861 TEST KB_FLAG, ALT_SHIFT ; CHECK FOR ALT_SHIFT
EA05 7404 1862 JNZ K22 ; JUMP FOR NUMERIC, NOT INSERT

1863 1------ CHECK FOR 1ST MAKE OF INSERT KEY
EA07 E85800 1864 TEST KB_FLAG, INSERT ; CHECK FOR 1ST MAKE OF INSERT KEY
EA09 7404 1865 JNZ K22 ; JUMP FOR 1ST MAKE OF INSERT KEY

1866 1------ CHECK FOR 1ST MAKE OF KB_FLAG, 1
EA0B 84261800 R 1867 TEST KB_FLAG, 1 ; IS KEY ALREADY DEPRESSED
EA0D 750003 1868 JNZ K25 ; JUMP IF KEY ALREADY DEPRESSED
EA11 02001000 R 1870 OR KB_FLAG, 1 ; INDICATE THAT THE KEY IS DEPRESSED
EA13 02001700 R 1871 XOR KB_FLAG, 1 ; TOGGLE THE SHIFT STATE
EA15 8500 1872 MOV AX, 5230H ; PUT OUT AN ASCII ZERO
EA17 754C 1873 JMP KB_FLAG, 1 ; TEST FOR 1ST MAKE OF INSERT KEY
EA19 7541 1874 JNE K26 ; JUMP IF NOT INSERT KEY

A-26
MOV AX, INH_KEY+256  ; SET SCAN CODE INTO AH, O INTO AL
JMP K57             ; PUT INTO OUTPUT BUFFER

--- BREAK SHIFT FOUND

BREAK-SHIFT-FOUND

IS THIS A TOGGLE KEY
YES, HANDLE BREAK TOGGLE
INVERT MASK
TURN OFF SHIFT BIT
IS THIS ALTERNATE SHIFT RELEASE
INTERRUPT_RETURN

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

MDV AL, ALT_INPUT
MDV AH, 0
ZERO OUT THE FIELD
HAS THE INPUT-0
IT WASN'T, SO PUT IN BUFFER

--- TEST FOR HOLD STATE

NO-SHIFT-FOUND
TEST FOR BREAK KEY
NOTHING FOR BREAK CHARMS FROM HERE ON
ARE WE IN HOLD STATE
BRANCH AROUND TEST IF NOT
CAN'T END HOLD ON NUM_LOCK
TURN OFF THE HOLD STATE BIT

--- TEST FOR HOLD STATE

CONTROL-ALT-DEL HAS BEEN FOUND.
DO I/O CLEANUP
SET FLAG FOR RESET FUNCTION
JUMP TO POWER ON DIAGNOSTICS

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

CONTROL-ALT-DEL

NOT-HOLD-STATE

ARE WE IN ALTERNATE SHIFT
JUMP IF ALTERNATE SHIFT
JUMP IF NOT ALTERNATE

--- TEST FOR RESET KEY SEQUENCE (CTL ALT DEL)

TEST-KB_FLAG, ALT_SHIFT
ARE WE IN CONTROL SHIFT ALSO
SHIFT STATE IS THERE, TEST KEY
NO_RESET

--- CTL-ALT-DEL HAS BEEN FOUND, DO I/O CLEANUP

MOV RESET_FLAG, 1234H
JMP RESET

--- ALT-INPUT-TABLE

LABEL BYTE
82, 79, 80, 81, 75, 76, 77

A-27
LOC OBJ

EA91 474B49 1951
1952
DB  71,72,73  ; 10 NUMBERS ON KEYPAD

EA94 111111134151617 1953
DB  16,17,18,19,20,21,22,23  ; A-Z TYPEWRITER CHARS

EA9C 1891EFPD2112223 1954
DB  24,25,30,31,32,33,34,35

EAAD 242562CD22F30 1955
DB  36,37,38,44,45,46,47,48

EAAC 3132 1956
DB  49,50

1957

EA9E 1958

EA9E 3C39 1959
CHP AL,57  ; TEST FOR SPACE KEY

EA90 7905 1960
JNE K32  ; NOT THERE

EA92 8020 1961
MOV AL,' '  ; SET SPACE CHAR

EAD4 E929501 1962
JMP K57  ; BUFFER_FILL

1963

EAD6 1964

EAD7 1966
LOOK FOR KEY PAD ENTRY

EAD8 1967

EAD9 1968
K32:  ; ALT-KEY-PAD

EA9A 1969
MOV DI,OFFSET K30  ; ALT-INPUT-TABLE

EA9B 590A00 1970
MOV CX,10  ; LOOK FOR ENTRY USING KEYPAD

EA9D F2 1971
REPNE SCASB  ; LOOK FOR MATCH

EA9D 7512 1972
JNE K33  ; NO_ALTERNATE

EA9E 187F8EA 1973
SUB DI,OFFSET K30+1  ; DI NOW HAS ENTRY VALUE

EAC5 401900 1974
MOV AL,ALT_INPUT  ; GET THE CURRENT BYTE

EACB 840A 1975
MOV AH,10  ; MULTIPLY BY 10

EAAD F646 1976
MUL AH

EACC 03C7 1977
ADD AX,DI  ; ADD IN THE LATEST ENTRY

EACE AD21900 1978
MOV ALT_INPUT,AL  ; STORE IT AWAY

EADF 1979
JMP K26  ; THROW AWAY THAT KEYSROKE

1980

EAD3 1981

EADD 5C66190000 1982
MOV AL,OFFSET_AL  ; ZERO ANY PREVIOUS ENTRY INTO INPUT

EAD8 B91400 1983
MOV CX,16  ; DI.IS ALREADY POINTING

EAD9 F2 1984
REPNE SCASB  ; LOOK FOR MATCH IN ALPHABET

EADC AE 1985

EADD 7545 1986
JNE K34  ; NOT FOUND, FUNCTION KEY OR OTHER

EADF 0000 1987
MOV AL,0  ; ASCII CODE OF ZERO

EAC1 E9F600 1988
JMP K57  ; PUT IT IN THE BUFFER

1990

EAA4 1991

EAA4 3C02 1992
CHP AL,2  ; ALT-TOP-ROW

EAA6 720C 1993
JB K35  ; KEY WITH '1' ON IT

EAA8 3C02 1994
JMP AL,14  ; NOT ONE OF INTERESTING KEYS

EAAE 7308 1995
JAE K35  ; ALT-FUNCTION

EACC 80C476 1996
ADD AH,86  ; CONVERT PSUEDO SCAN CODE TO RANGE

EAEF 0600 1997
MOV AL,0  ; INDICATE AS SUCH

EAF1 E9E800 1998
JMP K57  ; BUFFER_FILL

1999

EAA4 2000

EAA4 3C30 2001
CHP AL,59  ; TEST FOR IN TABLE

EAAE 730B 2002
JAE K37  ; ALT-CONTINUE

EAF8 8933F 2003
JMP K26  ; CLOSE-RETURN

EAFB E963F 2004
JMP AL,71  ; IN KEYPAD REGION

EAFD 73F9 2005
JAE K36  ; IF SO, IGNORE

EAEF B3AE9 2006
MOV BX,OFFSET K13  ; ALT SHIFT PSUEDO SCAN TABLE

EB02 E92801 2007
JMP K63  ; TRANSLATE THAT

2012

EB05 2013

EB05 F66170004 2014
TEST KB_FLAG.CTL_SHIFT  ; ARE WE IN CONTROL SHIFT

EB0A 745B 2015
JZ K44  ; NOT-CTL-SHIFT

2016

EB0D 2017

EB0E 3C46 2018
CHP AL,SCROLL_KEY  ; TEST FOR BREAK

EB0E 7518 2019
JNE K39  ; NO-BREAK
<table>
<thead>
<tr>
<th>LOC OBJ</th>
<th>LINE</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>EB13 081E00</td>
<td>2026</td>
<td>MOV BX,OFFSET KB_BUFFER ; RESET BUFFER TO EMPTY</td>
</tr>
<tr>
<td>EB13 081E100</td>
<td>2027</td>
<td>MOV BUFFER_HEAD,BX</td>
</tr>
<tr>
<td>EB17 091E1C0</td>
<td>2028</td>
<td>MOV BUFFER_TAIL,BX</td>
</tr>
<tr>
<td>EB19 0A6710000</td>
<td>2029</td>
<td>MOV BIOS_BREAK,60H ; TURN ON BIOS_BREAK BIT</td>
</tr>
<tr>
<td>EB20 0C16</td>
<td>2030</td>
<td>INT 1BH ; BREAK INTERRUPT VECTOR</td>
</tr>
<tr>
<td>EB22 0D0000</td>
<td>2031</td>
<td>MOV AX, 0 ; PUT OUT DTBHY CHARACTER</td>
</tr>
<tr>
<td>EB25 E90400</td>
<td>2032</td>
<td>JMP K57 ; BUFFER_FILL</td>
</tr>
<tr>
<td>EB28 2033</td>
<td>2034</td>
<td>K39: ; NO-BREAK</td>
</tr>
<tr>
<td>EB28 2035</td>
<td>2036</td>
<td>CMP AX,46H ; LOOK FOR PAUSE KEY</td>
</tr>
<tr>
<td>EB29 7621</td>
<td>2037</td>
<td>JNE K41 ; NO-PAUSE</td>
</tr>
<tr>
<td>EB2C 080E10000</td>
<td>2038</td>
<td>OR KB_FLAG_I,HOLD_STATE ; TURN ON THE HOLD FLAG</td>
</tr>
<tr>
<td>EB51 0B20</td>
<td>2039</td>
<td>MOV AL,EOI ; END OF INTERRUPT TO CONTROL PORT</td>
</tr>
<tr>
<td>EB53 E620</td>
<td>2040</td>
<td>OUT 020H,AL ; ALLOW FURTHER KEYSTROKE INTS</td>
</tr>
<tr>
<td>EB61 2041</td>
<td>2042</td>
<td>; DURING PAUSE INTERVAL, TURN CRT BACK ON</td>
</tr>
<tr>
<td>EB65 083F490007</td>
<td>2043</td>
<td>CMP CRT_MODE,7 ; IS THIS BLACK AND WHITE CARD</td>
</tr>
<tr>
<td>EB6A 7907</td>
<td>2044</td>
<td>JZ K4A ; YES, NOTHING TO DO</td>
</tr>
<tr>
<td>EB6C 0A5D03</td>
<td>2045</td>
<td>MOV DX,6300H ; PORT FOR COLOR CARD</td>
</tr>
<tr>
<td>EB6F A06500</td>
<td>2046</td>
<td>MOV AL,CRT_MODE_SET ; GET THE VALUE OF THE CURRENT MODE</td>
</tr>
<tr>
<td>EB72 EE0709</td>
<td>2047</td>
<td>OUT DX,AL ; SET THE CRT MODE, SO THAT CRT IS ON</td>
</tr>
<tr>
<td>EB75 2048</td>
<td>2049</td>
<td>K40: ; PAUSE-LOOP</td>
</tr>
<tr>
<td>EB77 F60618000</td>
<td>2050</td>
<td>TEST KB_FLAG_I,HOLD_STATE</td>
</tr>
<tr>
<td>EB79 765F9</td>
<td>2051</td>
<td>JNE K40 ; LOOP UNTIL FLAG TURNED OFF</td>
</tr>
<tr>
<td>EB84 E916FF</td>
<td>2052</td>
<td>JMP K27 ; INTERRUPT_RETURN_NO_EOI</td>
</tr>
<tr>
<td>EB8A 2053</td>
<td>2054</td>
<td>K41: ; NO-PAUSE</td>
</tr>
<tr>
<td>EB8E 2055</td>
<td>2056</td>
<td>; TEST SPECIAL CASE KEY 55</td>
</tr>
<tr>
<td>EB90 3C37</td>
<td>2057</td>
<td>CMP AL,55</td>
</tr>
<tr>
<td>EB94 750F6</td>
<td>2058</td>
<td>JNE K42 ; NOT-KEY-55</td>
</tr>
<tr>
<td>EB98 600072</td>
<td>2059</td>
<td>MOV AX,114*256 ; START/STOP PRINTING SWITCH</td>
</tr>
<tr>
<td>EB9C E95000</td>
<td>2060</td>
<td>JMP K57 ; BUFFER_FILL</td>
</tr>
<tr>
<td>EB9E 2061</td>
<td>2062</td>
<td>; SET UP TO TRANSLATE CONTROL SHIFT</td>
</tr>
<tr>
<td>EB9F 2063</td>
<td>2064</td>
<td>K42: ; NOT-KEY-55</td>
</tr>
<tr>
<td>EB9F 2065</td>
<td>2066</td>
<td>CMP AX,OFFSET KB ; SET UP TO TRANSLATE CTL</td>
</tr>
<tr>
<td>EBAC 3C39</td>
<td>2067</td>
<td>CMP AL,59 ; IS IT IN TABLE</td>
</tr>
<tr>
<td>EBE3 7303</td>
<td>2068</td>
<td>JAE K43 ; CTL-TABLE-TRANSLATE</td>
</tr>
<tr>
<td>EBEB 0E7009</td>
<td>2069</td>
<td>JMP K56 ; YES, GO TRANSLATE CHAR</td>
</tr>
<tr>
<td>EB61 206A</td>
<td>2070</td>
<td>K43: ; CTL-TABLE-TRANSLATE</td>
</tr>
<tr>
<td>EB64 080C88</td>
<td>2071</td>
<td>MOV BX,OFFSET K9 ; CTL TABLE SCAN</td>
</tr>
<tr>
<td>EB67 E93300</td>
<td>2072</td>
<td>JMP K63 ; TRANSLATE_SCAN</td>
</tr>
<tr>
<td>EB68 2073</td>
<td>2074</td>
<td>K44: ; NOT-CTL-SHIFT</td>
</tr>
<tr>
<td>EB69 3C37</td>
<td>2075</td>
<td>CMP AL,71 ; TEST FOR KEYPAD REGION</td>
</tr>
<tr>
<td>EB6E 7320</td>
<td>2076</td>
<td>JAE K46 ; HANDLE KEYPAD REGION</td>
</tr>
<tr>
<td>EB70 F604170000</td>
<td>2077</td>
<td>TEST KB_FLAG,I,LEFT_SHIFT+RIGHT_SHIFT</td>
</tr>
<tr>
<td>EB72 765B</td>
<td>2078</td>
<td>JZ K54 ; TEST FOR SHIFT STATE</td>
</tr>
<tr>
<td>EB74 2079</td>
<td>2080</td>
<td>K45: ; NOT-BACK-TAB</td>
</tr>
<tr>
<td>EB75 7320</td>
<td>2081</td>
<td>CMP AL,55 ; PRINT SCREEN KEY</td>
</tr>
<tr>
<td>EB7E 7509</td>
<td>2082</td>
<td>JNE K46 ; NOT-PRINT-SCREEN</td>
</tr>
<tr>
<td>EB7F 2083</td>
<td>2084</td>
<td>K46: ; NOT-PRINT-SCREEN</td>
</tr>
<tr>
<td>EB84 3C30</td>
<td>2085</td>
<td>CMP AL,19 ; FUNCTION KEYS</td>
</tr>
<tr>
<td>EB88 1206</td>
<td>2086</td>
<td>JB K47 ; NOT-UPPER-FUNCTION</td>
</tr>
<tr>
<td>EB8B BE59E9</td>
<td>2087</td>
<td>MOV BX,OFFSET K12 ; UPPER CASE PSEUDO SCAN CODES</td>
</tr>
<tr>
<td>EB8C 2088</td>
<td>2089</td>
<td>K47: ; NOT-UPPER-FUNCTION</td>
</tr>
<tr>
<td>EB8E 208A</td>
<td>208B</td>
<td>CMP AL,15 ; BACK TAB KEY</td>
</tr>
<tr>
<td>EB92 7556</td>
<td>208C</td>
<td>JNE K45 ; NOT-BACK-TAB</td>
</tr>
<tr>
<td>EB97 8000F</td>
<td>208D</td>
<td>MOV AX,15*256 ; SET PSEUDO SCAN CODE</td>
</tr>
<tr>
<td>EB9C E86199</td>
<td>208E</td>
<td>JMP K57 ; BUFFER_FILL</td>
</tr>
<tr>
<td>EB9F 208F</td>
<td>2090</td>
<td>K48: ; NOT-BACK-TAB</td>
</tr>
<tr>
<td>EBBC 3C37</td>
<td>2091</td>
<td>CMP AL,55 ; PRINT SCREEN KEY</td>
</tr>
<tr>
<td>EBCE 7509</td>
<td>2092</td>
<td>JNE K46 ; NOT-PRINT-SCREEN</td>
</tr>
<tr>
<td>EBDE 2093</td>
<td>2094</td>
<td>K49: ; NOT-PRINT-SCREEN</td>
</tr>
<tr>
<td>EBFB 765B</td>
<td>2095</td>
<td>CMP AL,19 ; FUNCTION KEYS</td>
</tr>
<tr>
<td>EBFC 2096</td>
<td>2097</td>
<td>JMP K47 ; NOT-UPPER-FUNCTION</td>
</tr>
<tr>
<td>EBFF BE59E9</td>
<td>2098</td>
<td>MOV BX,OFFSET K12 ; UPPER CASE PSEUDO SCAN CODES</td>
</tr>
</tbody>
</table>

A-29
EB90 E9700 2103 JMP K63 ; TRANSLATE_SCAN
2104
EB93 2105 K47: ; NOT-UPPER-FUNCTION
EB95 BB1E9 R 2106 MOV BX,OFFSET K11 ; POINT TO UPPER CASE TABLE
EB96 EB40 2107 JMP SHORT K56 ; OK, TRANSLATE THE CHAR
EB98 2108
EB9B F606170020 R 2110 ; ------ KEYBOARD KEYS, MUST TEST NUM LOCK FOR DETERMINATION
EB9D 7520 2111 JMP K52 ; TEST FOR SURE
EBF9 F606170003 R 2114 TEST KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT ; ARE WE IN SHIFT STATE
EB4A 7520 2115 JNZ K53 ; IF SHIFTED, REALLY NUM STATE
2116
EB4B 2117 ; ------ BASE CASE FOR KEYPAD
EB4E 2118 K49: ; BASE-CASE
2119
EB4E 3C4A 2120 CMP AL,74 ; SPECIAL CASE FOR A COUPLE OF KEYS
EB4E 740B 2122 JE K50 ; MINUS
EBA A 3C4E 2123 CMP AL,70
EBA C 740C 2124 JE K51
EBAC 2C47 2125 SUB AL,71 ; CONVERT ORIGIN
EBBD BB7E9 R 2126 MOV BX,OFFSET K15 ; BASE CASE TABLE
EBBE 877 2127 JMP SHORT K64 ; CONVERT TO PSEUDO SCAN
2128
EBBF 2128 B02A4 A MOV AX,74*256+'-'; MINUS
EBB9 822 2130 JMP SHORT K57 ; BUFFER_FILL
EBBB B02E9 2131
EBBC B010E 2132 K51: MOV AX,78*256+' '; PLUS
EBBE 2133 JMP SHORT K57 ; BUFFER_FILL
2134
EBBF 2135 ; ------ MIGHT BE NUM LOCK, TEST SHIFT STATUS
EBBF F606170003 R 2136 K52: ; ALMOST-NUM-STATE
EBC4 75E0 2138 TEST KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT
EBE6 2139 JNZ K49 ; SHIFTED TEMP OUT OF NUM STATE
2140
EBC6 2141 K53: ; REALLY_NUM_STATE
EBC6 2C46 2142 SUB AL,70 ; CONVERT ORIGIN
EBCC BB60E9 R 2143 MOV BX,OFFSET K14 ; NUM STATE TABLE
EBCE B008 2144 JMP SHORT K54 ; TRANSLATE_CHAR
2145
EBD0 2146 ; ------ PLAIN OLD LOWER CASE
EBD0 F60617000 D 2147 K54: ; NOT-SHIFT
EBDC 3C3B 2148 CMP AL,89 ; TEST FOR FUNCTION KEYS
EBEC 7204 2149 JB K55 ; NOT-LOWER-FUNCTION
EBD1 BB00 2150 MOV AL,0 ; SCAN CODE IN AL READY
EBD3 BB07 2151 JMP SHORT K57 ; BUFFER_FILL
2152
EBD5 2153 K55: ; NOT-LOWER-FUNCTION
EBD5 BB0E9 R 2154 MOV BX,OFFSET K10 ; LC TABLE
EBD5 2155 JMPSHORT K56 ; TRANSLATE_CHAR
EBD5 2156
EBD0 2157 ; ------ TRANSLATE THE CHARACTER
EBD0 2158 K56: ; TRANSLATE-CHAR
EBD0 FEB8 2159 DEC AL ; CONVERT ORIGIN
EBDA 2ED7 2160 XLAT CS:K11 ; CONVERT THE SCAN CODE TO ASCII
EBDC 2161
EBDC 2162 ; ------ PUT CHARACTER INTO BUFFER
EBDC 2163 K57: ; BUFFER-FILL
EBDC 3CFF 2164 CMP AL,-1 ; IS THIS AN IGNORE CHAR
EBDF 741F 2165 JE K59 ; YES, DO NOTHING WITH IT
EBE0 B08FF 2166 CMP AH,-1 ; LOOK FOR -1 PSEUDO SCAN
EBE3 741A 2167 JE K59 ; NEAR_INTERRUPT_RETURN
2169
EBE5 2170 ; ------ HANDLE THE CAPS LOCK PROBLEM
EBE5 F606170040 R 2171 K58: ; BUFFER-FILL-NOTEST
EBEA 7420 2172 TEST KB_FLAG,CAPS_STATE ; ARE WE IN CAPS LOCK STATE
EBE5 2173 JZ K61 ; SKIP IF NOT
2175
EBE6 2176 ; ------ IN CAPS LOCK STATE
EBE6 F606170003 R 2178 TEST KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT ; TEST FOR SHIFT STATE

A-30
LOC OBJ | LINE | SOURCE

EBF1 740F | 2180 | JZ K60 | IF NOT SHIFT, CONVERT LOWER TO UPPER
2181
2182 | -------- CONVERT ANY UPPER CASE TO LOWER CASE
2183
EBF3 3C41 | 2184 | CMP AL, 'A' | FIND OUT IF ALPHABETIC
2185 | JB K61 | NOT_CAPS_STATE
EBF5 7215 | 2186 | CMP AL, 'Z' | FIND OUT IF ALPHABETIC
2187 | JB K61 | NOT_CAPS_STATE
EBF7 3C5A | 2188 | ADD AL, 'a'-'A' | CONVERT TO LOWER CASE
2189 | JMP SHORT K61 | NOT_CAPS_STATE
2190
EBFF | 2191 | K59: | NEAR-INTERUPT-RETURN
2192 | JMP K26 | INTERRUPT_RETURN
2193
EBFF F99CFE | 2194 | -------- CONVERT ANY LOWER CASE TO UPPER CASE
2195
EBF 2196 | K60: | LOWER-TO-UPPER
ECO2 2197 | 2198 | CMP AL, 'a' | FIND OUT IF ALPHABETIC
2199 | JB K61 | NOT_CAPS_STATE
ECO4 7206 | 2200 | CMP AL, 'z' | FIND OUT IF ALPHABETIC
2201 | JB K61 | NOT_CAPS_STATE
ECO6 7702 | 2202 | SUB AL, 'a'-'A' | CONVERT TO LOWER CASE
2203
EC0 | 2204 | K61: | NOT-CAPS-STATE
EC0C BB1EC00 | 2205 | MOV BX, BUFFER_TAIL | GET THE END POINTER TO THE BUFFER
2206 | MOV SI, BX | SAVE THE VALUE
2207 | CALL K4 | ADVANCE THE TAIL
2208 | CMP BX, BUFFER_HEAD | HAS THE BUFFER WRAPPED AROUND
2209 | JE K62 | BUFFER_FULL_BEEP
2210 | MOV AX, BX | STORE THE VALUE
2211 | JMP K26 | INTERRUPT_RETURN
2212
EC21 493FE | 2213 | -------- BUFFER IS FULL, SOUND THE BEEPER
2214
EC24 60000 | 2215 | K62: | BUFFER-FULL-BEEP
2216 | CALL ERROR_BEEP | INTERRUPT_RETURN
2217 | JMP K26 | INTERRUPT_RETURN
2218
EC2A 2C3B | 2219 | -------- TRANSLATE SCAN FOR PSEUDO SCAN CODES
2220
EC2A 2EC0 | 2221 | K63: | TRANSLATE-SCAN
2222 | SUB AL, 59 | CONVERT ORIGIN TO FUNCTION KEYS
2223 | K64: | TRANSLATE-SCAN-ORGD
2224 | XLAT CS:K9 | CTL TABLE SCAN
2225 | MOV AX, AL | PUT VALUE INTO AH
2226 | MOV AL, 0 | ZERO ASCII CODE
2227 | JMP K57 | PUT IT INTO THE BUFFER
2228
EC30 8934 | 2229 | KB_INT END | ERROR_BEEP proc near
2230 | KB_INT | ERROR_BEEP proc near
2231 | PUSH AX | SAVE REGISTERS
2232 | PUSH BX | SAVE REGISTERS
2233 | PUSH CX | SAVE REGISTERS
2234 | MOV BX, 0C0H | NUMBER OF CYCLES FOR 1/8 SECOND TONE
2235 | IN AL, KB_CTL | GET CONTROL INFORMATION
2236 | PUSH AX | SAVE
2237 | K65: | BEEP-CYCLE
2238 | AMD AL, 0FCH | TURN OFF TIMER GATE AND SPEAKER DATA
2239 | OUT KB_CTL, AL | OUTPUT TO CONTROL
2240 | MOV CX, 0D8H | HALF CYCLE TIME FOR TONE
2241 | LOOP K66 | SPEAKER OFF
2242 | OR AL, 2 | TURN ON SPEAKER BIT
2243 | OUT KB_CTL, AL | OUTPUT TO CONTROL
2244 | MOV CX, 0D8H | SET UP COUNTER
2245 | LOOP K67 | ANOTHER HALF CYCLE
2246 | DEC BX | TOTAL TIME COUNT
2247 | JNZ K65 | DO ANOTHER CYCLE
2248 | POP AX | RECOVER CONTROL
2249 | OUT KB_CTL, AL | OUTPUT THE CONTROL
2250 | POP CX | RECOVER REGISTERS
2251 | POP BX | RECOVER REGISTERS
2252 | POP AX | RECOVER REGISTERS
2253 | RET |
LOC OBJ LINE SOURCE

2255 ;-- INT 13 -----------------------------------------
2256 1 DISKETTE I/O
2257 ; THIS INTERFACE PROVIDES ACCESS TO THE 5 1/4" DISKETTE DRIVES
2258 1 INPUT
2259 1 (AH)=0 RESET DISKETTE SYSTEM
2260 1 (AH)=1 READ THE STATUS OF THE SYSTEM INTO (AL)
2261 1 DISKETTE STATUS FROM LAST DP"H IS USED
2262 1 REGISTERS FOR READ/WRITE/VERIFY/FORMAT
2263 1 (DL) - DRIVE NUMBER (0-3 ALLOWED, VALUE CHECKED)
2264 1 (CH) - HEAD NUMBER (0-1 ALLOWED, NOT VALUE CHECKED)
2265 1 (CL) - SECTOR NUMBER (1-8, NOT VALUE CHECKED)
2266 1 (AL) - NUMBER OF SECTORS ( MAX = 8, NOT VALUE CHECKED)
2267 1 (DS:BX) - ADDRESS OF BUFFER ( NOT REQUIRED FOR VERIFY)
2268 1 READ THE DESIRED SECTORS INTO MEMORY
2269 1 WRITE THE DESIRED SECTORS FROM MEMORY
2270 1 VERIFY THE DESIRED SECTORS
2271 1 FORMAT THE DESIRED TRACK
2272 1 FOR THE FORMAT OPERATION, THE BUFFER POINTER (ES:BX) MUST
2273 1 POINT TO THE COLLECTION OF DESIRED ADDRESS FIELDS FOR THE
2274 1 TRACK. EACH FIELD IS COMPOSED OF 4 BYTES, (C,H,R,N), WHERE
2275 1 C = TRACK NUMBER, H = HEAD NUMBER, R = SECTOR NUMBER, N = NUMBER
2276 1 OF BYTES PER SECTOR (00=128, 01=256, 02=512, 03=1024, )
2277 1 THERE MUST BE ONE ENTRY FOR EVERY SECTOR ON THE TRACK.
2278 1 THIS INFORMATION IS USED TO FIND THE REQUESTED SECTOR DURING
2279 1 READ/WRITE ACCESS.
2280 1 DATA VARIABLE -- DISK_POINTER
2281 1 DOUBLE WORD POINTER TO THE CURRENT SET OF DISKETTE PARAMETERS
2282 1 OUTPUT
2283 1
2284 1 AH = STATUS OF OPERATION
2285 1 STATUS BITS ARE DEFINED IN THE EQUATES FOR DISKETTE_STATUS
2286 1 VARIABLE IN THE DATA SEGMENT OF THIS MODULE
2287 1 CY = 0 SUCCESSFUL OPERATION (AH=0 ON RETURN)
2288 1 CT = 1 FAILED OPERATION (AH HAS ERROR REASON)
2289 1 FOR READ/WRITE/VERIFY
2290 1 DS,BX,DX,CH,CL PRESERVED
2291 1 AL = NUMBER OF SECTORS ACTUALLY READ
2292 1 **** AL MAY NOT BE CORRECT IF TIME OUT ERROR OCCURS
2293 1 NOTE: IF AN ERROR IS REPORTED BY THE DISKETTE CODE, THE APPROPRIATE
2294 1 ACTION IS TO RESET THE DISKETTE, THEN RETRY THE OPERATION.
2295 1 ON READ ACCESSES, NO MOTOR START DELAY IS TAKEN, SO THAT
2296 1 THREE RETRIES ARE REQUIRED ON READS TO ENSURE THAT THE
2297 1 PROBLEM IS NOT DUE TO MOTOR START-UP.
2298 1
2299 1 PROC
2300 1 DISKETTE_IO
2301 1 DISKETTE_IO PROC FAR
2302 1 --- CS:CODE,DS:DATA,ES:DATA
2303 1 STI
2304 1 INTO BACK ON
2305 1 INTERRUPTS
2306 1 PUSH BX ; SAVE ADDRESS
2307 1 PUSH CX
2308 1 PUSH DS ; SAVE SEGMENT REGISTER VALUE
2309 1 PUSH SI ; SAVE ALL REGISTERS DURING OPERATION
2310 1 PUSH DI
2311 1 PUSH BP
2312 1 PUSH DX
2313 1 MOV BX,SP ; SET UP POINTER TO HEAD PARM
2314 1 MOV DS:DATA ; SET DATA REGION
2315 1 CALL JI ; CALL THE REST TO ENSURE DS RESTORED
2316 1 MOV BX,4 ; GET THE MOTOR WAIT PARAMETER
2317 1 CALL GET_PARM
2318 1 MOV MOTOR_COUNT,AH ; SET THE TIMER COUNT FOR THE MOTOR
2319 1 MOV AH,DISKETTE_STATUS ; GET STATUS OF OPERATION
2320 1 CMP AH,1 ; SET THE CARRY FLAG TO INDICATE
2321 1 CMP AH,1 ; SUCCESS OF FAILURE
2322 1 MOV DX ; RESTORE ALL REGISTERS
2323 1 POP BP
2324 1 POP DI
2325 1 POP DS
2326 1 POP DX
2327 1 POP CX
2328 1 POP BX ; RECOVER ADDRESS
2329 1 POP ES
2330 1 MOV AX,ES:DATA
2331 1 PROC NEAR
2332 1 DISKETTE_IO ENDP

A-32
LOC OBJ  |  LINE  |  SOURCE
---  |  ---  |  ---
EC87 BAFO  |  2332  |  MOV DH, AL  ; SAVE 7 SECTORS IN DH
EC89 00263F007F  |  2333  |  AND MOTOR_STATUS, 0FFH  ; INDICATE A READ OPERATION
EC8E 08A4  |  2334  |  OR AH, AH  ; AH= 0
EC90 74A2  |  2335  |  JZ DISK_RESET
EC92 FECC  |  2336  |  DEC AH  ; AH= 1
EC94 7474  |  2337  |  JZ DISK_STATUS
EC96 C60641000a  |  2338  |  MOV DISKETTE_STATUS, 0  ; RESET THE STATUS INDICATOR
EC98 08AF04  |  2339  |  CMP DL, 4  ; TEST FOR DRIVE IN 0-3 RANGE
EC9E 7131  |  2340  |  JAE J3  ; ERROR IF ABOVE
ECAC FECC  |  2341  |  DEC AH  ; AH= 2
ECB2 746A  |  2342  |  JZ DISK_READ
ECB4 FECC  |  2343  |  DEC AH  ; AH= 3
ECB6 7503  |  2344  |  JNZ J2  ; TEST_DISK_VERF
ECB8 E99600  |  2345  |  JMP DISK_WRITE  ; TEST_DISK_VERF
ECB4 2346  |  J2:  |  JMP DISK_WRITE
ECB8 FECC  |  2347  |  DEC AH  ; AH= 4
ECB8 7468  |  2348  |  JZ DISK_VERF
ECB8 FECC  |  2349  |  DEC AH  ; AH= 5
ECBA 7468  |  2350  |  JZ DISK_FORMAT
ECBC FECC  |  2351  |  J5:  ; BAD_COMMAND
ECBD C606410001  |  2352  |  MOV DISKETTE_STATUS, BAD_CMD  ; ERROR CODE, NO SECTORS TRANSFERRED
ECBE C3  |  2353  |  RET  ; UNDEFINED OPERATION
ECBF J1  |  2354  |  ENDP
ECBF 2355  |  ;------ RESET THE DISKETTE SYSTEM
ECBF 2356  |  DISK_RESET  |  PROC NEAR
ECBF BAF03  |  2357  |  MOV DX, 0032H  ; ADAPTER CONTROL PORT
ECBF FA  |  2358  |  CLI  ; NO INTERRUPTS
ECBF A03F000  |  2359  |  MOV AL, MOTOR_STATUS  ; WHICH MOTOR IS ON
ECBF B104  |  2360  |  MOV CL, 4  ; SHIFT COUNT
ECBF B340  |  2361  |  SAR AL, CL  ; MOVE MOTOR VALUE TO HIGH HYBRID
ECBF 7460  |  2362  |  TEST AL, 2H  ; SELECT CORRESPONDING DRIVE
ECBF C20C  |  2363  |  JNZ J5  ; JUMP IF MOTOR ONE IS ON
ECBF C4A4  |  2364  |  TEST AL, 40H  ; JUMP IF MOTOR TWO IS ON
ECBF CA04  |  2365  |  JNZ J4  ; JUMP IF MOTOR THREE IS ON
ECBF CC50  |  2366  |  TEST AL, 80H  ; JUMP IF MOTOR ZERO IS ON
ECBF CC40  |  2367  |  JZ J6  ; JUMP IF MOTOR ZER0 IS ON
ECBF E00E  |  2368  |  INC AL
ECBF E00F  |  2369  |  JNZ J4  ; INC AL
ECBF E010  |  2370  |  INC AL
ECBF E011  |  2371  |  J6:  OR AL, O  ; TURN ON INTERRUPT ENABLE
ECBF E012  |  2372  |  OUT DX, AL  ; RESET THE ADAPTER
ECBF E013  |  2373  |  MOV DX, 0DF2H  ; ADAPTER CONTROL PORT
ECBF E014  |  2374  |  MOV DISKETTE_STATUS, 0  ; SET OK STATUS FOR DISKETTE
ECBF E015  |  2375  |  MOV AL, 4  ; TURN OFF RESET
ECBF E016  |  2376  |  OUT DX, AL  ; TURN OFF THE RESET
ECBF E017  |  2377  |  STI  ; REENABLE THE INTERRUPTS
ECBF E02002  |  2378  |  CALL CMK_STAT_2  ; DO SENSE INTERRUPT STATUS FOLLOWING RESET
ECBF E040200  |  2379  |  MOV AL, NEC_STATUS  ; IGNORE ERROR RETURN AND DO OWN TEST
ECBF E050  |  2380  |  CMP AL, 00H  ; TEST FOR DRIVE READY TRANSITION
ECBF E047  |  2381  |  JZ J7  ; EVERYTHING OK
ECBF E00E410000  |  2382  |  OR DISKETTE_STATUS, BAD_NEC  ; SET ERROR CODE
ECBF E00E410020  |  2383  |  JMP SHORT J8  ; RESET RET
ECBF E00E  |  2384  |  ;------ SEND SPECIFY COMMAND TO NEC
ECBF E00E  |  2385  |  MOV AH, 03H  ; SPECIFY COMMAND
ECBF E00F  |  2386  |  CALL NEC_OUTPUT  ; OUTPUT THE COMMAND
ECBF E01000  |  2387  |  MOV BX, 1  ; FIRST BYTE PARM IN BLOCK
ECBF E01001  |  2388  |  CALL GET_PARM  ; TO THE NEC CONTROLLER
ECBF E01003  |  2389  |  MOV BX, 2  ; SECOND BYTE PARM IN BLOCK
ECBF E01004  |  2390  |  CALL GET_PARM  ; TO THE NEC CONTROLLER
ECBF E01005  |  2391  |  JBP J10  ; RETURN TO CALLER
ECBF E010C3  |  2392  |  RET  ; RETURN TO CALLER
ECBF E010C5  |  2393  |  DISK_RESET  ; ENDP
ECBF E010C6  |  2394  |  ;------ DISKETTE STATUS ROUTINE
ECBF E010C7  |  2395  |  MOV AL, DISKETTE_STATUS
ECBF E010C8  |  2396  |  RET  ; ENDP
DISK_READ PROC NEAR
  MOV AL,046H ; READ COMMAND FOR DMA
  CALL DMA_SETUP ; SET UP THE DMA
  JMP SHORT RW_OPN ; GO DO THE OPERATION
DISK_READ ENDP

DISK_VERIFY PROC NEAR
  MOV AL,042H ; VERIFY COMMAND FOR DMA
  JMP SHORT RW_OPN ; GO DO THE OPERATION
DISK_VERIFY ENDP

DISK_FORMAT PROC NEAR
  MOV AL,044H ; WRITE COMMAND
  CALL DMA_SETUP
  MOV AH,04DH ; ESTABLISH THE FORMAT COMMAND
  JMP SHORT RW_OPN ; DO THE R/O OPERATION

DISK_WRITE PROC NEAR
  OR MOTOR_STATUS,80H ; INDICATE WRITE OPERATION
  MOV AL,04AH ; DMA WRITE COMMAND
  CALL DMA_SETUP
  MOV AH,045H ; COMMAND TO WRITE TO DISKETTE
  JMP SHORT RW_OPN ; ALLOW WRITE ROUTINE TO FALL INTO RW_OPN

RH_OPN PROC NEAR
  JNC J11 ; TEST FOR DMA ERROR
  MOV DISK_STATUS,DATA ; TEST ERROR
  MOV AL,0 ; HD SECTORS TRANSFERRED
  RET ; RETURN TO MAIN ROUTINE
  J11: DD_RW_OPN

THIS ROUTINE PERFORMS THE READ/WRITE/VERIFY OPERATION

TURN ON THE MOTOR AND SELECT THE DRIVE

SAVE THE T/S PAVMS
GET DRIVE NUMBER AS SHIFT COUNT
MASK FOR DETERMINING MOTOR BIT
SHIFT THE MASK BIT
NO INTERRUPTS WHILE DETERMINING MOTOR STATUS
MOV MOTOR_COUNT,OFFH ; SET LARGE COUNT DURING OPERATION
TEST AL,MOTOR_STATUS ; TEST THAT MOTOR FOR OPERATING
JNZ J14 ; IF RUNNING, SKIP THE WAIT
AND MOTOR_STATUS,OFFH ; TURN OFF ALL MOTOR BITS
OR MOTOR_STATUS,AL ; TURN ON THE CURRENT MOTOR
INTERRUPTS BACK ON
MOV AL,10H ; MASK BIT
SAI AL,CL ; DEVELOP BIT PAK FOR MOTOR ENABLE
GET DRIVE SELECT BITS IN
NO RESET, DISABLE DMA/INT
SAVE REG
MOV DX,03F2H ; CONTROL PORT ADDRESS
OUT DX,AL

SAVE THE COMMAND
SAVE THE COMMAND
f082: 5A 2464 2485
   POP ox
   TEST MOTOR_STATUS.88H
   IS THIS A WRITE
   OR ah, ah
   TEST FOR NO WAIT
   JZ J13:
   WAIT FOR MOTOR

E081 F603 0080
E082 7412
JZ J14:
NO, CONTINUE WITHOUT WAIT
E084 00E4
CALL GET_PARM
PARAMETER
E085 28C9
OR AH, AH
TEST FOR NO WAIT
E086 7408
JZ J14:
EXIT WITH TIME EXPIRED
E087 2BC9
SUB CX, CX
SET UP 1/8 SECOND LOOP TIME
E088 FECE
J13:
LOOP J13
WAIT FOR THE REQUIRED TIME
E089 F788
DEC AH
DECREMENT TIME VALUE
E08A EB6F
JMP J12:
ARE WE DONE YET
E08B 7406
J14:
; MOTOR_RUNNING

E090 2498
; ------ WAIT FOR MOTOR IF WRITE OPERATION
E091 00
TEST MOTOR_STATUS.88H
JZ J14:
NO, CONTINUE WITHOUT WAIT

E095 28C9
OR AH, AH
TEST FOR NO WAIT
E096 7408
JZ J14:
EXIT WITH TIME EXPIRED
E097 2BC9
SUB CX, CX
SET UP 1/8 SECOND LOOP TIME
E098 FECE
J13:
LOOP J13
WAIT FOR THE REQUIRED TIME
E099 F788
DEC AH
DECREMENT TIME VALUE
E09A EB6F
JMP J12:
ARE WE DONE YET
E09B 7406
J14:
; MOTOR_RUNNING

E0A0 2498
; ------ DO THE SEEK OPERATION
E0A1 00
TEST MOTOR_STATUS.88H
JZ J14:
NO, CONTINUE WITHOUT WAIT

E0A5 28C9
OR AH, AH
TEST FOR NO WAIT
E0A6 7408
JZ J14:
EXIT WITH TIME EXPIRED
E0A7 2BC9
SUB CX, CX
SET UP 1/8 SECOND LOOP TIME
E0A8 FECE
J13:
LOOP J13
WAIT FOR THE REQUIRED TIME
E0A9 F788
DEC AH
DECREMENT TIME VALUE
E0AA EB6F
JMP J12:
ARE WE DONE YET
E0AB 7406
J14:
; MOTOR_RUNNING

E0B0 2498
; ------ SEND OUT THE PARAMETERS TO THE CONTROLLER
E0B1 00
TEST MOTOR_STATUS.88H
JZ J14:
NO, CONTINUE WITHOUT WAIT

E0B5 28C9
OR AH, AH
TEST FOR NO WAIT
E0B6 7408
JZ J14:
EXIT WITH TIME EXPIRED
E0B7 2BC9
SUB CX, CX
SET UP 1/8 SECOND LOOP TIME
E0B8 FECE
J13:
LOOP J13
WAIT FOR THE REQUIRED TIME
E0B9 F788
DEC AH
DECREMENT TIME VALUE
E0BA EB6F
JMP J12:
ARE WE DONE YET
E0BB 7406
J14:
; MOTOR_RUNNING

E0C0 2498
; ------ TEST FOR FORMAT COMMAND
E0C1 00
TEST MOTOR_STATUS.88H
JZ J14:
NO, CONTINUE WITHOUT WAIT

E0C5 28C9
OR AH, AH
TEST FOR NO WAIT
E0C6 7408
JZ J14:
EXIT WITH TIME EXPIRED
E0C7 2BC9
SUB CX, CX
SET UP 1/8 SECOND LOOP TIME
E0C8 FECE
J13:
LOOP J13
WAIT FOR THE REQUIRED TIME
E0C9 F788
DEC AH
DECREMENT TIME VALUE
E0CA EB6F
JMP J12:
ARE WE DONE YET
E0CB 7406
J14:
; MOTOR_RUNNING

E0D0 2498
; ------ LET THE OPERATION HAPPEN
E0D1 00
TEST MOTOR_STATUS.88H
JZ J14:
NO, CONTINUE WITHOUT WAIT

E0D5 28C9
OR AH, AH
TEST FOR NO WAIT
E0D6 7408
JZ J14:
EXIT WITH TIME EXPIRED
E0D7 2BC9
SUB CX, CX
SET UP 1/8 SECOND LOOP TIME
E0D8 FECE
J13:
LOOP J13
WAIT FOR THE REQUIRED TIME
E0D9 F788
DEC AH
DECREMENT TIME VALUE
E0DA EB6F
JMP J12:
ARE WE DONE YET
E0DB 7406
J14:
; MOTOR_RUNNING

E0E0 2498
; ------ CHECK THE RESULTS RETURNED BY THE CONTROLLER
E0E1 00
TEST MOTOR_STATUS.88H
JZ J14:
NO, CONTINUE WITHOUT WAIT

E0E5 28C9
OR AH, AH
TEST FOR NO WAIT
E0E6 7408
JZ J14:
EXIT WITH TIME EXPIRED
E0E7 2BC9
SUB CX, CX
SET UP 1/8 SECOND LOOP TIME
E0E8 FECE
J13:
LOOP J13
WAIT FOR THE REQUIRED TIME
E0E9 F788
DEC AH
DECREMENT TIME VALUE
E0EA EB6F
JMP J12:
ARE WE DONE YET
E0EB 7406
J14:
; MOTOR_RUNNING

E0F0 2498
; ------ TEST FOR NORMAL TERMINATION
E0F1 00
TEST MOTOR_STATUS.88H
JZ J14:
NO, CONTINUE WITHOUT WAIT

E0F5 28C9
OR AH, AH
TEST FOR NO WAIT
E0F6 7408
JZ J14:
EXIT WITH TIME EXPIRED
E0F7 2BC9
SUB CX, CX
SET UP 1/8 SECOND LOOP TIME
E0F8 FECE
J13:
LOOP J13
WAIT FOR THE REQUIRED TIME
E0F9 F788
DEC AH
DECREMENT TIME VALUE
E0FA EB6F
JMP J12:
ARE WE DONE YET
E0FB 7406
J14:
; MOTOR_RUNNING

E0FC 2498
; ------ SET THE CORRECT DIRECTION
E0FD 2498
; ------ WAIT FOR THE INTERRUPT
E0FE 2498
; ------ TEST FOR NORMAL TERMINATION
; TEST FOR ABNORMAL TERMINATION
; NOT ABNORMAL, BAD NEC

; TEST FOR RECORD NOT FOUND
; TEST FOR WRITE PROTECT
; TEST FOR DMA OVERRUN
; TEST FOR BAD ADDRESS MARK

; OPERATION WAS SUCCESSFUL
LOC OBJ  
LINE  
SOURCE

EE52 800E+10000  R  2639  OR  DISKETTE_STATUS,TIME_OUT
EE55 59  2640  POP  CX  SET ERROR CODE AND RESTORE REGS
EE55 5A  2641  POP  DX
EE59 5B  2642  POP  AX  DISCARD THE RETURN ADDRESS
EE5A F9  2643  STC  INDICATE ERROR TO CALLER
EE5B C5  2644  RET

EE5C 2645
EE5C 33C9  2646  XOR  CX,CX  RET

EE5E 2647
EE5E J26:  2648  IN  AL,DX  GET THE STATUS
EE5F A000  2649  TEST  AL,000H  IS IT READY
EE61 7504  2650  JNZ  J27  YES, GO OUTPUT
EE63 E2F9  2651  LOOP  J26  COUNT DOWN AND TRY AGAIN
EE65 EEBB  2652  JNP  J24  ERROR CONDITION
EE67 2653
EE67 J27:  2654  POP  AL,AH  GET BYTE TO OUTPUT
EE69 BAF503  2655  MOV  AL,DX  DATA PORT
EE6C EE  2656  OUT  DX,AL  OUTPUT THE BYTE
EE6D 59  2657  POP  CX  RECOVER REGISTERS
EE6E 5A  2658  POP  DX
EE6F C3  2659  RET  CY = 0 FROM TEST INSTRUCTION

EE70 2660
EE70 1E  2661  -----------------------------------------­
EE71 2BC0  2662 ; GET_PARM
EE75 C5367800  2663 ; THIS ROUTINE FETCHES THE INDEXED POINTER FROM
EE77 D1EB  2664 ; THE DISK_BASE BLOCK POINTED AT BY THE DATA
EE79 8406  2665 ; VARIABLE DISK_POINTER
EE81 06063E00  2666 ; A BYTE FROM THAT TABLE IS THEN MOVED INTO AH,
EE85 6407  2667 ; THE INDEX OF THAT BYTE BEING THE PARM IN BX
EE87 E6ACFF  2668 ; ENTRY --
EE89 06063E00  2669 ; BX = INDEX OF BYTE TO BE Fetched = 2
EE8F 7513  2670 IF THE LOW BIT OF BX IS ON, THE BYTE IS IMMEDIATELY
EE91 06063E00  2671 OUTPUT TO THE NEC CONTROLLER
EE93 06063E00  2672 EXIT --
EE95 72C4  2673 ; AH = THAT BYTE FROM BLOCK
EE96 C3  2674 ;-------------------------------------------­
EE98 C3  2675 ; GET_PARM ENOP

EEA0 2676
EEA0 1E  2677  -----------------------------------------­
EEA1 2BC0  2678 ; SEEK
EEA5 C5367800  2679 ; TO THE NAMED TRACK. IF THE DRIVE HAS NOT BEEN
EEA7 D1EB  2680 ; Sficssed, THE DRIVE WILL BE
EEA9 8406  2681 ; SINCE THE DRIVE RESET COMMAND WAS ISSUED, THE DRIVE WILL BE
EEAB E6ACFF  2682 ; RECALIBRATED.
EEAD 06063E00  2683 ; INPUT
EEAE 72C4  2684 ; (DL) = DRIVE TO SEEK ON
EEAF C3  2685 ; (CH) = TRACK TO SEEK TO
EEB1 1F  2686 ; OUTPUT
EEB2 06063E00  2687 ; CY = 0 SUCCESS
EEB4 72C4  2688 ; CY = 1 FAILURE -- DISKETTE_STATUS SET ACCORDINGLY
EEB5 C3  2689 ; (AX) DESTROYED

EEB8 2690
EEB8 1E  2691  -----------------------------------------­
EEB9 2BC0  2692 ; SEEK
EEBC C5367800  2693 ; ESTABLISH MASK FOR RECIAL TEST
EEBF 51  2694 
EEC4 84063E00  2695 ; MOV  AL,00H  SAVE INPUT VALUES
EEC7 7513  2696 ; MOV  CL,DL  GET DRIVE VALUE INTO CL
EECB 06063E00  2697 ; ROL  AL,CL  SHIFT IT BY THE DRIVE VALUE
EECE 6407  2698 ; POP  CX  RECOVER TRACK VALUE
EEF0 00063E00  2699 ; TEST AL,SEEK_STATUS  TEST FOR RECIAL REQUIRED
EEF2 7513  2700 ; JNZ  J26  NO_RECAL
EEF4 00063E00  2701 ; OR  SEEK_STATUS,AL  TURN ON THE NO RECIAL BIT IN FLAG
EEF6 B407  2702 ; MOV  AH,07H  RECALIBRATE COMMAND
EEF8 E6ACFF  2703 ; CALL  NEC_OUTPUT
EEF9 8406  2704 ; MOV  AH,DL
EEFA E6ACFF  2705 ; CALL  NEC_OUTPUT
EEFB 6407  2706 ; MOV  AH,DL  OUTPUT THE DRIVE NUMBER

A-37
Ef'TD £67200 2716  CALL  CHK_STAT_2  ; GET THE INTERRUPT AND SENSE INT STATUS
                      JC    J32  ; SEEK_ERROR

2719  i----- DRIVE IS IN SYNCH WITH CONTROLLER. SEEK TO TRACK

2720  

2721  J28:  

2722  MOV  AH,0FH  ; SEEK COMMAND TO NEC

2723  CALL  NEC_OUTPUT  ; DRIVE NUMBER

2724  MOV  AH,DL  ; TRACK NUMBER

2725  CALL  NEC_OUTPUT  ; GET ENDING INTERRUPT AND SENSE STATUS

2726  CALL  CHK_STAT_2  ; SEEK_ERROR

2727  JC    J32  ; SEEK ERROR

2728  ;----- WAIT FOR HEAD SETTLE

2729  PUSHF ; SAVE STATUS FLAGS

2730  MOV  BX,18 ; GET HEAD SETTLE PARAMETER

2731  MOV  AH,AL  ; TEST FOR TIME EXPIRED

2732  JMP  J30  ; DO IT SOME MORE

2733  INC  CH  ; CARRY MEANS HIGH 4 BITS MUST BE INC

2734  POP  AX  ; SAVE COUNT VALUE

2735  SHR  AX,1 ; -1 FOR DMA VALUE

2736  DEC  AX  ; TEST FOR CARRY FROM ADDITION

2737  JNC  J33

2738  J31:

2739  POP  AX  ; SAVE START ADDRESS

2740  MOV  AL,CH  ; GET HIGHEST NYBBLE OF ES TO CH

2741  MOV  CL,4  ; SHIFT COUNT

2742  JMP  J29

2743  POP  CX  ; RECOVER STATE

2744  POPF

2745  J32:

2746  ; RETURN TO CALLER

2747  SEEK ENDP

2748  ;-----------------------------

2749  ; DNA_SETUP

2750  THIS ROUTINE SETS UP THE DMA FOR READ/WRITE/VERIFY

2751  OPERATIONS.

2752  INPUT  

2753  OUTPUT  

2754  ; AX = MODE BYTE FOR THE DMA

2755  ; (ES:BX) = ADDRESS TO READ/ WRITE THE DATA

2756  ; AX = LENGTH OF DMA TRANSFER

2757  ; (AX) DESTROYED

2758  i-----------------------------

2759  ; DMA_SETUP  PROC NEAR

2760  PUSH  CX  ; SAVE REGISTRER

2761  MOV  DMA+12,AL  ; SET THE FIRST/LAST F/F

2762  MOV  DMA+11,AL  ; OUTPUT THE MODE BYTE

2763  MOV  AX,ES  ; GET THE ES VALUE

2764  MOV  CL,4  ; SHIFT COUNT

2765  ROL  AX,CL  ; ROTATE LEFT

2766  MOV  CH,4  ; GET HIGHEST NYBBLE OF ES TO CH

2767  AND  AL,0FH  ; ZERO THE LOW NYBBLE FROM SEGMENT

2768  ADD  AX,BX  ; TEST FOR CARRY FROM ADDITION

2769  

2770  ; INC CH  ; CARRY MEANS HIGH 4 Bits MUST BE INC

2771  J33:  

2772  PUSH  AX  ; SAVE START ADDRESS

2773  MOV  DMA+4,AL  ; OUTPUT LOW ADDRESS

2774  MOV  AL,AH  ; GET HIGH 4 BITS

2775  MOV  DMA+4,AL  ; OUTPUT HIGH ADDRESS

2776  MOV  AL,CH  ; GET HIGH 4 BITS

2777  MOV  AL,0FH  ; USE AS SHIFT COUNT (0128, 1256 ETC)

2778  OUT  08H,AL  ; OUTPUT THE HIGH 4 BITS TO PAGE REGISTER

2779  ;-----------------------------

2780  ; DETERMINE COUNT

2781  ; NUMBER OF SECTORS

2782  MOV  AH,0DH  ; TIMES 256 INTO AX

2783  SHR  AX,1  ; SECTORS + 128 INTO AX

2784  PUSH  AX

2785  MOV  BX,4  ; GET THE BYTES/SECTOR PARM

2786  CALL  GET_PARM

2787  MOV  CL,AM  ; USE AS SHIFT COUNT (0128, 1256 ETC)

2788  MOV  CL,AM  ; USE AS SHIFT COUNT (0128, 1256 ETC)

2789  POP  AX

2790  SHR  AX,CL  ; MULTIPLY BY CORRECT AMOUNT

2791  DEC  AX  ; -1 FOR DMA VALUE

2792  PUSH  AX  ; SAVE COUNT VALUE

2793  MOV  AL,CH

2794  SHR  AX,1

2795  PUSH  AX

2796  MOV  BX,4

2797  CALL  GET_PARM

2798  MOV  CL,AM

2799  MOV  CL,AM

2800  MOV  AX

2801  MOV  AH,0DH

2802  MOV  AX

2803  MOV  SUB

2804  SHR

2805  PUSH

2806  MOV

2807  CALL

2808  MOV

2809  POP

2810  SHR

2811  SHR

2812  MOV

2813  MOV

2814  MOV

2815  MOV

2816  MOV

2817  MOV

2818  MOV

2819  MOV

2820  MOV

2821  MOV

2822  MOV

2823  MOV

2824  MOV

2825  MOV

2826  MOV

2827  MOV

2828  MOV

2829  MOV

2830  MOV

2831  MOV

2832  MOV

2833  MOV

2834  MOV

2835  MOV

2836  MOV

2837  MOV

2838  MOV

2839  MOV

2840  MOV

2841  MOV

2842  MOV

2843  MOV

2844  MOV

2845  MOV

2846  MOV

2847  MOV

2848  MOV

2849  MOV

2850  MOV

2851  MOV

2852  MOV

2853  MOV

2854  MOV

2855  MOV

2856  MOV

2857  MOV

2858  MOV

2859  MOV

2860  MOV

2861  MOV

2862  MOV

2863  MOV

2864  MOV

2865  MOV

2866  MOV

2867  MOV

2868  MOV

2869  MOV

2870  MOV

2871  MOV

2872  MOV

2873  MOV

2874  MOV

2875  MOV

2876  MOV

2877  MOV

2878  MOV

2879  MOV

2880  MOV

2881  MOV

2882  MOV

2883  MOV

2884  MOV

2885  MOV

2886  MOV

2887  MOV

2888  MOV

2889  MOV

2890  MOV

2891  MOV

2892  MOV

2893  MOV

2894  MOV

2895  MOV

2896  MOV

2897  MOV

2898  MOV

2899  MOV

2900  MOV

2901  MOV

2902  MOV

2903  MOV

2904  MOV

2905  MOV

2906  MOV

2907  MOV

2908  MOV

2909  MOV

2910  MOV

2911  MOV

2912  MOV

2913  MOV

2914  MOV

2915  MOV

2916  MOV

2917  MOV

2918  MOV

2919  MOV

2920  MOV

2921  MOV

2922  MOV

2923  MOV

2924  MOV

2925  MOV

2926  MOV

2927  MOV

2928  MOV

2929  MOV

2930  MOV

2931  MOV

2932  MOV

2933  MOV

2934  MOV

2935  MOV

2936  MOV

2937  MOV

2938  MOV

2939  MOV

2940  MOV

2941  MOV

2942  MOV

2943  MOV

2944  MOV

2945  MOV

2946  MOV

2947  MOV

2948  MOV

2949  MOV

2950  MOV

2951  MOV

2952  MOV

2953  MOV

2954  MOV

2955  MOV

2956  MOV

2957  MOV

2958  MOV

2959  MOV

2960  MOV

2961  MOV

2962  MOV

2963  MOV

2964  MOV

2965  MOV

2966  MOV

2967  MOV

2968  MOV

2969  MOV

2970  MOV

2971  MOV

2972  MOV
LOC OBJ  LINE  SOURCE

EF02  E605  2793  OUT  DMA+,AL ; LOW BYTE OF COUNT
EF04  E605  2794  MOV  AL,AM
EF06  E605  2795  OUT  DMA+,AL ; HIGH BYTE OF COUNT
EF08  59  2796  POP  CX ; RECOVER COUNT VALUE
EF09  58  2797  POP  AX ; RECOVER ADDRESS VALUE
EF0A  03C1  2798  ADD  AX,CX ; ADD, TEST FOR 64K OVERFLOW
EF0C  59  2799  POP  CX ; RECOVER REGISTER
EF10  B002  2800  MOV  AL,A ; MODE FOR 8217
EF1A  E60A  2801  OUT  DMA+,AL ; INITIALIZE THE DISKETTE CHANNEL
EF1C  C3  2802  RET 1; RETURN TO CALLER, CFI SET BY ABOVE IF ERROR

2803  DMA_SETUP  ENDP

2804  ; -----------------------------
2805  ; CHK_STAT_2
2806  ; THIS ROUTINE HANDLES THE INTERRUPT RECEIVED AFTER
2807  ; A RECALIBRATE, SEEK, OR RESET TO THE ADAPTER.
2808  ; THE INTERRUPT IS WAITED FOR, THE INTERRUPT STATUS SENSED,
2809  ; AND THE RESULT RETURNED TO THE CALLER.
2810  ; INPUT
2811  1; NONE
2812  1; OUTPUT
2813  1; CY = 0 SUCCESS
2814  1; CY = 1 FAILURE -- ERROR IS IN DISKETTE_STATUS
2815  1; (AX) DESTROYED

2817  CHK_STAT_2  PROC NEAR
2818  1; CALL WAIT_INT ; WAIT FOR THE INTERRUPT
2819  JC  J34 ; IF ERROR, RETURN IT
2820  MOV  AH,00H ; SENSE INTERRUPT STATUS COMMAND
2821  CALL  NEC_OUTPUT
2822  CALL  RESULTS ; READ IN THE RESULTS
2823  JC  J34 ; CHK2_RETURN
2824  MOV  Al,4EC_STATUS ; GET THE FIRST STATUS BYTE
2825  ANO  AL,060H ; ISOLATE THE SITS
2826  CMP  AL,060H ; TEST FOR CORRECT VALUE
2827  JC  J3S ; IF ERROR, GO MARK IT
2828  CLC ; GOOD RETURN
2829  J34: 1; RETURN TO CALLER
2830  RET
2831  J35: ; CHK2_ERROR
2832  OR  o,DISKETTE_STATUS BAD SEEK
2833  STC ; ERROR RETURN CODE

2836  CHK_STAT_2  ENDP

2838  ; -----------------------------
2839  ; WAIT_INT
2840  ; THIS ROUTINE WAITS FOR AN INTERRUPT TO OCCUR
2841  ; A TIME OUT ROUTINE TAKES PLACE OUTSIDE THE WAIT, SO
2842  ; THAT AN ERROR MAY BE RETURNED IF THE DRIVE IS NOT READY
2843  ; INPUT
2844  1; NONE
2845  1; OUTPUT
2846  1; CY = 0 SUCCESS
2847  1; CY = 1 FAILURE -- DISKETTE_STATUS IS SET ACCORDINGLY
2848  1; (AX) DESTROYED

2849  WAIT_INT  PROC NEAR
284A  1; STI ; TURN ON INTERRUPTS, JUST IN CASE
284B  1; PUSH BX
284C  1; PUSH CX ; SAVE REGISTERS
284D  1; MOV  BL,2 ; CLEAR THE COUNTERS
284E  1; XOR  CX,CX ; FOR 2 SECOND WAIT

284F  J36: 1; TEST SEEK_STATUS,INT_FLAG ; TEST FOR INTERRUPT OCCURRING
2850  1; JNZ  J37 ; COUNT DOWN WHILE WAITING
2851  1; DEC  BL ; SECOND LEVEL COUNTER
2852  1; (AX) DESTROYED

2853  J37: 1; ERROR RETURN
2854  1; (AX) DESTROYED

285A  1; PUSHF ; SAVE CURRENT CARRY
285B  1; AND  SEEK_STATUS,NOT_INT_FLAG ; TURN OFF INTERRUPT FLAG
285C  1; POPF ; RECOVER CARRY
285D  1; POP  CX ; RECOVER REGISTERS
285E  1; POP  RX
285F  RET 1; GOOD RETURN CODE COMES FROM TEST INST

2869  WAIT_INT  ENDP

A-39
THIS ROUTINE HANDLES THE DISKETTE INTERRUPT

THE INTERRUPT FLAG IS SET IF SEEK_STATUS

RESULTS

RESULTS_ERROR

RESULTS_ERROR

RESULTS_ERROR

RESULTS_ERROR

RESULTS_ERROR

RESULTS_ERROR

RESULTS_ERROR

RESULTS_ERROR

RESULTS_ERROR

RESULTS_ERROR

RESULTS_ERROR

RESULTS_ERROR

RESULTS_ERROR

RESULTS_ERROR

RESULTS_ERROR

RESULTS_ERROR

RESULTS_ERROR

RESULTS_ERROR

RESULTS_ERROR

RESULTS_ERROR

RESULTS_ERROR

RESULTS_ERROR
LOC OBJ
LINE
SOURCE
2946
J43:
LOOP J43
2947
DEC DX
; POINT AT STATUS PORT
2948
IN AL,DX
; GET STATUS
2949
TEST AL,010H
; TEST FOR NEC STILL BUSY
2950
JZ J46
; RESULTS DONE
2951
DEC BL
; DECREMENT THE STATUS COUNTER
2952
JNZ J38
; GO BACK FOR MORE
2953
JMP J41
; CHIP HAS FAILED
2954

2955
; ------ RESULT OPERATION IS DONE
2956

2957
J44:
2958
POP BX
2959
POP DX
2960
POP CX
; RECOVER REGISTERS
2961
RET
; GOOD RETURN CODE FROM TEST INST
2962

2963
; NUM_TRANS PROC NEAR
2964

2965
; NUM_TRANS PROC NEAR
2966

2967
; THIS ROUTINE CALCULATES THE NUMBER OF SECTORS THAT
2968
; WERE ACTUALLY TRANSFERRED TO/FROM THE DISKETTE
2969
; INPUT
2970
; ; (CH) = CYLINDER OF OPERATION
2971
; ; (CL) = START SECTOR OF OPERATION
2972
; ; HD OTHER REGISTERS MODIFIED
2973

2974
MOV A.L,NEC_STATUS+3
; GET CYLINDER ENDED UP ON
2975
CMP AL,CH
; SAME AS WE STARTED
2976
MOV AL,NEC_STATUS+5
; GET ENDING SECTOR
2977
JZ J45
; IF ON SAME CYL, THEN NO ADJUST
2978
MOV DX,0
2979
CALL GET_PARM
; GET EOT VALUE
2980
MOV AL,AH
; INTO AL
2981
INC AL
; USE EOT+1 FOR CALCULATION
2982
J45:
SUB AL,CL
; SUBTRACT START FROM END
2983

2984

2985
RESULTS ENDP
2986

2987
; -----------------------------
2988

2989
; THIS IS THE SET OF PARAMETERS REQUIRED FOR
2990
; DISKETTE OPERATION. THEY ARE POINTED AT BY THE
2991
; DATA VARIABLE DISK_POINTER. TO MODIFY THE PARAMETERS,
2992
; BUILD ANOTHER PARAMETER BLOCK AND POINT AT IT
2993
; -----------------------------
2994

2995

2996
DB 1101111B
; SRT=C. HD UNLOAD=OF - 1ST SPECIFY BYTE
2997
DB 2
; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
2998
DB MOTOR_WAIT
; WAIT AFTER OPM TIL MOTOR OFF
2999
DB 2
; 512 BYTES/SECTOR
3000
DB 0
; EOT (LAST SECTOR ON TRACK)
3001
DB 02AH
; GAP LENGTH
3002
DB 0FH
; DTL
3003
DB 050H
; GAP LENGTH FOR FORMAT
3004
DB 06H
; FILL BYTE FOR FORMAT
3005
DB 25
; HEAD SETTLE TIME (MILLISECONDS)
3006
DB 4
; MOTOR START TIME (1/8 SECONDS)
This routine provides communication with the printer.

On return, AH=0 if character could not be printed (time out).

Other bits set as on normal status call.

Returns with AH set with printer status.

Input status into AH.

Time out.

Selected.

Out of paper.

Register 0x = printer to be used (0,1,2) corresponding to actual values.

Register 0x = printer base area.

Data area printer_base contains the base address of the printer card(s).

Available (located at beginning of data segment, 409h absolute, 3 words).

All others unchanged.

Assume CS:CODE, DS:DATA.

Print the character in AL.

Print the character in AL.

镓
LOC OBJ | LINE | SOURCE
--- | --- | ---
F01A 6E | 3083 | OUT DX, AL; SET THE STROBE LOW
F01B 00C | 3084 | MOV AL, 0CH
F01D EE | 3085 | OUT DX, AL
F01E 58 | 3086 | POP AX; RECOVER THE OUTPUT CHAR
F01F | 3087 | 1------- PRINTER STATUS
F020 | 3088 | B5:
F021 00000000 R | 3089 | MOV DX, PRINTER_BASE[311]
F024 42 | 3090 | INC DX
F025 EC | 3091 | IN AL, DX; GET PRINTER STATUS
F026 8A 00 | 3092 | MOV AH, AL
F028 0E 0F8 | 3093 | AND AH, 0FH; TURN OFF UNUSED BITS
F02B 57 | 3094 | B7: ; STATUS_SET
F02C 8A | 3095 | POP DX; RECOVER AL REG
F02D 8A 00 | 3096 | MOV AL, 00H; GET CHARACTER INTO AL
F02E B2 | 3097 | XOR AH, 48H; FLIP A COUPLE OF BITS
F030 | 3098 | JMP B1; RETURN FROM ROUTINE
F031 EBC2 | 3099 | 1------- INITIALIZE THE PRINTER PORT
F033 | 3100 | B8:
F034 53C202 | 3101 | MOV AX, 0000H
F035 0000 | 3102 | MOV AL, 00H; SET INIT CHAR.
F036 EE | 3103 | OUT DX, AL
F037 8E 003 | 3104 | MOV AX, 000H
F037 | 3105 | 1------- INT 10 ------------------------------------------
F039 | 3106 | Push AX; SAVE AL REG
F03A 8E 0003 | 3107 | MOV AX, 0000H
F03B | 3108 | PUSH AX; SAVE AL REG
F03C | 3109 | ADD DX, 2; POINT TO OUTPUT PORT
F03D 8000 | 3110 | MOV AL, 00H; SET INIT LINE LOW
F03E EE | 3111 | OUT DX, AL
F03F 8E 00 | 3112 | MOV AX, 00H
F040 89 | 3113 | DEC AX; LOOP FOR RESET TO TAKE
F041 89 | 3114 | JNZ .9; INIT_LOOP
F042 8A | 3115 | MOV AL, 00H; NO INTERRUPTS, NON AUTO LF, INIT HIGH
F043 82 | 3116 | OUT DX, AL
F044 | 3117 | JMP B6; PRT_STATUS_1
F045 | 3118 | 1------- INT 10 ------------------------------------------
F047 | 3119 | Push AX; SAVE AL REG
F048 | 3120 | Push AX; SAVE AL REG
F049 | 3121 | These routines provide the CRT interface
F04A | 3122 | The following functions are provided:
F04B | 3123 | (AH)=0 SET MODE (AL) CONTAINS MODE VALUE
F04C | 3124 | (AH)=0 0000H BW (POWER ON DEFAULT)
F04D | 3125 | (AL)=1 0001H BLACK COLOR
F04E | 3126 | (AL)=2 0010H B/W
F04F | 3127 | (AL)=3 0011H BW
F050 | 3128 | Graphics modes
F051 | 3129 | (AL|= 32000H COLOR
F052 | 3130 | (AL|= 32001H BW
F053 | 3131 | (AL|= 64000H BW
F054 | 3132 | CRT MODE = 7 60X25 BW CARD (USED INTERNAL TO VIDEO ONLY)
F055 | 3133 | *** NOTE BW MODES OPERATE SAME AS COLOR MODES, BUT COLOR
F056 | 3134 | BURST IS NOT ENABLED
F057 | 3135 | (AH)=1 SET CURSOR TYPE
F058 | 3136 | (CH) = BITS 4-0 = START LINE FOR CURSOR
F059 | 3137 | ** Hardware will always cause blinking
F05A | 3138 | ** Setting bit 5 or 6 will cause erratic blinking
F05B | 3139 | OR NO CURSOR AT ALL
F05C | 3140 | (CL) = BITS 4-0 = END LINE FOR CURSOR
F05D | 3141 | (AH)=2 SET CURSOR POSITION
F05E | 3142 | (DH,DL) = ROW,COLUMN (0,0) IS UPPER LEFT
F05F | 3143 | (DH) = PAGE 16/32SP (MUST BE 0 FOR GRAPHICS MODES)
F060 | 3144 | (AH)=3 READ CURSOR POSITION
F061 | 3145 | (BH) = PAGE NUMBER (MUST BE 0 FOR GRAPHICS MODES)
F062 | 3146 | ON EXIT (DH,DL) = ROW,COLUMN OF CURRENT CURSOR
F063 | 3147 | (CH,CL) = CURSOR MODE CURRENTLY SET
F064 | 3148 | (AH)=4 READ LIGHT PEN POSITION
F065 | 3149 | ON EXIT:
F066 | 3150 | (AH) = 0 -- LIGHT PEN SWITCH NOT DOWN/HOT TRIGGERED
F067 | 3151 | (AH) = 1 -- VALID LIGHT PEN VALUE IN REGISTERS
F068 | 3152 | (DH,DL) = ROW,COLUMN OF CHARACTER (P POSN
F069 | 3153 | (CH) = RASTER LINE (0-199)
F06A | 3154 | (BX) = PIXEL COLUMN (0-319, 0-A)
F06B | 3155 | (AH)=5 SELECT ACTIVE DISPLAY PAGE (VALID ONLY FOR ALPHA MODES)
F06C | 3156 | (AL)=NEW PAGE VALUE (0-7 FOR MODES 0&1, 0-3 FOR MODES 2&3)
3157 | (AH)=6 SCROLL ACTIVE PAGE UP
3158 | (AL) = NUMBER OF LINES, INPUT LINES BLANKED AT BOTTOM OF WINDOW
3159 | AL = 0 MEANS BLANK ENTIRE WINDOW
3160 | (CH,CL) = ROW, COLUMN OF UPPER LEFT CORNER OF SCROLL
3161 | (DH,DL) = ROW, COLUMN OF LOWER RIGHT CORNER OF SCROLL
3162 | (BH) = ATTRIBUTE TO BE USED ON BLANK LINE
3163 | (AH)=7 SCROLL ACTIVE PAGE DOWN
3164 | (AL) = NUMBER OF LINES, INPUT LINES BLANKED AT TOP OF WINDOW
3165 | AL = 0 MEANS BLANK ENTIRE WINDOW
3166 | (CH,CL) = ROW, COLUMN OF UPPER LEFT CORNER OF SCROLL
3167 | (DH,DL) = ROW, COLUMN OF LOWER RIGHT CORNER OF SCROLL
3168 | (BH) = ATTRIBUTE TO BE USED ON BLANK LINE
3169 |
3170 |
3171 |
3172 | (AH) = 8 READ ATTRIBUTE/CHARACTER AT CURRENT CURSOR POSITION
3173 | (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY)
3174 | ON EXIT:
3175 | (AL) = CHAR READ
3176 | (AH) = ATTRIBUTE OF CHARACTER READ (ALPHA MODES ONLY)
3177 | (AH) = 9 WRITE ATTRIBUTE/CHARACTER AT CURRENT CURSOR POSITION
3178 | (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY)
3179 | (CX) = COUNT OF CHARACTERS TO WRITE
3180 | (AL) = CHAR TO WRITE
3181 | (BL) = ATTRIBUTE OF CHARACTER (ALPHA)/COLOR OF CHAR (GRAPHICS)
3182 | SEE NOTE ON WRITE DOT FOR BIT 7 OF BL = 1.
3183 | (AH) = 10 WRITE CHARACTER ONLY AT CURRENT CURSOR POSITION
3184 | (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY)
3185 | (CX) = COUNT OF CHARACTERS TO WRITE
3186 | (AL) = CHAR TO WRITE
3187 | FOR READ/WRITE CHARACTER INTERFACE WHILE IN GRAPHICS MODE, THE
3188 | CHARACTERS ARE FORMED FROM A CHARACTER GENERATOR IMAGE
3189 | MAINTAINED IN THE SYSTEM ROM. ONLY THE 1ST 128 CHAR
3190 | ARE CONTAINED THERE. TO READ/WRITE THE SECOND 128 CHARs,
3191 | THE USER MUST INITIALIZE THE POINTER AT INTERRUPT 1FH
3192 | (LOCATION 007CH TO POINT TO THE 1K BYTE TABLE CONTAINING
3193 | THE CODE POINTS FOR THE SECOND 128 CHARs (128-255)
3194 | FOR WRITE CHARACTER INTERFACE IN GRAPHICS MODE, THE REPLICATION FACTOR
3195 | CONTAINED IN (CX) ON ENTRY WILL PRODUCE VALID RESULTS ONLY
3196 | FOR CHARACTERS CONTAINED ON THE SAME ROW. CONTINUATION TO
3197 | SUCCCEEDING LINES WILL NOT PRODUCE CORRECTLY.
3198 |
3199 |
3200 | (AH) = 11 SET COLOR PALETTE
3201 | (BH) = PALETTE COLOR ID BEING SET (0-127)
3202 | (BL) = COLOR VALUE TO BE USED WITH THAT COLOR ID
3203 | NOTE: FOR THE CURRENT COLOR CARD, THIS ENTRY POINT HAS
3204 | MEANING ONLY FOR 320X200 GRAPHICS.
3205 | COLOR ID = 0 SELECTS THE BACKGROUND COLOR (0-15)
3206 | COLOR ID = 1 SELECTS THE PALETTE TO BE USED:
3207 | 0 = GREEN(1)/RED(2)/YELLOW(3)
3208 | 1 = CYAN(1)/MAGENTA(2)/WHITE(3)
3209 | IN 40X25 OR 80X25 ALPHA Modes, THE VALUE SET FOR
3210 | PALLETTE COLOR 0 INDICATES THE BORDER COLOR
3211 | TO BE USED (VALUES 9-31, WHERE 16-31 SELECT THE
3212 | HIGH INTENSITY BACKGROUND SET.
3213 | (AH) = 12 WRITE DOT
3214 | (DX) = ROW NUMBER
3215 | (CX) = COLUMN NUMBER
3216 | (AL) = COLOR VALUE
3217 | IF BIT 7 OF AL = 1, THEN THE COLOR VALUE IS EXCLUSIVE
3218 | OR'ED WITH THE CURRENT CONTENTS OF THE DOT
3219 | (AH) = 13 READ DOT
3220 | (DX) = ROW NUMBER
3221 | (CX) = COLUMN NUMBER
3222 | (AL) RETURNS THE DOT READ
3223 |
3224 | ASCII TELETYPING ROUTINE FOR OUTPUT
3225 |
3226 | (AH) = 14 WRITE TELETYPING
3227 | (AL) = CHAR TO WRITE
3228 | (BL) = FOREGROUND COLOR IN GRAPHICS MODE
3229 | (BH) = DISPLAY PAGE IN ALPHA MODE
3230 | NOTE -- SCREEN WIDTH IS CONTROLLED BY PREVIOUS MODE SET
3231 |
; current video state
; returns the current video state
; (al) = mode currently set (see ah for explanation)
; (ah) = number of character columns on screen
; (bh) = current active display page

; cs, ss, ds, es, bx, cx, dx preserved during call
; all others destroyed

; ---------------
; assume cs: code, ds: data, es: video ram

; table of routines within video I/O

; set mode
; this routine initializes the attachment to the selected mode. the screen is blanked.
; input
; output
; none

; tables for use in setting of mode

; initial table

; video params label 'byte

; ******************
; set mode
; this routine initializes the attachment to the selected mode. the screen is blanked.
; input
; output
; none

; tables for use in setting of mode

; video params label 'byte

; initial table
115x549] INC ox ; POINT TO DATA PORT
115x542] INC AH ; NEXT REGISTER VALUE
115x535] MOV AL,[BX] ; GET TABLE VALUE
115x528] MOV DX,AL ; OUT TO CHIP
115x521] INC BX ; NEXT IN TABLE
115x514] MOV DX,AL ; BACK TO POINTER REGISTER
115x507] POP AX ; GET MODE BACK
115x500] POP DS ; RECOVER SEGMENT VALUE
115x493] ASSUME OS:DATA
115x487] ;------ FILL REGEN AREA WITH BLANK
115x480] ;------ ENABLE VIDEO AND CORRECT PORT SETTING
115x473] ;------ DETERMINE NUMBER OF COLUMNS, BOTH FOR ENTIRE DISPLAY
115x466] ;------ SET CURSOR POSITIONS
115x460] ;------ SET UP OVERSCAN REGISTER
115x453] ;------ NORMAL RETURN FROM ALL VIDEO RETURNS

SET_MODE ENDP

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

; SET_CTYPE

; THIS ROUTINE SETS THE CURSOR VALUE

; (CX) HAS CURSOR VALUE CH=START LINE, CL=STOP LINE

; OUTPUT

; NONE

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

; SET_CTYPE PROC NEAR

; MOV AH,10 ; 6845 REGISTER FOR CURSOR SET

; MOV CURSOR_MODE,CX ; SAVE IN DATA AREA

; CALL Min ; OUTPUT CX REG

; JMP VIDEO_RETURN

; 3482

; 3483

; 3484

; 3485

; 3486

; 3487

; 3488

; 3489

; 3490

; 3491

; 3492

; 3493

; 3494

; 3495

; 3496

; 3497

; 3498

; 3499

; RET ; ALL DONE

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

; SET_CTYPE ENDP

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

; SET_CPOS

; THIS ROUTINE SETS THE CURRENT CURSOR POSITION TO THE

; NEW X-Y VALUES PASSED

; INPUT

; DX = ROW-COLUMN OF NEW CURSOR

; BH = DISPLAY PAGE OF CURSOR

; OUTPUT

; CURSOR IS SET AT 6845 IF DISPLAY PAGE IS CURRENT DISPLAY

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

; SET_CPOS PROC NEAR

; MOV CL,BH

; MOV AX,OX ; GET ROW/COLUMN TO AX

; JMP CURSOR_SET

; SET_CPOS_RETURN

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

; XORM CH,CH ; ESTABLISH LOOP COUNT

; SAL CX,1 ; WORD OFFSET

; MOV SI,CX ; USE INDEX REGISTER

; MOV [SI+OFFSET CURSOR_POSNH].DX ; SAVE THE POINTER

; CMP ACTIVE_PAGE,BH

; JMP VIDEO_RETURN

; CURSOR_SET

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

; SET_CPOS ENDP

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

; SET_CURSOR POSITION, AX HAS ROW/COLUMN FOR CURSOR

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

; CALL POSITION ; DETERMINE LOCATION IN REGEN BUFFER

; MOV CX,AX

; ADD CX,CRT_START ; ADD IN THE START ADDRESS FOR THIS PAGE

; SAR CX,1 ; DIVIDE BY 2 FOR CHAR ONLY COUNT

; MOV AH,14 ; REGISTER NUMBER FOR CURSOR

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

; -------------------------- --------------------------
LaC OBJ
LINE SOURCE

F216 E8C1FF 3533 CALL M16 ; OUTPUT THE VALUE TO THE 6845
F219 C3 3534 RET
3535 M16 ENDP
3536 |---------------------------------------------|
3537 | READ_CURSOR
3538 | THIS ROUTINE READS THE CURRENT CURSOR VALUE FROM THE
3539 | 6845, FORMATS IT, AND SENDS IT BACK TO THE CALLER
3540 | INPUT
3541 | BH - PAGE OF CURSOR
3542 | OUTPUT
3543 | BX - ROW, COLUMN OF THE CURRENT CURSOR POSITION
3544 | CX - CURRENT CURSOR MODE
3545 |---------------------------------------------|
3546 F21A READ_CURSOR PROC NEAR
3547 F21A A0D' 3548 MOV BL,BH
3549 F21C 30F 3550 XOR BH,BH ; WORD OFFSET
3551 F220 08975000 R 3552 MOV DX,(BX+OFFSET CURSOR_POSH]
3553 F220 08066000 R 3554 MOV CX,CURSOR_MODE
3555 F22D 5F 3556 POP DI
3557 F229 9F 3558 POP SI
3559 F22A 5B 355A POP BX
355B F22B 5B 355C POP AX ; DISCARD SAVED CX AND DX
355D F22D 1F 355E POP DS
355F F22E 07 3560 POP ES
3561 F22F CF 3562 POP CX ; DISCARD SAVED CX AND
3563 F22:0 IF 3564 POP BX
3565 F22:8 58 3565 POP AX ; DISCARD SAVED CX AND
3566 F22:0 IF 3567 POP BX
3568 |---------------------------------------------|
3569 F230 ACT_DISP_PAGE PROC NEAR
3570 F230 A2E200 3571 MOV ACTlVE_PAGE,AL ; SAVE ACTIVE PAGE VALUE
3572 F233 A004C000 R 3573 MOV CX,CRT_LEN ; GET SAVED LENGTH OF REGEN BUFFER
3574 F237 9B 3575 MOV CX ; CONVERT AL TO WORD
3576 F233 5D 3577 PUSH AX ; SAVE PAUL VALUE
3578 F239 F7E1 3579 MUL CX ; DISPLAY PAGE TIMES REGEN LENGTH
3580 F23B A3400 3581 MOV AX,CRT_START,AX ; SAVE START ADDRESS FOR LATER REQUIRMENTS
3582 F23E 0BC6 3583 MOV CX,AX ; START ADDRESS TO CX
3584 F240 D1F9 3585 SAR CX,1 ; DIVIDE BY 2 FOR 6845 HANDLING
3586 F242 B40C 3587 MOV AX,12 ; 6845 REGISTER FOR START ADDRESS
3588 F244 E93F3F 3589 CALL M16
3590 F247 5B 3591 POP BX ; RECOVER PAGE VALUE
3592 F246 D13 3593 POP BX ; RECOVER PAGE VALUE
3594 F24A 08750000 R 3595 MOV AX,(BX+OFFSET CURSOR_POSH]; GET CURSOR FOR THIS PAGE
3596 F24E E80F0F 3597 CALL M16 ; SET THE CURSOR POSITION
3598 F251 :E973FF 3599 JMP VIDEO_RETURN
3600 |---------------------------------------------|
3601 | GET_COLOR PROC NEAR
3602 |---------------------------------------------|
3603 | F254 0816300 3604 MOV DX,ADDR_6845 ; I/O PORT FOR PALETTE
3605 F250 03205 3606 ADD DX,5 ; OVERSCAN PORT
3607 F25B A6600 R 3608 MOV AX,CRT_PALLETE ; GET THE CURRENT PALETTE VALUE
3609 F25E 0AFF 3610 OR AX,BH ; IS THIS COLOR 0?
LOC OBJ       LINE SOURCE
F260 75OE  3608  JNZ   M20  ; OUTPUT COLOR 1
F260 79EO  3609
F260 80E1F  3610  ;------- HANDLE COLOR 0 BY SETTING THE BACKGROUND COLOR
F267 8AC3  3611
F269 8E   3612  AND   ALO,DEHM  ; TURN OFF LOW 5 BITS OF CURRENT
F269 IE    3613  OR    BL,1FH  ; TURN OFF HIGH 3 BITS OF INPUT VALUE
F26A A5600  3614  OR    AL,1B  ; PUT VALUE INTO REGISTER
3615  M19:  ; OUTPUT THE PALLETTE
F26A A5660  3616  OUT   DX,AL  ; put color selection to 3DF port
F26A A5660  3617  MOV   CRT_PALLETTE,AL  ; SAVE THE COLOR VALUE
F269 E93FF  3618  JMP   VIDEO_RETURN
F269 E93F  3619
F269 E93F  3620  ;------- HANDLE COLOR 1 BY SELECTING THE PALLETTE TO BE USED
F270 75OE  3621
F270 24OE  3622  MOV   M20,AL  ; OUTPUT COLOR 1
F270 2D8E  3623  AND   ALO,0DFH  ; TURN OFF PALLETTE SELECT BIT
F270 73F3  3624  SHR   BL,1  ; TEST THE LOW ORDER BIT OF BL
F270 0C20  3625  JNC   M19  ; ALREADY DONE
F270 E93F  3626  OR    AL,20H  ; TURN ON PALLETTE SELECT BIT
F270 E93F  3627  JMP   M19  ; GO DO IT
F270 E93F  3628  SET_COLOR ENDP
F27A 8A2664A0  3629
F27A 8A2664A0  3630  ; VIDEO STATE
F27A 8A2664A0  3631  ; RETURNS THE CURRENT VIDEO STATE IN AX
F27A 8A2664A0  3632  ; AH = NUMBER OF COLUMNS ON THE SCREEN
F27A 8A2664A0  3633  ; AL = CURRENT VIDEO MODE
F27A 8A2664A0  3634  ; BH = CURRENT ACTIVE PAGE
F27A 8A2664A0  3635  ;----------------------------------------
F27A 8A2664A0  3636  VIDEO_STATE PROC NEAR
F27A 8A2664A0  3637  MOV   AH,BYTE PTR CRT_COLS  ; GET NUMBER OF COLUMNS
F27A 8A2664A0  3638  MOV   AL,CRT_MODE  ; CURRENT MODE
F27A 8A2664A0  3639  MOV   BH,ACTIVE_PAGE  ; GET CURRENT ACTIVE PAGE
F27A 8A2664A0  3640  POP   DI  ; RECOVER REGISTERS
F27A 8A2664A0  3641  POP   SI  ;
F27A 8A2664A0  3642  POP   CX  ; DISCARD SAVED BX
F27A 8A2664A0  3643  JMP   M15  ; RETURN TO CALLER
F27A 8A2664A0  3644  VIDEO_STATE ENDP
F27A 8A2664A0  3645  ;----------------------------------------
F27A 8A2664A0  3646  ; POSITION
F27A 8A2664A0  3647  ; THIS SERVICE ROUTINE CALCULATES THE REGEN BUFFER ADDRESS
F27A 8A2664A0  3648  ; OF A CHARACTER IN THE ALPHA MODE
F27A 8A2664A0  3649  ; INPUT
F27A 8A2664A0  3650  ; AX = ROW/COLUMN POSITION
F27A 8A2664A0  3651  ; OUTPUT
F27A 8A2664A0  3652  ; AX = OFFSET OF CHAR POSITION IN REGEN BUFFER
F27A 8A2664A0  3653  ;----------------------------------------
F27B 8A2664A0  3654  POSITION PROC NEAR
F27B 8A2664A0  3655  PUSH  BX  ; SAVE REGISTER
F27B 8A2664A0  3656  MOV   BX,AH  ; AX TO AL
F27B 8A2664A0  3657  MOV   AL,AH  ; AX TO AL
F27B 8A2664A0  3658  MUL   BYTE PTR CRT_COLS  ; DETERMINE BYTES TO ROW
F27B 8A2664A0  3659  XOR   BH,BH  ; ADD IN COLUMN VALUE
F27B 8A2664A0  3660  ADD   AX,BX  ; AX = ROW/COLUMN OF UPPER LEFT CORNER
F27B 8A2664A0  3661  SHR   AX,1  ; = 2 FOR ATTRIBUTE BYTES
F27B 8A2664A0  3662  POP   BX  ; DISCARD SAVED BX
F27B 8A2664A0  3663  RET
F27B 8A2664A0  3664  POSITION ENDP
F27B 8A2664A0  3665  ;----------------------------------------
F27B 8A2664A0  3666  ; SCROLL UP
F27B 8A2664A0  3667  ; THIS ROUTINE MOVES A BLOCK OF CHARACTERS UP
F27B 8A2664A0  3668  ; ON THE SCREEN
F27B 8A2664A0  3669  ; INPUT
F27B 8A2664A0  3670  ; (BH) = CURRENT CRT MODE
F27B 8A2664A0  3671  ; (AL) = NUMBER OF ROWS TO SCROLL
F27B 8A2664A0  3672  ; (CX) = ROW/COLUMN OF UPPER LEFT CORNER
F27B 8A2664A0  3673  ; (DX) = ROW/COLUMN OF LOWER RIGHT CORNER
F27B 8A2664A0  3674  ; (BH) = ATTRIBUTE TO BE USED ON BLANKED LINE
F27B 8A2664A0  3675  ; (DS) = DATA SEGMENT
F27B 8A2664A0  3676  ; (ES) = REGEN BUFFER SEGMENT
F27B 8A2664A0  3677  ; OUTPUT
F27B 8A2664A0  3678  ; NONE -- THE REGEN BUFFER IS MODIFIED
F27B 8A2664A0  3679  ;----------------------------------------
F27C 8A2664A0  3680  ASSUME CS:CODE,DS:DATA,ES:DATA
F27C 8A2664A0  3681  SCROLL_UP PROC NEAR
F27C 8A2664A0  3682  MOV   BL,AL  ; SAVE LINE COUNT IN BL
F27C 8A2664A0  3683  CPI   AH,4  ; TEST FOR GRAPHICS MODE
LOC OBJ  
LINE  
SOURCE

F2A1 7208  
3684  
JC N1  
; HANDLE SEPARATELY

F2A3 00FC07  
3685  
CMP AH,7  
; TEST FOR BW CARD

F2A6 7403  
3686  
JE N1  

F2A9 E9F301  
3687  
JMP GRAPHICS_UP

F2AB 53  
3688  
N1:  
; UP_CONTINUE

F2AC 80FC07  
3689  
PUSH BX  
; SAVE FILL ATTRIBUTE IN BM

F2AE E83900  
3690  
MDV AX,CX  
; UPPER LEFT POSITION

F2B1 74AE  
3692  
JZ N7  
; BLANK_FIELD

F2BF 403  
3693  
ADD SI,AX  
; FROM ADDRESS

F2BC 80FC07  
3694  
MDV AH,DH  
; # ROWS IN BLOCK

F2BE E9F301  
3695  
SUB AH,BL  
; # ROWS TO BE MOVED

F2F9  
3696  
N2:  
; ROW_LOOP

F2F9 E87500  
3697  
CALL N10  
; MOVE ONE ROW

F2FC 03F5  
3698  
ADD SI,BP

F2FE E83900  
3699  
ADD DI,BP  
; POINT TO NEXT LINE IN BLOCK

F300 75F5  
3700  
DEC AH  
; COUNT OF LINES TO MOVE

F302 75F5  
3701  
JNZ N2  
; ROW_LOOP

F304 56  
3702  
PDP AX  
; RECOVER ATTRIBUTE IN AH

F306 50  
3703  
POP AX  
; CLEAR_ENTRY

F306 B202  
3704  
MDV AL,' '  
; FILL WITH BLANKS

F308 56  
3705  
CLEAN_LOOP

F308 E87000  
3706  
CALL N11  
; CLEAR THE ROW

F30A 03F0  
3707  
ADD DI,BP  
; POINT TO NEXT LINE

F30C FECB  
3708  
REC BL  
; COUNTER OF LINES TO SCROLL

F310 75F7  
3709  
JNZ N4  
; CLEAR_LOOP

F312  
3710  
N5:  
; SCROLL_END

F312 B84000  
3711  
MDV AX,DATA  
; GET LOCATION

F314 0E06  
3712  
MDV DS,AX  

F316 00FC077  
3713  
CMP CRT_MODE,7  
; IS THIS THE BLACK AND WHITE CARD

F318 7407  
3714  
JE N6  
; IF SO, SKIP THE MODE RESET

F31A A04500  
3715  
MDV AL,CRT_MODE_SET  
; GET THE VALUE OF THE MODE SET

F31C BAD045  
3716  
MDV DX,00DH  
; ALWAYS SET COLOR CARD PORT

F31E EE  
3717  
OUT DX,AL  

F320 E91FE  
3718  
JMP VIDEO_RETURN

F322 040000  
3719  
N7:  
; BLANK_FIELD

F322 0A06  
3720  
MDV BL,DH  
; GET ROW COUNT

F324 E0DA  
3721  
JMP N3  
; GO CLEAR THAT AREA

F326 7712  
3722  
CLEAR_LOOP  
; CLEAR THE ROW

F328 7712  
3723  
SCROLL_UP  
; END

F32A  
3724  
HANDLE COMMON SCROLL SET UP HERE

F32C  
3725  
SCROLL_POSITION_PROC  
; NEAR

F32E 00FC0002  
3726  
CMP CRT_MODE,2  
; TEST FOR SPECIAL CASE HERE

F330 7219  
3727  
JB N9  
; HAVE TO HANDLE BOXES SEPARATELY

F332 00FC0003  
3728  
CMP CRT_MODE,3  

F334 7712  
3729  
N9  

F336 7712  
3730  
Boxx25_COLOR_CARD_SCROLL

F338  
3731  

F33A 52  
3732  
ADD AX,30AH  
; GUARANTEED TO BE COLOR CARD HERE

F33C 50  
3733  
PUSH AX  

F33E  
3734  

F340 B4A03  
3735  
MDV DX,30DAY  
; ADDRESS OF ACTIVE PAGE

F342 50  
3736  
PUSH AX  

F344  
3737  

F346 EC  
3738  
N8:  
; WAIT_DISP_ENABLE

F348 A096  
3739  
IN AL,DX  
; GET PORT

F34A 74FB  
3740  
TEST AL,6  
; WAIT FOR VERTICAL RETRACE

F34C 0005  
3741  
JE N8  
; WAIT_DISP_ENABLE

F34E 04AD  
3742  
MDV DX,30BH  

F350 E005  
3743  
MOV AL,13H  

F352 04AD  
3744  
MOV DX,00BH  

F354 EE  
3745  
OUT DX,AL  
; TURN OFF VIDEO

F356 50  
3746  
POP AX  
; DURING VERTICAL RETRACE

F358 5A  
3747  
PUSH DX  

F35A 0807FF  
3748  
CALL POSITION  
; CONVERT TO REGEN POINTER

F35C 03044000  
3749  
ADD AX,CRT_START  
; OFFSET OF ACTIVE PAGE

F35E 00FF  
3750  
MDV DX,AX  
; TO ADDRESS FOR SCROLL

F360 00F0  
3751  
MOV SI,AX  
; FROM ADDRESS FOR SCROLL

F362 2801  
3752  
SUB DX,CX  
; DX = #ROWS, #COLS IN BLOCK

F364 FC6  
3753  
INC DH  

F366  
3754  
INC DL  
; INCREMENT FOR 0 ORIGIN

F368 321D  
3755  
XOR CH,CH  
; SET HIGH BYTE OF COUNT TO ZERO

F36A 0024A00  
3756  
MOV BP,CRT_COLS  
; GET NUMBER OF COLUMNS IN DISPLAY

F36C 03ED  
3757  
ADD BP,BP  
; TIMES 2 FOR ATTRIBUTE BYTE

F36E 043C  
3758  
MOV AL,BL  
; GET LINE COUNT

F370 F6264000  
3759  
MUL BYTE PTR CRT_COLS  
; DETERMINE OFFSET TO FROM ADDRESS

F372 03C9  
3760  
ADD AX,AX  
; #2 FOR ATTRIBUTE BYTE
PUSH ES  ; ESTABLISH ADDRESSING TO REGEN BUFFER
POP DS  ; FOR BOTH POINTERS
CMP BL,0 ; O SCROLL MEANS BLANK FIELD
RET  ; RETURN WITH FLAGS SET

SCROLL_POSITION ENDP

;------ MOVE_ROW
PROC NEAR
MOV CL,DL  ; GET # OF COLS TO MOVE
PUSH SI
PUSH DI  ; SAVE START ADDRESS
REP MOVSW ; MOVE THAT LINE ON SCREEN
POP DI
POP SI  ; RECOVER ADDRESSES
RET
ENDP

;------ CLEAR_ROW
PROC NEAR
MOV Cl,DL  ; GET 
CCUJMNS TO CLEAR
PUSH 01
REP STQSW  ; STORE THE FILL CHARACTER
POP 01
POP 01  ; RECOVER ADDRESSES
RET
ENDP

; SCROLL_DOWN
THIS ROUTINE MOVES THE CHARACTERS WITHIN A DEFINED BLOCK DOWN ON THE SCREEN, FILLING THE TOP LINES WITH A DEFINED CHARACTER

; INPUT
(AH) = CURRENT CRT MODE
(AL) = NUMBER OF LINES TO SCROLL
(CX) = UPPER LEFT CORNER OF REGION
(0X) = LOWER RIGHT CORNER OF REGION
(BH) = FILL CHARACTER
(0S) = DATA SEGMENT
(ES) = REGEN SEGMENT

; OUTPUT
SCREEN IS SCROLLED

;Continue_Down
STD  ; DIRECTION FOR SCROLL DOWN
NOV BL,AL ; LINE COUNT TO BL
CMP AH,4  ; TEST FOR GRAPHICS
JC HI2
CNP AH,7 ; TEST FOR BW CARD
JZ HI2
JMP GRAPHICS_DOWN

JNZ HIS
SUB SI,AX ; SI IS FROM ADDRESS
MOV AH,OH ; GET TOTAL # ROWS
SUB AH,Bl ; COUNT TO MOVE IN SCROLL
JNZ HIS
SUB AI,BP ; GO TO NEXT ROW
JNZ HIS
JMP HS ; SCROLL.END

; Continue_Down
STD ; DIRECTION FOR SCROLL DOWN
MOV BL,AL  ; LINE COUNT TO BL
CMP AH,4 ; TEST FOR GRAPHICS
JC HI2
Mov AH,DH ; GET ADDRESS
SUB SI,AX ; SI IS FROM ADDRESS
SUB AI,BP ; GO TO NEXT ROW
JNZ HIS
JMP HS ; SCROLL.END
LOC OBJ  LINE  SOURCE

F37D BE00 3834 JMP NI4
3835 SCROLL_DOWN ENDP
3836 ;-------------
3837 ; READ_AC_CURRENT
3838 ; THIS ROUTINE READS THE ATTRIBUTE AND CHARACTER AT THE CURRENT
3839 ; CURSOR POSITION AND RETURNS THEM TO THE CALLER
3840 ; INPUT
3841 ; (AH) = CURRENT CRT MODE
3842 ; (BH) = DISPLAY PAGE (ALPHA MODES ONLY)
3843 ; (DS) = DATA SEGMENT
3844 ; (ES) = REGEN SEGMENT
3845 ; OUTPUT
3846 ; (AL) = CHAR READ
3847 ; (AH) = ATTRIBUTE READ
3848 ; ----------------------
3849 ; ASSUME CS:CODE,DS:DATA,ES:DATA
F37D 80FC04 3850 READ_AC_CURRENT PROC NEAR
F37D 7208 3851 CMP AH,4 ; IS THIS GRAPHICS
F37D 7207 3852 JC P1
F37D 7403 3853 CMP AH,7 ; IS THIS 84 CARD
F37D 7402 3854 JE P1
F37D E9A902 3855 JMP GRAPHICS_READ
P1: ; READ_AC_CONTINUE
F360 E01A00 3857 CALL FIND_POSITION ; READ_AC_CONTINUE
F360 80F3 3858 MOV SI,DX ; ESTABLISH ADDRESSING IN SI
F359 3859
3860 ;----- WAIT FOR HORIZONTAL RETRACE
F35F BB16D300 R 3861 MOV DX,ADDR_6845 ; GET BASE ADDRESS
3862 ADD DX,6 ; POINT AT STATUS PORT
3863 3864 PUSH ES
3865 MOV DS,SS
3866 ; GET SEGMENT FOR QUICK ACCESS
3867 P2: ; WAIT FOR RETRACE LOW
F35F EC 3868 IN AL,DX ; GET STATUS
3869 JNZ P2 ; WAIT UNTIL IT IS
F35F 75FB 3870 JMP GRAPHICS_READ
F35F 75FB 3871 CLI ; NO MORE INTERRUPTS
F35F 7571 3872 IN AL,DX ; GET STATUS
F35F 7570 3873 TEST AL,1 ; IS IT HIGH
F35F AD 3874 JZ P3 ; WAIT UNTIL IT IS
F35F 74FB 3875 JCXZ P5
F35F 747F 3876 XOR CH,CH
F35F 746F 3877 MOV CX,CRT_LEN LENGTH OF BUFFER
F35F 746B 3878 LOOP P4
F35F 7469 3879 JMP VIDEO_RETURN
F344 EB20FE 3880 READ_AC_CURRENT ENDP
F344 3881
F347 8ACF 3882 FIND_POSITION PROC NEAR
F347 3883 MOV CL,BH ; DISPLAY PAGE TO CX
F347 3884 MOV CH,CH
F347 3885 MOV CX,CRT_LEN ; MOVE TO SI FOR INDEX
F347 3886 ADD SI,CL
F347 3887 JCXZ P5 ; MOVE TO START OF BUFFER OFFSET
F347 3888 MOV AX,[SI+OFFSET CURSOR_POSN] ; GET ROW/COLUMN OF THAT PAGE
F347 3889 XOR BX,BX ; GET SET START ADDRESS TO ZERO
F347 388A E036 3890 JCXZ P5 ; NO_PAGE
F347 3889 LOOP P4
F347 3888 ADD BX,CRT_LEN ; LENGTH OF BUFFER
F347 3887 P4: ; PAGE_LOOP
F347 3886 ADD BX,CRT_LEN ; LENGTH OF BUFFER
F347 3885 LOOP P4
F347 3884 P5: ; NO_PAGE
F347 3883 CALL POSITION ; DETERMINE LOCATION IN REGEN
F347 3882 ADD BX,AX ; ADD TO START OF REGEN
F347 3881 PET
F347 3880 FIND_POSITION ENDP
F347 3879
F342 C3 3885 ;----------------------
F342 3886 ; WRITE_AC_CURRENT
F342 3887 ; THIS ROUTINE WRITES THE ATTRIBUTE AND CHARACTER AT
F342 3888 ; THE CURRENT CURSOR POSITION
F342 3889 ; INPUT
F342 388A ; (AH) = CURRENT CRT MODE
F342 388B ; (BH) = DISPLAY PAGE
F342 388C ; (CX) = COUNTER OF CHAR TO WRITE
F342 388D ; (AL) = CHAR TO WRITE
F342 388E ; (BL) = ATTRIBUTE OF CHAR TO WRITE
F342 388F ; (DS) = DATA SEGMENT
F342 3890 ; (ES) = REGEN SEGMENT
F342 3891 ; OUTPUT
F342 3892 ; NONE
F342 3893 ;----------------------
F342 3894 A-53
WRITE_AC_CURRENT PROC NEAR

; THIS ROUTINE WRITES THE CHARACTER AT
; THE CURRENT CURSOR POSITION, ATTRIBUTE UNCHANGED

; INPUT
; (AH) = CURRENT CRT MODE
; (BH) = DISPLAY PAGE
; (CX) = COUNT OF CHARACTERS TO WRITE
; (DS) = DATA SEGMENT
; (ES) = REGION SEGMENT

; OUTPUT
; NONE

WRITE_AC_CURRENT PROC NEAR

CMP AH, 4
JNZ P6
JMP GRAPHICS_WRITE

MOV AH, 4
JMP P6:

CMP AH, 7
JNZ P9
JMP P6:

JMP P7:

; ------- WAIT FOR HORIZONTAL RETRACE

MOV DX, ADDR_6845
ADD DX, 6
PUSH AX
PUSH CX
CALL FIND_POSITION
MOV DI, DX
POP CX
POP BX
CHARACTER IN BX REG
PUSH BX
PUSH AX
JMP GRAPHICS_WRITE

; ------ WAIT FOR HORIZONTAL RETRACE

MOV DX, ADDR_6845
ADD DX, 6
PUSH AX
PUSH CX
MOV DI, DX
POP CX
POP BX
JMP VIDEO_RETURN

; ------- WRITE_LOOP

CMP AL, 0
JNZ P9
TEST AL, 1
JZ P6:
TEST AL, 1
JNZ P9
MOV AX, ES
MOV CX, DS
STOSW
JMP P7:

; ------ WAIT FOR HORIZONTAL RETRACE

MOV DX, ADDR_6845
ADD DX, 6
PUSH AX
PUSH CX
CALL FIND_POSITION
MOV DI, DX
POP CX
POP BX
CHARACTER IN BX REG
PUSH BX
PUSH AX
JMP GRAPHICS_WRITE

; ------- WRITE_LOOP

CMP AL, 0
JNZ P9
TEST AL, 1
JZ P6:
TEST AL, 1
JNZ P9
MOV AX, ES
MOV CX, DS
STOSW
JMP P7:

; ------ WAIT FOR HORIZONTAL RETRACE

MOV DX, ADDR_6845
ADD DX, 6
PUSH AX
PUSH CX
CALL FIND_POSITION
MOV DI, DX
POP CX
POP BX
CHARACTER IN BX REG
PUSH BX
PUSH AX
JMP VIDEO_RETURN

; ------- WRITE_LOOP

CMP AL, 0
JNZ P9
TEST AL, 1
JZ P6:
TEST AL, 1
JNZ P9
MOV AX, ES
MOV CX, DS
STOSW
JMP P7:

; ------ WAIT FOR HORIZONTAL RETRACE

MOV DX, ADDR_6845
ADD DX, 6
PUSH AX
PUSH CX
CALL FIND_POSITION
MOV DI, DX
POP CX
POP BX
CHARACTER IN BX REG
PUSH BX
PUSH AX
JMP VIDEO_RETURN

; ------- WRITE_LOOP

CMP AL, 0
JNZ P9
TEST AL, 1
JZ P6:
TEST AL, 1
JNZ P9
MOV AX, ES
MOV CX, DS
STOSW
JMP P7:

; ------ WAIT FOR HORIZONTAL RETRACE

MOV DX, ADDR_6845
ADD DX, 6
PUSH AX
PUSH CX
CALL FIND_POSITION
MOV DI, DX
POP CX
POP BX
CHARACTER IN BX REG
PUSH BX
PUSH AX
JMP VIDEO_RETURN

; ------- WRITE_LOOP

CMP AL, 0
JNZ P9
TEST AL, 1
JZ P6:
TEST AL, 1
JNZ P9
MOV AX, ES
MOV CX, DS
STOSW
JMP P7:

; ------ WAIT FOR HORIZONTAL RETRACE

MOV DX, ADDR_6845
ADD DX, 6
PUSH AX
PUSH CX
CALL FIND_POSITION
MOV DI, DX
POP CX
POP BX
CHARACTER IN BX REG
PUSH BX
PUSH AX
JMP VIDEO_RETURN

; ------- WRITE_LOOP

CMP AL, 0
JNZ P9
TEST AL, 1
JZ P6:
TEST AL, 1
JNZ P9
MOV AX, ES
MOV CX, DS
STOSW
JMP P7:
THESE ROUTINES WILL WRITE A DOT, OR READ THE
DOT AT THE INDICATED LOCATION.

ENTRY --

DO = ROW (0-199) (THE ACTUAL VALUE DEPENDS ON THE MODE)

CX = COLUMN (0-639) (THE VALUES ARE NOT RANGE CHECKED)

AL = DOT VALUE TO WRITE (1, 2 OR 4 BITS DEPENDING ON MODE).

REQ'D FOR WRITE DOT ONLY, RIGHT JUSTIFIED.

BIT 7 OF AL : 1 INDICATES XOR THE VALUE INTO THE LOCATION.

DS = DATA SEGMENT

ES = REGEN SEGMENT

EXIT

AL = DOT VALUE READ, RIGHT JUSTIFIED, READ ONLY

入口

PROC NEAR

CALL R3 ; DETERMINE BYTE POSITION OF DOT

MOV AL,ES:[SI] ; GET THE BYTE

AND AL,CL ; MASK OFF THE OTHER BITS IN THE BYTE

SHL AL,CL ; LEFT JUSTIFY THE VALUE

MOV CL,ES:[SI] ; GET NUMBER OF BITS IN RESULT

ROL AL,CL ; RIGHT JUSTIFY THE RESULT

JMP VIDEO.RETURN ; RETURN FROM VIDEO

READ_DOT PROC NEAR

PUSH CX ; SAVE OURING OPERATION

PUSH AX ; WILL SAVE AL DURING OPERATION

MOV AL,40

PUSH OX ; SAVE ROW VALUE

MOV ES:[SI].AL ; RESTORE THE BYTE IN MEMORY

POP AX

JMP VIDEO.RETURN ; RETURN FROM VIDEO

WRITE_DOT PROC NEAR

PUSH AX ; SAVE DOT VALUE

PUSH AX ; THICE

CALL R3 ; DETERMINE BYTE POSITION OF THE DOT

SHR AL,CL ; SHIFT TO SET UP THE BITS FOR OUTPUT

AND AL,CL ; STRIP OFF THE OTHER BITS

MOV CL,ES:[SI] ; GET THE CURRENT BYTE

POP BX ; RECOVER XOR FLAG

TEST BL,80H ; IS IT ON

JNZ ; YES. XOR THE DOT.

NOT AH ; SET THE MASK TO REMOVE THE INDICATED BITS

AND CL,AH

OR AL,CL ; OR IN THE NEW VALUE OF THOSE BITS

JMP FINISH_DOT ; FINISHING OPERATION

READ_DOT ENDP

WRITE_DOT ENDP

READ_DOT}

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

Determine the REGEN byte location of the indicated row/column value in Graphics Mode.

ENTRY --

DX = ROW VALUE (0-199)

CX = COLUMN VALUE (0-639)

EXIT --

SI = OFFSET INTO REGEN BUFFER FOR BYTE OF INTEREST

AH = MASK TO STRIP OFF THE BITS OF INTEREST

CL = BITS TO SHIFT TO RIGHT JUSTIFY THE MASK IN AH

DH = # BITS IN RESULT

F45B D02E

MDV AL,40

F45F S2

PUSH DX ; SAVE ROW VALUE

F45D B02E

MDV AX

F454 E90FFD

JMP VIDEO_RETURN

F453 D2CO

ROL AL,CL

F452 E9AOFD

JMP VIDEO.RETURN

F451 4AC4

MOV AL,ES:[SI]

F450 E83100

CALL 03 ; DETERMINE BYTE POSITION OF DOT

F44F 22C4

AND AL,AH ; MASK OFF THE OTHER BITS IN THE BYTE

F44E QAel

OR0 AL,CL ; OR IN THE NEW VALUE OF THOSE BITS

F44D 268A04

MOV AL,ES:[SI]

F44C 02E8

SHR AL,CL ; LEFT JUSTIFY THE VALUE

F44B 268AOC

MOV CL,ES:[SI] ; GET NUMBER OF BITS IN RESULT

F44A E90FFD

JMP VIDEO.RETURN ; RETURN FROM VIDEO

F449 E90FD

MOV AL,40

PUSH AX ; SAVE DOT VALUE

PUSH AX ; THICE

CALL R3 ; DETERMINE BYTE POSITION OF THE DOT

SHR AL,CL ; SHIFT TO SET UP THE BITS FOR OUTPUT

AND AL,CL ; STRIP OFF THE OTHER BITS

MOV CL,ES:[SI] ; GET THE CURRENT BYTE

POP BX ; RECOVER XOR FLAG

TEST BL,80H ; IS IT ON

JNZ ; YES. XOR THE DOT.

NOT AH ; SET THE MASK TO REMOVE THE INDICATED BITS

AND CL,AH

OR AL,CL ; OR IN THE NEW VALUE OF THOSE BITS

JMP FINISH_DOT ; FINISHING OPERATION

WRITE_DOT ENDP

READ_DOT}

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

Determine the REGEN byte location of the indicated row/column value in Graphics Mode.

ENTRY --

DX = ROW VALUE (0-199)

CX = COLUMN VALUE (0-639)

EXIT --

SI = OFFSET INTO REGEN BUFFER FOR BYTE OF INTEREST

AH = MASK TO STRIP OFF THE BITS OF INTEREST

CL = BITS TO SHIFT TO RIGHT JUSTIFY THE MASK IN AH

DH = # BITS IN RESULT

F45B D02E

MDV AL,40

F45F S2

PUSH DX ; SAVE ROW VALUE

F45D B02E

MDV AX
LOC OBJ LINE SOURCE

F468 0DEFE 4062 AND DL,0FH  ; STRIP OFF ODD/EVEN BIT
F463 F462 4063 MUL DL  ; AX HAS ADDRESS OF 1ST BYTE OF INDICATED ROW
F465 5A 4064 POP DX  ; RECOVER IT
F466 F4C201 4065 TEST DL,1  ; TEST FOR EVEN/ODD
F469 7403 4066 JZ 94  ; JUMP IF EVEN ROW
F468 050020 4067 ADD AX,2000H  ; OFFSET TO LOCATION OF ODD ROWS
F46E 4068 R4:  ; EVEN_ROW
F45E 8F69 4069 MOV SI,AX  ; MOVE POINTER TO SI
F470 59 4070 POP AX  ; RECOVER AL VALUE
F471 8D01 4071 MOV DX,CX  ; COLUMN VALUE TO DX

F472 4072  ; ------ DETERMINE GRAPHICS MODE CURRENTLY IN EFFECT
F473 BDC042 4073  ; SET UP THE REGISTERS ACcORDING TO THE MODE
F476 B97206 4074 CH = MASK FOR LOW OF COLUMN ADDRESS (7/3 FOR HIGH/MED RES)
F479 03E490006 R 4075 CL = # OF ADDRESS BITS IN COLUMN VALUE (1/2 FOR H/M)
F47E 7206 4076 BL = MASK TO SELECT BITS FROM POINTED BYTE (16H/COH FOR H/M)
F480 B88001 4077 BH = NUMBER OF VALID BITS IN POINTED BYTE (1/2 FOR H/M)
F483 E97307 4078
F484 4079
F486 4080
F486 2EEA 4081
F487 4082
F488 4083
F489 4084
F48A 4085
F48C 4086
F48E 4087
F490 4088
F492 4089
F493 4090
F494 4091
F494 4092
F495 4093
F496 4094
F497 4095
F498 4096
F499 4097
F49A 4098
F49B 4099
F49C 4100
F49D 4101
F49E 4102
F49F 4103
F4A0 FFFC 4104
F4A2 4105
F4A4 4106
F4A6 4107
F4A7 4108
F4A8 4109
F4A9 4110
F4AA 4111
F4AB 4112
F4AC 4113
F4AD 4114
F4AE 4115
F4AF 4116
F4B0 4117
F4B1 4118
F4B2 4119
F4B3 4120
F4B4 4121
F4B5 4122
F4B6 4123
F4B7 4124
F4B8 4125
F4B9 4126
F4BA 4127
F4BB 4128
F4BC 4129
F4BD 4130
F4BE 4131
F4BF 4132
F4C0 4133
F4C1 4134
F4C2 4135
F4C3 4136

A-56
LOC OBJ

LINE

SOURCE

F447 2D01 4137 SUB DX,CX ; ADJUST VALUES
F448 81C20101 4138 ADD DX,10H ; MULTIPLY # ROWS BY 4 SINCE 8 VERT DOTS/CHAR
F44D DE06 4139 SAL DH,1 ; AND EVEN/ODD ROWS
F44F DE06 4140 SAL DH,1
F450 4141 ----- DETERMINE CRT MODE
F451 003C400006 4142 CMP CRT_MODE,6 ; TEST FOR MEDIUM RES
F456 7D04 4143 JNC R7 ; FIND_SOURCE
F457 4144 ----- MEDIUM RES UP
F45B DE02 4145 SAL DL,1 ; # COLUMNS = 10 SINCE 2 BYTES/CHAR
F45B 1E17 4146 SAL DI,1 ; OFFSET #2 SINCE 2 BYTES/CHAR
F45C 4150
F45D 4151 ----- DETERMINE THE SOURCE ADDRESS IN THE BUFFER
F45C 4152 R7: ; FIND_SOURCE
F460 06 4153 PUSH ES ; GET SEGMENTS BOTH POINTING TO REGEN
F460 1F 4154 POP DS
F462 1AE3 4155 MOV CH,CH ; ZERO TO HIGH OF COUNT REG
F463 0E03 4156 SAL BL,1 ; MULTIPLY NUMBER OF LINES BY 4
F463 2E03 4157 SAL BL,1
F464 7A0D 4158 JZ R11 ; IF ZERO, THEN BLANK ENTIRE FIELD
F465 94C3 4159 MOV A1,DL ; GET NUMBER OF LINES IN AL
F465 9449 4160 MOV AH,DL ; # BYTES/ROM
F467 F6E4 4161 MUL AH ; DETERMINE OFFSET TO SOURCE
F46C 87F7 4162 MOV SI,01 ; SET UP SOURCE
F46D 0300 4163 ADD SI,A1 ; ADD IN OFFSET TO IT
F46D BAE6 4164 MOV AH,01 ; NUMBER OF ROWS IN FIELD
F470 2AE3 4165 SUB AH,DL ; DETERMINE NUMBER TO MOVE
F47B 4166
F47B 4167 ----- LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD FIELDS
F47C 4168 R8: ; ROW_LOOP
F47D 4169 CALL R17 ; MOVE ONE ROW
F47E 4170 SUB SI,2000H-80 ; MOVE TO NEXT ROW
F47F 4171 SUB DI,2000H-80
F480 7F01 4172 DEC AH ; NUMBER OF ROWS TO MOVE
F481 7F13 4173 JNZ R8 ; CONTINUE TILL ALL MOVED
F482 4174
F483 4175 ----- FILL IN THE VACATED LINE(S)
F483 4176 R9: ; CLEAR_ENTRY
F484 4177 MOV AL,AH ; ATTRIBUTE TO FILL WITH
F485 4177
F486 4177 R10: MOV AL,OH
F487 4178 DEC AL ; NUMBER OF LINES TO FILL
F488 4179 CALL A18 ; CLEAR THAT ROW
F489 4179 CALL R18 ; CLEAR_ENTRY
F48A 417A CALL A18 ; CLEAR_ENTRY
F48B 417A CALL A18 ; CLEAR_ENTRY
F48C 417B CALL A18 ; CLEAR_ENTRY
F48D 417B CALL A18 ; CLEAR_ENTRY
F48E 417C CALL A18 ; CLEAR_ENTRY
F48F 417C CALL A18 ; CLEAR_ENTRY
F490 417D CALL A18 ; CLEAR_ENTRY
F491 417D CALL A18 ; CLEAR_ENTRY
F492 417E CALL A18 ; CLEAR_ENTRY
F493 417E CALL A18 ; CLEAR_ENTRY
F494 417F CALL A18 ; CLEAR_ENTRY
F495 417F CALL A18 ; CLEAR_ENTRY
F496 4180 CALL A18 ; CLEAR_ENTRY
F497 4180 CALL A18 ; CLEAR_ENTRY
F498 4181 CALL A18 ; CLEAR_ENTRY
F499 4181 CALL A18 ; CLEAR_ENTRY
F49A 4182 CALL A18 ; CLEAR_ENTRY
F49B 4182 CALL A18 ; CLEAR_ENTRY
F49C 4183 CALL A18 ; CLEAR_ENTRY
F49D 4183 CALL A18 ; CLEAR_ENTRY
F49E 4184 CALL A18 ; CLEAR_ENTRY
F49F 4184 CALL A18 ; CLEAR_ENTRY
F4A0 4185 CALL A18 ; CLEAR_ENTRY
F4A1 4185 CALL A18 ; CLEAR_ENTRY
F4A2 4186 MOV BH,08H ; SET BLANK COUNT TO EVERYTHING IN FIELD
F4A2 E8EC 4187 JMP R9 ; CLEAR THE FIELD
F4A3 E8EC 4187 GRAPHICS_UP ENDP
F4A4 4188 -----------------------------------------------
F4A4 4189 | SCROLL DOWN
F4A4 4190 | SCROLL_DOWN PROC NEAR
F4A5 4191 | THIS ROUTINE SCROLLS DOWN THE INFORMATION ON THE CRT
F4A6 4192 | ENTRY --
F4A7 4193 | CH.CL = UPPER LEFT CORNER OF REGION TO SCROLL
F4A8 4194 | DH.CL = LOWER RIGHT CORNER OF REGION TO SCROLL
F4A9 4195 | BOTH OF THE ABOVE ARE IN CHARACTER POSITIONS
F4AA 4196 | BH = FILL VALUE FOR BLANKED LINES
F4AB 4197 | AL = # LINES TO SCROLL (AL=0 MEANS BLANK THE ENTIRE FIELD)
F4AC 4198 | DS = DATA SEGMENT
F4AD 4199 | ES = REGEN SEGMENT
F4AE 4200 | EXIT --
F4AF 4201 | NOTHING, THE SCREEN IS SCROLLED
F4B0 4202 |-----------------------------------------------
F4B1 4203
F4B2 4204 |
F4B3 4204 | GRAPHICS_DOWN PROC NEAR
F4B4 4205 | STD ; SET DIRECTION
F4B5 4206 | MOV BL,AL ; SAVE LINE COUNT IN BL
F4B6 4206 | MOV AH,DX ; GET LOWER RIGHT POSITION INTO AX REG
F4B7 4207 |
F4B8 4208 |
F4B9 4209 |----- USE CHARACTER SUBROUTINE FOR POSITIONING
F4BA 4209 |
F4BB 4210 |----- ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE
F4BC 4211
F4BD 4212 | CALL GRAPH_PSN

A-57
; DETERMINE SIZE OF WINDOW
F4FF 0FF8 4213 MOV DI,AX
; SAVE RESULT AS DESTINATION ADDRESS
4214
4215
;------ DETERMINE SIZE OF WINDOW
F501 2BD1 4217 SUB DX,AX
F503 61C20101 MOV DI,101H
F507 0E16 SAL DI,1
F509 0E16 SAL DI,1
; AND EVEN/ODD ROWS
4218
4219
F50B 93E49006 R 4220 CMP CR1_MODE,:6 ; TEST FOR MEDIUM RES
F510 7315 JNC R12 ; FIND_SOURCE_DOWN
4221
4222
;------ DETERMINE CRT MODE
F512 0E02 4223 CMP ; TEST FOR MEDIUM RES
F514 1D17 4224 SAL DI,1
F516 47 INC DI ; POINT TO LAST BYTE
4225
4226
;------ DETERMINE THE SOURCE ADDRESS IN THE BUFFER
F517 4227 R12 ! ; FIND_SOURCE_DOWN
4228
4229
;------ ROUTINE TO MOVE ONE ROW OF INFORMATION
F553 422A R13: ; ROW_LOOP_DOWN
F553 E82900 JMP VIDEO_RETURN ; EVERYTHING DONE
4230
4231
;------ LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD FIELDS
F523 742E 4232 JZ R14 ; IF ZERO, THEN BLANK ENTIRE FIELD
F525 8AC3 MOV AL,BL ; GET NUMBER OF LINES IN AL
F527 F6E4 MOV AH,80 ; 80 BYTES/ROW
F529 8BF7 MOV SI,DI ; SET UP SOURCE
F52A 0F80 SUB SI,AX ; SUBTRACT THE OFFSET
F52B 0EA6 MOV AH,0DH ; NUMBER OF ROWS IN FIELD
F531 2A13 MOV SUB AH,DL ; DETERMINE NUMBER TO MOVE
4233
4234
;------ FILL IN THE VACATED LINE(S)
F52F 8B0E MOV SI,AX ; SUBTRACT THE OFFSET
F530 0A66 MOV AH,DL ; DETERMINE NUMBER TO MOVE
4235
4236
;------ ROUTINE TO MOVE ONE ROW OF INFORMATION
F533 8AC7 MOV AL,8H ; ATTRIBUTE TO FILL WITH
F534 BACA MOV CL,DL
;------ DETERMINE CRT ADDRESS
F555 E974FC JMP VIDEO_RETURN ; EVERYTHING DONE
4237
4238
;------ DETERMINE CRT ADDRESS
F53A 8BF7 MOV SI,DI ; SET UP SOURCE
F53B 0F80 SUB SI,AX ; SUBTRACT THE OFFSET
F53C 0EA6 MOV AH,0DH ; NUMBER OF ROWS IN FIELD
F540 75F1 JNZ R13 ; CONTINUE TILL ALL MOVED
4239
4240
;------ DETERMINE CRT ADDRESS
F542 8AC7 MOV AL,BH
;------ DETERMINE CRT ADDRESS
F544 8B0E MOV SI,AX ; SUBTRACT THE OFFSET
F545 0A66 MOV AH,DL ; DETERMINE NUMBER TO MOVE
4241
4242
;------ ROUTINE TO MOVE ONE ROW OF INFORMATION
F553 8AC7 MOV AL,8H ; ATTRIBUTE TO FILL WITH
F554 BACA MOV CL,DL
;------ DETERMINE CRT ADDRESS
F555 E974FC JMP VIDEO_RETURN ; EVERYTHING DONE
4243
4244
;------ ROUTINE TO MOVE ONE ROW OF INFORMATION
F557 8ACA MOV CL,DL ; NUMBER OF BYTES IN THE ROW
F559 56 PUSH SI
F56A 57 PUSH SI ; SAVE POINTERS
F56B 13 REP NEAR ; MOVE THE EVEN FIELD
4245
4246
;------ ROUTINE TO MOVE ONE ROW OF INFORMATION
F55C 4A MOV SI,0
F55D 6F MOV SI,DL
;------ DETERMINE CRT ADDRESS
F55F 81C60010 ADD SI,2000H
F563 81C70020 ADD DI,2000H
F567 56 PUSH SI
F568 57 PUSH SI ; SAVE THE POINTERS
F569 8ACA MOV CL,DL ; COUNT BACK
FS6B F3 407 REP MOVSB ; MOVE THE ODD FIELD
FS6C A4 408 POP DI ; POINTERS BACK
FS6E SE 409 POP SI ; RETURN TO CALLER
FS6F C3 410 R17 ENDP

FS70 R18 PROC NEAR
FS70 BACA 411 MOV Cl,DL ; NUMBER OF BYTES IN FIELD
FS72 S7 412 PUSH DI ; SAVE POINTER
FS73 F3 413 REP STOSB ; STORE THE NEW VALUE
FS74 AA 414 OR FS75 SF 415 POP OR FS76 B1C70020 416 ADD DI,1000H ; POINT TO ODD FIELD
FS76 S7 417 PUSH DI
FS77 BACA 418 MOV Cl,DL
FS77 F3 419 REP STOSB ; FILL THE ODD FIELD
FS78 AA 420 OR FS79 SF 421 POP OR FS80 C3 422 RET ; RETURN TO CALLER

FS80 C3 423 ; ------ CLEAR A SINGLE ROW
FS81 424 ;------------------------
FS81 GRAPHICS_WRITE PROC NEAR
FS82 B400 425 MOV AH,0 ; ZERO TO HIGH Of CODE POINT
FS83 50 426 PUSH AX ; SAVE CODE POINT VALUE
FS85 3C80 427 JMP SHORT S2 ; DETERMINE_NODE
FS86 438 A-59
FS94   4359  ; IMAGE IS IN SECOND HALF, IN USER RAM
FS94  ZC00   4360  ; EXTEND_CHAR
FS96  IE    4361  ; ZERO ORIGIN FOR SECOND HALF
FS97  2BF6   4362  ; SAVE DATA POINTER
FS99  BEDE   4363  ; ESTABLISH VECTOR ADDRESSING
FS9B  CS47C00  4364  ; GET THE OFFSET OF THE TABLE
FS9F  8CDA   4365  ; GET THE SEGMENT OF THE TABLE
FSAI  IF    4366  ; RECOVER DATA SEGMENT
FSAE  52    4367  ; SAVE TABLE SEGMENT ON STACK

--- DETERMINE GRAPHICS MODE IN OPERATION ---

FS43   4368  ; DETERMINE_MODE
FS43  D1E0   4369  ; MULTIPLY CODE POINT
FS45  D1E0   4370  ; VALUE BY 8
FS47  D1E0   4371  ; SI HAS OFFSET OF DESIRED CODES
FS49  D9F0   4372  ; CPTR_MODE
F500  IF    4373  ; RECOVER TABLE POINTER SEGMENT
F501  72C   4374  ; TEST FOR MEDIUM RESOLUTION MODE
F503   4375  ; HIGH_CHAR
F503  57    4376  ; SAVE REGEN POINTER
F504  56    4377  ; SAVE CODE POINTER
F505  8604   4378  ; NUMBER OF TIMES THROUGH LOOP
F507   4379  ; GET BYTE FROM CODE POINTS
F50B  F4380   4380  ; SHOULD WE USE THE FUNCTION
F50B  7516   4381  ; TO PUT CHAR IN
F50D  AA    4382  ; STORE IN REGEN BUFFER
F50E  AC    4383  ; MORE CHARS TO WRITE
F50F   4384  ; BACK TO MAINSTREAM

--- HIGH RESOLUTION MODE ---

F50F  260695FF1F  4385  ; STORE IN SECOND HALF
F50F  B3C74F   4386  ; STORE SECOND HALF
F50F  FECE   4387  ; MOVE TO NEXT ROW IN REGEN
F50C  75EC   4388  ; DONE WITH LOOP
F50D  5E     4389  ; GET CODE POINT
F50D  20369   4390  ; DOUBLE UP ALL THE BITS
F50E  E80F   4391  ; BACK TO MAINSTREAM

--- MEDIUM RESOLUTION WRITE ---

F50F  2AD5   4392  ; MED_RES_WRITE
F50F  B1E7   4393  ; SAVE HIGH COLOR BIT
F50F  E80100  4394  ; OFFSET+2 SINCE 2 BYTES/CHAR
F50F  6604   4395  ; MED_CHAR
F50F  57    4396  ; ADD DI,79
F50F  59    4397  ; MOVE NEXT ROW IN REGEN
F50F  67    4398  ; DONE WITH LOOP
F50F  6604   4399  ; GET CODE POINT
F50F  B452   439A  ; DOUBLE UP ALL THE BITS
F50F  B519   439B  ; CONVERT THEM TO FOREGROUND COLOR ( 0 BACK )
F50F  BE55   439C  ; THIS XOR FUNCTION
F50F  621    439D  ; NO, STORE IT IN AS IT IS
F50F  58    439E  ; DO FUNCTION WITH HALF
F50F  52    439F  ; AND WITH OTHER HALF
F50F  55    439G  ; GET CODE POINT
F50F  4E00   439H  ; STORE FIRST BYTE
F50F  6E00   439I  ; STORE SECOND BYTE
F50F  E8500  439J  ; GET CODE POINT

A-60
AND AX,BX ; CONVERT TO COLOR
TEST DL,A0H ; AGAIN, IS THIS XOR FUNCTION
JZ SI1 ; NO, JUST STORE THE VALUES
XOR AH,ES:DI+2000H ; FUNCTION WITH FIRST HALF
XOR AL,ES:DI+2001H ; AND WITH SECOND HALF
MOV ES:DI+2000H,AH ; STORE IN SECOND PORTION OF BUFFER
ADD DI,00 ; POINT TO NEXT LOCATION
DEC DH ; LOOP S8 ; MORE TO WRITE
MOV ES:(DH+2000H),AH
MOV ES: [01+2000H+1]AH ; STORE IN SECOND PORTION OF BUFFER
ADO 01,60 ; POINT TO NEXT LOCATION
DEC OH ; KEEP GOING
JNZ S11 ; NO
I JUST STORE THE VALUES
XOR AH.ES:{DH+2000H} ; FUNCTION WITH FIRST HALF
XOR AL,ES:[01+2000H+1]AH ; AND WITH SECOND HALF
MOV ES:DI+2000H,AH
MOV ES:[01+2000H+1]AH ; STORE IN SECOND PORTION OF BUFFER
ADD DI,2 ; POINT TO NEXT CHAR POSITION
LOOP 58
JMP VIDEO_RETURN
GRAPHICS_WRITE ENDP
GRAPHICS_READ PROC NEAR
CALL 52& ; CONVERTED TO OFFSET IN REGEN
MOV SI,AX ; SAVE IN SI
MOV BP,SP POINTER TO SAVE AREA
DETERMINE GRAPHICS MODES
MOV AL,4 ; NUMBER OF PASSES
MOV BP,AL ; ADJUST AND STORE
INC BP
MOV SI,60 ; POINTER INTO REGEN
MOV DH ; LOOP CONTROL
DEC DH
JHZ S14
JSIP SI; MATCH THE SAVED CODE POINTS
MOV DI,0F6EH ; OFFSET CRT_CHAR_GENERATE ADDRESSING
PUSH CS ; FOR THE STRING COMPARE
MOV DX,12& ; NUMBER TO TEST AGAINST
PUSH 51 ; SAVE SAVE AREA POINTER
LOC OBJ

F669 57 4511  PUSH  DI   ; SAVE CODE POINTER
F66A 89000 4512  MOV  CX,8   ; NUMBER OF BYTES TO MATCH
F66D F3 4513  REP  CMPSB  ; COMPARE THE 8 BYTES
F66E A6 4514  POP  DI   ; RECOVER THE POINTERS
F66F 5F 4515  POP  SI   ; IF ZERO FLAG SET, THEN MATCH OCCURRED
F671 741E 4516  JZ 518   ; NO MATCH, MOVE ON TO NEXT
F673 FEC0 4517  INC  AL   ; NEXT CODE POINT
F675 63C700 4518  ADD  DI,8   ; NEXT CODE POINT
F678 4A 4519  DEC  DX   ; LOOP CONTROL
F679 75ED 4520  JNZ 517   ; DO ALL OF THEM
F67B 4521  
F67C 3C00 4522  CMP  AL,0   ; AL<> 0 IF ONLY 1ST HALF SCANNED
F67D 7412 4523  JE 518   ; IF = 0, THEN ALL HAS BEEN SCANNED
F67E 26C0 4524  SUB  AX,AX   ; ESTABLISH ADDRESSING TO VECTOR
F680 4E0B 4525  MOV  DS,AX   ; ASSUME DS:AX
F681 C357C00 4526  LES  DI,EXT_PTR  ; GET POINTER
F682 5C00 4527  MOV  AX,ES   ; SEE IF THE POINTER REALLY EXISTS
F683 0BC7 4528  OR  AX,DI   ; IF ALL 0, THEN DOESN'T EXIST
F684 7404 4529  JZ 51.   ; NO SENSE LOOKING
F685 B000 4530  MOV  AL,128   ; ORIGIN FOR SECOND HALF
F686 EBD2 4531  JMP 51.   ; GO BACK AND TRY FOR IT
F687 ASSUME DS:OATA 4532  
F688 E910FB 4533  JNP VIDEO.RETURN  ; ALL DONE
F689 4534  
F68A 08C408 4535  ADD  SP,8   ; READJUST THE STACK. THROW AWAY SAVE
F68B 83CF 4536  ADI  SP,6   ; CHARACTER IS FOUND ( AL=0 IF NOT FOUND )
F68C 41 4537  JMP VIDEO_READ  ; ALL DONE
F68D 51. Etmp 4538  
F68E 80E303 4539  ADD  AL,0   ; READJUST THE STACK. THROW AWAY SAVE
F68F 80E30B 4540  JMP VIDEO_READ  ; ALL DONE
F690 80E303 4541  
F691 51. PROC NEAR 4542  
F692 80E30B 4543  AND  BL,3   ; ISOLATE THE COLOR BITS
F693 80E313 4544  MOV  AL,BL   ; COPY TO AL
F694 86C5 4545  PUSH  CX   ; SAVE REGISTER
F695 80E310 4546  MOV  CX,3   ; NUMBER OF TIMES TO DO THIS
F696 80E30F 4547  S20:  ; ISO COLOR BITS
F697 80E314 4548  SAL  AL,1   ; LEFT SHIFT BY 2
F698 80E315 4549  SAL  AL,1   ; LEFT SHIFT BY 2
F699 80E316 4550  OR  BL,AL   ; ANOTHER VERSION INTO BL
F69A E2F8 4551  LOOP S20   ; FILL ALL OF BL
F69B 8AFB 4552  MOV  BH,BL   ; FILL UPPER PORTION
F69C 59 4553  POP  CX   ; REGISTER BACK
F69D 83C4 4554  RET   ; ALL DONE
F69E S19 ENDP 4555  
F69F 51. PROC NEAR 4556  
F6A0 80E30B 4557  JNZ 51.   ; THE RESULT IS LEFT IN AX
F6A1 51. ENDP 4558  
F6A2 80E308 4559  AND  BL,3   ; ISO COLOR BITS
F6A3 80E30B 4560  MOV  AL,BL   ; COPY TO AL
F6A4 80E30F 4561  S20:  ; ISO COLOR BITS
F6A5 80E314 4562  SAL  AL,1   ; LEFT SHIFT BY 2
F6A6 80E315 4563  SAL  AL,1   ; LEFT SHIFT BY 2
F6A7 80E316 4564  OR  BL,AL   ; ANOTHER VERSION INTO BL
F6A8 E2F8 4565  LOOP S20   ; FILL ALL OF BL
F6A9 8AFB 4566  MOV  BH,BL   ; FILL UPPER PORTION
F6AA 59 4567  POP  CX   ; REGISTER BACK
F6AB 83C4 4568  RET   ; ALL DONE
F6AC S19 ENDP 4569  
F6AD 51. PROC NEAR 4570  
F6AE 80E30B 4571  JNZ 51.   ; THE RESULT IS LEFT IN AX
F6AF 51. ENDP 4572  
F6B0 80E30B 4573  AND  BL,3   ; ISO COLOR BITS
F6B1 80E30B 4574  MOV  AL,BL   ; COPY TO AL
F6B2 80E30F 4575  S20:  ; ISO COLOR BITS
F6B3 80E315 4576  SAL  AL,1   ; LEFT SHIFT BY 2
F6B4 80E316 4577  SAL  AL,1   ; LEFT SHIFT BY 2
F6B5 80E317 4578  OR  BL,AL   ; ANOTHER VERSION INTO BL
F6B6 E2F8 4579  LOOP S20   ; FILL ALL OF BL
F6B7 8AFB 4580  MOV  BH,BL   ; FILL UPPER PORTION
F6B8 59 4581  POP  CX   ; REGISTER BACK
F6B9 83C4 4582  RET   ; ALL DONE
F6BA S19 ENDP 4583  
F6BB 51. PROC NEAR 4584  
F6BC 80E30B 4585  JNZ 51.   ; THE RESULT IS LEFT IN AX
F6BD 51. ENDP 4586  
F6BE 80E30B 4587  AND  BL,3   ; ISO COLOR BITS
F6BF 80E30B 4588  MOV  AL,BL   ; COPY TO AL
F6C0 80E30F 4589  S20:  ; ISO COLOR BITS
F6C1 80E314 4590  SAL  AL,1   ; LEFT SHIFT BY 2
F6C2 80E315 4591  SAL  AL,1   ; LEFT SHIFT BY 2
F6C3 80E316 4592  OR  BL,AL   ; ANOTHER VERSION INTO BL
F6C4 E2F8 4593  LOOP S20   ; FILL ALL OF BL
F6C5 8AFB 4594  MOV  BH,BL   ; FILL UPPER PORTION
F6C6 59 4595  POP  CX   ; REGISTER BACK
F6C7 83C4 4596  RET   ; ALL DONE
F6C8 S19 ENDP 4597  
F6C9 51. PROC NEAR 4598  
F6CA 80E30B 4599  JNZ 51.   ; THE RESULT IS LEFT IN AX
F6CB 51. ENDP 459A  
F6CC 51. PROC NEAR 459B  
F6CD 80E30B 459C  JNZ 51.   ; THE RESULT IS LEFT IN AX
F6CE 51. ENDP 459D  
F6CF 51. PROC NEAR 459E  
F6D0 80E30B 459F  JNZ 51.   ; THE RESULT IS LEFT IN AX
F6D1 51. ENDP 45A0  
F6D2 B40000 45A1  MOV  DX,0   ; RESULT REGISTER
F6D3 890100 45A2  MOV  CX,1   ; MASK REGISTER
F6D4 860B 45A3  MOV  BX,AX   ; BASE INTO TEMP
F6D5 2309 45A4  AND  BX,CX   ; USE MASK TO EXTRACT A BIT
F6D6 0803 45A5  OR  DX,BX   ; PUT INTO RESULT REGISTER
F6D7 D1E0 45A6  SHL  AX,1   ; SHIFT BASE AND MASK BY 1
F6D8 D1E1 45A7  SHL  CX,1   ; SHIFT BASE AND MASK BY 1
F6D9 80DB 45A8  MOV  BX,AX   ; BASE INTO TEMP
F6DA 2309 45A9  AND  BX,CX   ; EXTRACT THE SAME BIT
F6DB 0803 45AA  OR  DX,BX   ; PUT INTO RESULT

A-62
Hough, the document contains several assembly language instructions and comments explaining the functionality of the code. It appears to be related to video card operations, specifically dealing with cursor position, foreground color, and other related tasks. The text is not in a natural language format but rather in a technical, programming-oriented style. The comments are written in English and provide context and explanations for the assembly instructions. The document includes various labels and procedures, which are common in assembly language code. The overall purpose seems to be managing video output, possibly for a graphics interface.
THE COLOR IS USED.

ENTRY --

CURRENT CRT MODE

NOTE THAT BACK SPACE, CAR RET, BELL AND LINE FEED ARE HANDLED

AS COMMANDS RATHER THAN AS DISPLAYER GRAPHICS

CURRENTLY IN A GRAPHICS MODE

EXIT --

ALL REGISTERS SAVED

--- DX NOW HAS THE CURRENT CURSOR POSITION

WRITE THE CHAR TO THE SCREEN

-------- POSITION THE CURSOR FOR NEXT CHAR

-------- SCROLL REQUIRED

-------- DETERMINE VALUE TO FILL WITH DURING SCROLL

-------- SCROLL-UP

-------- SCROLL-UP

-------- VIDEO-CALL-RETURN

-------- TTY-RETURN

RESTORE THE CHARACTER
LIGHT PEN
THIS ROUTINE TESTS THE LIGHT PEN SWITCH AND THE LIGHT PEN TRIGGER. IF BOTH ARE SET, THE LOCATION OF THE LIGHT PEN IS DETERMINED. OTHERWISE, A RETURN WITH NO INFORMATION IS MADE.

ON EXIT:

IF NO LIGHT PEN INFORMATION IS AVAILABLE

IF LIGHT PEN IS AVAILABLE

ROW, COLUMN OF CURRENT LIGHT PEN POSITION

B = BEST GUESS AT PIXEL HORIZONTAL POSITION

ASSUME CS:CODE, DS:DATA

SUBTRACT_TABLE

VI  LABEL BYTE

DB  3,3,5,5,3,3,3,4

READ_LPEN  PROC  NEAR

WAIT FOR LIGHT PEN TO BE DEPRESSED

MOV AL,0 ; SET NO LIGHT PEN RETURN CODE

MOV DX,ADDR_6545 ; GET BASE ADDRESS OF 6545

ADD DX,6 ; POINT TO STATUS REGISTER

IN AL,DX ; GET STATUS REGISTER

TEST AL,4 ; TEST LIGHT PEN SWITCH

JNZ V6 ; NOT SET, RETURN

NOW TEST FOR LIGHT PEN TRIGGER

TEST AL,2 ; TEST LIGHT PEN TRIGGER

JZ V7 ; RETURN WITHOUT RESETTING TRIGGER

TRIGGER HAS BEEN SET, READ THE VALUE IN

MOV AH,14 ; LIGHT PEN REGISTERS ON 6845

IN AL,DX ; GET STATUS REGISTER

TEST AL,4 ; TEST LIGHT PEN SWITCH

JNZ V6 ; NOT SET, RETURN

NOW TEST FOR LIGHT PEN TRIGGER

TEST AL,2 ; TEST LIGHT PEN TRIGGER

JZ V7 ; RETURN WITHOUT RESETTING TRIGGER

INPUT REGS POINTED TO BY AH, AND CONVERT TO ROW COLUMN IN DX

MOV DX,ADDR_6545 ; ADDRESS REGISTER FOR 6545
MOV AL, AH  ; Register to Read
OUT DX, AL  ; SET IT UP
INC DX  ; DATA REGISTER
IN AL, DX  ; GET THE VALUE
MOV CH, AL  ; SAVE IN CX
DEC DX  ; ADDRESS REGISTER
INC AH  ; SECOND DATA REGISTER
OUT DX, AL  ; POINT TO DATA REGISTER
IN AL, DX  ; GET SECOND DATA VALUE
MOV AH, CH  ; AX HAS INPUT VALUE

i----- AX HAS THE VALUE READ IN FROM THE 6845

MOV BL, CRT_MODE  ; Mode value to BX
MOV BL, CS: [0BH]  ; Determine amount to subtract
MOV AX, AX  ; Take it away
MOV AL, [0CH]  ; Convert to correct page origin
MOV AH, CH  ; AX has input value

i----- Determine mode of operation

V2:  ; Determine_MODE
MOV CL, 1  ; SET #6 SHIFT COUNT
CMP CRT_MODE, 4  ; Determine if graphics or Alpha
JE V4  ; ALPHA_PEN

; ------ Graphics mode
MOV DL,  04  ; DIVISOR FOR GRAPHICS
MOV DL,  80  ; DETERMINE ROW COLUMN VALUE
MOV DL,  AH  ; ROWS TO DH
MOV AH, AL  ; COLUMNS TO DL
ADD AL, CL  ; Multiply rows, 8
MOV CH, AL  ; GET RASTER VALUE TO RETURN REG
XOR BH, BH  ; COLUMN VALUE TIMES 4 FOR HIGH RES
SAL BX, CL  ; COLUMN VALUE TIMES 2 FOR MEDIUM RES
SHL DX, CL  ; COLUMN VALUE TIMES 1 FOR LOW RES

i----- Determine Graphics Row Position

; ------ Determine ALPHA CHAR POSITION
MOV DL, AH  ; COLUMN VALUE FOR RETURN
MOV DL, AL  ; ROW VALUE
MOV DL, DH  ; DIVIDE BY 4
MOV DL, DH  ; FOR VALUE IN 0-24 RANGE
SAL SHORT V5  ; LIGHT_PEN_RETURN_SET

i----- ALPHA_MODE ON LIGHT PEN

MOV AH, 1  ; LIGHT_PEN_RETURN_SET
MOV AH, 0  ; INDICATE EVERYTHING SET
MOV AH, 1  ; LIGHT_PEN RETURN
MOV AH, 0  ; SAVE RETURN VALUE (IN CASE)
MOV AX, [0CH]  ; GET BASE ADDRESS
ADD DX, 7  ; POINT TO RESET PARAMETER
LaC OBJ
LINE SOURCE
F838 5A
F859
F839 5F
F83A 5E
F83B 5F
F83C 1F
F83D 1F
F83E 1F
F83F 07
F840 CF

READ_LPEN ENDP

1 --- INT 12 -------------------------------

; MEMORY_SIZE_DETERMINE
; 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

MEMORY_SIZE_DETERMINE PROC FAR

STI ; INTERRUPTS BACK ON

PUSH DS ; SAVE SEGMENT

MOV AX,DATA ; ESTABLISH ADDRESSING

IF

POP OS ; RECOVER SEGMENT

CF IRET ; RETURN TO CALLER

MEMORY_SIZE_DETERMINE ENDP

;--- INT 11 ----------------------------------

; EQUIPMENT DETERMINATION
; THIS ROUTINE ATTEMPTS TO DETERMINE WHAT OPTIONAL
; DEVICES ARE ATTACHED TO THE SYSTEM.

; INPUT
; NO REGISTERS
; THE EQUIP_FLAG VARIABLE IS SET DURING THE POWER ON DIAGNOSTICS
; USING THE FOLLOWING HARDWARE ASSUMPTIONS:
; PORT 60 = LOW ORDER BYTE Of EQUIPMENT
; PORT 3FA = INTERRUPT REGISTER OF 8250
; BITS 7-3 ARE ALWAYS 0
; PORT 376 = OUTPUT PORT Of PRINTER -- 6255 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
; 00=1, 01=2, 10=3, 11=4 ONLY IF BIT 0 = 1
; BIT 5,4 = INITIAL VIDEO MODE
; 00 = UNUSED
; 01 - 40X25 BW USING COLOR CARD
; 10 - 80X25 BW USING COLOR CARD
; 11 - 80X25 BW USING BW CARD
; BIT 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

MEMORY_SIZE_DETERMINE PROC FAR

ASSUME CS:CODE,DS:DATA

MEMORY_SIZE_DETERMINE PROC FAR
EQUIPMENT

PROC FAR

; INTERRUPTS BACK ON
STI

; SAVE SEGMENT REGISTER
PUSH DS

; ESTABLISH ADDRESSING
MOV AX, DS

; GET THE CURRENT SETTINGS
MOV AX, EQUIP_FLAG

; RECOVER SEGMENT
POP DS

; RETURN TO CALLER
IRET

ENDP

;--- INT 15 ---------------------------------

; PURPOSE:
; TO CALL APPROPRIATE ROUTINE DEPENDING ON REG AH

; 0 MOTOR ON
; 1 MOTOR OFF
; 2 READ CASSETTE BLOCK
; 3 WRITE CASSETTE BLOCK

; TURN ON MOTOR?
OR AH, AH

; YES, DO IT
JZ MOTOR_ON

; TURN OFF MOTOR?
DEC AH

; YES, DO IT
JZ MOTOR_OFF

; READ CASSETTE BLOCK?
DEC AH

; YES, DO IT
JZ READ_BLOCK

; WRITE CASSETTE BLOCK?
; NOT_DEFINED
JNZ WRITE_BLOCK

; YES, DO IT
JZ WRITE_BLOCK

; COMMAND NOT DEFINED
MOV AH, 080H

; ERROR, UNDEFINED OPERATION
JMP ERROR

; ERROR FLAG
Halt

ENDP
```assembly
F885 5043 MOTOR_ON PROC NEAR
F885 5044 ; PURPOSE:
F885 5045 ; TO TURN ON CASSETTE MOTOR
F885 5046 ; ----------------- .. ------------------
F885 5047 ; PURPOSE:
F885 5048 ; TO TURN OFF CASSETTE MOTOR
F885 5049 ; ----------------- .. ------------------
F885 5050 IN AL,PORT_B ; READ CASSETTE OUTPUT
F885 5051 OR AL,00H ; SET BIT TO TURN ON MOTOR
F885 5052 JMP W3 ; RETURN IT TRUE
F885 5053 RET
F885 5054 READ_BLOCK PROC NEAR
F885 5055 ; PURPOSE:
F885 5056 ; TO READ 1 OR MORE 256 BYTE BLOCKS FROM CASSETTE
F885 5057 ; ON ENTRY:
F885 5058 ; ES IS SEGMENT FOR MEMORY BUFFER (FOR COMPACT CODE)
F885 5059 ; BX POINTS TO START OF MEMORY BUFFER
F885 5060 ; CX CONTAINS NUMBER OF BYTES TO READ
F885 5061 ; ON EXIT:
F885 5062 ; BX POINTS 1 BYTE PAST LAST BYTE PUT IN MEM
F885 5063 ; CX CONTAINS DECREMENTED BYTE COUNT
F885 5064 ; Ox CONTAINS NUMBER OF BYTES ACTUALLY READ
F885 5065 ; -------------------- .. -----------------
F885 5066 PUSH BX ; SAVE BX
F885 5067 PUSH CX ; SAVE CX
F885 5068 MOV SI,7 ; SET UP RETRY COUNT FOR LEADER
F885 5069 CALL BEGIN_OP ; BEGIN BY STARTING MOTOR
F885 5070 IN AL,PORT_C ; GET INITIAL VALUE
F885 5071 ANO AL,00H ; MASK OFF EXTRANEOUS BITS
F885 5072 MOV DX,16250 ; N OF TRANSITIONS TO LOOK FOR
F885 5073 W5: TEST BIOS.BREAK, 80H ; CHECK FOR BREAK KEY
F885 5074 JZ W6 ; JUMP IF NO BREAK KEY
F885 5075 JMP W17 ; JUMP IF BREAK KEY HIT
F885 5076 W6: DEC DX ; INCLUDE THE LAST EDGE
F885 5077 JMP W17 ; JUMP IF BREAK KEY DETECTED
F885 5078 JMP W17 ; JUMP IF BREAK KEY HIT
F885 5079 DEC CX ; CHECK FOR TRANSITION
F885 5080 .POP CX ; RESTORE ONE BIT COUNTER
F885 5081 JZ W4 ; JUMP IF NO TRANSITION
F885 5082 CMP DX,BX ; CHECK PULSE WIDTH
F885 5083 JCXZ W3 ; JUMP IF NO ERROR DETECTED
F885 5084 MOV DX,16250 ; N OF TRANSITIONS TO LOOK FOR
F885 5085 W3: READ BLOCK PROC NEAR
F885 5086 ; PURPOSE:
F885 5087 ; TO READ 1 OR MORE 256 BYTE BLOCKS FROM CASSETTE
F885 5088 ; ON ENTRY:
F885 5089 ; ES IS SEGMENT FOR MEMORY BUFFER (FOR COMPACT CODE)
F885 5090 ; BX POINTS TO START OF MEMORY BUFFER
F885 5091 ; CX CONTAINS NUMBER OF BYTES TO READ
F885 5092 ; ON EXIT:
F885 5093 ; BX POINTS 1 BYTE PAST LAST BYTE PUT IN MEM
F885 5094 ; CX CONTAINS DECREMENTED BYTE COUNT
F885 5095 ; Ox CONTAINS NUMBER OF BYTES ACTUALLY READ
F885 5096 ; -------------------- .. -----------------
```

IF ex::o THEN WE CAN LOOK

IF QR SYNC BIT (0)

JHC W4 JUMP IF ZERO BIT (NOT GOOD LEADER)

Loop ex AND READ ANOTHER HALF ONE BIT

ED 72 Eb JUMP IF ONE BIT (STILL LEADER)

FIND-SYNC

ED 72 Eb SKIP OTHER HALF OF SYNC BIT (0)

CALL READ_HALF_BIT

READ SYN CHRACTER:

CALL READ_BYTE

CMP AL, 16H SYNCHRONIZATION CHARACTER

JHE W16 JUMP IF BAD LEADER FOUND.

POP AL; RESTORE REGS

READ 1 OR MORE 256 BYTE BLOCKS FROM CASSETTE

ON ENTRY:

ES IS SEGMENT FOR MEMORY BUFFER (FOR COMPACT CODE)

BX POINTS TO START OF MEMORY BUFFER

CX CONTAINS NUMBER OF BYTES TO READ

ON EXIT:

BX POINTS 1 BYTE PAST LAST BYTE PUT IN MEMORY

CX CONTAINS DECREMENTED BYTE COUNT

DX CONTAINS NUMBER OF BYTES ACTUALLY READ

PUSH CX;SAVE BYTE COUNT

COME HERE BEFORE EACH 256 BYTE BLOCK IS READ

MOV CRC_REG, OFFFFH ;INIT CRC REG

MOV DX, 256 ;SET DX TO DATA BLOCK SIZE

TEST BIDS_BREAK, 80H , CHECK FOR BREAK KEY

JHZ W13 JUMP IF BREAK KEY HIT

CALL READ_BYTE ;READ BYTE FROM CASSETTE

JC X1 ;IF WE'VE ALREADY REACHED END OF MEMORY BUFFER

MOV ES:BX, Al ;STORE DATA BYTE AT BYTE PTR

INC BX ;INC BUFFER PTR

DEC CX ;DEC BYTE COUNTER

RD_BLK

JG W11 RD_BLK

CALL READ_BYTE ;NOW READ HID CRC BYTES

SUB AH, AH ;CLEAR AH

CMP CRC_REG, IDOFH IS THE CRC CORRECT

JNE W14 IF NOT EQUAL CRC IS BAD

JCXZ W15 IF BYTE COUNT IS ZERO

WE HAVE READ ENOUGH

JMP WID STILL MORE, SO READ ANOTHER BLOCK

MOV AH, 02H ;SET AH=02 TO INDICATE DATA TIMEOUT

TEST 03H

JtiZ W18 JUMP IF ERROR DETECTED

CALL READ_BYTE ;READ TRAILER

SKIP TO TURN OFF MOTOR

BAD-LEADER
<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>LINE</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>F936</td>
<td>7403</td>
<td>5194</td>
<td>JZ W17</td>
</tr>
<tr>
<td>F938</td>
<td>E962FF</td>
<td>5195</td>
<td>JMP W4</td>
</tr>
<tr>
<td>F938</td>
<td>5196</td>
<td>W17:</td>
<td>NO VALID DATA FOUND</td>
</tr>
<tr>
<td>F938</td>
<td>E962FF</td>
<td>5197</td>
<td>IF TOO MANY RETRIES</td>
</tr>
<tr>
<td>F938</td>
<td>5198</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F938</td>
<td>5199</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F93E</td>
<td>59</td>
<td>5200</td>
<td>POP CX</td>
</tr>
<tr>
<td>F93D</td>
<td>58</td>
<td>5201</td>
<td>POP BX</td>
</tr>
<tr>
<td>F93C</td>
<td>59</td>
<td>5202</td>
<td>MOV DX,DX</td>
</tr>
<tr>
<td>F940</td>
<td>8404</td>
<td>5203</td>
<td>MOV AH,04H</td>
</tr>
<tr>
<td>F942</td>
<td>50</td>
<td>5204</td>
<td>PUSH AX</td>
</tr>
<tr>
<td>F943</td>
<td>W18:</td>
<td>5205</td>
<td>NOT-OFF</td>
</tr>
<tr>
<td>F943</td>
<td>E962FF</td>
<td>5206</td>
<td>JMP W4</td>
</tr>
<tr>
<td>F944</td>
<td>E421</td>
<td>5207</td>
<td>IN AL,021H</td>
</tr>
<tr>
<td>F945</td>
<td>408</td>
<td>5208</td>
<td>OUT 021H,AL</td>
</tr>
<tr>
<td>F949</td>
<td>8042FF</td>
<td>5209</td>
<td>CALL MOTOR_OFF</td>
</tr>
<tr>
<td>F94C</td>
<td>58</td>
<td>5210</td>
<td>POP AX</td>
</tr>
<tr>
<td>F94D</td>
<td>80FC00</td>
<td>5211</td>
<td>CALL MOTOR_OFF</td>
</tr>
<tr>
<td>F94F</td>
<td>53</td>
<td>5212</td>
<td>PUSH AX</td>
</tr>
<tr>
<td>F950</td>
<td>5213</td>
<td>PUSH BX</td>
<td></td>
</tr>
<tr>
<td>F952</td>
<td>5214</td>
<td>POP CX</td>
<td></td>
</tr>
<tr>
<td>F953</td>
<td>5215</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F956</td>
<td>5216</td>
<td>PROC NEAR</td>
<td></td>
</tr>
<tr>
<td>F956</td>
<td>PURPOSE:</td>
<td>5217</td>
<td></td>
</tr>
<tr>
<td>F956</td>
<td>TO READ A BYTE FROM CASSETTE</td>
<td>5218</td>
<td></td>
</tr>
<tr>
<td>F956</td>
<td>ON EXIT REG AL CONTAINS READ DATA BYTE</td>
<td>5219</td>
<td></td>
</tr>
<tr>
<td>F957</td>
<td>5220</td>
<td>PUSH BX</td>
<td></td>
</tr>
<tr>
<td>F958</td>
<td>5221</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F959</td>
<td>5222</td>
<td>PUSH CX</td>
<td></td>
</tr>
<tr>
<td>F959</td>
<td>5223</td>
<td>PUSH CX</td>
<td></td>
</tr>
<tr>
<td>F95A</td>
<td>408</td>
<td>5224</td>
<td>MOV CL,8H</td>
</tr>
<tr>
<td>F956</td>
<td>W19:</td>
<td>5225</td>
<td>BYTE-ASH</td>
</tr>
<tr>
<td>F956</td>
<td>5226</td>
<td>PUSH CX</td>
<td></td>
</tr>
<tr>
<td>F957</td>
<td>5227</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F957</td>
<td>5228</td>
<td>PROC NEAR</td>
<td></td>
</tr>
<tr>
<td>F957</td>
<td>PURPOSE:</td>
<td>5229</td>
<td></td>
</tr>
<tr>
<td>F957</td>
<td>TO READ A BYTE FROM CASSETTE</td>
<td>5230</td>
<td></td>
</tr>
<tr>
<td>F957</td>
<td>ON EXIT REG AL CONTAINS READ DATA BYTE</td>
<td>5231</td>
<td></td>
</tr>
<tr>
<td>F957</td>
<td>CALL READ_HALF_BIT</td>
<td>5232</td>
<td>READ ONE PULSE</td>
</tr>
<tr>
<td>F957</td>
<td>JCXZ W11</td>
<td>5233</td>
<td>IF CX=0 THEN TIMEOUT</td>
</tr>
<tr>
<td>F957</td>
<td>JZ W17</td>
<td>5234</td>
<td>BECAUSE OF NO DATA TRANSITIONS</td>
</tr>
<tr>
<td>F957</td>
<td>5235</td>
<td>PUSH BX</td>
<td></td>
</tr>
<tr>
<td>F957</td>
<td>5236</td>
<td>SAVE 1ST HALF BIT'S</td>
<td></td>
</tr>
<tr>
<td>F957</td>
<td>5237</td>
<td>PULSE WIDTH (IN BX)</td>
<td></td>
</tr>
<tr>
<td>F957</td>
<td>CALL READ_HALF_BIT</td>
<td>5238</td>
<td>READ COMPLEMENTARY PULSE</td>
</tr>
<tr>
<td>F957</td>
<td>POP AX</td>
<td>5239</td>
<td>COMPUTE DATA BIT</td>
</tr>
<tr>
<td>F957</td>
<td>5240</td>
<td>COMPUTE DATA BIT</td>
<td></td>
</tr>
<tr>
<td>F957</td>
<td>JCXZ W11</td>
<td>5241</td>
<td>IF C0 0 THEN TIMEOUT DUE TO</td>
</tr>
<tr>
<td>F957</td>
<td>CALL READ_HALF_BIT</td>
<td>5242</td>
<td>NO DATA TRANSITIONS</td>
</tr>
<tr>
<td>F957</td>
<td>ADD BX,AX</td>
<td>5243</td>
<td></td>
</tr>
<tr>
<td>F957</td>
<td>JNZ W19</td>
<td>5244</td>
<td></td>
</tr>
<tr>
<td>F957</td>
<td>CMP BX,06F0H</td>
<td>5245</td>
<td>COUNT FOR ZERO BIT</td>
</tr>
<tr>
<td>F957</td>
<td>5246</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F957</td>
<td>CMP BX,06F0H</td>
<td>5247</td>
<td></td>
</tr>
<tr>
<td>F957</td>
<td>5248</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F957</td>
<td>CMP BX,06F0H</td>
<td>5249</td>
<td></td>
</tr>
<tr>
<td>F957</td>
<td>5250</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F957</td>
<td>CMP BX,06F0H</td>
<td>5251</td>
<td></td>
</tr>
<tr>
<td>F957</td>
<td>5252</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F957</td>
<td>5253</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F957</td>
<td>5254</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F957</td>
<td>5255</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F957</td>
<td>5256</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F957</td>
<td>5257</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F957</td>
<td>5258</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F957</td>
<td>5259</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F957</td>
<td>5260</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F957</td>
<td>5261</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F957</td>
<td>5262</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F957</td>
<td>5263</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F957</td>
<td>5264</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F957</td>
<td>5265</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F957</td>
<td>5266</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F957</td>
<td>5267</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A-71
F97E E0F9
5268 JMP M20 ; P_OV_EX
5249 READ_BYTE ENDP
5270

F996
5271 READ_HALF_BIT PROC NEAR
5272 ; PURPOSE:
5273 ; TO COMPUTE TIME TILL NEXT DATA
5274 ; TRANSITION (EDGE)
5275
5276 ; ON ENTRY:
5277 ; EDGE_CNT CONTAINS LAST EDGE COUNT
5278
5279 ; ON EXIT:
5280 ; AX CONTAINS OLD LAST EDGE COUNT
5281 BX CONTAINS PULSE WIDTH
5282
5283 MOV CX, 100 ; SET TIME TO WAIT FOR BIT
5284 MOV AH, LAST_VAL ; GET PRESENT INPUT VALUE
5285 M22: ; RD-H-BIT
5286 IN AL, PORT_C ; INPUT DATA BIT
5287 AND AL, 01H ; MASK OFF EXTRANEOUS BITS
5288 CMP AL, 0 ; SAME AS BEFORE?
5289 LOOP M22 ; LOOP TILL IT CHANGES
5290 MOV LAST_VAL, AL ; UPDATE LAST_VAL WITH NEW VALUE
5291 MOV AL, 0 ; READ TIMER'S COUNTER
5292 OUT TIM_CTL, AL ; LATCH COUNTER
5293 MOV AL, OB6H ; SET UP TIMER -- MODE 3 SQUARE WAVE
5294 OUT TIM_CTL, AL
5295 MOV EDGE_CNT, AX ; BX GETS LAST EDGE COUNT
5296 MOV CX,Edge_CNT ; UPDATE EDGE COUNT;
5297 RET

F9A8
5298 PUSH BX
5299 PUSIt CX
5300 IN AL, PORT_B ;DISABLE SPEAKER
5301 AND AL, NOT 02H
5302 OR AL, 01H ; ENABLE TIMER
5303 OUT PORT_B, AL
5304 MOV AL, OB6H ; SET UP TIMER -- MODE 3 SQUARE WAVE
5305 OUT TIM_CTL, AL
5306 CALL BEGIN_OP ; START MOTOR AND DELAY
5307 MOV AX, 1184 ; SET NORMAL BIT SIZE
5308 CALL WRITE_BLOCK ; WRITE LEADER BYTE COUNT
5309 CALL WRITE_BYTE ; WRITE LEADER
5310 CALL WRITE_BYTE ; WRITE SYM CHARACTER
I WRITE 1 OR MORE 256 BYTE BLOCKS TO CASSETTE.

ENTRY:

1. AX POINTS TO MEMORY BUFFER ADDRESS
   2. EX CONTAINS NUMBER OF BYTES TO WRITE

ON EXIT:

1. BX POINTS 1 Byte Past Last Byte Written to CASSETTE
   2. EX IS ZERO

; WR_BLOCK:

F9D3 C7066900fFFF  ; INIT CRC
F9D9 BFG00001 ; FOR 256 BYTES
F9DC E04fFFF  ; WR_BLK
F9D0 E04fFFF  ; READ BYTE FROM MEM
F9DD E03500  ; WRITE BYTE TO CASSETTE
F9E2 2f6b  ; JUMP IF NOT DONE YET
F9E4 2f6b  ; SKP-ADV
F9E4 2f6b  ; LOOP TILL 256 BYTE BLOCK IS WRITTEN TO TAPE
F9E9 A16900  ; WRITE THE ONE'S COMPLEMENT OF THE:
F9E9 A16900  ; 2ND BYTE CRC TO TAPE
F9E8 2f6b  ; SAVE IT
F9EE F7D0  ; FOR 1'S COMPLEMENT
F9EE 2f6b  ; DATA BIT WRITTEN FIRST
F9E9 E02a00  ; DISASSEMBLE THE DATA BIT
F9E9 E02a00  ; MS BIT INTO CARRY
FA06 080f  ; READ DATA BITS IN BYTE.
FA06 080f  ; DISASSEMBLE THE DATA BIT
FA07 8a0f  ; TURN MOTOR OFF
FA07 8a0f  ; TURN TIMER OFF
FA08 8a0f  ; NO ERRORS REPORTED ON WRITE OP
FA16 C3  ; FINISHED

WRITE_BLOCK ENDP

WRITE_BYTE PROC NEAR

; WRITE A BYTE TO CASSETTE.
; WRITE TO WRITE IS IN REG AL.

FA17 57  ; SAVE REGS CX.AX
FA17 56  ; WRITE TO WRITE IS IN REG AL.

FA17 51  ; SAVE REGS CX.AX
FA19 0A4b  ; AL=BYTE TO WRITE.
FA1b B108  ; FOR 8 DATA BITS IN BYTE.
FA1b B108  ; NOTE: TWO EDGES PER BIT
FA1d 82a9  ; DISASSEMBLE THE DATA BIT
FA1d 82a9  ; MS BIT INTO CARRY
FA1f 9c  ; SAVE FLAGS.
FA20 82b600  ; WRITE DATA BIT
FA23 9c  ; RESTORE CARRY FOR CRC CALC
FA27 D000  ; CRC_GEN
FA27 D000  ; LOP TILL ALL 8 BITS DONE
FA29 75f2  ; JUMP IF NOT DONE YET
FA2b 5b  ; RESTORE REG AX.CX
FA2c 59  ; RESTORE REG AX.CX

A-73
FA2E C3
5415 RET
5416 WRITE_BYTE ENDP
5417 ; WE ARE FINISHED

FAZE
5418 WRITE_BIT PROC NEAR
5419 ; PURPOSE:
5420 ; TO WRITE A DATA BIT TO CASSETTE
5421 ; CARRY FLAG CONTAINS DATA BIT
5422 ; I.E. IF SET DATA BIT IS A ONE
5423 ; IF CLEAR DATA BIT IS A ZERO
5424 ;
5425 ; NOTE: TWO EDGES ARE WRITTEN PER BIT
5426 ; ONE BIT HAS 500 USEC BETWEEN EDGES
5427 ; FOR A 1000 USEC PERIOD (1 MILLISEC)
5428 ; ZERO BIT HAS 250 USEC BETWEEN EDGES
5429 ; FOR A 500 USEC PERIOD (.5 MILLISEC)
5430 ; CARRY FLAG IS DATA BIT
5431 ;
5432 ; ASSUME IT'S A '1'
5433 ; SET AX TO NOMINAL ONE SIZE
5434 MOV AX,1104
5435 JC W26
5436 MOV AX,592
5437 ; WRITE-BIT-AX
5438 MOV AX,592
5439 PUSH AX
5440 JC W26
5441 IN AL,PORT_C
5442 AND AL,020H
5443 JL W29
5444 LOOP TILL HIGH
5445 IN AL,PORT_C
5446 AND AL,020H
5447 JNZ W30
5448 ; RELOAD TIMER WITH PERIOD FOR NEXT DATA BIT
5449 POP AX
5450 ; RESTORE PERIOD COUNT
5451 MOV AL, AH
5452 OUT 042H, AL
5453 MOV AL, AH
5454 OUT 042H, AL
5455 RET
5456 ;
5457 ; UPDATE CRC REGISTER WITH NEXT DATA BIT
5458 ; CRC IS USED TO DETECT READ ERRORS
5459 ; ASSUMES DATA BIT IS IN CARRY
5460 ; REG AX IS MODIFIED
5461 ; FLAGS ARE MODIFIED
5462 ;
5463 CRe_Gen PROC NEAR
5464 ; UPDATE CRC REGISTER WITH NEXT DATA BIT
5465 ; CRC IS USED TO DETECT READ ERRORS
5466 ; ASSUMES DATA BIT IS IN CARRY
5467 ; REG AX IS MODIFIED
5468 ; FLAGS ARE MODIFIED
5469 ;
5470 MOV AX,CRC_REG
5471 R
5472 MOV AX,CRC_REG
5473 RCR AX,1
5474 RCL AX,1
5475 CLC
5476 JNO W32
5477 ; CRe REG BIT IS ONE
5478 XOR AX,0810H
5479 ; THEN XOR CRC REG WITH OH
5480 ; SET CARRY
5481 W32:
5482 RCL AX,1
5483 ; ROTATE CARRY (DATA BIT)
5484 ; INTO CRC REG
5485 MOV CRC_REG,AX
5486 R
5487 CRC_Gen ENDP
5488 ;
5489 BEGIN_OP PROC NEAR
5490 ; START TAPE AND DELAY
5491 ;
5641 |--- INT IA -------------------------------
5642 | TIME_OF_DAY
5643 | THIS ROUTINE ALLOWS THE CLOCK TO BE SET/READ
5644 |
5645 | INPUT
5646 | (AH) = 0 READ THE CURRENT CLOCK SETTING
5647 | RETURNS CX = HIGH PORTION OF COUNT
5648 | DX = LOW PORTION OF COUNT
5649 | AL = 0 IF TIMER HAS NOT PASSED 24 HOURS SINCE LAST READ
5650 | <>0 IF ON ANOTHER DAY
5651 | (AH) = 1 SET THE CURRENT CLOCK
5652 | CX = HIGH PORTION OF COUNT
5653 | DX = LOW PORTION OF COUNT
5654 | NOTE: COUNTS OCCUR AT THE RATE OF 1193180/65536 COUNTS/SEC
5655 | (OR ABOUT 18.2 PER SECOND -- SEE EQUATES BELOW)
5656 |
5657 | --------------------------------------
5658 | Assumes CS: CODE, DS: DATA
5659 | TIME_OF_DAY PROC FAR
5660 | READ_TIME
5661 | SET_TIME
5662 |
5663 | STI ; INTERRUPTS BACK ON
5664 | PUSH DS ; SAVE SEGMENT
5665 | PUSH AX ; SAVE PARM
5666 |
5667 | MOVB AX, Data
5668 | MOV BX, DS
5669 |
5670 | OR AH, AH ; AH=0
5671 | JZ T2 ; READ_TIME
5672 |
5673 | JNZ T3 ; SET_TIME
5674 |
5675 | STI ; INTERRUPTS BACK ON
5676 | POP DS ; RECOVER SEGMENT
5677 | IRET ; RETURN TO CALLER
5678 |
5679 | Timer_OFl
5680 | MOV Timer_OFl
5681 |
5682 | MOV Timer_High
5683 |
5684 | MOV Timer_Low
5685 |
5686 | MOV Timer_High
5687 |
5688 | T1: ; TOO_RETURN
5689 |
5690 | Timer_INT PROC
5691 | STI ; INTERRUPTS BACK ON
5692 | PUSH DS ; SAVE SEGMENT
5693 | PUSH AX ; SAVE PARM
5694 |
5695 | MOV AX, Data
5696 |
5697 | OR AH, AH ; AH=0
5698 | JZ T4 ; TOO_RETURN
5699 |
5700 | JNZ T5 ; TOO_RETURN
5701 |
5702 | Timer_OFl
5703 |
5704 | MOV Timer_OFl
5705 |
5706 | MOV Timer_High
5707 |
5708 | MOV Timer_Low
5709 |
5710 | MOV AX, Data
5711 |
5712 | JNZ T5 ; TOO_RETURN
5713 |
5714 | T4: ; TOO_RETURN
5715 |
LOC OBJ    LINE SOURCE

FE88 0336E0018 R 5715  CMP TIMER_HIGH,018H ; TEST FOR COUNT EQUIVALLING 24 HOURS
FE8B 7519 R 5716  JLZ T5 ; DISKETTE_CTL
FE8F 836E0000 R 5717  CMP TIMER_LOW,000H
FE95 7511 R 5718  JLZ T5 ; DISKETTE_CTL

5719

I------- TIMER HAS GONE 24 HOURS

FE97 C706E00000 R 5720  MOV TIMER_HIGH,0
FE9D C706E00000 R 5721  MOV TIMER_LOW,0
FE9F C606700000 R 5722  MOV TIMER_OFL,1

5723

I------- TEST FOR DISKETTE TIME OUT

FE9B 5724  T5: ; DISKETTE_CTL
FE9D 5725  DEC MOTOR_COUNT
FEB0 B50 R 5726  JNZ T6 ; RETURN IF COUNT NOT OUT
FEBE 8026F00F R 5727  AND MOTOR_STATUS,0F0H ; TURN OFF MOTOR RUNNING BITS
FEC3 850C R 5728  MOV AL,0CH
FEE5 BAF203 R 5729  MOV DX,03F2H
FEE7 5730  OUT DX,AL ; TURN OFF THE MOTOR
FEE9 5731  T6: ; TIMER_RET:
FEEO 5732  MOV AL,EOI
FEEF CD1C R 5733  INTO 1CH ; TRANSFER CONTROL TO A USER ROUTINE
FEB0 B520 R 5734  MOV AL,EOI
FEBE 5735  OUT 020H,AL ; END OF INTERRUPT TO 8259
FEEF 5736  POP AX
FEEF 5737  POP DS ; SET MACHINE STATE
FEF8 000000000 R 5738  JIRET ; RETURN FROM INTERRUPT

5740

; THESE ARE THE VECTORS WHICH ARE MOVED INTO
; THE 8086 INTERRUPT AREA DURING POWER ON

5742

; VECTOR_TABLE LABEL WORD ; VECTOR TABLE FOR MOVE TO INTERRUPTS

5744

FEF3 5745  ASH VECTOR_TABLE
FEF9 5746  ASH OFFSET TIMER_INT ; INTERRUPT 8
FEF9 5747  ASH OFFSET KB_INT ; INTERRUPT 9
FEF9 5748  ASH OFFSET KB_INT ; INTERRUPT 9
FEF9 5749  ASH OFFSET KB_INT ; INTERRUPT 9
FEF9 5750  ASH OFFSET KB_INT ; INTERRUPT 9
FEF9 5751  ASH OFFSET KB_INT ; INTERRUPT 9
FEF9 5752  ASH OFFSET KB_INT ; INTERRUPT 9
FEF9 5753  ASH OFFSET KB_INT ; INTERRUPT 9
FEF9 5754  ASH OFFSET KB_INT ; INTERRUPT 9
FEF9 5755  ASH OFFSET KB_INT ; INTERRUPT 9
FEF9 5756  ASH OFFSET KB_INT ; INTERRUPT 9
FEF9 5757  ASH OFFSET KB_INT ; INTERRUPT 9
FEF9 5758  ASH OFFSET KB_INT ; INTERRUPT 9
FEF9 5759  ASH OFFSET KB_INT ; INTERRUPT 9
FEF9 5760  ASH OFFSET KB_INT ; INTERRUPT 9
FEF9 5761  ASH OFFSET KB_INT ; INTERRUPT 9
FEF9 5762  ASH OFFSET KB_INT ; INTERRUPT 9
FEF9 5763  ASH OFFSET KB_INT ; INTERRUPT 9
FEF9 5764  ASH OFFSET KB_INT ; INTERRUPT 9
FEF9 5765  ASH OFFSET KB_INT ; INTERRUPT 9
FEF9 5766  ASH OFFSET KB_INT ; INTERRUPT 9
FEF9 5767  ASH OFFSET KB_INT ; INTERRUPT 9
FEF9 5768  ASH OFFSET KB_INT ; INTERRUPT 9
FEF9 5769  ASH OFFSET KB_INT ; INTERRUPT 9
FEF9 5770  ASH OFFSET KB_INT ; INTERRUPT 9
FEF9 5771  ASH OFFSET KB_INT ; INTERRUPT 9
FEF9 5772  ASH OFFSET KB_INT ; INTERRUPT 9
FEF9 5773  ASH OFFSET KB_INT ; INTERRUPT 9
FEF9 5774  ASH OFFSET KB_INT ; INTERRUPT 9
FEF9 5775  ASH OFFSET KB_INT ; INTERRUPT 9
FEF9 5776  ASH OFFSET KB_INT ; INTERRUPT 9
FEF9 5777  ASH OFFSET KB_INT ; INTERRUPT 9
FEF9 5778  ASH OFFSET KB_INT ; INTERRUPT 9
FEF9 5779  ASH OFFSET KB_INT ; INTERRUPT 9
FEF9 5780  ASH OFFSET KB_INT ; INTERRUPT 9
FEF9 5781  ASH OFFSET KB_INT ; INTERRUPT 9
FEF9 5782  ASH OFFSET KB_INT ; INTERRUPT 9
FEF9 5783  ASH OFFSET KB_INT ; INTERRUPT 9
FEF9 5784  ASH OFFSET KB_INT ; INTERRUPT 9
FEF9 5785  ASH OFFSET KB_INT ; INTERRUPT 9
FEF9 5786  ASH OFFSET KB_INT ; INTERRUPT 9
FEF9 5787  ASH OFFSET KB_INT ; INTERRUPT 9
FEF9 5788  ASH OFFSET KB_INT ; INTERRUPT 9
FEF9 5789  ASH OFFSET KB_INT ; INTERRUPT 9
FEF9 5790  ASH OFFSET KB_INT ; INTERRUPT 9

A-78
LOC OBJ  | LINE  | SOURCE
---------|-------|--------
FF33 0000 | 5791  | DN 00000H ; INTERRUPT 1BH
FF35 00F3 | 5792  | DN 0F600H ; ROM BASIC ENTRY POINT
FF37 1266 | 5793  |
FF39 00F0 | 5794  | DN OFFSET BOOT_STRAP ; INTERRUPT 1BH
FF39 00F0 | 5795  | DN CODE
FF3B 00F6 | 5796  | DN TIME_OF_DAY ; INTERRUPT 1AH -- TIME OF DAY
FF3F 53F7 | 5797  | DN DUMMY_RETURN ; INTERRUPT 1BM -- KEYBOARD BREAK ADDR
FF41 00F0 | 5798  | DN CODE
FF43 00F0 | 5799  | DN CODE
FF45 00F0 | 5800  | DN VIDEO_PARMS ; INTERRUPT 10 -- VIDEO PARAMETERS
FF47 00F0 | 5801  | DN CODE
FF48 00F0 | 5802  | DN CODE
FF4A 00000000 | 5803  | DUMMY_RETURN:
FF53 CF | 5804  | IRET ; DUMMY RETURN FOR BREAK FROM KEYBOARD
FF54 | 5805  | --- INT 5 ----------------------------
FF54 | 5806  | --- THIS LOGIC WILL BE INVOKED BY INTERRUPT 05H TO PRINT
FF54 | 5807  | --- THE SCREEN, THE CURSOR POSITION AT THE TIME THIS ROUTINE
FF54 | 5808  | --- IS INVOKED WILL BE SAVED AND RESTORED UPON COMPLETION. THE
FF54 | 5809  | --- ROUTINE IS INTENDED TO RUN WITH INTERRUPTS ENABLED.
FF54 | 5810  | --- IF A SUBSEQUENT 'PRINT SCREEN' KEY IS DEPRESSED DURING THE
FF54 | 5811  | --- TIME THIS ROUTINE IS PRINTING IT WILL BE IGNORED.
FF54 | 5812  | --- ADDRESS 50:0 CONTAINS THE STATUS OF THE PRINT SCREEN:
FF54 | 5813  | --- 0:0 EITHER PRINT SCREEN HAS NOT BEEN CALLED
FF54 | 5814  | --- OR UPON RETURN FROM A CALL THIS INDICATES
FF54 | 5815  | --- A SUCCESSFUL OPERATION.
FF54 | 5816  | --- +1 PRINT SCREEN IS IN PROGRESS
FF54 | 5817  | --- =377 ERROR ENCOUNTERED DURING PRINTING
FF54 | 5818  | --- I-------------------------------
FF54 | 5819  | --- ASSUME CS:CODE,DS:XXDATA
FF54 | 5820  | --- PRINT_SCREEN PROC FAR
FF54 | 5821  | --- STI
FF54 | 5822  | FF54 0B FB | 5823  | PUSH DS
FF54 | 5824  | FF54 1E | 5825  | PUSH AX
FF54 | 5826  | FF54 50 | 5827  | PUSH BX
FF54 | 5828  | FF54 53 | 5829  | PUSH CX
FF54 | 5830  | FF54 51 | 5831  | PUSH DX
FF54 | 5832  | FF54 550000 | 5833  | MOV AX,XXDATA INDEX 50
FF54 | 5834  | FF54 0800 | 5835  | MOV DS,AX
FF54 | 5836  | FF54 003FOO00 | 5837  | MOV AH,1 ; SEE IF PRINT ALREADY IN PROGRESS
FF54 | 5838  | FF54 74F5 | 5839  | JZ EXIT ; JUMP IF PRINT ALREADY IN PROGRESS
FF54 | 5840  | FF54 06000000 | 5841  | MOV STATUS_BYTE,1 ; INDICATE PRINT NOW IN PROGRESS
FF54 | 5842  | FF54 B40F | 5843  | MOV AH,15 ; REQUEST THE CURRENT SCREEN MODE
FF54 | 5844  | FF54 0D10 | 5845  | INT 10H ; [AH]=MODE
FF54 | 5846  | FF54 09 | 5847  | XOR OX,OX ; SET CURSOR POSITION TO (0,0)
FF54 | 5848  | FF54 0A | 5849  | MOV CL,AH ; WILL MAKE USE OF IX REGISTER TO
FF54 | 5850  | FF54 0519 | 5851  | MOV CM,23 ; CONTROL ROW & COLUMNS
FF54 | 5852  | FF54 05500 | 5853  | CALL CRWF ; CARRY RETURN LINE FEED ROUTINE
FF54 | 5854  | FF54 51 | 5855  | PUSH CX ; SAVE SCREEN BOUNDS
FF54 | 5856  | FF54 8403 | 5857  | MOV AH,3 ; WILL NOW READ THE CURSOR,
FF54 | 5858  | FF54 0D10 | 5859  | INT 10H ; AND PRESERVE THE POSITION
FF54 | 5860  | FF54 89 | 5861  | POP CX ; RECALL SCREEN BOUNDS
FF54 | 5862  | FF54 52 | 5863  | PUSH DX ; PRECALL [OH]-VISUAL PAGE
FF54 | 5864  | FF54 3302 | 5865  | XOR DX,DX ; WILL SET CURSOR POSITION TO (0,0)
THE LOOP FROM PRI10 TO THE INSTRUCTION PRIOR TO PA120
IS THE LOOP TO READ EACH CURSOR POSITION FROM THE SCREEN
AND PRINT.

PRI10: MOV AH,2  ;TO INDICATE CURSOR SET REQUEST

PRI2: MOV AH,0  ;TO INDICATE READ CHARACTER

INT 10H  ;FIRST CURSOR POSITION ESTABLISHED

INT 10H  ;SECOND CURSOR POSITION ESTABLISHED

JNZ PRI10  ;JUMP IF NOT PROCEED

PRI10: MOV AH,2  ;TO INDICATE CURSOR SET REQUEST

JMP NEAR CRLF  ;EXIT THE ROUTINE

POWER ON RESET VECTOR

0000 EA580000FF  JMP RESET

0005 30342F32342F38  DB '04/24/81'  ;RELEASE MARKER

CRLF PROC NEAR

CRLF  DB 5,13  ;PRINT LF, CR

CRLF ENDP
Notes For The BIOS Listing

1. The wait loop for the printer times out on form feed of > 51 lines. - line ref (3069)

2. Mode controls for the 320 x 200 video have Color/BW reversed. - line ref (3338)

3. The RS232 Timeout is 80 decimal, not 80 hexadecimal. - line ref (1566)

4. The Base Pointer register is destroyed by some video calls.

5. D_04 ◇ character in the character generator has 08 as it’s last value, S/80. - line ref (5511)

6. If you hit print screen in the Color/Graphics 80x25 Character Mode, the screen may not display during the print cycle.
Appendix B. 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>BX:</td>
<td>BH</td>
<td>BL</td>
<td>BASE</td>
</tr>
<tr>
<td>CX:</td>
<td>CH</td>
<td>CL</td>
<td>COUNT</td>
</tr>
<tr>
<td>DX:</td>
<td>DH</td>
<td>DL</td>
<td>DATA</td>
</tr>
<tr>
<td></td>
<td>SP</td>
<td></td>
<td>STACK POINTER</td>
</tr>
<tr>
<td></td>
<td>BP</td>
<td></td>
<td>BASE POINTER</td>
</tr>
<tr>
<td></td>
<td>SI</td>
<td></td>
<td>SOURCE INDEX</td>
</tr>
<tr>
<td></td>
<td>DI</td>
<td></td>
<td>DESTINATION INDEX</td>
</tr>
<tr>
<td>IP:</td>
<td></td>
<td></td>
<td>INSTRUCTION POINTER</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>STATUS FLAGS</td>
</tr>
<tr>
<td></td>
<td>CS</td>
<td></td>
<td>CODE SEGMENT</td>
</tr>
<tr>
<td></td>
<td>DS</td>
<td></td>
<td>DATA SEGMENT</td>
</tr>
<tr>
<td></td>
<td>SS</td>
<td></td>
<td>STACK SEGMENT</td>
</tr>
<tr>
<td></td>
<td>ES</td>
<td></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:

\[
\begin{array}{cccccccccccc}
15 & 7 & 0 \\
X & X & X & X & OF & DF & IF & TF & SF & ZF & X & AF & X & PF & X & CF \\
\end{array}
\]

X = Don't Care

**8080 FLAGS**

- AF: AUXILIARY CARRY - BCD
- CF: CARRY FLAG
- PF: PARITY FLAG
- SF: SIGN FLAG
- ZF: ZERO FLAG

**8088 FLAGS**

- DF: DIRECTION FLAG (STRINGS)
- IF: INTERRUPT ENABLE FLAG
- OF: OVERFLOW FLAG (CF ⊕ SF)
- TF: TRAP - SINGLE STEP FLAG
OPERAND SUMMARY

"reg" field Bit Assignments:

<table>
<thead>
<tr>
<th>16-Bit (w=1)</th>
<th>8-Bit (w=0)</th>
<th>Segment</th>
</tr>
</thead>
<tbody>
<tr>
<td>000 AX</td>
<td>000 AL</td>
<td>00 ES</td>
</tr>
<tr>
<td>001 CX</td>
<td>001 CL</td>
<td>01 CS</td>
</tr>
<tr>
<td>010 DX</td>
<td>010 DL</td>
<td>10 SS</td>
</tr>
<tr>
<td>011 BX</td>
<td>011 BL</td>
<td>11 DS</td>
</tr>
<tr>
<td>100 SP</td>
<td>100 AH</td>
<td></td>
</tr>
<tr>
<td>101 BP</td>
<td>101 CH</td>
<td></td>
</tr>
<tr>
<td>110 SI</td>
<td>110 DH</td>
<td></td>
</tr>
<tr>
<td>111 DI</td>
<td>111 BH</td>
<td></td>
</tr>
</tbody>
</table>

SECOND INSTRUCTION BYTE SUMMARY

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

<table>
<thead>
<tr>
<th>r/m</th>
<th>Operand Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>(BX) + (SI) + DISP</td>
</tr>
<tr>
<td>001</td>
<td>(BX) + (DI) + DISP</td>
</tr>
<tr>
<td>010</td>
<td>(BP) + (SI) + DISP</td>
</tr>
<tr>
<td>011</td>
<td>(BP) + (DI) + DISP</td>
</tr>
<tr>
<td>100</td>
<td>(SI) + DISP</td>
</tr>
<tr>
<td>101</td>
<td>(DI) + DISP</td>
</tr>
<tr>
<td>110</td>
<td>(BP) + DISP*</td>
</tr>
<tr>
<td>111</td>
<td>(BX) + DISP</td>
</tr>
</tbody>
</table>

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

LOGICAL MEMORY SPACE

OFFSET ADDRESS

SELECTED SEGMENT REGISTER

CS, SS, DS, ES OR NONE FOR I/O, INT

DISPLACEMENT

ADDRES

0

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 incl. strings)</td>
<td>DS</td>
<td>ES, SS, or CS</td>
</tr>
<tr>
<td>SI (implicit source addr for strings)</td>
<td>DS</td>
<td>ES, SS, or CS</td>
</tr>
<tr>
<td>DI (implicit dest addr for strings)</td>
<td>ES</td>
<td>Never</td>
</tr>
</tbody>
</table>
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 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

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

Register with accumulator

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

IN = Input to AL/A.X from
Fixed port

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

Variable port (DX)

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

OUT = Output from AL/A.X to
Fixed port

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

Variable port (DX)

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

XLAT = Translate byte to AL

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

LEA = Load EA to register

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

LDS = Load pointer to DS

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

LES = Load pointer to ES

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

LAHF = Load AH with flags

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

SAHF = Store AH into flags

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

PUSHF = Push flags

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

POPF = Pop flags

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

B-6
ARITHMETIC

**ADD** = Add
Reg./memory with register to either

\[
\begin{array}{cccc}
0 & 0 & 0 & 0 \\
\hline
\text{mod} & \text{reg} & \text{r/m}
\end{array}
\]

Immediate to register/memory

\[
\begin{array}{cccc}
1 & 0 & 0 & 0 \\
\hline
\text{mod} & 0 & 0 & \text{r/m} \\
data
\end{array}
\]
data if s:w=01

Immediate to accumulator

\[
\begin{array}{cccc}
0 & 0 & 0 & 1 \\
\hline
\text{data}
\end{array}
\]
data if w=1

**ADC** = Add with carry
Reg./memory with register to either

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

Immediate to register/memory

\[
\begin{array}{cccc}
1 & 0 & 0 & 0 \\
\hline
\text{mod} & 0 & 1 & \text{r/m} \\
data
\end{array}
\]
data if s:w=01

Immediate to accumulator

\[
\begin{array}{cccc}
0 & 0 & 1 & 0 \\
\hline
\text{data}
\end{array}
\]
data if w=1

**INC** = Increment
Register/memory

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

Register

\[
\begin{array}{cccc}
0 & 1 & 0 & 0 \\
\hline
\text{reg}
\end{array}
\]

**AAA** = ASCII adjust for add

\[
\begin{array}{cccc}
0 & 0 & 1 & 1 \\
\hline
\text{111}
\end{array}
\]

**DAA** = Decimal adjust for add

\[
\begin{array}{cccc}
0 & 0 & 1 & 0 \\
\hline
\text{1111}
\end{array}
\]

**SUB** = Subtract
Reg./memory and register to either

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

Immediate from register/memory

\[
\begin{array}{cccc}
1 & 0 & 0 & 0 \\
\hline
\text{mod} & 1 & 0 & \text{r/m} \\
data
\end{array}
\]
data if s:w=01

Immediate from accumulator

\[
\begin{array}{cccc}
0 & 0 & 1 & 0 \\
\hline
\text{data}
\end{array}
\]
data if w=1
SBB = Subtract with borrow
Reg./memory and register to either

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

Immediate from register/memory

\[ \begin{array}{cccc}
1 & 0 & 0 & 0 \ 0 \ 0 \ s \ w \\
\ \ \ \ \ \ \ mod \ 0 \ 1 \ 1 \ r/m \ \ \ \ \ data \ \ \ \ \ \ data \ if \ s:w=01
\end{array} \]

Immediate from accumulator

\[ \begin{array}{cccc}
0 & 0 & 0 & 1 \ 1 \ 1 \ 0 \ w \\
\ \ \ \ \ \ \ data \ \ \ \ \ \ data \ if \ w=1
\end{array} \]

DEC = Decrement
Register/memory

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

Register

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

NEG = Change sign

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

CMP = Compare
Register/memory and register

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

Immediate with register/memory

\[ \begin{array}{cccc}
1 & 0 & 0 & 0 \ 0 \ 0 \ s \ w \\
\ \ \ \ \ \ \ mod \ 1 \ 1 \ 1 \ r/m \ \ \ \ \ data \ \ \ \ \ \ data \ if \ s:w=01
\end{array} \]

Immediate with accumulator

\[ \begin{array}{cccc}
0 & 0 & 1 & 1 \ 1 \ 1 \ 1 \ 0 \ w \\
\ \ \ \ \ \ \ data \ \ \ \ \ \ data \ if \ w=1
\end{array} \]

AAS = ASCII adjust for subtract

\[ \begin{array}{cccc}
0 & 0 & 1 & 1 \ 1 \ 1 \ 1 \ 1
\end{array} \]

DAS = Decimal adjust for subtract

\[ \begin{array}{cccc}
0 & 0 & 1 & 0 \ 1 \ 1 \ 1 \ 1
\end{array} \]

MUL = Multiply (unsigned)

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

IMUL = Integer multiply (signed)

\[ \begin{array}{cccc}
1 & 1 & 1 & 1 \ 0 \ 1 \ 1 \ 1 \ w \\
\ \ \ \ \ \ \ mod \ 1 \ 0 \ 1 \ r/m
\end{array} \]
AAM = ASCII adjust for multiply

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

DIV = Divide (unsigned)

```
1 1 1 1 0 1 1 w \mod 1 1 0 r/m
```

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
Reg./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 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
Reg./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 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
Reg./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 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 |

B-10
STRING MANIPULATION

**REP** = Repeat

1 1 1 1 0 0 1 z

**MOVS** = Move String

1 0 1 0 0 1 0 w

**CMPS** = Compare String

1 0 1 0 0 1 1 w

**SCAS** = Scan String

1 0 1 0 1 1 1 w

**LODS** = Load String

1 0 1 0 1 1 0 w

**STOS** = Store String

1 0 1 0 1 0 1 w

CONTROL TRANSFER

**CALL** = Call

Direct within segment

1 1 1 0 1 0 0 0 disp-low disp-high

Indirect within segment

1 1 1 1 1 1 1 1 mod 0 1 0 r/m

Direct intersegment

1 0 0 1 1 0 1 0 offset-low offset-high

seg-low seg-high

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

```
11111111 mod 100 r/m
```

Direct intersegment

```
11101010 offset-low  offset-high
seg-low        seg-high
```

Indirect intersegment

```
11111111 mod 101 r/m
```

**RET = Return from CALL**

Within segment

```
110000011
```

Within seg. adding immed to SP

```
11000010 data-low data-high
```

Intersegment

```
11001011
```

Intersegment, adding immediate to SP

```
11001010 data-low data-high
```

**JE/JZ = Jump on equal/zero**

```
01110100 disp
```

**JL/JNGE = Jump on less/not greater or equal**

```
01111100 disp
```

**JLE/JNG = Jump on less or equal/not greater**

```
01111110 disp
```

**JB/JNAE = Jump on below/not above or equal**

```
01111010 disp
```

**JBE/JNA = Jump on below or equal/not above**

```
01110110 disp
```

**JP/JPE = Jump on parity/parity even**

```
01111010 disp
```

**JO = Jump on overflow**

```
01110000 disp
```

B-12
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
0 1 1 1 1 1 0 1 disp

JNLE/JG = Jump on not less or equal/greater
0 1 1 1 1 1 1 1 disp

JNB/JAE = Jump on not below/above or equal
0 1 1 1 0 0 1 1 disp

JNBE/JA = Jump on not below or equal/above
0 1 1 1 0 1 1 1 disp

JNP/JPO = Jump on not parity/parity odd
0 1 1 1 1 0 1 1 disp

JNO = Jump on not overflow
0 1 1 1 0 0 0 1 disp

JNS = Jump on not sign
0 1 1 1 1 0 0 1 disp

LOOP = Loop CX times
1 1 1 0 0 0 1 0 disp

LOOPZ/LOOPE = Loop while zero/equal
1 1 1 0 0 0 0 1 disp

LOOPNZ/LOOPNE = Loop while not zero/not equal
1 1 1 0 0 0 0 0 disp

JCXZ = Jump on CX zero
1 1 1 0 0 0 1 1 disp
### 8088 CONDITIONAL TRANSFER OPERATIONS

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Condition</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>JE or JZ</td>
<td>ZF = 1</td>
<td>&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>(SP xor OF) or ZF = 1</td>
<td>&quot;less or equal&quot; or &quot;not greater&quot;</td>
</tr>
<tr>
<td>JB or JNAE</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&quot; or &quot;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</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 3
```
1 1 0 0 1 1 0 1
```

**INTO** = Interrupt on overflow
```
1 1 0 0 1 1 1 0
```

**IRET** = Interrupt return
```
1 1 0 0 1 1 1 1
```

### PROCESSOR CONTROL

<table>
<thead>
<tr>
<th>CLC = Clear carry</th>
<th>STC = Set carry</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 1 1 1 1 0 0</td>
<td>1 1 1 1 1 0 0 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CMC = Complement carry</th>
<th>NOP = No operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 1 1 0 1 0 1</td>
<td>1 0 0 1 0 0 0 0</td>
</tr>
</tbody>
</table>

B-14
CLD = Clear direction
1 1 1 1 1 1 0 0

STD = Set direction
1 1 1 1 1 1 0 1

CLI = Clear interrupt
1 1 1 1 1 1 0 1 0

STI = Set interrupt
1 1 1 1 1 1 0 1 1

HLT = Halt
1 1 1 1 1 0 1 0 0

WAIT = Wait
1 0 0 1 1 1 0 1 1

LOCK = Bus lock prefix
1 1 1 1 0 0 0 0 0

ESC = Escape (to external device)
1 1 0 1 1 x x x mod x x x r/m

Footnotes:
if d = 1 then "to"; if d = 0 then "from"
if w = 1 then word instruction; if w = 0 then byte instruction
if s:w = 01 then 16 bits of immediate data from the operand
if s:w = 11 then an immediate data byte is sign extended to form the
16-bit operand
if v = 0 then "count" = 1; if v = 1 then "count" in (CL)
x = don’t care
z is used for some string primitives to compare with ZF FLAG
AL = 8-bit accumulator
AX = 16-bit accumulator
CX = Count register
DS = Data segment
DX = Variable port register
ES = Extra segment
Above/below refers to unsigned value
Greater = more positive;
Less = less positive (more negative) signed values
<table>
<thead>
<tr>
<th>LO</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ADD</td>
<td>ADD</td>
<td>ADD</td>
<td>ADD</td>
<td>ADD</td>
<td>Add</td>
<td>PUSH</td>
<td>POP</td>
</tr>
<tr>
<td></td>
<td>b.t/r/m</td>
<td>w.f.r/m</td>
<td>b.t/r/m</td>
<td>w.f.r/m</td>
<td>b.t/r/m</td>
<td>w.f.r/m</td>
<td>w.t/r/m</td>
<td>w.t/r/m</td>
</tr>
<tr>
<td>1</td>
<td>ADC</td>
<td>ADC</td>
<td>ADC</td>
<td>ADC</td>
<td>ADC</td>
<td>Add</td>
<td>PUSH</td>
<td>POP</td>
</tr>
<tr>
<td></td>
<td>b.t/r/m</td>
<td>w.f.r/m</td>
<td>b.t/r/m</td>
<td>w.f.r/m</td>
<td>b.t/r/m</td>
<td>w.f.r/m</td>
<td>w.t/r/m</td>
<td>w.t/r/m</td>
</tr>
<tr>
<td>2</td>
<td>AND</td>
<td>AND</td>
<td>AND</td>
<td>AND</td>
<td>AND</td>
<td>Add</td>
<td>PUSH</td>
<td>SEG</td>
</tr>
<tr>
<td></td>
<td>b.t/r/m</td>
<td>w.f.r/m</td>
<td>b.t/r/m</td>
<td>w.f.r/m</td>
<td>b.t/r/m</td>
<td>w.f.r/m</td>
<td>w.t/r/m</td>
<td>w.t/r/m</td>
</tr>
<tr>
<td>3</td>
<td>XOR</td>
<td>XOR</td>
<td>XOR</td>
<td>XOR</td>
<td>XOR</td>
<td>Add</td>
<td>SEG</td>
<td>AAA</td>
</tr>
<tr>
<td></td>
<td>b.t/r/m</td>
<td>w.f.r/m</td>
<td>b.t/r/m</td>
<td>w.f.r/m</td>
<td>b.t/r/m</td>
<td>w.f.r/m</td>
<td>w.t/r/m</td>
<td>w.t/r/m</td>
</tr>
<tr>
<td>4</td>
<td>INC</td>
<td>INC</td>
<td>INC</td>
<td>INC</td>
<td>INC</td>
<td>Add</td>
<td>INC</td>
<td>INC</td>
</tr>
<tr>
<td></td>
<td>AX</td>
<td>CX</td>
<td>DX</td>
<td>EX</td>
<td>SP</td>
<td>BP</td>
<td>SI</td>
<td>DI</td>
</tr>
<tr>
<td>5</td>
<td>PUSH</td>
<td>PUSH</td>
<td>PUSH</td>
<td>PUSH</td>
<td>PUSH</td>
<td>Add</td>
<td>PUSH</td>
<td>PUSH</td>
</tr>
<tr>
<td>AX</td>
<td>CX</td>
<td>DX</td>
<td>BX</td>
<td>SP</td>
<td>BP</td>
<td>SI</td>
<td>DI</td>
<td>DI</td>
</tr>
<tr>
<td>6</td>
<td>JO</td>
<td>JNO</td>
<td>JB/</td>
<td>JNB/</td>
<td>JE/</td>
<td>JNE/</td>
<td>JBE/</td>
<td>JNB/</td>
</tr>
<tr>
<td></td>
<td>JZ</td>
<td>JZ</td>
<td>JZ</td>
<td>JZ</td>
<td>JZ</td>
<td>JZ</td>
<td>JZ</td>
<td>JA</td>
</tr>
<tr>
<td>7</td>
<td>Immed</td>
<td>Immed</td>
<td>TEST</td>
<td>TEST</td>
<td>XCHG</td>
<td>XCHG</td>
<td>XCHG</td>
<td>XCHG</td>
</tr>
<tr>
<td></td>
<td>b.t/r/m</td>
<td>w.f.r/m</td>
<td>b.t/r/m</td>
<td>w.f.r/m</td>
<td>b.t/r/m</td>
<td>w.f.r/m</td>
<td>w.t/r/m</td>
<td>w.t/r/m</td>
</tr>
<tr>
<td>8</td>
<td>NOP</td>
<td>XCHG</td>
<td>XCHG</td>
<td>XCHG</td>
<td>XCHG</td>
<td>Add</td>
<td>Add</td>
<td>Add</td>
</tr>
<tr>
<td></td>
<td>AX</td>
<td>DX</td>
<td>BX</td>
<td>SP</td>
<td>BP</td>
<td>SI</td>
<td>DI</td>
<td>DI</td>
</tr>
<tr>
<td>9</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>m AL</td>
<td>m AX</td>
<td>AX m</td>
<td>AX m</td>
<td>AX m</td>
<td>AX m</td>
<td>AX m</td>
<td>AX m</td>
<td></td>
</tr>
<tr>
<td>10</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>i AL</td>
<td>i AL</td>
<td>i AL</td>
<td>i AL</td>
<td>i AL</td>
<td>i AL</td>
<td>i AL</td>
<td>i AL</td>
<td></td>
</tr>
<tr>
<td>11</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>(i+SP)</td>
<td>(i+SP)</td>
<td>(i+SP)</td>
<td>(i+SP)</td>
<td>(i+SP)</td>
<td>(i+SP)</td>
<td>(i+SP)</td>
<td>(i+SP)</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Shift</td>
<td>Shift</td>
<td>Shift</td>
<td>Shift</td>
<td>Shift</td>
<td>Shift</td>
<td>Shift</td>
<td>Shift</td>
</tr>
<tr>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
</tr>
<tr>
<td>13</td>
<td>LOOPNZ/</td>
<td>LOOPZ/</td>
<td>LOOP</td>
<td>LOOP</td>
<td>LOOP</td>
<td>LOOP</td>
<td>LOOP</td>
<td>LOOP</td>
</tr>
<tr>
<td>LOOPNE</td>
<td>LOOPNE</td>
<td>LOOPNE</td>
<td>LOOPNE</td>
<td>LOOPNE</td>
<td>LOOPNE</td>
<td>LOOPNE</td>
<td>LOOPNE</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>LOCK</td>
<td>LOCK</td>
<td>LOCK</td>
<td>LOCK</td>
<td>LOCK</td>
<td>LOCK</td>
<td>LOCK</td>
<td>LOCK</td>
</tr>
<tr>
<td></td>
<td>REP</td>
<td>REP</td>
<td>REP</td>
<td>REP</td>
<td>REP</td>
<td>REP</td>
<td>REP</td>
<td>REP</td>
</tr>
</tbody>
</table>

b = byte operation  
d = direct  
f = from CPU reg  
i = immediate  
ia = imm. to accum.  
id = indirect  
is = imm. 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
## 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</td>
<td>OR</td>
<td>OR</td>
<td>OR</td>
<td>OR</td>
<td>OR</td>
<td>PUSH</td>
<td>CS</td>
</tr>
<tr>
<td></td>
<td>b.f.r/m</td>
<td>w.f.r/m</td>
<td>b.f.r/m</td>
<td>w.f.r/m</td>
<td>b.t,r/m</td>
<td>w.t,r/m</td>
<td>b.i</td>
<td>w.i</td>
</tr>
<tr>
<td>1</td>
<td>SBB</td>
<td>SBB</td>
<td>SBB</td>
<td>SBB</td>
<td>SBB</td>
<td>SBB</td>
<td>PUSH</td>
<td>POP</td>
</tr>
<tr>
<td></td>
<td>b.f.r/m</td>
<td>w.f.r/m</td>
<td>b.t,r/m</td>
<td>w.t,r/m</td>
<td>b.i</td>
<td>w.i</td>
<td>b.t,r/m</td>
<td>w.t,r/m</td>
</tr>
<tr>
<td>2</td>
<td>SUB</td>
<td>SUB</td>
<td>SUB</td>
<td>SUB</td>
<td>SUB</td>
<td>SUB</td>
<td>SEG</td>
<td>DAS</td>
</tr>
<tr>
<td></td>
<td>b.f.r/m</td>
<td>w.f.r/m</td>
<td>b.t,r/m</td>
<td>w.t,r/m</td>
<td>b.i</td>
<td>w.i</td>
<td>b.t,r/m</td>
<td>w.t,r/m</td>
</tr>
<tr>
<td>3</td>
<td>CMP</td>
<td>CMP</td>
<td>CMP</td>
<td>CMP</td>
<td>CMP</td>
<td>CMP</td>
<td>SEG</td>
<td>AAS</td>
</tr>
<tr>
<td></td>
<td>b.f.r/m</td>
<td>w.f.r/m</td>
<td>b.t,r/m</td>
<td>w.t,r/m</td>
<td>b.i</td>
<td>w.i</td>
<td>b.t,r/m</td>
<td>w.t,r/m</td>
</tr>
<tr>
<td>4</td>
<td>DEC</td>
<td>DEC</td>
<td>DEC</td>
<td>DEC</td>
<td>DEC</td>
<td>DEC</td>
<td>DEC</td>
<td>DEC</td>
</tr>
<tr>
<td></td>
<td>AX</td>
<td>CX</td>
<td>DX</td>
<td>BX</td>
<td>SP</td>
<td>BP</td>
<td>SI</td>
<td>DI</td>
</tr>
<tr>
<td>5</td>
<td>POP</td>
<td>POP</td>
<td>POP</td>
<td>POP</td>
<td>POP</td>
<td>POP</td>
<td>SI</td>
<td>POP</td>
</tr>
<tr>
<td></td>
<td>AX</td>
<td>CX</td>
<td>DX</td>
<td>BX</td>
<td>SP</td>
<td>BP</td>
<td>SI</td>
<td>DI</td>
</tr>
<tr>
<td>6</td>
<td>JS</td>
<td>JNS</td>
<td>JNP/</td>
<td>JPO</td>
<td>JL/</td>
<td>JNL/</td>
<td>JLE/</td>
<td>JNG</td>
</tr>
<tr>
<td></td>
<td>JPE</td>
<td>JPO</td>
<td>JNGE</td>
<td>JGE</td>
<td>JNGE</td>
<td>JLE/</td>
<td>JNG</td>
<td>JG</td>
</tr>
<tr>
<td>7</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></td>
<td>b.f.r/m</td>
<td>w.f.r/m</td>
<td>b.t,r/m</td>
<td>w.t,r/m</td>
<td>b.i</td>
<td>w.t,r/m</td>
<td>b.i</td>
<td>w.t,r/m</td>
</tr>
<tr>
<td>8</td>
<td>CBW</td>
<td>CWD</td>
<td>CALL</td>
<td>WAIT</td>
<td>PUSH</td>
<td>POPF</td>
<td>SAHF</td>
<td>LAHF</td>
</tr>
<tr>
<td></td>
<td>l.d</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>TEST</td>
<td>TEST</td>
<td>STOS</td>
<td>STOS</td>
<td>LODS</td>
<td>LODS</td>
<td>SCAS</td>
<td>SCAS</td>
</tr>
<tr>
<td></td>
<td>b.i</td>
<td>w.i</td>
<td>b</td>
<td>w</td>
<td>b</td>
<td>w</td>
<td>b</td>
<td>w</td>
</tr>
<tr>
<td>A</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></td>
<td>i→AX</td>
<td>i→AX</td>
<td>i→DX</td>
<td>i→DX</td>
<td>i→BX</td>
<td>i→BX</td>
<td>i→SP</td>
<td>i→BP</td>
</tr>
<tr>
<td>B</td>
<td>RET</td>
<td>RET</td>
<td>INT</td>
<td>INT</td>
<td>INT</td>
<td>INTO</td>
<td>INTO</td>
<td>INTO</td>
</tr>
<tr>
<td></td>
<td>l,(i+SP)</td>
<td>l</td>
<td>Type 3</td>
<td>(Any)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>ESC</td>
<td>ESC</td>
<td>ESC</td>
<td>ESC</td>
<td>ESC</td>
<td>ESC</td>
<td>ESC</td>
<td>ESC</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>D</td>
<td>CALL</td>
<td>JMP</td>
<td>JMP</td>
<td>JMP</td>
<td>JMP</td>
<td>JMP</td>
<td>JMP</td>
<td>JMP</td>
</tr>
<tr>
<td></td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>i.d</td>
<td>i.d</td>
<td>i.d</td>
<td>i.d</td>
<td>i.d</td>
</tr>
<tr>
<td>E</td>
<td>CLC</td>
<td>STC</td>
<td>CLI</td>
<td>STI</td>
<td>CLD</td>
<td>STD</td>
<td>Grp 2</td>
<td>Grp 2</td>
</tr>
<tr>
<td></td>
<td>000</td>
<td>001</td>
<td>010</td>
<td>011</td>
<td>100</td>
<td>101</td>
<td>100</td>
<td>111</td>
</tr>
<tr>
<td>F</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**where**

- **mod r/m**
- **Immed**
- **Shift**
- **Grp 1**
- **Grp 2**
<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAA</td>
<td>6</td>
</tr>
<tr>
<td>AAD</td>
<td>8</td>
</tr>
<tr>
<td>AAM</td>
<td>8</td>
</tr>
<tr>
<td>AAS</td>
<td>7</td>
</tr>
<tr>
<td>ADC</td>
<td>6</td>
</tr>
<tr>
<td>ADD</td>
<td>6</td>
</tr>
<tr>
<td>AND</td>
<td>9</td>
</tr>
<tr>
<td>CALL</td>
<td>10</td>
</tr>
<tr>
<td>CBW</td>
<td>8</td>
</tr>
<tr>
<td>CLC</td>
<td>13</td>
</tr>
<tr>
<td>CLD</td>
<td>14</td>
</tr>
<tr>
<td>CLI</td>
<td>14</td>
</tr>
<tr>
<td>CMC</td>
<td>13</td>
</tr>
<tr>
<td>CMP</td>
<td>7</td>
</tr>
<tr>
<td>CMPS</td>
<td>10</td>
</tr>
<tr>
<td>CWD</td>
<td>8</td>
</tr>
<tr>
<td>DAA</td>
<td>6</td>
</tr>
<tr>
<td>DAS</td>
<td>7</td>
</tr>
<tr>
<td>DEC</td>
<td>7</td>
</tr>
<tr>
<td>DIV</td>
<td>8</td>
</tr>
<tr>
<td>ESC</td>
<td>14</td>
</tr>
<tr>
<td>HLT</td>
<td>14</td>
</tr>
<tr>
<td>IDIV</td>
<td>8</td>
</tr>
<tr>
<td>IMUL</td>
<td>7</td>
</tr>
<tr>
<td>IN</td>
<td>5</td>
</tr>
<tr>
<td>INC</td>
<td>6</td>
</tr>
<tr>
<td>INT</td>
<td>13</td>
</tr>
<tr>
<td>INTO</td>
<td>13</td>
</tr>
<tr>
<td>IRET</td>
<td>13</td>
</tr>
<tr>
<td>JA</td>
<td>12</td>
</tr>
<tr>
<td>JAE</td>
<td>12</td>
</tr>
<tr>
<td>JB</td>
<td>11</td>
</tr>
<tr>
<td>JBE</td>
<td>11</td>
</tr>
<tr>
<td>JCXZ</td>
<td>12</td>
</tr>
<tr>
<td>JE</td>
<td>11</td>
</tr>
<tr>
<td>JG</td>
<td>12</td>
</tr>
<tr>
<td>JGE</td>
<td>12</td>
</tr>
<tr>
<td>JL</td>
<td>11</td>
</tr>
<tr>
<td>JLE</td>
<td>11</td>
</tr>
<tr>
<td>JMP</td>
<td>10</td>
</tr>
<tr>
<td>JNA</td>
<td>11</td>
</tr>
<tr>
<td>JNAE</td>
<td>11</td>
</tr>
<tr>
<td>JNB</td>
<td>12</td>
</tr>
<tr>
<td>JNBE</td>
<td>12</td>
</tr>
<tr>
<td>JNE</td>
<td>12</td>
</tr>
<tr>
<td>JNG</td>
<td>11</td>
</tr>
<tr>
<td>JNGE</td>
<td>11</td>
</tr>
<tr>
<td>JNL</td>
<td>12</td>
</tr>
<tr>
<td>JNLE</td>
<td>12</td>
</tr>
<tr>
<td>JNO</td>
<td>12</td>
</tr>
<tr>
<td>JNP</td>
<td>12</td>
</tr>
<tr>
<td>JNS</td>
<td>12</td>
</tr>
<tr>
<td>JNZ</td>
<td>12</td>
</tr>
<tr>
<td>JO</td>
<td>11</td>
</tr>
<tr>
<td>JP</td>
<td>11</td>
</tr>
<tr>
<td>JPE</td>
<td>11</td>
</tr>
<tr>
<td>JPO</td>
<td>12</td>
</tr>
<tr>
<td>JS</td>
<td>12</td>
</tr>
<tr>
<td>JZ</td>
<td>11</td>
</tr>
<tr>
<td>LAHF</td>
<td>5</td>
</tr>
<tr>
<td>LDS</td>
<td>5</td>
</tr>
<tr>
<td>LEA</td>
<td>5</td>
</tr>
<tr>
<td>LES</td>
<td>5</td>
</tr>
<tr>
<td>LOCK</td>
<td>14</td>
</tr>
<tr>
<td>LODS</td>
<td>10</td>
</tr>
<tr>
<td>LOOP</td>
<td>12</td>
</tr>
<tr>
<td>LOOPB</td>
<td>12</td>
</tr>
<tr>
<td>LOOPNE</td>
<td>12</td>
</tr>
<tr>
<td>LOOPNZ</td>
<td>12</td>
</tr>
<tr>
<td>LOOPZ</td>
<td>12</td>
</tr>
<tr>
<td>MOV</td>
<td>4</td>
</tr>
<tr>
<td>MOVS</td>
<td>10</td>
</tr>
<tr>
<td>MUL</td>
<td>7</td>
</tr>
<tr>
<td>NEG</td>
<td>7</td>
</tr>
<tr>
<td>NOP</td>
<td>13</td>
</tr>
<tr>
<td>NOT</td>
<td>3</td>
</tr>
<tr>
<td>OR</td>
<td>9</td>
</tr>
<tr>
<td>OUT</td>
<td>5</td>
</tr>
<tr>
<td>POP</td>
<td>4</td>
</tr>
<tr>
<td>POPF</td>
<td>5</td>
</tr>
<tr>
<td>PUSH</td>
<td>4</td>
</tr>
<tr>
<td>PUSHF</td>
<td>5</td>
</tr>
<tr>
<td>RCL</td>
<td>8</td>
</tr>
<tr>
<td>RCR</td>
<td>8</td>
</tr>
<tr>
<td>REP</td>
<td>10</td>
</tr>
<tr>
<td>RET</td>
<td>11</td>
</tr>
<tr>
<td>ROL</td>
<td>8</td>
</tr>
<tr>
<td>ROR</td>
<td>8</td>
</tr>
<tr>
<td>SAHF</td>
<td>5</td>
</tr>
<tr>
<td>SAL</td>
<td>8</td>
</tr>
<tr>
<td>SAR</td>
<td>8</td>
</tr>
<tr>
<td>SBB</td>
<td>7</td>
</tr>
<tr>
<td>SCAS</td>
<td>10</td>
</tr>
<tr>
<td>SHL</td>
<td>8</td>
</tr>
<tr>
<td>SHR</td>
<td>8</td>
</tr>
<tr>
<td>STC</td>
<td>13</td>
</tr>
<tr>
<td>STD</td>
<td>14</td>
</tr>
<tr>
<td>STI</td>
<td>14</td>
</tr>
<tr>
<td>STOS</td>
<td>10</td>
</tr>
<tr>
<td>SUB</td>
<td>6</td>
</tr>
<tr>
<td>TEST</td>
<td>9</td>
</tr>
<tr>
<td>WAIT</td>
<td>14</td>
</tr>
<tr>
<td>XCHG</td>
<td>5</td>
</tr>
<tr>
<td>XLAT</td>
<td>5</td>
</tr>
<tr>
<td>XOR</td>
<td>9</td>
</tr>
</tbody>
</table>

B-18
## Appendix C. Of Characters Keystrokes and Color

<table>
<thead>
<tr>
<th>VALUE</th>
<th>AS CHARACTERS</th>
<th>AS TEXT ATTRIBUTES</th>
<th>IBM MONOCHROME DISPLAY ADAPTER</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>COLOR/GRAPHICS MONITOR ADAPTER</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>BACKGROUND</td>
<td>FOREGROUND</td>
</tr>
<tr>
<td>HEX</td>
<td>DEC</td>
<td>SYMBOL</td>
<td>KEYSTROKES</td>
</tr>
<tr>
<td>00</td>
<td>0</td>
<td>BLANK</td>
<td>CTRL 2</td>
</tr>
<tr>
<td>01</td>
<td>1</td>
<td>☺</td>
<td>CTRL A</td>
</tr>
<tr>
<td>02</td>
<td>2</td>
<td>☻</td>
<td>CTRL B</td>
</tr>
<tr>
<td>03</td>
<td>3</td>
<td>⋆</td>
<td>CTRL C</td>
</tr>
<tr>
<td>04</td>
<td>4</td>
<td>♦</td>
<td>CTRL D</td>
</tr>
<tr>
<td>05</td>
<td>5</td>
<td>♣</td>
<td>CTRL E</td>
</tr>
<tr>
<td>06</td>
<td>6</td>
<td>♠</td>
<td>CTRL F</td>
</tr>
<tr>
<td>07</td>
<td>7</td>
<td>✪</td>
<td>CTRL G</td>
</tr>
<tr>
<td>08</td>
<td>8</td>
<td>⚫</td>
<td>CTRL H, BACKSPACE, SHIFT BACKSPACE</td>
</tr>
<tr>
<td>09</td>
<td>9</td>
<td>☀</td>
<td>CTRL I</td>
</tr>
<tr>
<td>0A</td>
<td>10</td>
<td>☾</td>
<td>CTRL J, CTRL ↓</td>
</tr>
<tr>
<td>0B</td>
<td>11</td>
<td>♂</td>
<td>CTRL K</td>
</tr>
<tr>
<td>0C</td>
<td>12</td>
<td>♀</td>
<td>CTRL L,</td>
</tr>
<tr>
<td>0D</td>
<td>13</td>
<td>♊</td>
<td>CTRL M,</td>
</tr>
<tr>
<td>0E</td>
<td>14</td>
<td>♋</td>
<td>CTRL N</td>
</tr>
<tr>
<td>0F</td>
<td>15</td>
<td>☋</td>
<td>CTRL O</td>
</tr>
<tr>
<td>10</td>
<td>16</td>
<td>→</td>
<td>CTRL P</td>
</tr>
<tr>
<td>11</td>
<td>17</td>
<td>←</td>
<td>CTRL Q</td>
</tr>
<tr>
<td>12</td>
<td>18</td>
<td>†</td>
<td>CTRL R</td>
</tr>
<tr>
<td>13</td>
<td>19</td>
<td>‼️</td>
<td>CTRL S</td>
</tr>
<tr>
<td>14</td>
<td>20</td>
<td>⌈</td>
<td>CTRL T</td>
</tr>
<tr>
<td>15</td>
<td>21</td>
<td>⎙</td>
<td>CTRL U</td>
</tr>
<tr>
<td>16</td>
<td>22</td>
<td>⬇️</td>
<td>CTRL V</td>
</tr>
<tr>
<td>17</td>
<td>23</td>
<td>⬇️</td>
<td>CTRL W</td>
</tr>
</tbody>
</table>

C-1
<table>
<thead>
<tr>
<th>VALUE</th>
<th>AS CHARACTERS</th>
<th>AS TEXT ATTRIBUTES</th>
<th>COLOR/GRAFICS</th>
<th>MONITOR ADAPTER</th>
<th>IBM MONOCHROME</th>
<th>DISPLAY ADAPTER</th>
</tr>
</thead>
<tbody>
<tr>
<td>HEX</td>
<td>DEC</td>
<td>SYMBOL</td>
<td>KEYSTROKES</td>
<td>MODES</td>
<td>BACKGROUND</td>
<td>FOREGROUND</td>
</tr>
<tr>
<td>18</td>
<td>24</td>
<td>↑</td>
<td>CTRL X</td>
<td></td>
<td>BLUE</td>
<td>DARK GREY</td>
</tr>
<tr>
<td>19</td>
<td>25</td>
<td>↓</td>
<td>CTRL Y</td>
<td></td>
<td>BLUE</td>
<td>LIGHT BLUE</td>
</tr>
<tr>
<td>1A</td>
<td>26</td>
<td>←</td>
<td>CTRL Z</td>
<td></td>
<td>BLUE</td>
<td>LIGHT GREEN</td>
</tr>
<tr>
<td>1B</td>
<td>26</td>
<td>←</td>
<td>CTRL [ , ESC, SHIFT ESC, CTRL ESC</td>
<td></td>
<td>BLUE</td>
<td>LIGHT CYAN</td>
</tr>
<tr>
<td>1C</td>
<td>28</td>
<td>↓</td>
<td>CTRL \</td>
<td></td>
<td>BLUE</td>
<td>LIGHT RED</td>
</tr>
<tr>
<td>1D</td>
<td>29</td>
<td>←</td>
<td>CTRL ]</td>
<td></td>
<td>BLUE</td>
<td>LIGHT MAGENTA</td>
</tr>
<tr>
<td>1E</td>
<td>30</td>
<td>▲</td>
<td>CTRL 6</td>
<td></td>
<td>BLUE</td>
<td>YELLOW</td>
</tr>
<tr>
<td>1F</td>
<td>31</td>
<td>▼</td>
<td>CTRL -</td>
<td></td>
<td>BLUE</td>
<td>WHITE</td>
</tr>
<tr>
<td>20</td>
<td>32</td>
<td>BLANK (SPACE</td>
<td>SPACE BAR, SHIFT SPACE, CTRL SPACE, ALT SPACE</td>
<td></td>
<td>GREEN</td>
<td>BLACK</td>
</tr>
<tr>
<td>21</td>
<td>33</td>
<td>! !</td>
<td>SHIFT</td>
<td></td>
<td>GREEN</td>
<td>BLUE</td>
</tr>
<tr>
<td>22</td>
<td>34</td>
<td>&quot; &quot;</td>
<td>SHIFT</td>
<td></td>
<td>GREEN</td>
<td>GREEN</td>
</tr>
<tr>
<td>23</td>
<td>35</td>
<td># #</td>
<td>SHIFT</td>
<td></td>
<td>GREEN</td>
<td>CYAN</td>
</tr>
<tr>
<td>24</td>
<td>36</td>
<td>$ $</td>
<td>SHIFT</td>
<td></td>
<td>GREEN</td>
<td>RED</td>
</tr>
<tr>
<td>25</td>
<td>37</td>
<td>% %</td>
<td>SHIFT</td>
<td></td>
<td>GREEN</td>
<td>MAGENTA</td>
</tr>
<tr>
<td>26</td>
<td>38</td>
<td>&amp; &amp;</td>
<td>SHIFT</td>
<td></td>
<td>GREEN</td>
<td>BROWN</td>
</tr>
<tr>
<td>27</td>
<td>39</td>
<td>' '</td>
<td>SHIFT</td>
<td></td>
<td>GREEN</td>
<td>LIGHT GREY</td>
</tr>
<tr>
<td>28</td>
<td>40</td>
<td>( (</td>
<td>SHIFT</td>
<td></td>
<td>GREEN</td>
<td>DARK GREY</td>
</tr>
<tr>
<td>29</td>
<td>41</td>
<td>) )</td>
<td>SHIFT</td>
<td></td>
<td>GREEN</td>
<td>LIGHT BLUE</td>
</tr>
<tr>
<td>2A</td>
<td>42</td>
<td>* *</td>
<td>NOTE 1</td>
<td></td>
<td>GREEN</td>
<td>LIGHT GREEN</td>
</tr>
<tr>
<td>2B</td>
<td>43</td>
<td>+ +</td>
<td>SHIFT</td>
<td></td>
<td>GREEN</td>
<td>LIGHT CYAN</td>
</tr>
<tr>
<td>2C</td>
<td>44</td>
<td>. .</td>
<td>GREEN</td>
<td>LIGHT RED</td>
<td>HIGH INTENSITY</td>
<td></td>
</tr>
<tr>
<td>2D</td>
<td>45</td>
<td>- -</td>
<td>GREEN</td>
<td>LIGHT MAGENTA</td>
<td>HIGH INTENSITY</td>
<td></td>
</tr>
<tr>
<td>2E</td>
<td>46</td>
<td>.</td>
<td>NOTE 2</td>
<td></td>
<td>GREEN</td>
<td>YELLOW</td>
</tr>
<tr>
<td>VALUE</td>
<td>AS CHARACTERS</td>
<td>COLOR/GRAPHICS MONITOR ADAPTER</td>
<td>IBM MONOCHROME DISPLAY ADAPTER</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>---------------</td>
<td>--------------------------------</td>
<td>--------------------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HEX</td>
<td>DEC</td>
<td>AS TEXT ATTRIBUTES</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SYMBOL KEYS</td>
<td>BACKGROUND</td>
<td>FOREGROUND</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2F 47</td>
<td>/ /</td>
<td>GREEN</td>
<td>WHITE</td>
<td>HIGH INTENSITY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30 48</td>
<td>0</td>
<td>CYAN</td>
<td>BLACK</td>
<td>NORMAL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>31 49</td>
<td>1</td>
<td>CYAN</td>
<td>BLUE</td>
<td>UNDERLINE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32 50</td>
<td>2</td>
<td>CYAN</td>
<td>GREEN</td>
<td>NORMAL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>33 51</td>
<td>3</td>
<td>CYAN</td>
<td>CYAN</td>
<td>NORMAL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>34 52</td>
<td>4</td>
<td>CYAN</td>
<td>RED</td>
<td>NORMAL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>35 53</td>
<td>5</td>
<td>CYAN</td>
<td>MAGENTA</td>
<td>NORMAL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>36 54</td>
<td>6</td>
<td>CYAN</td>
<td>BROWN</td>
<td>NORMAL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>37 55</td>
<td>7</td>
<td>CYAN</td>
<td>LIGHT GREY</td>
<td>NORMAL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>38 56</td>
<td>8</td>
<td>CYAN</td>
<td>DARK GREY</td>
<td>HIGH INTENSITY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>39 57</td>
<td>9</td>
<td>CYAN</td>
<td>LIGHT BLUE</td>
<td>HIGH INTENSITY UNDERLINE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3A 58</td>
<td>:</td>
<td>SHIFT</td>
<td>CYAN</td>
<td>HIGH INTENSITY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3B 59</td>
<td>;</td>
<td>CYAN</td>
<td>LIGHT CYAN</td>
<td>HIGH INTENSITY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3C 60</td>
<td>&lt; &lt;</td>
<td>SHIFT</td>
<td>CYAN</td>
<td>HIGH INTENSITY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3D 61</td>
<td>= =</td>
<td>CYAN</td>
<td>LIGHT MAGENTA</td>
<td>HIGH INTENSITY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3E 62</td>
<td>&gt; &gt;</td>
<td>SHIFT</td>
<td>CYAN</td>
<td>YELLOW</td>
<td>HIGH INTENSITY</td>
<td></td>
</tr>
<tr>
<td>3F 63</td>
<td>? ?</td>
<td>SHIFT</td>
<td>CYAN</td>
<td>WHITE</td>
<td>HIGH INTENSITY</td>
<td></td>
</tr>
<tr>
<td>40 64</td>
<td>@ @</td>
<td>SHIFT</td>
<td>RED</td>
<td>BLACK</td>
<td>NORMAL</td>
<td></td>
</tr>
<tr>
<td>41 65</td>
<td>A</td>
<td>NOTE 4</td>
<td>RED</td>
<td>BLUE</td>
<td>UNDERLINE</td>
<td></td>
</tr>
<tr>
<td>42 66</td>
<td>B</td>
<td>NOTE 4</td>
<td>RED</td>
<td>GREEN</td>
<td>NORMAL</td>
<td></td>
</tr>
<tr>
<td>43 67</td>
<td>C</td>
<td>NOTE 4</td>
<td>RED</td>
<td>CYAN</td>
<td>NORMAL</td>
<td></td>
</tr>
<tr>
<td>44 68</td>
<td>D</td>
<td>NOTE 4</td>
<td>RED</td>
<td>RED</td>
<td>NORMAL</td>
<td></td>
</tr>
<tr>
<td>45 69</td>
<td>E</td>
<td>NOTE 4</td>
<td>RED</td>
<td>MAGENTA</td>
<td>NORMAL</td>
<td></td>
</tr>
<tr>
<td>46 70</td>
<td>F</td>
<td>NOTE 4</td>
<td>RED</td>
<td>BROWN</td>
<td>NORMAL</td>
<td></td>
</tr>
<tr>
<td>47 71</td>
<td>G</td>
<td>NOTE 4</td>
<td>RED</td>
<td>LIGHT GREY</td>
<td>NORMAL</td>
<td></td>
</tr>
<tr>
<td>48 72</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 73</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 74</td>
<td>J</td>
<td>NOTE 4</td>
<td>RED</td>
<td>LIGHT GREEN</td>
<td>HIGH INTENSITY</td>
<td></td>
</tr>
<tr>
<td>VALUE</td>
<td>AS CHARACTERS</td>
<td>AS TEXT ATTRIBUTES</td>
<td>COLOR/GRAPHICS MONITOR ADAPTER</td>
<td>IBM MONOCROME DISPLAY ADAPTER</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>---------------</td>
<td>--------------------</td>
<td>-------------------------------</td>
<td>-----------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HEX</td>
<td>DEC</td>
<td>SYMBOL</td>
<td>KEYSTROKES</td>
<td>MODES</td>
<td>BACKGROUND</td>
<td>FOREGROUND</td>
</tr>
<tr>
<td>4B 75</td>
<td>K</td>
<td>K</td>
<td>NOTE 4</td>
<td>RED</td>
<td>LIGHT CYAN</td>
<td></td>
</tr>
<tr>
<td>4C 76</td>
<td>L</td>
<td>L</td>
<td>NOTE 4</td>
<td>RED</td>
<td>LIGHT RED</td>
<td></td>
</tr>
<tr>
<td>4D 77</td>
<td>M</td>
<td>M</td>
<td>NOTE 4</td>
<td>RED</td>
<td>LIGHT MAGENTA</td>
<td></td>
</tr>
<tr>
<td>4E 78</td>
<td>N</td>
<td>N</td>
<td>NOTE 4</td>
<td>RED</td>
<td>YELLOW</td>
<td></td>
</tr>
<tr>
<td>4F 79</td>
<td>O</td>
<td>O</td>
<td>NOTE 4</td>
<td>RED</td>
<td>WHITE</td>
<td></td>
</tr>
<tr>
<td>50 80</td>
<td>P</td>
<td>P</td>
<td>NOTE 4</td>
<td>MAGENTA</td>
<td>BLACK</td>
<td>NORMAL</td>
</tr>
<tr>
<td>51 81</td>
<td>Q</td>
<td>Q</td>
<td>NOTE 4</td>
<td>MAGENTA</td>
<td>BLUE</td>
<td>UNDERLINE</td>
</tr>
<tr>
<td>52 82</td>
<td>R</td>
<td>R</td>
<td>NOTE 4</td>
<td>MAGENTA</td>
<td>GREEN</td>
<td>NORMAL</td>
</tr>
<tr>
<td>53 83</td>
<td>S</td>
<td>S</td>
<td>NOTE 4</td>
<td>MAGENTA</td>
<td>CYAN</td>
<td>NORMAL</td>
</tr>
<tr>
<td>54 84</td>
<td>T</td>
<td>T</td>
<td>NOTE 4</td>
<td>MAGENTA</td>
<td>RED</td>
<td>NORMAL</td>
</tr>
<tr>
<td>55 85</td>
<td>U</td>
<td>U</td>
<td>NOTE 4</td>
<td>MAGENTA</td>
<td>MAGENTA</td>
<td>NORMAL</td>
</tr>
<tr>
<td>56 86</td>
<td>V</td>
<td>V</td>
<td>NOTE 4</td>
<td>MAGENTA</td>
<td>BROWN</td>
<td>NORMAL</td>
</tr>
<tr>
<td>57 87</td>
<td>W</td>
<td>W</td>
<td>NOTE 4</td>
<td>MAGENTA</td>
<td>LIGHT GREY</td>
<td>NORMAL</td>
</tr>
<tr>
<td>58 88</td>
<td>X</td>
<td>X</td>
<td>NOTE 4</td>
<td>MAGENTA</td>
<td>DARK GREY</td>
<td>HIGH INTENSITY</td>
</tr>
<tr>
<td>59 89</td>
<td>Y</td>
<td>Y</td>
<td>NOTE 4</td>
<td>MAGENTA</td>
<td>LIGHT BLUE</td>
<td>HIGH INTENSITY UNDERLINE</td>
</tr>
<tr>
<td>5A 90</td>
<td>Z</td>
<td>Z</td>
<td>NOTE 4</td>
<td>MAGENTA</td>
<td>LIGHT GREEN</td>
<td>HIGH INTENSITY</td>
</tr>
<tr>
<td>5B 91</td>
<td>1</td>
<td>1</td>
<td>NOTE 4</td>
<td>MAGENTA</td>
<td>LIGHT CYAN</td>
<td>HIGH INTENSITY</td>
</tr>
<tr>
<td>5C 92</td>
<td>\</td>
<td>\</td>
<td>NOTE 4</td>
<td>MAGENTA</td>
<td>LIGHT RED</td>
<td>HIGH INTENSITY</td>
</tr>
<tr>
<td>5D 93</td>
<td>]</td>
<td>]</td>
<td>NOTE 4</td>
<td>MAGENTA</td>
<td>LIGHT MAGENTA</td>
<td>HIGH INTENSITY</td>
</tr>
<tr>
<td>5E 94</td>
<td>\</td>
<td>\</td>
<td>SHIFT</td>
<td>MAGENTA</td>
<td>YELLOW</td>
<td>HIGH INTENSITY</td>
</tr>
<tr>
<td>5F 95</td>
<td>—</td>
<td>—</td>
<td>SHIFT</td>
<td>MAGENTA</td>
<td>WHITE</td>
<td>HIGH INTENSITY</td>
</tr>
<tr>
<td>60 96</td>
<td>'</td>
<td>'</td>
<td></td>
<td>YELLOW</td>
<td>BLACK</td>
<td>NORMAL</td>
</tr>
<tr>
<td>61 97</td>
<td>a</td>
<td>a</td>
<td>NOTE 5</td>
<td>YELLOW</td>
<td>BLUE</td>
<td>UNDERLINE</td>
</tr>
<tr>
<td>62 98</td>
<td>b</td>
<td>b</td>
<td>NOTE 5</td>
<td>YELLOW</td>
<td>GREEN</td>
<td>NORMAL</td>
</tr>
<tr>
<td>63 99</td>
<td>c</td>
<td>c</td>
<td>NOTE 5</td>
<td>YELLOW</td>
<td>CYAN</td>
<td>NORMAL</td>
</tr>
<tr>
<td>64 100</td>
<td>d</td>
<td>d</td>
<td>NOTE 5</td>
<td>YELLOW</td>
<td>RED</td>
<td>NORMAL</td>
</tr>
<tr>
<td>65 101</td>
<td>e</td>
<td>e</td>
<td>NOTE 5</td>
<td>YELLOW</td>
<td>MAGENTA</td>
<td>NORMAL</td>
</tr>
<tr>
<td>66 102</td>
<td>f</td>
<td>f</td>
<td>NOTE 5</td>
<td>YELLOW</td>
<td>BROWN</td>
<td>NORMAL</td>
</tr>
</tbody>
</table>

C-4
<table>
<thead>
<tr>
<th>VALUE</th>
<th>AS CHARACTERS</th>
<th>AS TEXT ATTRIBUTES</th>
<th>COLOR/GRAPHICS MONITOR ADAPTER</th>
<th>18M MONOCROME DISPLAY ADAPTER</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>BACKGROUND</td>
<td>FOREGROUND</td>
</tr>
<tr>
<td>67  103</td>
<td>g</td>
<td>NOTE 5</td>
<td>YELLOW</td>
<td>LIGHT GREY</td>
</tr>
<tr>
<td>68  104</td>
<td>h</td>
<td>NOTE 5</td>
<td>YELLOW</td>
<td>DARK GREY</td>
</tr>
<tr>
<td>69  105</td>
<td>i</td>
<td>NOTE 5</td>
<td>YELLOW</td>
<td>LIGHT BLUE</td>
</tr>
<tr>
<td>6A  106</td>
<td>j</td>
<td>NOTE 5</td>
<td>YELLOW</td>
<td>LIGHT GREEN</td>
</tr>
<tr>
<td>6B  107</td>
<td>k</td>
<td>NOTE 5</td>
<td>YELLOW</td>
<td>LIGHT CYAN</td>
</tr>
<tr>
<td>6C  108</td>
<td>l</td>
<td>NOTE 5</td>
<td>YELLOW</td>
<td>LIGHT RED</td>
</tr>
<tr>
<td>6D  109</td>
<td>m</td>
<td>NOTE 5</td>
<td>YELLOW</td>
<td>LIGHT MAGENTA</td>
</tr>
<tr>
<td>6E  110</td>
<td>n</td>
<td>NOTE 5</td>
<td>YELLOW</td>
<td>YELLOW</td>
</tr>
<tr>
<td>6F  111</td>
<td>o</td>
<td>NOTE 5</td>
<td>YELLOW</td>
<td>WHITE</td>
</tr>
<tr>
<td>70  112</td>
<td>p</td>
<td>NOTE 5</td>
<td>WHITE</td>
<td>BLACK</td>
</tr>
<tr>
<td>71  113</td>
<td>q</td>
<td>NOTE 5</td>
<td>WHITE</td>
<td>BLUE</td>
</tr>
<tr>
<td>72  114</td>
<td>r</td>
<td>NOTE 5</td>
<td>WHITE</td>
<td>GREEN</td>
</tr>
<tr>
<td>73  115</td>
<td>s</td>
<td>NOTE 5</td>
<td>WHITE</td>
<td>CYAN</td>
</tr>
<tr>
<td>74  116</td>
<td>t</td>
<td>NOTE 5</td>
<td>WHITE</td>
<td>RED</td>
</tr>
<tr>
<td>75  117</td>
<td>u</td>
<td>NOTE 5</td>
<td>WHITE</td>
<td>MAGENTA</td>
</tr>
<tr>
<td>76  118</td>
<td>v</td>
<td>NOTE 5</td>
<td>WHITE</td>
<td>BROWN</td>
</tr>
<tr>
<td>77  119</td>
<td>w</td>
<td>NOTE 5</td>
<td>WHITE</td>
<td>LIGHT GREY</td>
</tr>
<tr>
<td>78  120</td>
<td>x</td>
<td>NOTE 5</td>
<td>WHITE</td>
<td>DARK GREY</td>
</tr>
<tr>
<td>79  121</td>
<td>y</td>
<td>NOTE 5</td>
<td>WHITE</td>
<td>LIGHT BLUE</td>
</tr>
<tr>
<td>7A  122</td>
<td>z</td>
<td>NOTE 5</td>
<td>WHITE</td>
<td>LIGHT GREEN</td>
</tr>
<tr>
<td>7B  123</td>
<td>i</td>
<td>SHIFT</td>
<td>WHITE</td>
<td>LIGHT CYAN</td>
</tr>
<tr>
<td>7C  124</td>
<td>i</td>
<td>SHIFT</td>
<td>WHITE</td>
<td>LIGHT RED</td>
</tr>
<tr>
<td>7D  125</td>
<td>i</td>
<td>SHIFT</td>
<td>WHITE</td>
<td>LIGHT MAGENTA</td>
</tr>
<tr>
<td>7E  126</td>
<td>~</td>
<td>SHIFT</td>
<td>WHITE</td>
<td>YELLOW</td>
</tr>
<tr>
<td>7F  127</td>
<td>Δ CTRL</td>
<td></td>
<td>WHITE</td>
<td>WHITE</td>
</tr>
<tr>
<td>VALUE</td>
<td>AS CHARACTERS</td>
<td>AS TEXT ATTRIBUTES</td>
<td>IBM MONOCROME DISPLAY ADAPTER</td>
<td></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>80</td>
<td>128</td>
<td>♂</td>
<td>ALT 128</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>81</td>
<td>129</td>
<td>♂</td>
<td>ALT 129</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>82</td>
<td>130</td>
<td>♂</td>
<td>ALT 130</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>83</td>
<td>131</td>
<td>♂</td>
<td>ALT 131</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>84</td>
<td>132</td>
<td>♂</td>
<td>ALT 132</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>85</td>
<td>133</td>
<td>♂</td>
<td>ALT 133</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>86</td>
<td>134</td>
<td>♂</td>
<td>ALT 134</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>87</td>
<td>135</td>
<td>♂</td>
<td>ALT 135</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>88</td>
<td>136</td>
<td>♂</td>
<td>ALT 136</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>89</td>
<td>137</td>
<td>♂</td>
<td>ALT 137</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>8A</td>
<td>138</td>
<td>♂</td>
<td>ALT 138</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>8B</td>
<td>139</td>
<td>♂</td>
<td>ALT 139</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>8C</td>
<td>140</td>
<td>♂</td>
<td>ALT 140</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>8D</td>
<td>141</td>
<td>♂</td>
<td>ALT 141</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>8E</td>
<td>142</td>
<td>♂</td>
<td>ALT 142</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>8F</td>
<td>143</td>
<td>♂</td>
<td>ALT 143</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>90</td>
<td>144</td>
<td>♂</td>
<td>ALT 144</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>91</td>
<td>145</td>
<td>♂</td>
<td>ALT 145</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>92</td>
<td>146</td>
<td>♂</td>
<td>ALT 146</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>93</td>
<td>147</td>
<td>♂</td>
<td>ALT 147</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>94</td>
<td>148</td>
<td>♂</td>
<td>ALT 148</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>95</td>
<td>149</td>
<td>♂</td>
<td>ALT 149</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>96</td>
<td>150</td>
<td>♂</td>
<td>ALT 150</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>97</td>
<td>151</td>
<td>♂</td>
<td>ALT 151</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>98</td>
<td>152</td>
<td>♂</td>
<td>ALT 152</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>99</td>
<td>153</td>
<td>♂</td>
<td>ALT 153</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>9A</td>
<td>154</td>
<td>♂</td>
<td>ALT 154</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>VALUE</td>
<td>AS CHARACTERS</td>
<td>AS TEXT ATTRIBUTES</td>
<td></td>
<td></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>9B</td>
<td>155</td>
<td>ccording</td>
<td>ALT 155</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>9C</td>
<td>156</td>
<td>è</td>
<td>ALT 156</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>9D</td>
<td>157</td>
<td>½</td>
<td>ALT 157</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>9E</td>
<td>158</td>
<td>Pt</td>
<td>ALT 158</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>9F</td>
<td>159</td>
<td>ǎ</td>
<td>ALT 159</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>A0</td>
<td>160</td>
<td>a</td>
<td>ALT 160</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>A1</td>
<td>161</td>
<td>i</td>
<td>ALT 161</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>A2</td>
<td>162</td>
<td>o</td>
<td>ALT 162</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>A3</td>
<td>163</td>
<td>u</td>
<td>ALT 163</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>A4</td>
<td>164</td>
<td>Ĳ</td>
<td>ALT 164</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>A5</td>
<td>165</td>
<td>Ñ</td>
<td>ALT 165</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>A6</td>
<td>166</td>
<td>å</td>
<td>ALT 166</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>A7</td>
<td>167</td>
<td>o</td>
<td>ALT 167</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>A8</td>
<td>168</td>
<td>ï</td>
<td>ALT 168</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>A9</td>
<td>169</td>
<td>Ы</td>
<td>ALT 169</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>AA</td>
<td>170</td>
<td>½</td>
<td>ALT 170</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>AB</td>
<td>171</td>
<td>½</td>
<td>ALT 171</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>AC</td>
<td>172</td>
<td>¼</td>
<td>ALT 172</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>AD</td>
<td>173</td>
<td>¾</td>
<td>ALT 173</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>AE</td>
<td>174</td>
<td>¾</td>
<td>ALT 174</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>AF</td>
<td>175</td>
<td>½</td>
<td>ALT 175</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>B0</td>
<td>176</td>
<td>½</td>
<td>ALT 176</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>B1</td>
<td>177</td>
<td>½</td>
<td>ALT 177</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>B2</td>
<td>178</td>
<td>¼</td>
<td>ALT 178</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>B3</td>
<td>179</td>
<td>¾</td>
<td>ALT 179</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>B4</td>
<td>180</td>
<td>¾</td>
<td>ALT 180</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>B5</td>
<td>181</td>
<td>¼</td>
<td>ALT 181</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>B6</td>
<td>182</td>
<td>¾</td>
<td>ALT 182</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>HEX DEC</td>
<td>SYMBOL</td>
<td>KEYSTROKES</td>
<td>MODES</td>
<td>BACKGROUND</td>
</tr>
<tr>
<td>---------</td>
<td>--------</td>
<td>------------</td>
<td>-------</td>
<td>------------</td>
</tr>
<tr>
<td>B7 183</td>
<td>ALT 183</td>
<td>NOTE 6</td>
<td>CYAN</td>
<td>LIGHT GREY</td>
</tr>
<tr>
<td>B8 184</td>
<td>ALT 184</td>
<td>NOTE 6</td>
<td>CYAN</td>
<td>DARK GREY</td>
</tr>
<tr>
<td>B9 185</td>
<td>ALT 185</td>
<td>NOTE 6</td>
<td>CYAN</td>
<td>LIGHT BLUE</td>
</tr>
<tr>
<td>BA 186</td>
<td>ALT 186</td>
<td>NOTE 6</td>
<td>CYAN</td>
<td>LIGHT GREEN</td>
</tr>
<tr>
<td>BB 187</td>
<td>ALT 187</td>
<td>NOTE 6</td>
<td>CYAN</td>
<td>LIGHT CYAN</td>
</tr>
<tr>
<td>BC 188</td>
<td>ALT 188</td>
<td>NOTE 6</td>
<td>CYAN</td>
<td>LIGHT RED</td>
</tr>
<tr>
<td>BD 189</td>
<td>ALT 189</td>
<td>NOTE 6</td>
<td>CYAN</td>
<td>LIGHT MAGENTA</td>
</tr>
<tr>
<td>BE 190</td>
<td>ALT 190</td>
<td>NOTE 6</td>
<td>CYAN</td>
<td>YELLOW</td>
</tr>
<tr>
<td>BF 191</td>
<td>ALT 191</td>
<td>NOTE 6</td>
<td>CYAN</td>
<td>WHITE</td>
</tr>
<tr>
<td>C0 192</td>
<td>ALT 192</td>
<td>NOTE 6</td>
<td>RED</td>
<td>BLACK</td>
</tr>
<tr>
<td>C1 193</td>
<td>ALT 193</td>
<td>NOTE 6</td>
<td>RED</td>
<td>BLUE</td>
</tr>
<tr>
<td>C2 194</td>
<td>ALT 194</td>
<td>NOTE 6</td>
<td>RED</td>
<td>GREEN</td>
</tr>
<tr>
<td>C3 195</td>
<td>ALT 195</td>
<td>NOTE 6</td>
<td>RED</td>
<td>CYAN</td>
</tr>
<tr>
<td>C4 196</td>
<td>ALT 196</td>
<td>NOTE 6</td>
<td>RED</td>
<td>RED</td>
</tr>
<tr>
<td>C5 197</td>
<td>ALT 197</td>
<td>NOTE 6</td>
<td>RED</td>
<td>MAGENTA</td>
</tr>
<tr>
<td>C6 198</td>
<td>ALT 198</td>
<td>NOTE 6</td>
<td>RED</td>
<td>BROWN</td>
</tr>
<tr>
<td>C7 199</td>
<td>ALT 199</td>
<td>NOTE 6</td>
<td>RED</td>
<td>LIGHT GREY</td>
</tr>
<tr>
<td>C8 200</td>
<td>ALT 200</td>
<td>NOTE 6</td>
<td>RED</td>
<td>DARK GREY</td>
</tr>
<tr>
<td>C9 201</td>
<td>ALT 201</td>
<td>NOTE 6</td>
<td>RED</td>
<td>LIGHT BLUE</td>
</tr>
<tr>
<td>CA 202</td>
<td>ALT 202</td>
<td>NOTE 6</td>
<td>RED</td>
<td>LIGHT BLUE</td>
</tr>
<tr>
<td>CB 203</td>
<td>ALT 203</td>
<td>NOTE 6</td>
<td>RED</td>
<td>LIGHT CYAN</td>
</tr>
<tr>
<td>CC 204</td>
<td>ALT 204</td>
<td>NOTE 6</td>
<td>RED</td>
<td>LIGHT RED</td>
</tr>
<tr>
<td>CD 205</td>
<td>ALT 205</td>
<td>NOTE 6</td>
<td>RED</td>
<td>LIGHT MAGENTA</td>
</tr>
<tr>
<td>CE 206</td>
<td>ALT 206</td>
<td>NOTE 6</td>
<td>RED</td>
<td>YELLOW</td>
</tr>
<tr>
<td>CF 207</td>
<td>ALT 207</td>
<td>NOTE 6</td>
<td>RED</td>
<td>WHITE</td>
</tr>
<tr>
<td>DO 208</td>
<td>ALT 208</td>
<td>NOTE 6</td>
<td>MAGENTA</td>
<td>BLACK</td>
</tr>
<tr>
<td>VALUE</td>
<td>AS CHARACTERS</td>
<td>AS TEXT ATTRIBUTES</td>
<td>COLOR/GRAPHICS MONITOR ADAPTER</td>
<td>IBM MONOCHROME DISPLAY ADAPTER</td>
</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></td>
</tr>
<tr>
<td>D2</td>
<td>210</td>
<td>ALT 210</td>
<td>NOTE 6</td>
<td></td>
</tr>
<tr>
<td>D3</td>
<td>211</td>
<td>ALT 211</td>
<td>NOTE 6</td>
<td></td>
</tr>
<tr>
<td>D4</td>
<td>212</td>
<td>ALT 212</td>
<td>NOTE 6</td>
<td></td>
</tr>
<tr>
<td>D5</td>
<td>213</td>
<td>ALT 213</td>
<td>NOTE 6</td>
<td></td>
</tr>
<tr>
<td>D6</td>
<td>214</td>
<td>ALT 214</td>
<td>NOTE 6</td>
<td></td>
</tr>
<tr>
<td>D7</td>
<td>215</td>
<td>ALT 215</td>
<td>NOTE 6</td>
<td></td>
</tr>
<tr>
<td>D8</td>
<td>216</td>
<td>ALT 216</td>
<td>NOTE 6</td>
<td></td>
</tr>
<tr>
<td>D9</td>
<td>217</td>
<td>ALT 217</td>
<td>NOTE 6</td>
<td></td>
</tr>
<tr>
<td>DA</td>
<td>218</td>
<td>ALT 218</td>
<td>NOTE 6</td>
<td></td>
</tr>
<tr>
<td>DB</td>
<td>219</td>
<td>ALT 219</td>
<td>NOTE 6</td>
<td></td>
</tr>
<tr>
<td>DC</td>
<td>220</td>
<td>ALT 220</td>
<td>NOTE 6</td>
<td></td>
</tr>
<tr>
<td>DD</td>
<td>221</td>
<td>ALT 221</td>
<td>NOTE 6</td>
<td></td>
</tr>
<tr>
<td>DE</td>
<td>222</td>
<td>ALT 222</td>
<td>NOTE 6</td>
<td></td>
</tr>
<tr>
<td>DF</td>
<td>223</td>
<td>ALT 223</td>
<td>NOTE 6</td>
<td></td>
</tr>
<tr>
<td>E0</td>
<td>224</td>
<td>a</td>
<td>ALT 224</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>E1</td>
<td>225</td>
<td>b</td>
<td>ALT 225</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>E2</td>
<td>226</td>
<td>g</td>
<td>ALT 226</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>E3</td>
<td>227</td>
<td>π</td>
<td>ALT 227</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>E4</td>
<td>228</td>
<td>s</td>
<td>ALT 228</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>E5</td>
<td>229</td>
<td>θ</td>
<td>ALT 229</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>E6</td>
<td>230</td>
<td>ρ</td>
<td>ALT 230</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>E7</td>
<td>231</td>
<td>ρ</td>
<td>ALT 231</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>E8</td>
<td>232</td>
<td>Φ</td>
<td>ALT 232</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>E9</td>
<td>233</td>
<td>θ</td>
<td>ALT 233</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>EA</td>
<td>234</td>
<td>Ω</td>
<td>ALT 234</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>EB</td>
<td>235</td>
<td>δ</td>
<td>ALT 235</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>HEX</td>
<td>DEC</td>
<td>SYMBOL</td>
<td>KEYSTROKES</td>
<td>MODES</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>--------</td>
<td>------------</td>
<td>-------</td>
</tr>
<tr>
<td>EC</td>
<td>236</td>
<td>∞</td>
<td>ALT 236</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>ED</td>
<td>237</td>
<td>φ</td>
<td>ALT 237</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>EE</td>
<td>238</td>
<td>∈</td>
<td>ALT 238</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>EF</td>
<td>239</td>
<td>∪</td>
<td>ALT 239</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>F0</td>
<td>240</td>
<td>≡</td>
<td>ALT 240</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>F1</td>
<td>241</td>
<td>±</td>
<td>ALT 241</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>F2</td>
<td>242</td>
<td>≈</td>
<td>ALT 242</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>F3</td>
<td>243</td>
<td>⋈</td>
<td>ALT 243</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>F4</td>
<td>244</td>
<td>≠</td>
<td>ALT 244</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>F5</td>
<td>245</td>
<td>∨</td>
<td>ALT 245</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>F6</td>
<td>246</td>
<td>÷</td>
<td>ALT 246</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>F7</td>
<td>247</td>
<td>≈</td>
<td>ALT 247</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>F8</td>
<td>248</td>
<td>∞</td>
<td>ALT 248</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>F9</td>
<td>249</td>
<td>⋈</td>
<td>ALT 249</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>FA</td>
<td>250</td>
<td>⋈</td>
<td>ALT 250</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>FB</td>
<td>251</td>
<td>√</td>
<td>ALT 251</td>
<td>NOTE 6</td>
</tr>
<tr>
<td>FC</td>
<td>252</td>
<td>√</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>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>
NOTE 1  Asterisk (*) can easily be keyed using two methods: 1) hit the [PRTSC] key or 2) in shift mode hit the [8] key.

NOTE 2  Period (.) can easily be keyed using two methods: 1) hit the [.] key or 2) in shift or NUM LOCK mode hit the [del] key.

NOTE 3  Numeric characters (0—9) can easily be keyed using two methods: 1) hit the numeric keys on the top row of the typewriter portion of the keyboard or 2) in shift or NUM LOCK mode hit the numeric keys in the 10-key pad portion of the keyboard.

NOTE 4  Upper case alphabetic characters (A—Z) can easily be keyed in two modes: 1) in shift mode hit 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 alphabetic 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.
## Character Set (00-7F) Quick Reference

<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>HEXA-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>0</td>
<td>0</td>
<td>0</td>
<td>!</td>
<td>!</td>
<td>!</td>
<td>!</td>
<td>!</td>
<td>!</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>11</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>13</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td>14</td>
<td>E</td>
<td>E</td>
<td>E</td>
<td>E</td>
<td>E</td>
<td>E</td>
<td>E</td>
<td>E</td>
</tr>
<tr>
<td>15</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

- **BLANK (NULL)**: Represents space or null character.
- **BLANK (SPACE)**: Represents blank or space.
- **@**: Represents the at symbol.
- **P**: Represents the capital letter P.
- **Q**: Represents the capital letter Q.
- **A**: Represents the capital letter A.
- **B**: Represents the capital letter B.
- **C**: Represents the capital letter C.
- **D**: Represents the capital letter D.
- **E**: Represents the capital letter E.
- **F**: Represents the capital letter F.
- **G**: Represents the capital letter G.
- **H**: Represents the capital letter H.
- **I**: Represents the capital letter I.
- **J**: Represents the capital letter J.
- **K**: Represents the capital letter K.
- **L**: Represents the capital letter L.
- **M**: Represents the capital letter M.
- **N**: Represents the capital letter N.
- **O**: Represents the capital letter O.
- **P**: Represents the capital letter P.
- **Q**: Represents the capital letter Q.
# Character Set (80-FF) Quick Reference

<table>
<thead>
<tr>
<th>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>8</td>
<td>9</td>
<td>A</td>
<td>B</td>
<td>D</td>
<td>C</td>
<td>E</td>
<td>F</td>
</tr>
<tr>
<td>1</td>
<td>ü</td>
<td>Æ</td>
<td>i</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>é</td>
<td>FE</td>
<td>ò</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>á</td>
<td>A</td>
<td>û</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>à</td>
<td>ô</td>
<td>ñ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>à</td>
<td>ò</td>
<td>À</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>à</td>
<td>ü</td>
<td>Ñ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>ç</td>
<td>ù</td>
<td>Ò</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>ë</td>
<td>ÿ</td>
<td>Ò</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>ë</td>
<td>Ø</td>
<td>Ù</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>ë</td>
<td>Ô</td>
<td>Ù</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>ë</td>
<td>Ô</td>
<td>Ù</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>ë</td>
<td>Ô</td>
<td>Ù</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>ë</td>
<td>Ô</td>
<td>Ù</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>ë</td>
<td>Ô</td>
<td>Ù</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>ë</td>
<td>Ô</td>
<td>Ù</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>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>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
</tr>
<tr>
<td>1</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
</tr>
<tr>
<td>2</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
</tr>
<tr>
<td>3</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
</tr>
<tr>
<td>4</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
</tr>
<tr>
<td>5</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
</tr>
<tr>
<td>6</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
</tr>
<tr>
<td>7</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
</tr>
<tr>
<td>8</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
</tr>
<tr>
<td>9</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
</tr>
<tr>
<td>10</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
</tr>
<tr>
<td>11</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
</tr>
<tr>
<td>12</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
</tr>
<tr>
<td>13</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
</tr>
<tr>
<td>14</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
</tr>
<tr>
<td>15</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
</tr>
</tbody>
</table>
APPENDIX D LOGIC DIAGRAMS

Contents:

System Board ................................... D-2
Keyboard ........................................... D-12
IBM Monochrome Display And Parallel Printer Adapter .................................. D-14
IBM Monochrome Display .................................. D-24
Color/Graphics Monitor Adapter .................................. D-25
IBM 80 CPS Matrix Printer .................................. D-31
Parallel Printer Adapter .................................. D-34
5 1/4” Diskette Drive Adapter .................................. D-35
5 1/4” Diskette Drive .................................. D-39
32 KB Memory Expansion .................................. D-42
64 KB Memory Expansion .................................. D-45
Asynchronous Communications Adapter .................................. D-48
Game Control Adapter .................................. D-49
Note: Logics one and two of twelve are not applicable
System Board (Device Decodes) Logic 5 of 12
SYSTEM BOARD (DMA)
SYSTEM BOARD (ROS AND BUS DRIVER)
SYSTEM BOARD (KEYBOARD/SENSE/CONTROL)
Note: Logics one and two of twelve are not applicable.
IBM Monochrome Display And Parallel Printer Adapter Logic 5 of 12
IBM Monochrome Display And Parallel Printer Adapter Logic 9 of 12
IBM Monochrome Display And Parallel Printer Adapter Logic 10 of 12
IBM Monochrome Display And Parallel Printer Adapter Logic 11 of 12
DANGER
HAZARDOUS VOLTAGES UP TO 450 VOLTS EXIST ON THE PRINTED CIRCUIT BOARDS
Color/Graphics Monitor Adapter Logic 5 of 6
IBM 80 CPS MATRIX PRINTER

IBM 80 CPS Dot Matrix Printer Diagram of Control Circuit
DISKETTE DRIVE ADAPTER

Note: Logics one and two of six are not applicable.

- ENABLE DRIVE 1
- ENABLE DRIVE 2
- DRIVE SELECT 1
- DRIVE SELECT 2
- MOTOR ENABLE 1
- MOTOR ENABLE 2
- STEP
- DIRECTION
- SELECT HEAD 1
- WRITE DATA
- WRITE ENABLE
- DRIVE SELECT 3
- DRIVE SELECT 4
- MOTOR ENABLE 3
- MOTOR ENABLE 4

5% Diskette Drive Adapter Logic 3 of 6
5¼" Diskette Drive Adapter Logic 4 of 6
5¼" Diskette Drive Adapter Logic 6 of 6
NOTES:
1. RESISTORS ARE IN OHMS, ± 5%, 1/4 W.
2. 1% RESISTORS ARE 1/8 W.
3. CAPACITORS ARE IN µF, ± 20%, 35 V.

5¼" Diskette Drive Logic 3 of 3
Appendix E
Unit Specifications

System Unit

Size:
- Length--19.6” (500 mm)
- Depth--16.1” (410 mm)
- Height--5.5” (142 mm)

Weight:
- Without Diskette Drive Unit-20.9 lbs (9.5 kg)
- With Diskette Drive Unit-25 lbs (11.4 kg)

Power Cable:
- Length--6’0” (1.83 mm)
- 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 (Maximum)

Noise Levels:
- Without Printer, 59 DBS
- With Printer, 66 DBS

Electrical:
- Nominal-120 VAC
- Minimum-104 VAC
- Maximum-127 VAC
- KVA-.3175 maximum

Keyboard

Size:
- Length--19.6” (500 mm)
- Depth--7.87” (200 mm)
- Height--2.2” (57 mm)

Weight:
- 6.5 lbs (14.3 kg)
IBM Monochrome Display

Size:
- Length--14.9” (380 mm)
- Depth--13.7” (350 mm)
- Height--11” (280 mm)

Weight:
- 17.3 lbs (7.9 kg)

Heat Output:
- 325 BTU/HR

Power Cable:
- Length--3.0” (914 mm)
- Size--18 AWG

Signal Cable:
- Length--4’0” (1.22 mm)
- Size--22 AWG

IBM 80 CPS Matrix Printer

Size:
- Length--15.7” (400 mm)
- Depth--14.5” (370 mm)
- Height--4.3” (110 mm)

Weight:
- 12.9 lbs (5.9 kg)

Power Cable:
- Length--6.0” (1.83 mm)
- Size--18 AWG

Signal Cable:
- Length--6’0” (1.83 mm)
- Size--22 AWG

Heat Output:
- 341 BTU/HR (Max.)

Electrical:
- Nominal-120 VAC
- Minimum-104 VAC
- Maximum-127 VAC
1. Address Buss: A set of wires or signals carrying the binary-coded address from the Intel-8088 microprocessor throughout the rest of the IBM Personal Computer System Unit.

2. AEN: Address Enable. (Refer to System Board I/O Channel Descriptions).

3. ALE: Address Latch Enable. (Refer to System Board I/O Channel Descriptions).

4. Analog: (1) Pertaining to representation by means of continuously variable physical quantities. (2) Contrast with digital.

5. A/N: Alphanumeric: Pertaining to a character set that contains letters, digits, and usually other characters, such as punctuation marks. Synonymous with alphanetic.

6. A0-A19: Address bits 0-19. (Refer to System Board I/O Channel Descriptions).

7. APA: All points addressable graphics.

8. ASCII: American Standard Code of Information Interchange. 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.

9. Assembler: A computer program used to assemble. Synonymous with assembly program.

10. BASIC: (Beginner's all-purpose symbolic instruction code). A programming language with a small repertoire of commands and a simple syntax, primarily designed for numerical application.

11. BAUD: (1) A unit of signaling speed equal to the number of discrete conditions or signal events 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, i.e. if the duration of the unit interval is 20 milliseconds, the modulation rate is 50 baud.
12. Binary: (1) Pertaining to a selection, choice, or condition that that has two possible values or states. (2) Pertaining to a fixed radix numeration system having a radix of two.

13. BIOS: Basic Input/Output System.

14. Bootstrap: A technique or device designed to bring itself into a desired state by means of its own action, e.g. a machine routine whose first few instructions are sufficient to bring the rest of itself into the computer from an input device.

15. Buffer: 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. A portion of storage for temporarily holding input or output data.

16. Bus: One or more conductors used for transmitting signals or power.

17. Byte: (1) A binary character operated upon as a unit and usually shorter than a computer word. (2) The representation of a character.

18. CLK: Clock. (Refer to System Board I/O Channel Descriptions).

19. 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.

20. Computer: A data processor that can perform substantial computation, including numerous arithmetic operations, or logic operations, without intervention by a human operator during the run.

21. CPS: Characters per second.

22. CRC: The cyclic redundancy check character.

23. CRT: (1) A Cathode ray tube display. (2) A display device, such as the IBM Monochrome Display, that uses a cathode ray tube.

24. CTS: Conversational Terminal System. (2) Clear to Send. Associated with modem control.

25. DACK0-DACK3: DMA Acknowledge 0 to 3. (Refer to System Board I/O Channel Description).
26. 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.

27. Din Connectors: One of the connectors specified by the Din standardization committee.

28. DIP: “Dual In-Line Package.” A widely used container for an integrated circuit. DIP’s are pins usually in two parallel rows. These pins are spaced on 1/10” inters and come in different configurations ranging from a 14-pin assembly to a 40-pin configuration.

29. Display: A visual presentation of data.

30. DMA: Direct Memory Access.

31. DO-D7: Data Bits 0 to 7. (Refer to System Board I/O Channel Descriptions).

32. DRQ1-DRQ3: DMA Request 1 to 3. (Refer to System Board I/O Channel Descriptions).

33. DSR: Data Set Ready, associated with modem control.

34. DTR: Distribution Tape Reel.

35. Edge Connector: An opening which joins with the end of a circuit board. The purpose of this interface is to send electrical signals back and forth.

36. EIA/CCITT Drives: Electronic Industries Association/Consultative Committee on International Telegraphy and Telephony Drives.

EPROM or ‘PROM’: Term for “Programmable Read-Only Memory.” An EPROM or ‘PROM’ is actually Read-Only Memory (ROM) but the contents may be changed by electrical means. EPROM or ‘PROM’ information is not destroyed when the power is cut off.

37. Firmware: Memory chips with the software programs already built in.

38. Graphics: Symbols Produced by a process such as handwriting, drawing or printing. Synonymous with graphic symbol.

39. Hexadecimal: Pertaining to a selection, choice, or condition that has sixteen possible values or states. These values or states usually contain 10 digits and six letters A through F. Hexadecimal digits are equivalent to a power of 16.
40. Hertz (Hz.): A unit of frequency equal to one cycle per second.

41. High order position: The leftmost position in a string of characters.

42. Input/Output (I/O): Pertaining to a device or to a channel that may be involved in an input process, and, at a different time, in an output process. (2) Pertaining to a device whose parts can be performing an input process and an output process at the same time.

43. Integrated Circuit: A combination of interconnected circuit elements inseparably associated on or within a continuous substrate.

44. Interpreter: A computer program used to interpret. Synonymous with interpretive program.

45. Interrupt: (1) A suspension of a process, such as the execution of a computer program, in such a way that the process can be resumed. (2) To stop a process in such a way that it can be resumed. (3) In data transmission, to take an action at a receiving station that causes the transmitting station to terminate a transmission.

46. I/O Channel: Input/Output Channel. In a data processing system, a functional unit, controlled by the processing unit, that handles the transfer of data between main storage and peripheral equipment.

47. I/O CH CK: I/O Channel Check. (Refer to System Board I/O Channel Descriptions).

48. I/O CH RDY: I/O Channel Ready. (Refer to System Board I/O Channel Descriptions).

49. IMR: Interruption Mask Register.

50. IOR: I/O Read Command. (Refer to System Board I/O Channel Descriptions).

51. IOW: I/O Write Command: (Refer to System Board I/O Channel Descriptions).

52. IRQ2-IRQ7: Interrupt Request 2 to 7. (Refer to System Board I/O Channel Descriptions).

53. K: An abbreviation for the prefix kilo, i.e. 1000 in decimal notation. To the tenth power, 1024 in decimal notation.

54. KB: Kilobyte.

55. Khz: Kilohertz. A unit of frequency equal to 1,000 hertz.
56. Low order position: The rightmost position in a string of characters.

57. Machine Language: (1) A language that is used directly by a machine. (2) Another term for computer instruction code.

58. Memory Address: A two-byte value selecting one specific memory location on a memory map.

59. Memory Location: The most specific part of a memory map that the computer can refer to.

60. Memory Map: The list of memory locations addressed directly by the microprocessor.

61. MEMR: Memory Read Command. (Refer to System Board I/O Channel Descriptions).

62. MEMW: Memory Write Command. (Refer to System Board I/O Channel Descriptions).

63. MFM Coded: Modified Frequency Modulation. It is double density encoding of information on a diskette.

64. Mhz: Megahertz. A unit of frequency equal to one million Hertz.

65. Microprocessor: A processing unit, or part of a processing unit, that consists of microcode. In the IBM Personal Computer, the microprocessor is the Intel-8088.

66. Mnemonic: Symbol or symbols used instead of terminology more difficult to remember. Usually a mnemonic has two or three letters.

67. Mode: (1) A method of operation; for example, the binary mode, the interpretive mode, the alphanumeric mode. (2) The most frequent value in the statistical sense.

68. 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.

69. Multiplexer: A device capable of interleaving the events of two or more activities or capable of distributing the events of an interleaved sequence to their respective activities.
70. 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. P OR Q is often represented by P+Q, PVQ. The term is synonymous with boolean add; logic add.

71. “ORed”: Past tense of OR.

72. OSC: Oscillator. (Refer to System Board I/O Channel Descriptions).

73. Output: Pertaining to a device, process, or channel involved in an output process, or to the data or states involved in an output process.

74. Personal Computer: A small home or business computer complete with a System Unit, keyboard, and available with a variety of options such as monochrome display and a dot matrix printer.

75. Pinout: A diagram of functioning pins on a pinboard.

76. Printed Circuit Board: A piece of material, usually fiberglass, which contains a layer of conductive material, usually metal. The metallic layer is then etched and electronic equipment is then attached to the fiberglass. The electronic equipment then has the capacity to transmit electronic signals through the board by way of the etched metal tracks.

77. Program: (1) A series of actions designed to achieve a certain result. (2) To design, write and test computer programs.

78. Read/Write Memory: Random access storage.

79. Reset Drv: Reset Driver. (Refer to System Board I/O Channel Descriptions).

80. RF Modulator: The device used to convert the composite video signal to the antenna level input of a home TV.

81. ROM: Read-only Memory.

82. ROM BIOS: Read-only Memory/Basic Input Output System.

83. RS 232 Port: Asynchronous Type Communications.

84. RTS: Ready to Send. Associated with modem control.
85. Scan Line: The use of a cathode beam to test the cathode ray tube of a display used with a personal computer.

86. Schematic: The description, usually in diagram form, of the logical structure and physical structure of an entire data base according to a conceptual model.

87. Software: (1) Computer programs, procedures, rules, and possibly associated documentation concerned with the operation of a data processing system. (2) Contrast with hardware.

88. Strobe: (1) An instrument used to determine the exact speed of circular or cyclic movement. (2) A flashing signal displaying an exact event.

89. 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.

90. TX Data: Transmit Data. External connections of the RS 232 Asynchronous Communications Adapter interface.

91. Video: Computer data shown or displayed on a cathode ray tube monitor or display.
# BIBLIOGRAPHY

## IBM Publications

1. **IBM Personal Computer Guide to Operations-PN 6025000**
   General information on using the IBM Personal Computer.

2. **IBM Personal Computer Hardware and Service-PN 6025072**
   Information on hardware and steps necessary when servicing this IBM Personal Computer.

3. **IBM Personal Computer BASIC-PN 6025010**
   Information for programmers who are using BASIC.

4. **IBM Personal Computer Disk Operating System (DOS)-PN 6024001**
   Information for programmers who are using DOS.

5. **IBM Personal Computer MACRO Assembler-PN 6024002**
   Information for experienced assembly language programmers using the Macro Assembler.

6. **IBM Personal Computer Pascal Compiler-PN 6024010**
   Information for programmers who are familiar with the Pascal language.
Other Related Publications

7. NATIONAL SEMICONDUCTOR
   INS 8250 Asynchronous Communications Element
   This book documents physical and operating characteristics
   of the INS 8250.

8. INTEL
   The 8086 Family Users Manual
   This manual introduces the 8086 family of microcomputing
   components and serves as a reference in system design
   and implementation.

9. INTEL
   8086/8087/8088 Macro
   ASSEMBLY Language
   Reference Manual for 8088/8085 Based Development
   System
   The manual describes the 8086/8087/8088 Macro Assembly
   Language, and is intended for use by persons who are familiar
   with assembly language.

10. MOTOROLA
    The complete Microcomputer Data Library
    This book can provide additional information on the Motorola
    6845 CRT Controller used in the IBM Monochrome Display
    and Parallel Printer Adapter, and the Color/Graphics
    Monitor Adapter.
INDEX

A

A0-A19 (Address Bits 0-19) 2-10
A.C. Output 2-34
Adapters
  Asynchronous Communications (RS232) 2-123
  Color/Graphics 2-45
  5 1/4” Diskette Drive 2-89
  Game Control 2-117
  IBM Monochrome Display and Parallel Printer 2-37
  Parallel Printer 2-65
Adapter Attribute Relationship 2-51
Adapter Inputs 2-107
Adapter Outputs 2-106
Address Bits (A0-A19) 2-10
Address Decode 2-118
Address Enable (AEN) 2-12
Address Latch Enable (ALE) 2-10
Address Strobe (ADS) 2-129
ADS (Address Strobe) 2-129
AEN (Address Enable) 2-12
ALE (Address Latch Enable) 2-10
Algorithms 3-8
All Points Addressable (APA) 2-45
Alphanumeric Mode 2-49
American Standard Code for Information Interchange (ASCII) 1-2
Analog Input 2-122
A/N (Alphanumeric) 2-49
Appendices A-O
  A - ROM BIOS Listing A-1
  B - Assembly Instruction Set Reference B-1
  C - Of Characters, Keystrokes and Color C-1
  D - Logic Diagrams D-1
  E - Unit Specifications E-1
APA (All Points Addressable) 2-45
ASCII - (See American Standard Code for Information Interchange) 1-2
ASCII Coding Table 2-78
ASCII Control Codes 2-79
ASCII (Extended) 3-11
Assembly Instruction Set Reference Appendix B
Assembly Language Reference Guide (See Bibliography)
Assessable Registers 2-154
Asynchronous Communications Adapter
- Block Diagram 2-124
- Current Loop Interface 2-127
- Input/Output Decode 2-125
- Input/Output Signals 2-133
- Input Signals 2-129
- Interface Description 2-126
- Interface Specifications 2-147
- Interrupt 2-126
- Interrupt Enable Register 2-141
- Interrupt Identification Register 2-139
- INS 8250 Accessible Registers 2-134
- INS 8250 Functional Pin Description 2-129
- INS 8250 Line Control Register 2-134
- INS 8250 Programmable Baud Rate Generator 2-135
- Line Status Register 2-137
- Logic Diagram D-48
- Modem Control Register 2-142
- Modem Status Register 2-143
- Modes of Operation 2-125
- Output Signals 2-132
- Programming Considerations 2-133
- Receiver Buffer Register 2-144
- Reset Functions 2-133
- Selecting the Interface Format 2-146
- Transmitter Holding Register 2-145

BASIC (Beginner’s all-purpose symbolic instruction code)
- 80 Interpreter 1-1
- Reserved Interrupts 3-23
- Screen Editor Special Functions 3-19
- Workspace Variables 3-23
- Baud (See INS 8250 Programmable Baud Rate Generator) 2-132
- Baud Out (BAUDOUT) 2-132
- BEL (Bell) 2-82
- Bell (BEL) 2-82
- Berg Pin Connectors 2-63

BIOS
- (Basic Input Output System) 3-2
- Cassette Logic 3-8
- Memory Map 3-7
- Interrupt Vector Listing 3-3
- Parameter Passing 3-2

I-2
BIOS (continued)
  Programming Tip  3-2
  ROM (Read Only Memory)  3-2
  Use of  3-2
  Vectors With Special Meaning  3-5

BIT
  I/O Address  2-42
  I/O Bit Map  2-24
  Output Port Cassette  2-19
  Output Port Speaker  2-22
  Status Register  2-59

Block Diagrams
  Asynchronous Communications Adapter  2-124
  Cassette Motor Control  2-20
  Cassette Interface Read  2-19
  Cassette Interface Write  2-20
  Color/Graphics Monitor Adapter  2-47
  5 1/4” Diskette Drive Adapter  2-90
  Game Control Adapter  2-117
  Keyboard Interface  2-15
  IBM Monochrome Display Adapter  2-38
  Parallel Printer Adapter  2-66
  System  1-4

Bootstrap  3-3

Byte:
  Attribute Definition  2-49
  Display Buffer  2-61

C

CAN (Cancel)  2-81
Cancel (CAN)  2-81
Carriage Return(CR)  2-79
Cassette
  Circuit Block Diagrams  2-19
  Data Record Architecture  3-10
  Error Recovery  3-10
  Interface Connector Specifications  2-21
Interrupt 15  3-8
Jumpers  2-19
Logic (BIOS)  3-8
Read  3-9
Write  3-8
Character
Codes 3-11
Generator 2-48

"Of Characters, Keystrokes and Color Appendix C
Set (00-7F) Quick Reference C-12
Set (80-FF) C-13

Chip Select Out (CSOUT) 2-132
Clear To Send (CTS) 2-130
Clock (4.77 mhz) 2-10
CLK (Clock) 2-10
CNTRL (Control) 3-14
Coding Table (ASCII) 2-78

Color/Graphics Monitor Adapter
Block Diagram 2-47
Character Generator 2-48
Color/Graphics Mode 2-51
Color Select Register 2-57
Composite Color Generator 2-48
Description of Basic Operations 2-54
Display Buffer 2-48
Graphics Storage Map 2-52
I/O Address and Bit Map 2-61
Interrupt Level 2-60
Logic Diagrams D-25
Major Component Definitions 2-48
Memory Requirements 2-60
Mode Register Summary 2-58
Mode Select Register 2-58
Mode Set and Status Registers 2-48
Modes of Operation 2-49
Monitor Adapter Auxiliary Connectors 2-63
Monitor Adapter Direct Drive and Composite Interface
Pin Assignment 2-62
Monochrome Display Adapter Vs. Color/Graphics Adapter
Attribute Relationship 2-51
Motorola 6845 CRT Controller 2-48
Programming the 6845 Controller 2-48
Programming the Mode Control and Status Registers 2-57
6845 Register Description 2-56
Sequence of Events 2-59
Timing Generator 2-48

Composite Phone Jack 2-62
Computer (IBM Personal) 1-1
Command Phase 2-93
Command Status Registers
- Register 0 2-100
- Register 1 2-101
- Register 2 2-102
- Register 3 2-103

Control Codes
- CPS (Characters Per Second) 2-70
- CPU (Central Processing Unit) see System Board, Microprocessor 2-3
- CR (Carriage Return) 2-79

CRT:
- (Cathode Ray Tube) 2-37
- Output Port 1 (I/O Address ‘3B8’) 2-42
- Status Port (I/O Address ‘3BA’) 2-43

CTS (Clear To Send) 2-130
CSOUT (Chip Select Out) 2-132
Current 2-34
Cursor 2-14

D

D0-D7 (Data Bits 0-7) 2-10
DACK0 - DACK3 (DMA Acknowledge 0 to 3) 2-12

DATA
- Bit (D0-D7) 2-10
- Bus Buffer/Driver 2-118
- Input Strobe (DISTR, DISTR) 2-129
- Output Strobe (DOSTR, DOSTR) 2-129
- Rates 2-39
- Record Architecture 3-10
- Set Ready (DSR) 2-131

Data Flow (System) 2-6

DC 1 (Device Control 1) 2-81
DC 2 (Device Control 2) 2-81
DC 3 (Device Control 3) 2-81
DC 4 (Device Control 4) 2-80

DC Output 2-34

DEL (Delete) 2-81
Delete (DEL) 2-81
Device Control 1 (DC 1) 2-81
Device Control 2 (DC 2) 2-81
Device Control 3 (DC 3) 2-81
Device Control 4 (DC 4) 2-80

Digital Output Register (DOR) 2-91
DIN (Connectors) 2-5
DIP (Dual In-Line Package) 2-28
Diskettes 2-111
Diskette Drive (5 1/4") 2-110
Diskette Drive (5 1/4") Adapter
  Adapter Inputs 2-106
  Adapter Outputs 2-107
  Block Diagram 2-90
  Command Status Registers 2-100
  Command Summary 2-96
  Comments (Programming) 2-104
Digital Output Register 2-91
Drive A and B Interface 2-106
Drive Constants 2-104
DPC Registers 2-103
External Interface Specifications 2-109
Floppy Disk Controller 2-91
Functional Description 2-91
Internal Interface Specifications 2-108
Logic Diagrams D-35
Programming Considerations 2-94
Programming Summary 2-103
System I/O Channel Interface 2-104
Display (See IBM Monochrome Display) 2-43
Display Buffer 2-48
DISTR, DISTR (Data Input Strobe) 2-129
Divisor Latch 2-136
DMA (Direct Memory Access) 2-4
Dos Special Functions 3-19
DOSTR, DOSTR (Data Output Strobe) 2-129
Drive Constants 2-104
Drive Disable 2-132
DRQ1 - DRQ3 (DMA Request 1 to 3) 2-12
DSR (Data Set Ready) 2-131
DTR (Data Terminal Ready) 2-132
Dual In-Line Package Switches (DIP) 2-28

Edge Connector 2-108
Encoding 3-11
Error Recovery 3-10
ESC (Escape) 2-82
ESC A (Escape A) 2-83
ESC B (Escape B) 2-84
ESC C (Escape C) 2-85
ESC D (Escape D) 2-85
ESC E (Escape E) 2-86
ESC F (Escape F) 2-86
ESC G (Escape G) 2-86
ESC H (Escape H) 2-87
ESC 0 (Escape 0) 2-82
ESC 1 (Escape 1) 2-82
ESC 2 (Escape 2) 2-83
ESC 8 (Escape 8) 2-83
ESC 9 (Escape 9) 2-83
Escape (ESC) 2-82
Escape A (ESC A) 2-83
Escape B (ESC B) 2-84
Escape C (ESC C) 2-85
Escape D (ESC D) 2-85
Escape E (ESC E) 2-86
Escape F (ESC F) 2-86
Escape G (ESC G) 2-86
Escape H (ESC H) 2-87
Escape 0 (ESC 0) 2-82
Escape 1 (ESC 1) 2-82
Escape 2 (ESC 2) 2-83
Escape 8 (ESC 8) 2-83
Escape 9 (ESC 9) 2-83
Execution Phase 2-93
Extended Codes 3-13

F

FDC (Floppy Disk Controller) 2-91
Floppy Disk Controller (FDC) 2-91
Fonts 2-48
Functions and Conditions of DIP Switch 1 2-72
Functions and Conditions of DIP Switch 2 2-73
Functional Description
5 1/4” Diskette Drive Adapter 2-91
Game Control Adapter 2-118
INS 8250 2-129

G

Game Control Adapter
Address Decode 2-118
Block Diagram 2-117
Game Control Adapter (continued)
  Connector Specifications  2-122
  Data Bus Buffer/Driver  2-118
  Functional Description  2-118
  Interface Description  2-119
  Joystick Positions  2-118
  Joystick Schematic  2-121
  Trigger Buttons  2-118
Glossary  G-1
GND (Ground)  2-12
Graphics Character Extensions (Interrupt 1FH)  3-6
Graphics Mode (Color)  2-51
Graphics Storage Map  2-52

H

Hardware
  Asynchronous Communications Adapter  2-123
  Color/Graphics Monitor Adapter  2-45
  5 1/4” Diskette Drive  2-110
  5 1/4” Diskette Drive Adapter  2-89
  Game Control Adapter  2-117
  IBM Monochrome Display  2-43
  IBM Monochrome Display and Parallel Printer Adapter  2-37
  Memory Expansion Options 32KB and 64KB  2-113
  Printer  2-70
  Parallel Printer Adapter  2-65
  Power Supply  2-33
  System Board  2-3

Hardware
  Overview  1-1
  Data Flow  2-6
Hertz (Hz)  1-2
Horizontal Drive  2-43
Horizontal Tab (HT)  2-81
HT (Horizontal Tab)  2-81

I

IBM 80 CPS Matrix Printer  2-70
IBM Monochrome Display  2-43
IBM Monochrome Display and Parallel Printer Adapter
  Block Diagram  2-38
  Data Rates  2-39
  Direct Drive Interface and Pin Assignment  2-44
I-8
IBM Monochrome Display and Parallel Printer Adapter (continued)

DMA Channel 2-42
Interrupt and DMA Response Requirements 2-39
Interrupt Levels 2-42
I/O Address and Bit Map 2-42
Lines Used 2-39
Loads 2-39
Memory Requirements 2-41
Modes of Operation 2-40
Programming the 6845 CRT Controller 2-41
Sequence of Events 2-41

Important Operating Characteristics 2-36
IER (Interrupt Enable Register) 2-141
IIR (Interrupt Identification Register) 2-139

Input/Output Signals:
Data (D0, D7) 2-133
External Clock Input/Output (XTAL1, XTAL2) 2-133

Input Requirements (Power Supply) 2-34

Input Signals
Address Strobe (ADS) 2-129
Chip Select (SC0, CS1, CS2) 2-129
Clear to Send (CTS) 2-130
Data Input Strobe (DISTR, DISTR) 2-129
Data Output Strobe (DOSTR, DOSTR) 2-129
Data Set Ready (DSR) 2-131
Master Reset (MR) 2-130
Received Line Signal (RCLK) 2-130
Receiver Clock (RCLK) 2-130
Register Select (A0, A1, A2) 2-130
Ring Indicator (RI) 2-131
Serial Input (SIN) 2-130

Interface Diagram
Asynchronous Communications Adapter 2-147
Cassette Connector Specifications 2-21
Color/Graphics Monitor Adapter 2-62
5 1/4" Diskette Drive Adapter External 2-109
5 1/4" Diskette Drive Adapter Internal 2-108
Game Control 2-122
IBM Monochrome Display Direct Drive 2-44
Keyboard Connector Specifications 2-18
Parallel Printer 2-69
INTRPT (Interrupt) 2-132
Interrupt Control Functions 2-140
Intel 8048 (Keyboard Microcomputer) 2-14
Intel 8088 (System Unit Microprocessor) 2-3
I/O (Input/Output)
  Address Map  2-33
  Channel  2-8
  Channel Description (System Board)  2-10
  Diagram  2-9
I/O CH CK (I/O Channel Check)  2-10
I/O CH RDY (I/O Channel Ready)  2-11
IOR (I/O Read Command)  2-11
IOW (I/O Write Command)  2-11
Interrupts
  And DMA Response Requirements  2-39
  Enable Register  2-141
  Identification Register  2-139
  Levels (0-XX)  2-42
  Vector Listing  3-3
  Vectors (0-7F)  3-21
  1 CH - Timer Tick  3-5
  1 DH - Video Parameters  3-5
  1 EH - Diskette Parameters  3-5
  1 FH - Graphics Character Extensions  3-6
  15  3-8

K

KB - Kilobyte (See Memory Expansion Options)
Keyboard
  Break  3-16
  Character Codes  3-11
  Diagram  2-16
  Encoding  3-11
  Extended Codes  3-13
  Extended Functions  3-13
  Interface Block Diagram  2-15
  Interface Connector Specifications  2-18
  Pause  3-16
  Print Screen  3-16
  Scan Codes  2-17
  Shift States  3-14
  Shift Key Priorities  3-15
  Special Handling  3-15
  System Reset  3-15
  Usage  3-17
Kilobyte (KB) (See Memory Expansion Options)
L

LF (Line Feed) 2-79
Light Pen
   Interface 2-63
   Mode Control and Status Register 2-57
   Register Description 2-56
Line Control Register (LCR) 2-134
Line Feed 2-79
Line Status Register (LSR) 2-137
Lines Used 2-39
Loads 2-39
Logic Diagrams Appendix D
Low Memory Maps
   BASIC and DOS Reserved Interrupts (80-3FF) 3-22
   BASIC Workspace Variables 3-23
   Interrupt Vectors (0-7F) 3-21
   Reserved Memory Locations (400-5FF) 3-22

M

Major Component Definitions 2-48
Matrix Printer (IBM 80 CPS) 2-70
Megabyte 2-3
Memory
   BIOS Map 3-7
   Map (System) 2-25
   Module Description 2-114
   Module Pin Configuration 2-114
   Other Read/Write Usage 3-6
   Requirements (Color/Graphics) 2-60
   Requirements (IBM Monochrome) 2-41
   System Board Switch Settings 2-30
   32/64 KB Expansion Option Switch settings 2-31
Memory Expansion Options
   Memory Module Description 2-14
   Memory Module Pin Configuration 2-114
   Operating Characteristics 2-113
   Switch Configurable Starting Address 2-115
MEMR (Memory Read Command) 2-11
MEMW (Memory Write Command) 2-11
MFM (Modified Frequency Modulation) 2-110
Mhz (Megahertz) 2-43
Microprocessor 2-3
Microsecond 2-3
Mnemonic B-18
Mode Set and Status Register 2-48
Modem Control Register 2-142
Modem Status Register 2-143
Modes of Operation
   Asynchronous Communications Adapter 2-125
   Color/Graphics Monitor Adapter 2-49
   IBM Monochrome Display Adapter 2-40
Mode Register Summary 2-58
Mode Select Register 2-58
Monitor Type Switch Settings 2-29
Monochrome Display (IBM) 2-43
Monochrome Display and Parallel Printer Adapter (See IBM)
Motorola 6845 CRT Controller 2-48
MR (Master Reset) 2-130

N

NMI (Non-Maskable Interrupt)
   of the 8088 2-4
   to the 8088 2-8
NMI Mask. Reg. - (I/O Address Map) 2-23
Nominal Power Requirements 2-24
NULL (Null) 2-82
Null (NUL) 2-82
NUM LOCK 3-14

O

Of Characters, Keystrokes and Color  C-1
OHM Resistors 2-67
Operating Characteristics
   Memory Expansion Options 2-113
   Monochrome Display 2-43
   Power Supply 2-36
Options
   Asynchronous Communications Adapter 2-12
   Color/Graphics Monitor Adapter 2-45
   5 1/4” Diskette Drive 2-110
   5 1/4” Diskette Drive Adapter 2-89
   Game Control Adapter 2-117
   IBM 80 CPS Matrix Printer 2-70
   IBM Monochrome Display 2-43
   32/64 KB Memory Expansion Options 2-113

I-12
Options (continued)

Parallel Printer Adapter 2-65
"ORed" 2-65
OSC (Oscillator) 2-10
Other Read/Write Memory Usage 3-6
OUT PORT 2-39

Output

AC 2-34
Address 2-67
DC 2-34
Port 2-65
OUT 1 2-132
OUT 2 2-132

Output Signals

Baud Out (BAUDOUT) 2-132
Chip Select Out (CSOUT) 2-132
Data Terminal Ready (DTR) 2-132
Driver Disable (DDIS) 2-132
Interrupt (INTRPT) 2-132
Output 1 (OUT1) 2-132
Output 2 (OUT2) 2-132
Request to Send (RTS) 2-132
Serial Output (SOUT) 2-132

Overview (Hardware) 1-1
Over Voltage/Current Protection 2-36

P

Parallel Printer Adapter

ASCII Coding Table 2-78
ASCII Control Codes 2-79
Block Diagram 2-66
Description 2-37
DMA Channel 2-42
IBM 80 CPS Matrix Printer 2-70
I/O Address and Bit Map 2-42
Interrupt Levels 2-42
Logic Diagram D-31
Parallel Interface Description 2-73
Printer Specifications 2-71
Programming Considerations 2-67
Programming the 6845 CRT Controller 2-41
Sequence of Events 2-41
Setting the DIP Switches 2-72
Timing 2-77
Parameters, 6845 Initialization 2-41
Parameter Passing, ROM BIOS 3-2
Parity Flag, 8080 Flags B-1
Pause, BIOS Cassette Logic Special Handling 3-16
Pin Connectors
P2-6 Pin Berg Strip for Light Pen Connector 2-63
P1-4 Pin Berg Strip for RF Modulator 2-63
9 Pin Connector, Color Direct Drive 2-61
9 Pin Connector, IBM Monochrome Display 2-44
25 Pin Connector, Parallel Printer Adapter Block Diagram 2-66
25 Pin ‘D’ Shell Connector, Asynchronous Adapter Block
  Diagram 2-124
15 Pin ‘D’ Shell Connector, Game Controller Adapter (Analog
  Input Connector Specifications) 2-122
25 Pin ‘D’ Shell Connector, Parallel Printer Adapter 2-69
5 Pin Din Connector, Keyboard Interface Connector
  Specifications 2-18
15 Pin Male ‘D’ Shell Connector, Joystick Schematic 2-121
Planar (See System Board 2-3; Intel 8088 2-3)
Power-on Self-Test
  System Board 2-4
  Keyboard 2-14
Power Supply 2-33
  AC Output 2-34
  DC Output 2-34
  Important Operating Characteristics 2-36
  Over Voltage/Current Protection 2-36
  Signal Requirements 2-36
  Input Requirements 2-34
  Power Supply Connectors and Pin Assignments 2-35
  Power Supply Location 2-34
Preface i
Print Screen, Special Handling 3-16
Printer, IBM 80 CPS Matrix 2-70
Printer Specifications 2-71
Programmable Peripheral Interface (PPI) 8255A-5 2-19
Programming Considerations
  Asynchronous Communications Adapter
    Asynchronous Communications Reset Functions 2-133
    INS 8250 Accessable Registers 2-134
    INS 8250 Line Control Register 2-134
    INS 8250 Programmable Baud Rate Generator 2-135
    Interrupt Identification Register 2-139
    Interrupt Enable Register 2-141
    Line Status Register 2-137
    Modem Control Register 2-142
Programming Considerations
Asynchronous Communications Adapter (continued)
  Modem Status Register  2-143
  Receiver Buffer Register  2-144
  Transmitter Holding Register  2-145
Color/Graphics Monitor Adapter
  Color Select Register  2-57
  Control and Status Register  2-57
  I/O Address and Bit Map  2-61
  Interrupt Level  2-60
  Mode Register Summary  2-58
  Mode Select Register  2-58
  Programming the 6845 Controller  2-55
  Programming the Modem Control and Status Register  2-57
  6845 Register Descriptions  2-56
  Status Register  2-59
  Sequence of Events  2-59
5 1/4” Diskette Drive Adapter
  Command Status Registers  2-100
  Command Summary  2-96
  Symbol Descriptions  2-94
IBM Monochrome Display and Parallel Printer Adapter
  DMA Channel  2-42
  I/O Address and Bit Map  2-42
  Interrupt Levels  2-42
  Memory Requirements  2-41
  Sequence of Events  2-41
  Programming the CRT Controller  2-41
  Programming the 6845 CRT Controller  2-41
  Comments  2-104
  DPC Registers  2-103
  Drive Constants  2-104

R
  RAS  2-114
  Rating Amps, Over Voltage/Current Protection  2-36
  Ready Line  2-8
  Read Block  3-9
  Read Data  2-107, 2-108 (5 1/4” Diskette Drives, External)  2-109
  Read Status (Parallel Printer Adapter Block Diagram)  2-66
  Read/Write Memory
    Color Monitor  2-50
    Color TV  2-49
    Future Expansion in I/O Channel, 384 KB  2-26
Read/Write Memory (continued)
  Graphics Storage Map  2-53
  Hardware Overview  1-1
  I/O Address Map  2-24
  Memory Address Space  2-61
  Memory Expansion Options  2-113
  User  3-7
  Receive Circuit  2-127
  Receiver Clock, (RCLK)  2-128
  Recording Medium  2-111
  Refresh Cycles  2-8
  Registers, Address
    Command Status  2-100
    Data  2-9
    Initialization
      Parameters  2-41
      INS 8250 Accessible  2-134
      INS 8250 Line Control  2-134
      Interrupt Enable  2-141
      Line Control  2-134
      Line Status  2-137
      Main Status  2-9
      Modem Control  2-142
      Modem Status  2-143
      Receiver Buffer  2-144
      Transmitter Holding  2-145
  6845  2-55
  Register File
    Color Select  2-57
    6845 Data  2-41
    Description  2-56
    6845 Index  2-42
    6845 Initialization Parameters  2-41
    Mode Select  2-58
    Status  2-57, 2-59
  Requirements
    Input (Power Supply)  2-34
    Memory  2-60
    Signal (Power Supply)  2-36
  RESET DRV (Reset Drive)  2-10
    I/O Channel Description  2-119
  ROM Address Space, 256 KB  2-25
  =RESET  2-105
  Reserved Memory Locations (400-5FF)  3-22
  Result Phase  2-93
Response Requirements, Interrupt and DMA  2-39
Reverse Video
   Modes of Operation  2-40
   Color Graphics Monitor Adapter  2-45
RF Modulator
   Auxiliary Video Connector P1-4 Pin Berg Strip  2-63
   Color Graphics Monitor Adapter  2-45
   Interface  2-63
Ring Indicator  2-127
ROM
   Character Generator  2-48
   Color Graphics Mode  2-52
   Color Graphics Monitor Adapter  2-46
   Diagram  2-13
   Hardware Overview  1-1
   Keyboard  2-14
   Memory Expansion Options  2-113
   System Board  2-3, 2-4
   System Board Component Diagram  2-13
   System Memory Map  2-27
ROM and System Usage  3-1
ROM BIOS
   BIOS Cassette Logic  3-8
   BIOS Memory Map  3-8
   Cassette Read  3-9
   Cassette Write  3-9
   Data Record Architecture  3-10
   Description  3-2
   Error Recovery  3-10
   Interrupt 15  3-8
   Interrupt ICH Timer Tick  3-5
   Interrupt IDH Video Parameters  3-5
   Interrupt IEH Diskette Parameters  3-5
   Interrupt IFH Graphics Character Extensions  3-6
   Interrupt Vector Listing  3-3
   Other Read/Write Memory Usage  3-7
ROM BIOS Listing  Appendix A
ROS (Read Only Storage) (See ROM)
ROM, Request for Master  2-92
RS232-C (See Asynchronous Communications Adapter)  2-123
RTS (Ready to Send)  2-123
R/W (Read/Write)
   Symbol Description  2-95
Scan Codes 2-17
Screen 2-43
Schematic (See Logic Diagrams)
SCROLL LOCK 3-15
Selecting the Interface Format 2-146
Sequence of Events 2-59
Serial Input (SIN) 2-130
Serial Output (SOUT) 2-132
Setting the DIP Switches 2-72
Shift In 2-80
Shift Out 2-80
Shift States 3-14
SI (Shift In) 2-80
Signal Requirements 2-36
SIN (Serial Input) 2-130
SO 2-80
Software Algorithms 3-8
SOUT (Serial Output) 3-132
Speaker
  Drive System Block Diagram 2-22
  Interface 2-22
Special Handling 3-15
Special Timing 2-39
Specifications
  5 1/4" Diskette Drive 2-112
  Printer 2-71
  System Appendix E
Status Registers
  Color/Graphics 2-59
  0 (5 1/4" Diskette Drive Adapter) 2-100
  1 (5 1/4" Diskette Drive Adapter) 2-101
  2 (5 1/4" Diskette Drive Adapter) 2-102
  3 (5 1/4" Diskette Drive Adapter) 2-103
Storage (See Memory)
Strobe
  Address 2-129
  Data Output 2-129
Summary of Available Colors 2-55
Switch Settings
  Configurable Start Address 2-115
  5 1/4” Diskette Drives 2-29
  32/64 KB Memory Expansion Option 2-31
  Monitor Type 2-29
  System Board Memory 2-30
I-18
System Board
   Cassette Interface Connector Specifications  2-21
   Cassette Jumpers  2-19
   Cassette User Interface  2-19
   Circuit Block Diagrams (Cassette)  2-19
   Component Diagram  2-13
   Data Flow  2-6
   5 1/4" Diskette Drive Switch Settings  2-29
   I/O Address Map  2-23
   I/O Channel  2-8
   I/O Channel Description  2-10
   I/O Channel Diagram  2-9
   Keyboard  2-14
   Keyboard Diagram  2-16
   Keyboard Interface Block Diagram  2-15
   Keyboard Interface Connector Specifications  2-18
   Keyboard Scan Codes  2-17
   32/64 KB Memory Expansion Option Switch Settings  2-31
   Memory Switch Settings  2-30
   Monitor Type Switch Settings  2-29
   Speaker Drive System Block Diagram  2-22
   Speaker Interface  2-22
   System Memory Map  2-25
   System Expansion Slots (See I/O Slots)  2-3
   System Memory Map  2-25
   System Memory Map (16 KB Increments)  2-26
   System Unit  1-1
   System Unit Power Connector  2-35
   System Usage (ROM and)  3-1

T

   Tab (Vertical)  2-79
   T/C (Terminal Count)  2-12
   Terminal Count (T/C)  2-12
   THR (Transmitter Holding Register)  2-145
   Timing Generator  2-48
   Transmit Circuit  2-127
   Transmit Data (TX)  2-127
   Transmitter Holding Register (THR)  2-145
   Transmitter Output and Receiver Input  2-126
   Trigger Buttons  2-118
   TX Data (Transmit Data)  2-127
U

Unit Specifications  Appendix E
Usage (Keyboard)  3-17
Use of BIOS  3-2
User Interface (Cassette)  2-19

V

Vectors
(0-7F Interrupt)  3-21
(Interrupt Listing)  3-3
Vectors With Special Meaning
  Interrupt 1CH Timer Tick  3-5
  Interrupt 1DH Video Parameters  3-5
  Interrupt 1EH Diskette Parameters  3-5
  Interrupt 1FH Graphics Character Extensions  3-6
  Other Read/Write Memory Usage  3-6
Vertical Drive  2-43
Video
  Monitor  2-62
  (Reverse)  2-45
  Signal  2-43
Voltage Interchange Information  2-128
Voltage
  Power Supply  2-33
  I/O Channel  2-8
  System Board I/O Channel Description  2-10
  Keyboard Interface Connector Specifications  2-18
  Cassette Interface Connector Specifications  2-21
  Speaker Interface  2-22
  Power Supply  2-33
    Power Supply Location  2-34
    Important Operating Characteristics  2-36
  Color Graphics Monitor Adapter Direct Drive and
  Composite Interface Pin Assignment  2-62
  Color/Graphics Monitor Adapter Auxiliary Video
  Connectors  2-62
  Printer Specifications  2-71
    Game Controller Adapter (Analog Input) Connector
    Specifications  2-122
  Voltage Interchange Information, Asynchronous Communications
  Adapter  2-128
  Selecting the Interface Format, Asynchronous Communications
  Adapter  2-146
W

Workspace (BASIC Variables)  3-23
Write (Cassette)  3-8
Write (Cassette Interface Hardware)  2-20
Write Data  2-106
Write Enable  2-106
Write Protect  2-107

Numerics

5 1/4" Diskette Drive  2-110
5 1/4" Diskette Drive Adapter  2-91
32/64 KB Memory Expansion Options  1-3
80 CPS Matrix Printer, IBM  2-70
80 Interpreter, BASIC  1-1
6845 CRT Controller  2-48
8080 Parity Flags  B-1
8088, Intel  2-3
8250 INS Accessible Registers  2-134
RS232C-A Asynchronous Communications Adapter  2-123
Product Comment Form

TECHNICAL REFERENCE 6025008

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.

Comments:

If you wish a reply, provide your name and address in this space.

Name ________________________________
Address _______________________________
City ___________ State _________________
Zip Code ___________