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

Personal Computer
Hardware Reference
Library

Technical Reference

6025008

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.

Personal Computer
Hardware Reference
Library

Technical Reference

FEDERAL COMMUNICATIONS COMMISSION RADIO FREQUENCY INTERFERENCE STATEMENT

WARNING: This equipment has been certified to comply with the limits for a Class B computing device, pursuant to Subpart J of Part 15 of FCC rules. Only peripherals (computer input/output devices, terminals, printers, etc.) certified to comply with the Class B limits may be attached to this computer. Operation with non-certified peripherals is likely to result in interference to radio and TV reception.

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.

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

FIGURE LISTING

TABLE LISTING

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 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).

System Block Diagram

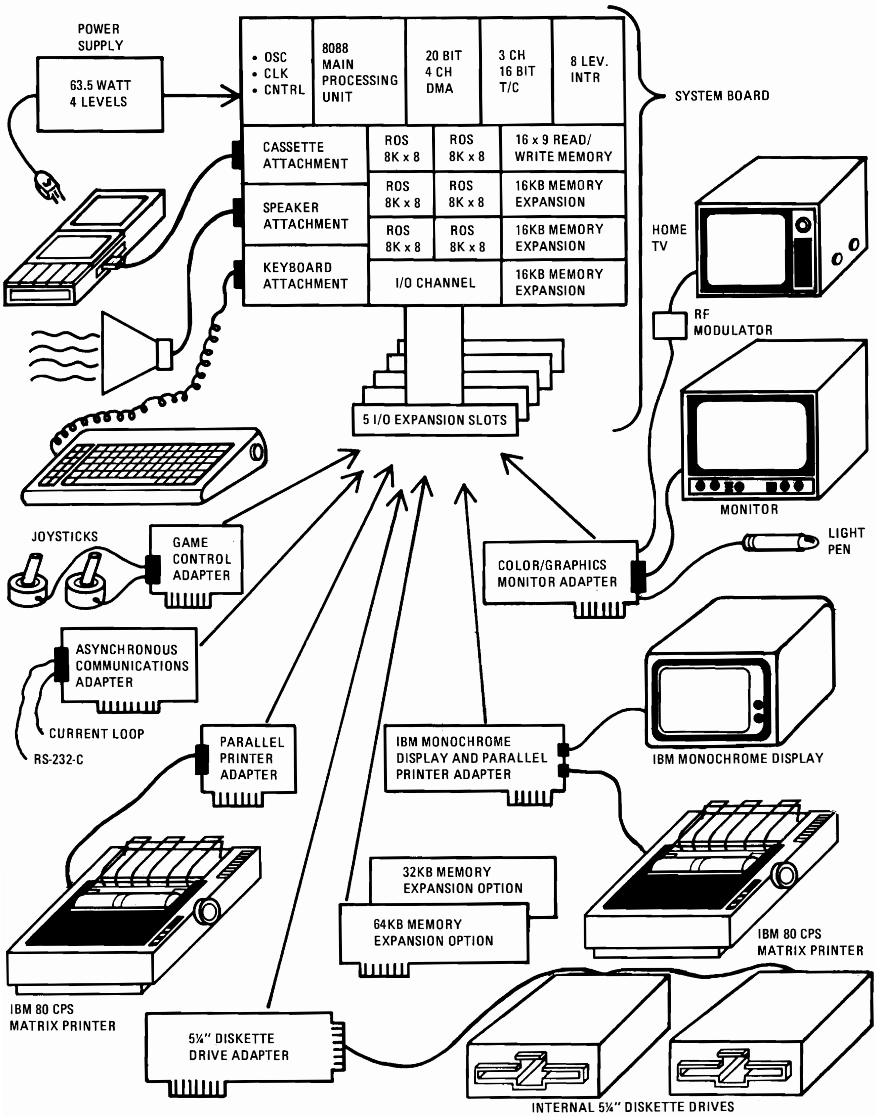


Figure 1. SYSTEM BLOCK DIAGRAM

SECTION 2. HARDWARE

Contents:

HARDWARE

NOTES

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).

HARDWARE

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 inn 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

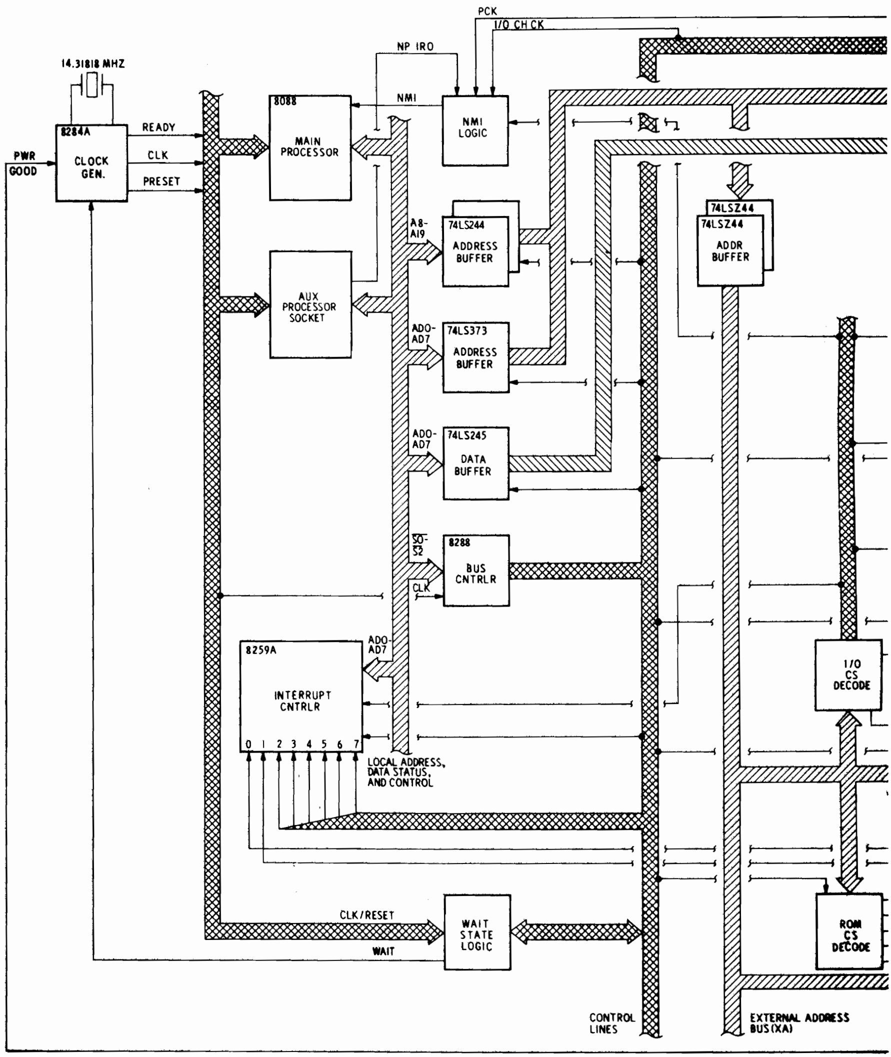


Figure 2. SYSTEM BOARD DATA FLOW (SHEET 1 OF 2)

System Board Data Flow

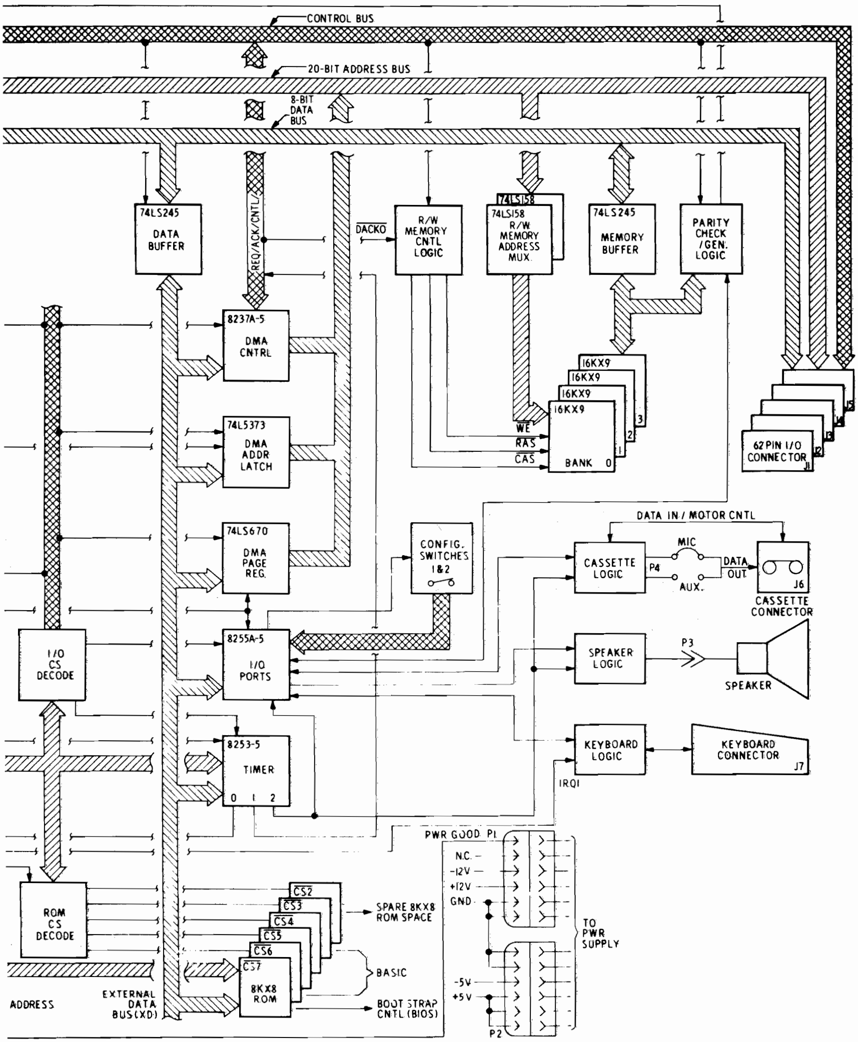


Figure 2. SYSTEM BOARD DATA FLOW (SHEET 2)

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.

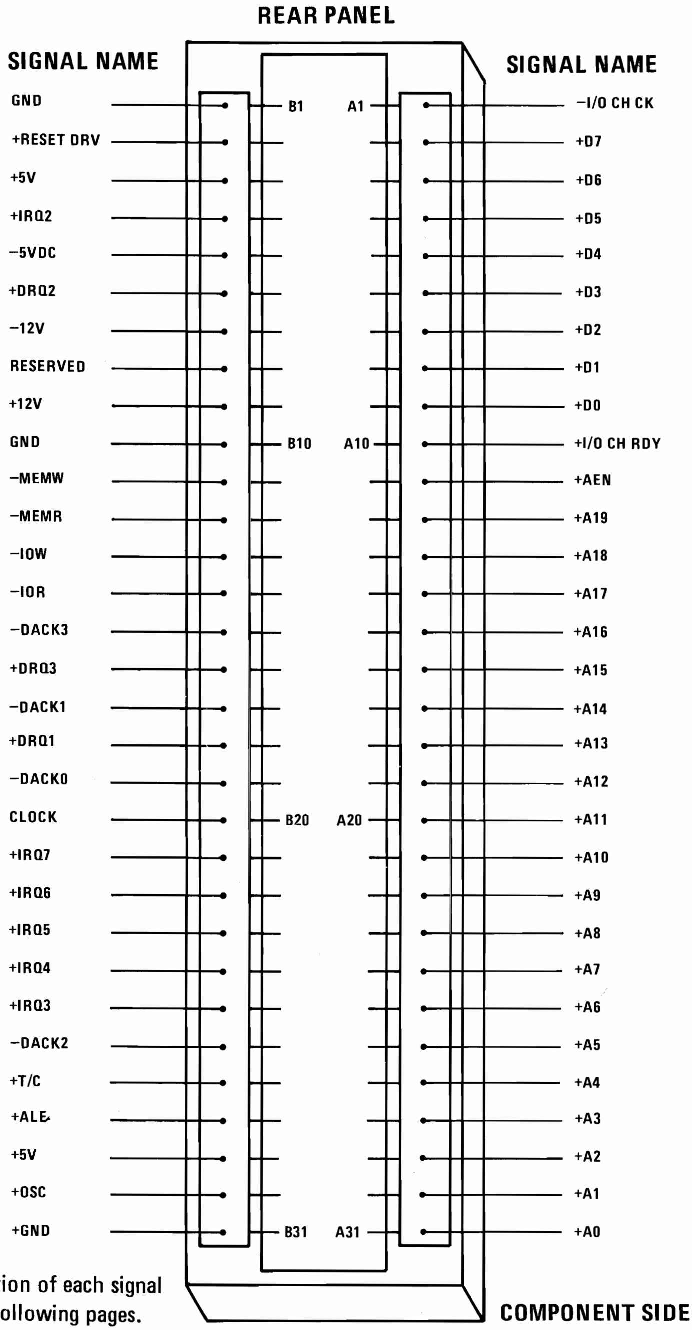
A ready line is available on the I/O channel to allow operation with slow I/O or memory devices. If the channel's Ready line is not activated by an addressed device, all processor generated memory read and write cycles take four 210 ns clock or 840 ns/byte. All processor-generated I/O read and write 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.

I/O Channel Diagram



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.

Signal	I/O	Description
OSC	O	Oscillator: This signal is a high speed clock with a 70 nsec. period (14.31818 MHz.) It has a 50% duty cycle.
CLK	O	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.
RESET DRV	O	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.
A0-A19	O	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.
D0-D7	I/O	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.
ALE	O	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.
$\overline{\text{I/O CH CK}}$	I	-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.

I/O CH RDY	I	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	I	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).
$\overline{\text{IOR}}$	O	-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.
$\overline{\text{IOW}}$	O	-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.
$\overline{\text{MEMR}}$		-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.
$\overline{\text{MEMW}}$	O	-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.
$\overline{\text{DACK0}}$-$\overline{\text{DACK3}}$	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.

System Board Component Diagram

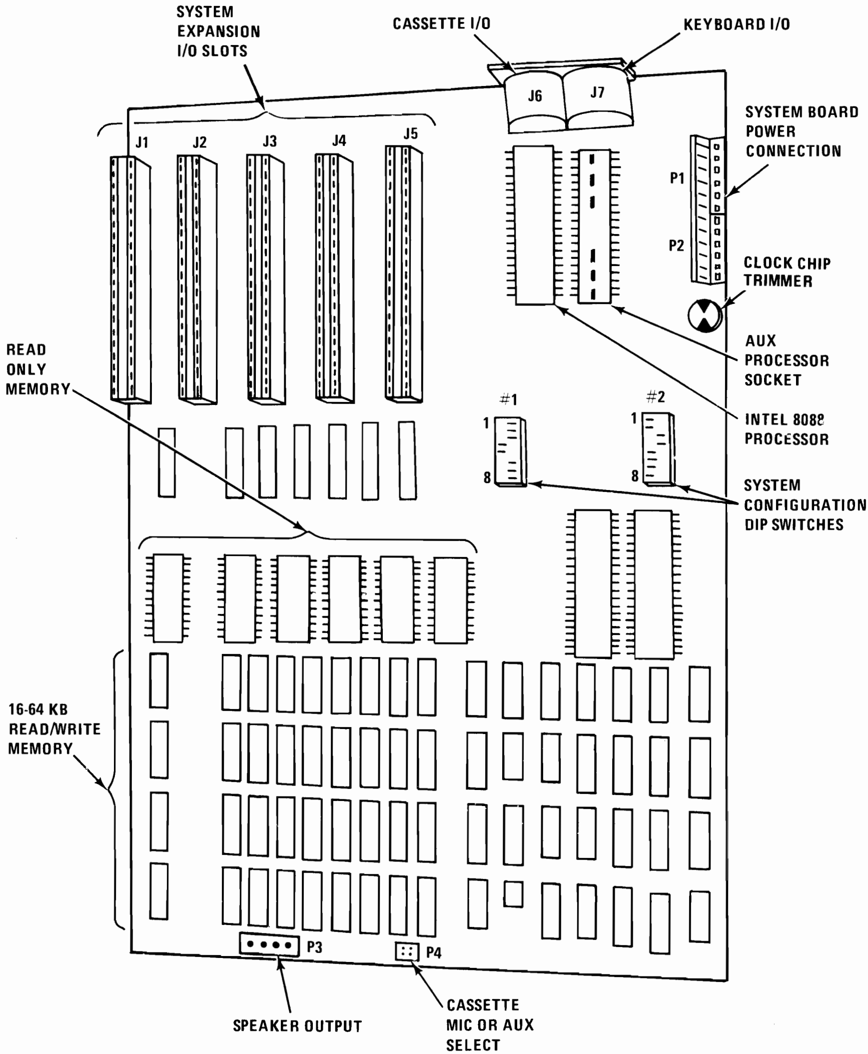


Figure 4. SYSTEM BOARD COMPONENT DIAGRAM

HARDWARE

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.

Keyboard Interface Block Diagram

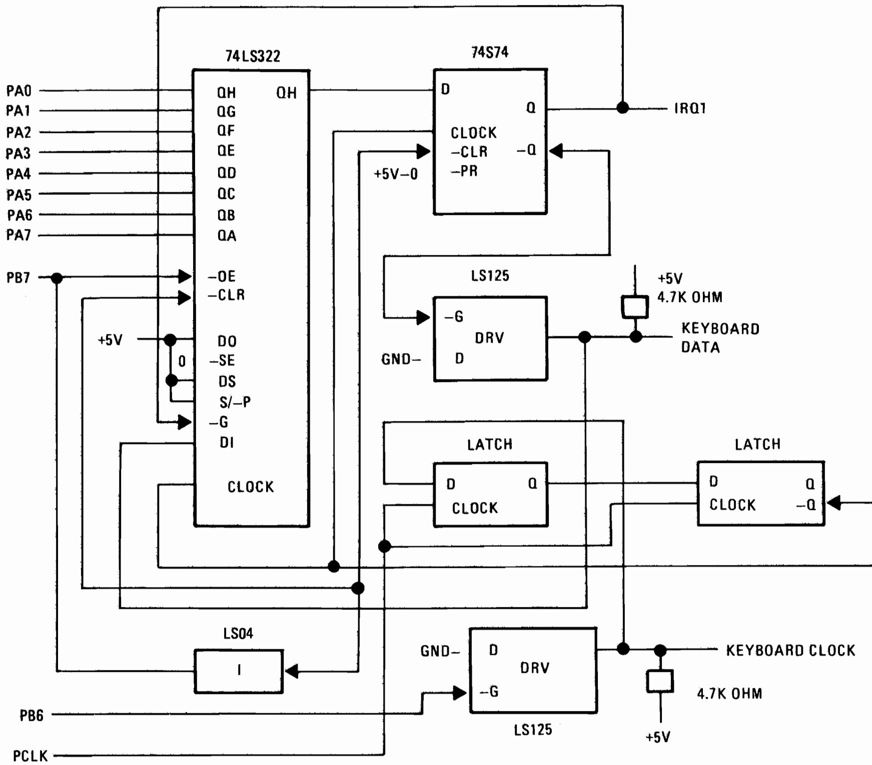
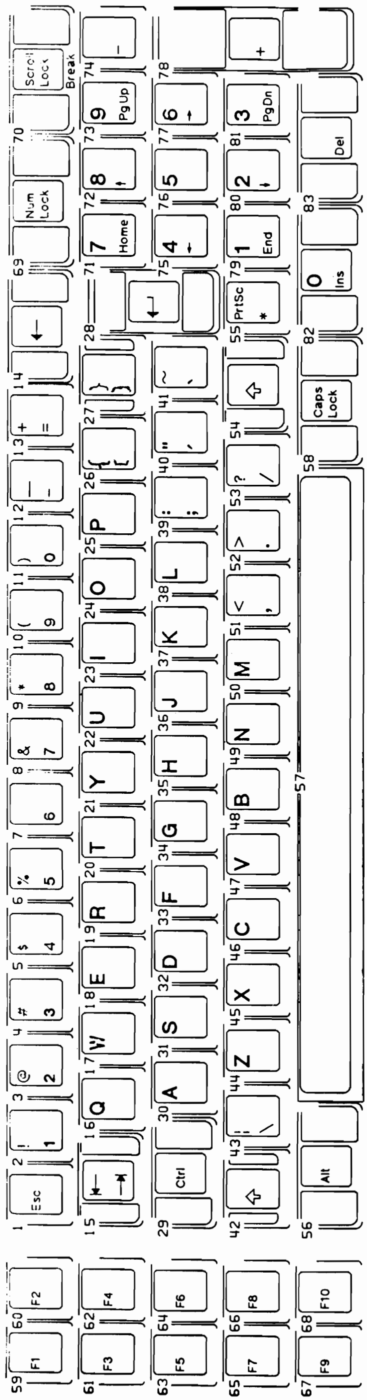


Figure 5. KEYBOARD INTERFACE BLOCK DIAGRAM

HARDWARE

Keyboard Diagram



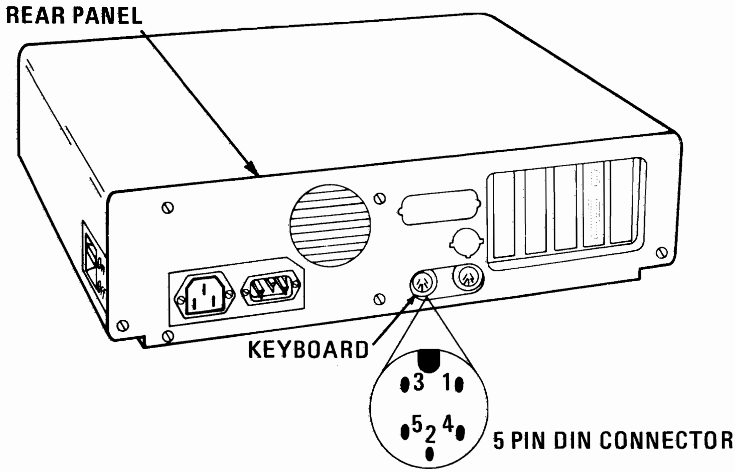
NOTE
1 NOMENCLATURE IS ON BOTH TOP AND FRONT FACE OF KEYBUTTON AS SHOWN. THE NUMBER TO THE UPPER LEFT DESIGNATES THE BUTTON POSITION.

Figure 6. KEYBOARD DIAGRAM

Table 1. Keyboard Scan Codes

Key Position	Scan Code in Hex	Key Position	Scan Code in Hex
1	01	43	2B
2	02	44	2C
3	03	45	2D
4	04	46	2E
5	05	47	2F
6	06	48	30
7	07	49	31
8	08	50	32
9	09	51	33
10	0A	52	34
11	0B	53	35
12	0C	54	36
13	0D	55	37
14	0E	56	38
15	0F	57	39
16	10	58	3A
17	11	59	3B
18	12	60	3C
19	13	61	3D
20	14	62	3E
21	15	63	3F
22	16	64	40
23	17	65	41
24	18	66	42
25	19	67	43
26	1A	68	44
27	1B	69	45
28	1C	70	46
29	1D	71	47
30	1E	72	48
31	1F	73	49
32	20	74	4A
33	21	75	4B
34	22	76	4C
35	23	77	4D
36	24	78	4E
37	25	79	4F
38	26	80	50
39	27	81	51
40	28	82	52
41	29	83	53
42	2A		

Keyboard Interface Connector Specifications



PIN	SIGNAL
1	+ Keyboard Clock
2	+ Keyboard Data
3	- Keyboard Reset (Not used by keyboard)
4	Ground
5	+5 Volts

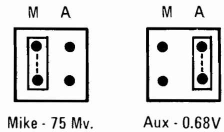
Cassette User Interface

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 casette 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.

JUMPER DIAGRAM



Circuit Block Diagrams

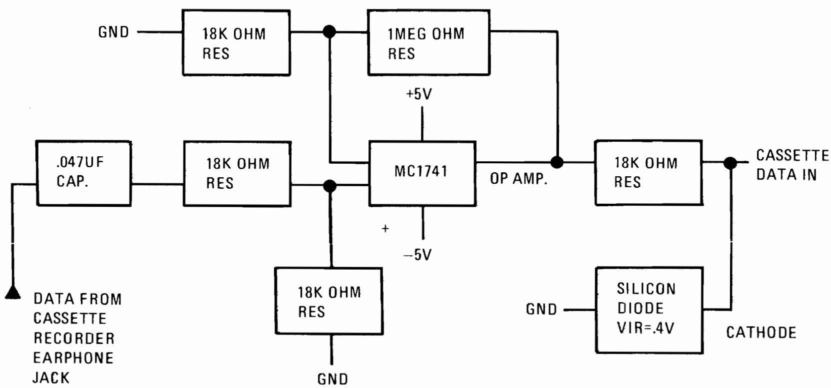


Figure 7. CASSETTE INTERFACE READ HARDWARE

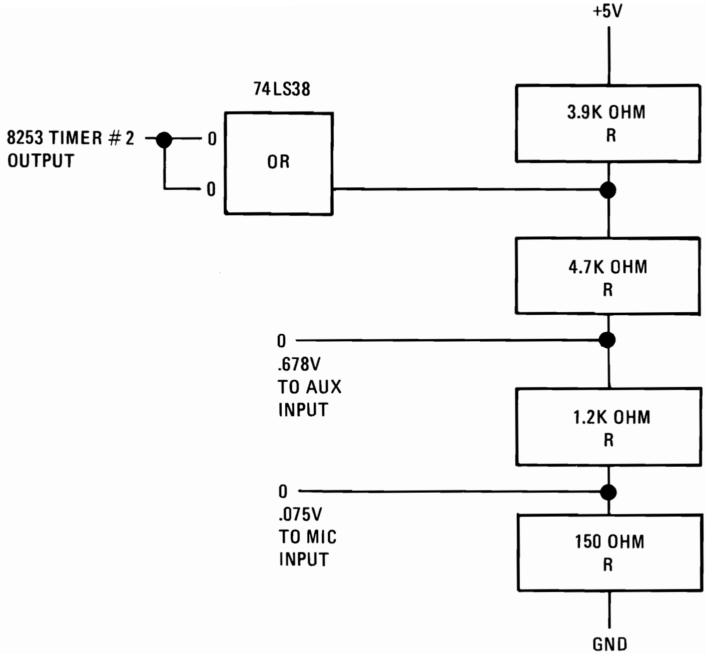


Figure 8. CASSETTE INTERFACE WRITE HARDWARE

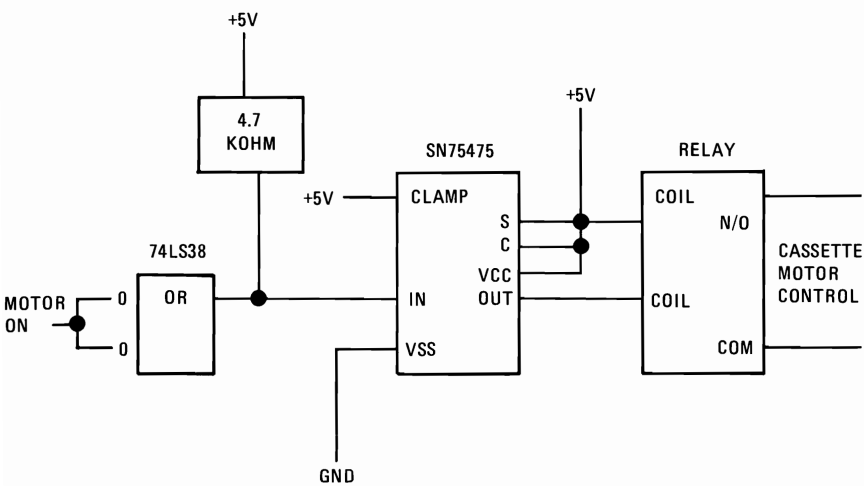
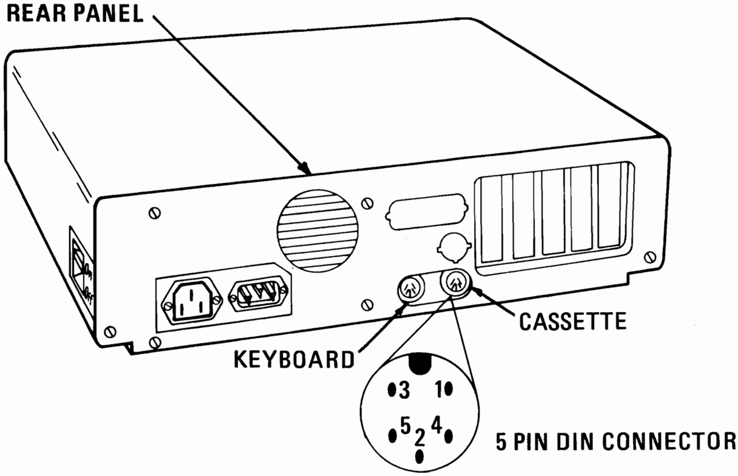


Figure 9. CASSETTE MOTOR CONTROL

Cassette Interface Connector Specifications



PIN	SIGNAL	ELECTRICAL CHARACTERISTICS*
1	Motor Control	Common from Relay
2	Ground	
3	Motor Control	6 VDC; 1A (Relay N.O.)
4	Data In	500nA at ±13V - at 1,000 - 2,000 Baud
5	Data Out (Mic or Aux)	250 μA at { .68V or 75mv } **

*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.

HARDWARE

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.

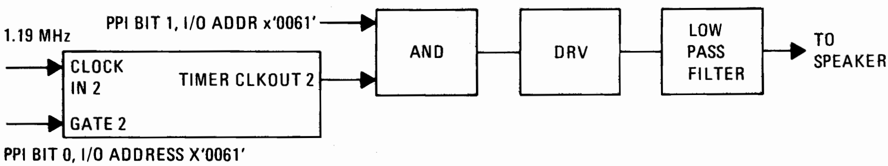


Figure 10. SPEAKER DRIVE SYSTEM BLOCK DIAGRAM

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.

PIN	FUNCTION
1	DATA
2	KEY
3	GROUND
4	+5 VOLTS

I/O Address Map

HEX RANGE	9 8	7 6 5 4	3	2	1	0	DEVICE
00–0F	0 0	0 0 0 Z	A3	A2	A1	A0	DMA CHIP 8237–2
20–21	0 0	0 0 1 Z	Z	Z	Z	A0	INTERRUPT 8259A
40–43	0 0	0 1 0 Z	Z	Z	A1	A0	TIMER 8253–5
60–63	0 0	0 1 1 Z	Z	Z	A1	A0	PPI 8255A–5
80–83	0 0	1 0 0 Z	Z	Z	A1	A0	DMA PAGE REGS
* AX	0 0	1 0 1					NMI MASK REG
CX	0 0	1 1 0					RESERVED
EX	0 0	1 1 1					RESERVED
3F8–3FF	1 1	1 1 1 1	1	A2	A1	A0	TP RS–232–C CD
3F0–3F7	1 1	1 1 1 1	0	A2	A1	A0	5 1/4" DRV ADAPTER
2F8–2FF	1 0	1 1 1 1	1	A2	A1	A0	RESERVED
378–37F	1 1	0 1 1 1	1	Z	A1	A0	PARALLEL PRTR PRT
3D0–3DF	1 1	1 1 0 1	A3	A2	A1	A0	COLOR/GRAPHICS ADAPTER
278–27F	1 0	0 1 1 1	1	Z	A1	A0	RESERVED
200–20F	1 0	0 0 0 0	A3	A2	A1	A0	GAME I/O ADAPTER
3B0–3BF	1 1	1 0 1 1	A3	A2	A1	A0	IBM MONOCHROME DISPLAY PARALLEL PRINTER ADAPTER

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

Port		Bit					
X'0060'	INPUT	PA0	+KBD SCAN CODE 0	OR	IPL 5 1/4 DISKETTE DRIVE		(SW1–1)
		1	1		RESERVED		(SW1–2)
		2	2		SYS. BD. READ/WRITE MEMORY SIZE	*	(SW1–3)
		3	3		SYS. BD. READ/WRITE MEMORY SIZE	*	(SW1–4)
		4	4		+DISPLAY TYPE 1	**	(SW1–5)
		5	5		+DISPLAY TYPE 2	**	(SW1–6)
		6	6		NO. OF 5 1/4 DRVS	***	(SW1–7)
		7	7		NO. OF 5 1/4 DRVS	***	(SW1–8)

Port		Bit	
X'0061'	OUTPUT	PB0	+TIMER 2 GATE SPEAKER
		1	+SPEAKER DATA
		2	+(READ READ/WRITE MEMORY SIZE) OR (READ SPARE KEY)
		3	+CASSETTE MOTOR OFF
		4	-ENABLE READ/WRITE MEMORY
		5	-ENABLE I/O CH CK
		6	-HOLD KBD CLK LOW
		7	-(ENABLE KBD) OR + (CLR KBD & ENABLE SENSE SW'S)

Port		Bit				
X'0062'	INPUT	PC0	I/O READ/WRITE MEMORY (SW2–1)	BINARY VALUE X 32KB	OR	SPARE KEY (SW2–5)
		1	I/O READ/WRITE MEMORY (SW2–2)			
		2	I/O READ/WRITE MEMORY (SW2–3)			
		3	I/O READ/WRITE MEMORY (SW2–4)			
		4	+CASSETTE DATA IN			
		5	+TIMER CHANNEL 2 OUT			
		6	+I/O CHANNEL CHECK			
		7	+READ/WRITE MEMORY PCK			

X'0063' CMD/MODE REGISTER

MODE REG VALUE

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

X'99'

*	PA3 SW1–4	PA2 SW1–3	AMOUNT OF MEMORY LOCATED ON SYS. BD.
	0	0	16K BYTES
	0	1	32K BYTES
	1	0	48K BYTES
	1	1	64K BYTES

**	PA5 SW1–6	PA4 SW1–5	TYPE OF DISPLAY
	0	0	RESERVED
	0	1	COLOR CARD 40X25 (BW MODE)
	1	0	COLOR CARD 80X25 (BW MODE)
	1	1	IBM MONOCHROME DISPLAY (80 X 25)

***	PA7 SW1–8	PA6 SW1–7	NUMBER OF 5–1/4" DRIVES IN SYSTEM
	0	0	1
	0	1	2
	1	0	3
	1	1	4

8255A – I/O BIT MAP

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

System Memory Map

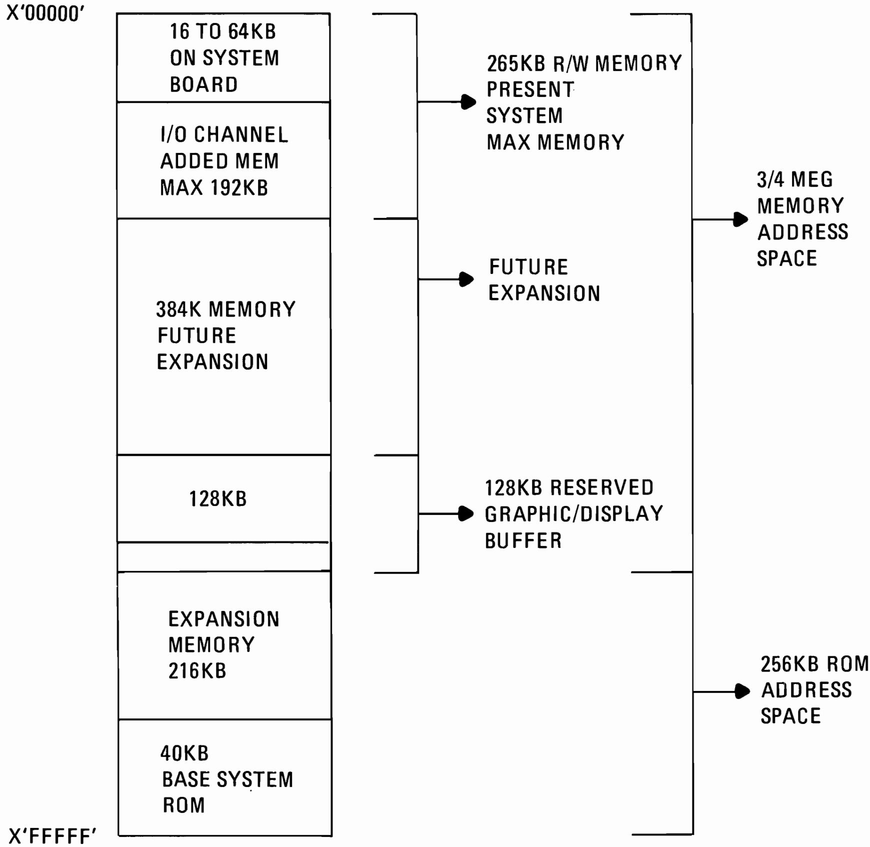


Figure 11. SYSTEM MEMORY MAP

HARDWARE

System Memory Map (Increments of 16KB)

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

Figure 12. SYSTEM MEMORY MAP (INCREMENTS OF 16KB) (SHEET 1 OF 2)

System Memory Map Cont.

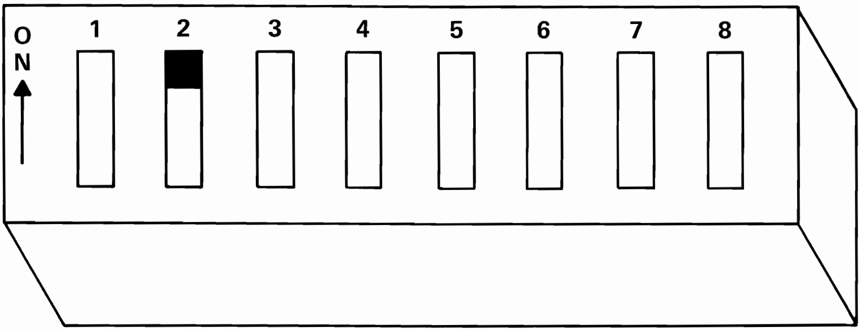
START ADDRESS: DECIMAL	HEX	FUNCTION:	
640K	A0000	RESERVED	
656K	A4000		112 KB GRAPHICS/DISPLAY VIDEO BUFFER
672K	A8000		
688K	AC000		
704K	B0000	MONOCHROME	
720K	B4000		
736K	B8000	COLOR/GRAPHICS	
752K	BC000		
768K	C0000	192 KB MEMORY EXPANSION AREA	
784K	C4000		
800K	C8000		
816K	CC000		
832K	D0000		
848K	D4000		
864K	D8000		
880K	DC000		
896K	E0000		
912K	E4000		
928K	E8000		
944K	EC000		
960K	F0000	RESERVED	
976K	F4000	48 KB BASE SYSTEM ROM	
992K	F8000		
1.008M	FC000		

Figure 12. SYSTEM MEMORY MAP (16KB) (SHEET 2)

System Board and Memory Expansion Switch Settings

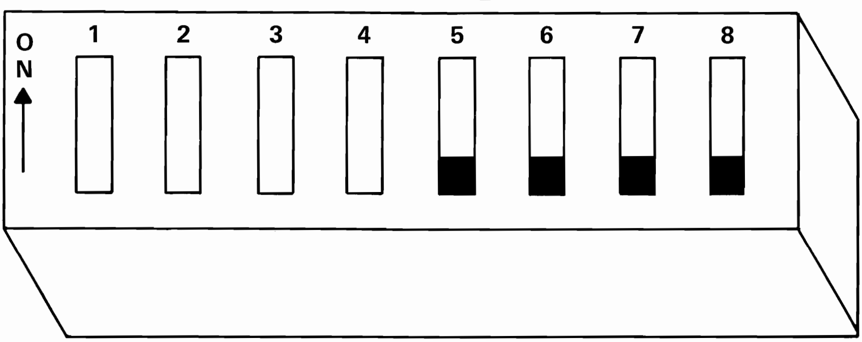
On the following four pages are graphic illustrations of switch settings. These are necesary for the system to address components attached, and to specify the amount of memory installed both on the System Board and in the System Expansion Slots. Refer to the System Board Component Diagram (page 13) for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

SWITCH 1



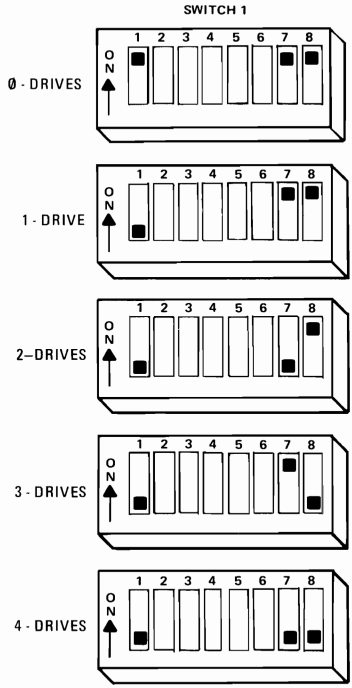
POSITION	FUNCTION
1-7-8	NUMBER OF 5¼" DISKETTE DRIVES INSTALLED; PAGE 2-29
2	UNUSED-MUST BE ON (RESERVED FOR CO-PROCESSOR)
3-4	AMOUNT OF MEMORY ON SYSTEM BOARD; PAGE 2-30
5-6	TYPE OF MONITOR YOU ARE USING; PAGE 2-29

SWITCH 2

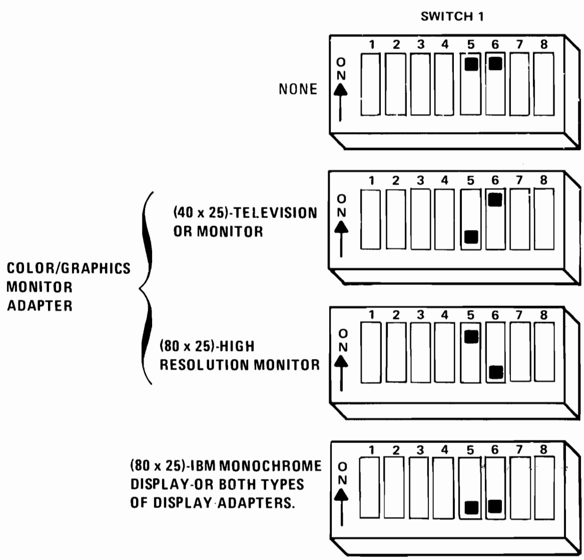


POSITION	FUNCTION
1-2-3-4	AMOUNT OF MEMORY OPTIONS INSTALLED; PAGE 2-30
5-6-7-8	ALWAYS IN THE OFF POSITION

5-1/4" Diskette Drives Switch Settings

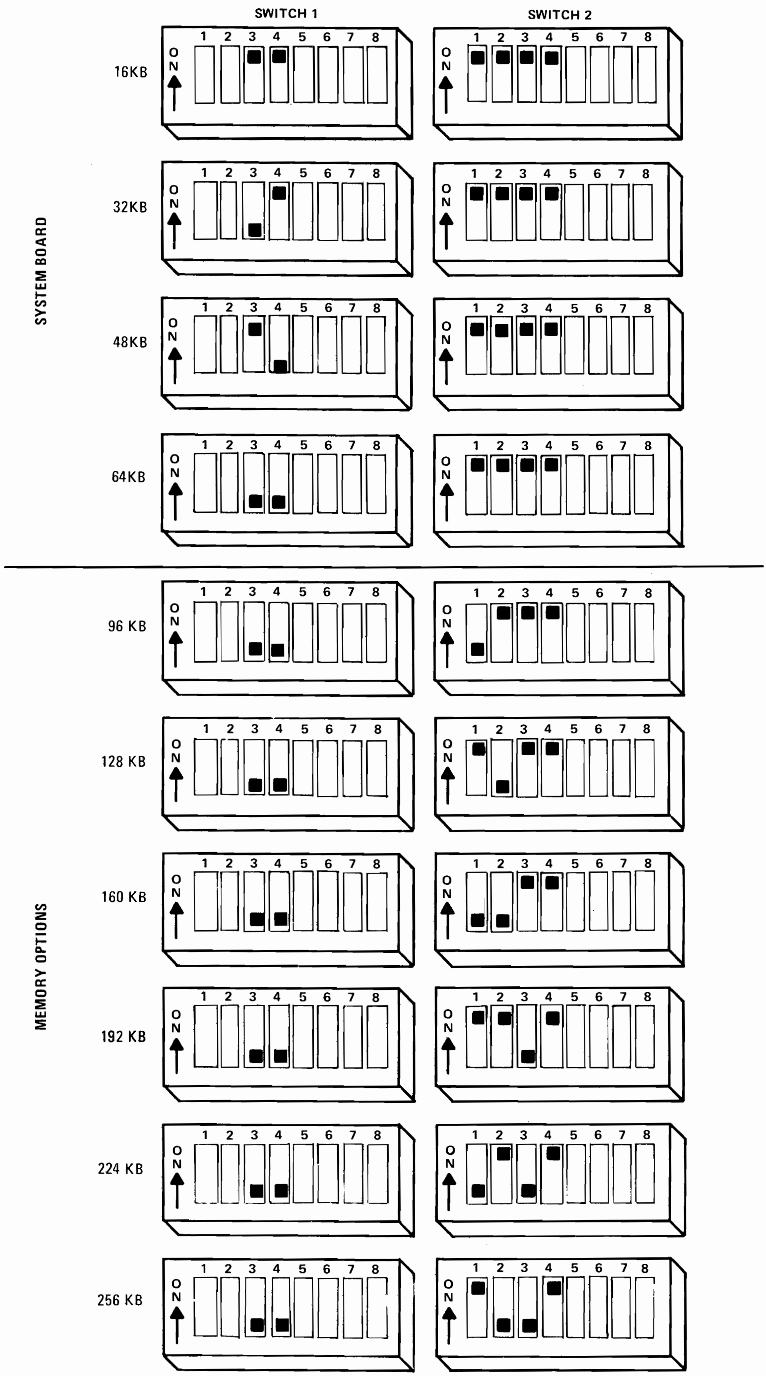


Monitor Type Switch Settings



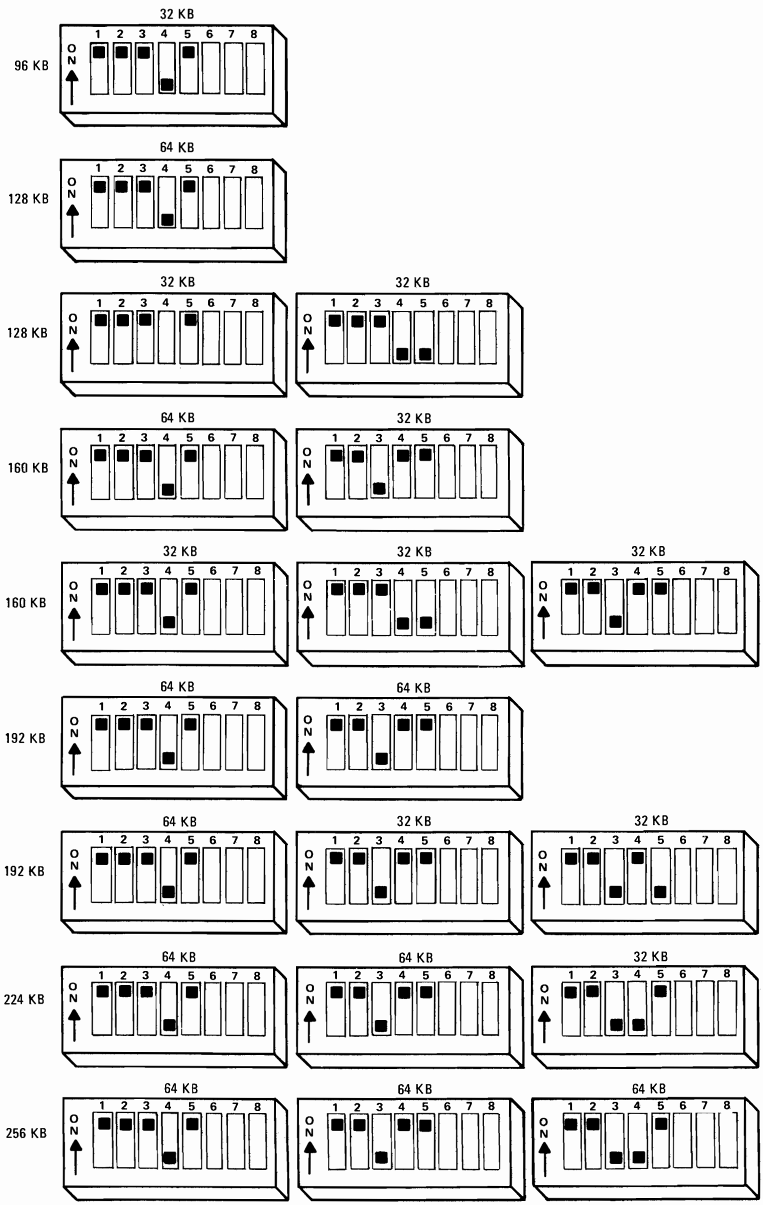
NOTE: SOME TELEVISIONS AND MONITORS OPERATED IN (80 x 25) MODE MAY HAVE CHARACTER LOSS.

System Board Memory Switch Settings



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.



NOTES

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 +12Vdc, ±5% 300 ma of -5Vdc, ±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.

HARDWARE

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, annd provides two seperate 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

VOLTAGE @ 60 Hz

NOMINAL Vac	MINIMUM Vac	MAXIMUM Vac
120	104	127

Frequency

60 Hz +/− .5 Hz

Current

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

DC Output

VOLTAGE Vdc	CURRENT AMPS		REGULATION TOLERANCE	
NOMINAL	MIN	MAX	±%	−%
+ 5.0	2.3	7.0	5	4
− 5.0	0.0	0.3	10	8
+12.0	0.0	2.0	5	4
−12.0	0.0	0.25	10	9

AC Output

VOLTAGE Vac	CURRENT AMPS		VOLTAGE LIMITS Vac	
NOMINAL	MIN	MAX	MIN	MAX
120.0	0.0	.75	101.0	130.0

Power Supply Connectors And Pin Assignment

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:

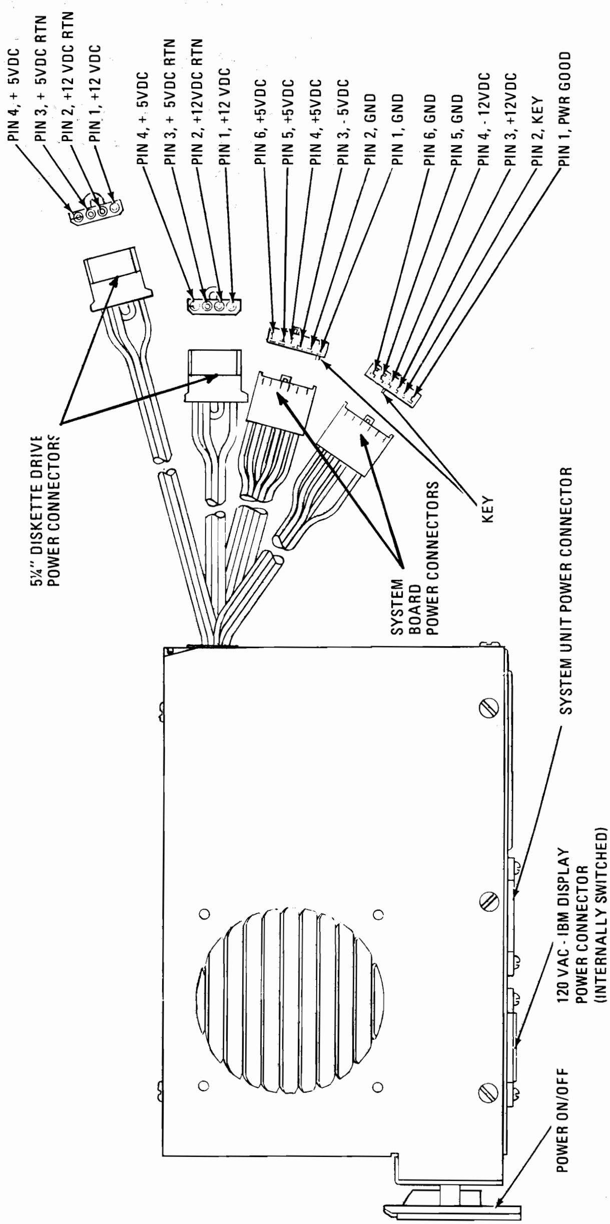


Figure 13. POWER SUPPLY AND CONNECTORS

Important Operating Characteristics

Over Voltage/Current Protection

PRIMARY (INPUT)

VOLTAGE NOMINAL VAC	TYPE PROTECTION	RATING AMPS
120	60 Hz FUSE TYPE 2 SOC SD4	2 AMPS

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:

OUTPUT	MIN	SENSE VOLTAGE NOMINAL	MAX
+5V	+3.7	+4.0	+4.3
−5V	−3.7	−4.0	−4.3
+12V	+8.5	+9.6	+10.5
−12V	−8.5	−9.6	−10.5

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 Phospor, will not support a light pen!

Parallel Interface Description

This topic is discussed in full on pages 2-65 through page 2-69.

IBM Monochrome Display Adapter Block Diagram

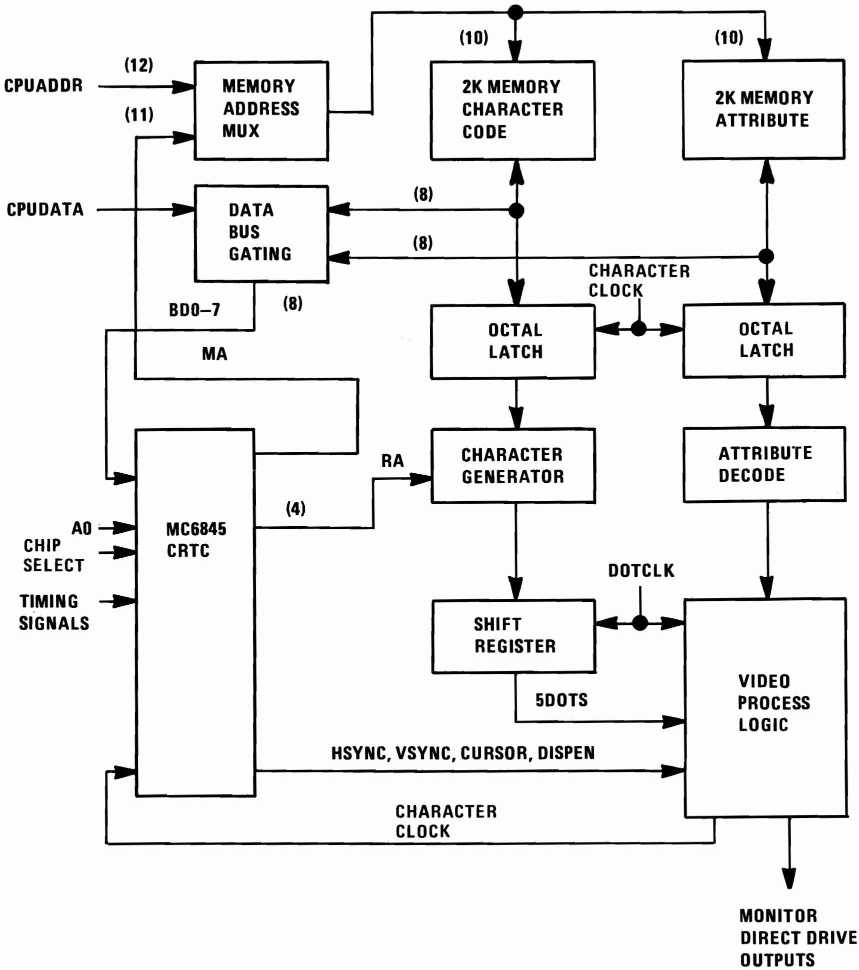


Figure 14. IBM MONOCHROME DISPLAY ADAPTER BLOCK DIAGRAM

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

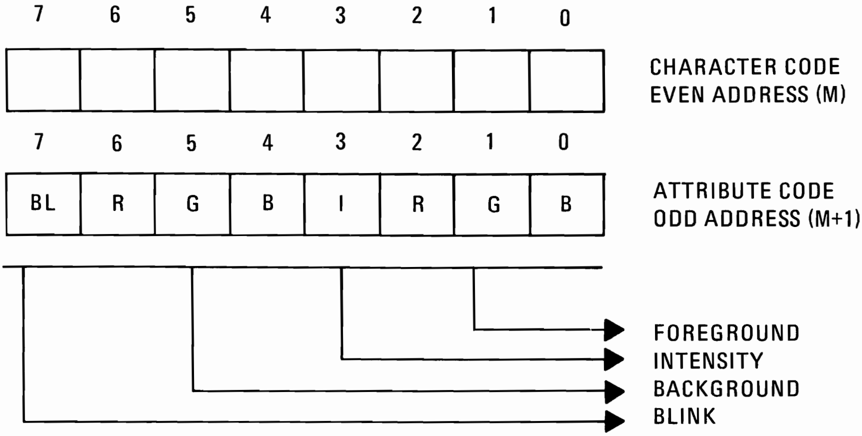
For the IBM Monochrome Display Adapter, two bytes are fetched from the display buffer in 553 ns providing a data rate of 1.8M bytes/second.

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.

BACKGROUND R G B	FOREGROUND R G B	FUNCTION
0 0 0	0 0 0	NON DISPLAY
0 0 0	0 0 1	UNDERLINE
0 0 0	1 1 1	WHITE CHARACTER/ BLACK BACKGROUND
1 1 1	0 0 0	REVERSE VIDEO

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

REGISTER #	REGISTER FILE	PROGRAM UNIT	80x25 MONOCHROME
R0	HORIZONTAL TOTAL	CHARACTERS	61H
R1	HORIZONTAL DISPLAYED	CHARACTERS	50H
R2	HSYNC POSITION	CHARACTERS	52H
R3	HSYNC WIDTH	CHARACTERS	FH
R4	VERTICAL TOTAL	CHAR ROWS	19H
R5	VTOTAL ADJUST	SCAN LINE	6H
R6	VERTICAL DISPLAYED	CHAR ROW	19H
R7	VSYNC POSITION	CHAR ROW	19H
R8	INTERLACE MODE	---	02
R9	MAX SCAN LINE ADDRESS	SCAN LINE	DH
R10	CURSOR START	SCAN LINE	BH
R11	CURSOR END	SCAN LINE	CH
R12	START ADDRESS (H)	---	00H
R13	START ADDRESS (L)	---	00H
R14	CURSOR (H)	---	00H
R15	CURSOR (L)	---	00H
R16	RESERVED	---	---
R17	RESERVED	---	---

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:

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

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')

Bit #	Function
0	+high resolution mode
1	Not used
2	Not used
3	+ video enable
4	Not used
5	+ enable blink
6,7	Not used

- CRT Status Port (I/O Address '3BA')

Bit	Function
0	+Horizontal Drive
1	Reserved
2	Reserved
3	+B/W Video

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.

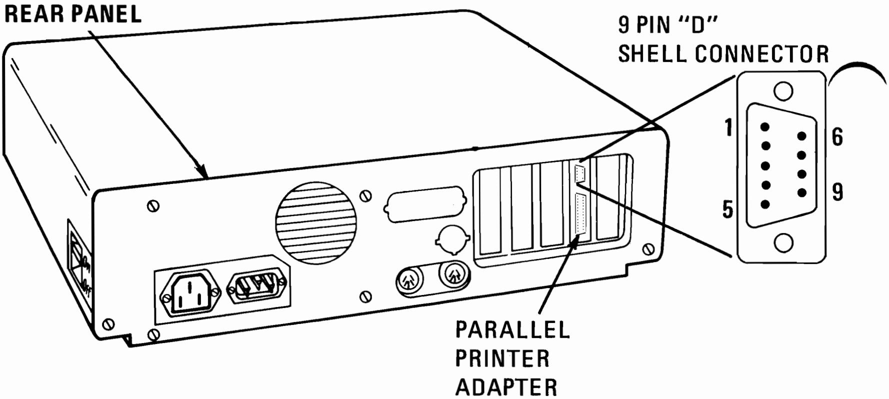
Vertical Drive

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.

IBM Monochrome Direct Drive Interface and Pin Assignment



At Standard TTL Levels

IBM Monochrome Display

Ground	1
Ground -	2
Not Used	3
Not Used	4
Not Used	5
+ Intensity	6
+ Video	7
+ Horizontal	8
- Vertical	9

IBM Monochrome Display and Parallel Printer Adapter

NOTE: Signal voltages are 0 - .6 Vdc at down level
+5 Vdc at high level

Color/Graphics Monitor Adapter

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 monitor 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

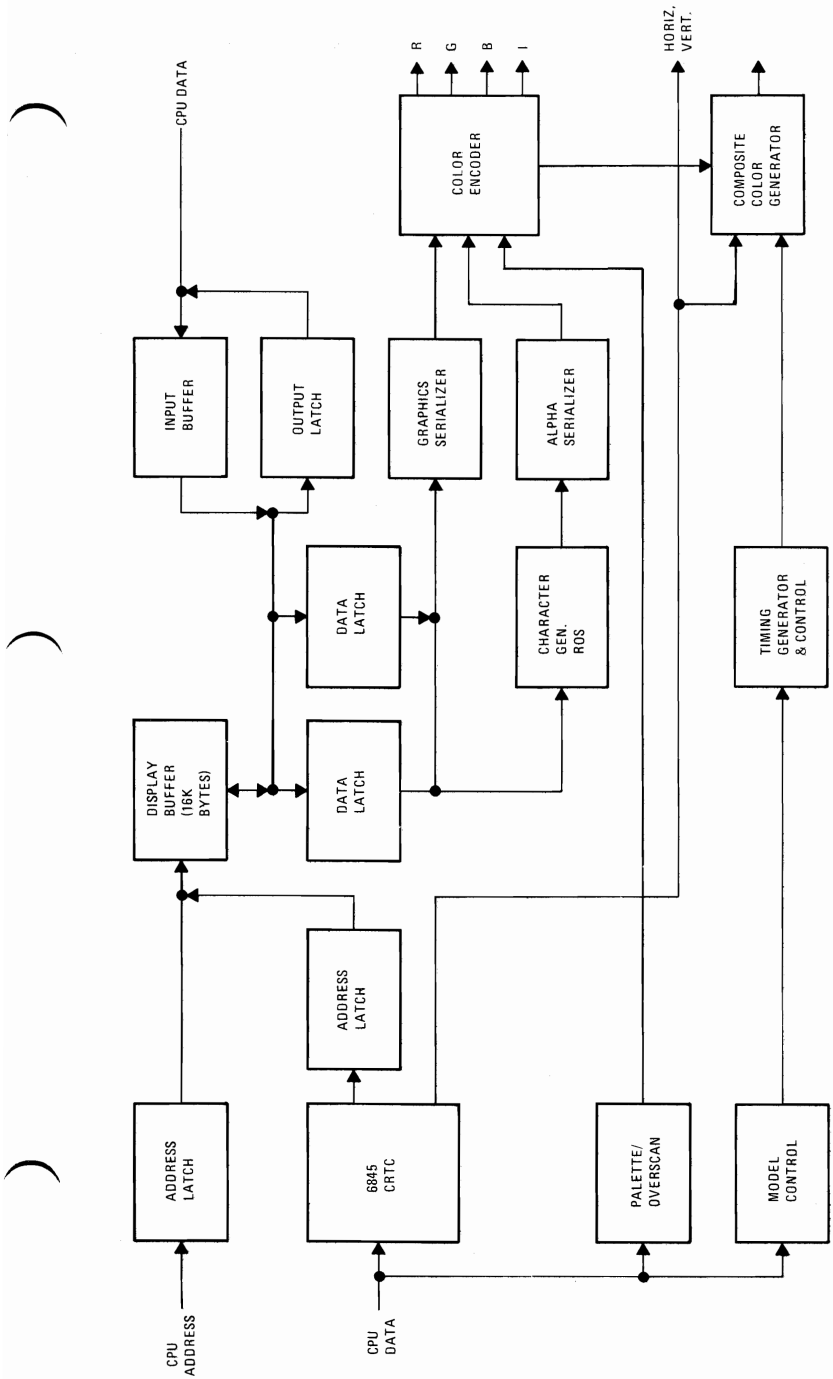


Figure 15. COLOR/GRAPHICS MONITOR ADAPTER BLOCK DIAGRAM

HARDWARE

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.

DISPLAY CHAR CODE BYTE								ATTRIBUTE BYTE							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

Attribute Byte Definition

	ATTRIBUTE BYTE							
	7	6	5	4	3	2	1	0
	B	R	G	B	I	R	G	B
ATTRIBUTE FUNCTION	FG	BACKGROUND			FOREGROUND			
NORMAL	B	0	0	0	I	1	1	1
REVERSE VIDEO	B	1	1	1	I	0	0	0
NON DISPLAY (BLK)	B	0	0	0	I	0	0	0
NON DISPLAY (WHITE)	B	1	1	1	I	1	1	1

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

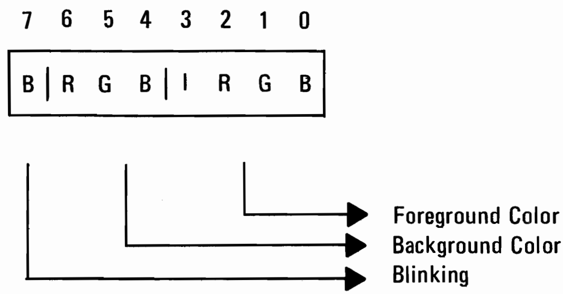
Color TV

- 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)

7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
CHARACTER CODE	ATTRIBUTE CODE
EVEN ADDRESS (M)	ODD ADDRESS (M+1)

ATTRIBUTE BYTE DEFINITIONS

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



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

	7	6	5	4	3	2	1	0	ON THE MONOCHROME DISPLAY ADAPTER		ON THE COLOR/GRAPHIC DISPLAY ADAPTER	
	B	R	G	B	I	R	G	B	CHAR. COLOR	BKGD. COLOR	CHAR. COLOR	BKGD. COLOR
	FG	BACKGROUND			FOREGROUND							
NORMAL	B	0	0	0	I	1	1	1	WHITE	BLACK	WHITE	BLACK
RVV	B	1	1	1	I	0	0	0	BLACK	WHITE	BLACK	WHITE
NON DISP (BLK)	B	0	0	0	I	0	0	0	BLACK	BLACK	BLACK	BLACK
NON DISP (WHT)	B	1	1	1	I	1	1	1	WHITE	WHITE	WHITE	WHITE

ALL OTHER CODES DEFINE FOREGROUND BACKGROUND COLOR COMBINATIONS

ALL OTHER CODES RESULT IN WHITE CHAR ON BLACK BACKGROUND

ALL OTHER CODES CHANGE FOREGROUND BACKGROUND COLOR TO SELECTED VALUE

R	G	B		
0	0	0	=	BLACK
0	0	1	=	BLUE
0	1	0	=	GREEN
0	1	1	=	CYAN
1	0	0	=	RED
1	0	1	=	MAGENTA
1	1	0	=	YELLOW
1	1	1	=	WHITE

* AN ADDITIONAL 8 COLOR (ACTUAL) DIFFERENT SHADES OF THE ABOVE) ARE SELECTED BY SETTING THE (I) BIT

* CODE WRITTEN WITH AN UNDERLINE ATTRIBUTE FOR THE IBM MONOCHROME DISPLAY WHEN EXECUTED ON A COLOR/GRAPHICS ADAPTER WILL RESULT IN A BLUE CHARACTER WHERE THE UNDERLINE ATTRIBUTES ARE ENCOUNTERED.

CODE WRITTEN ON A COLOR/GRAPHICS ADAPTER WITH BLUE CHARACTERS, WILL BE DISPLAYED AS WHITE CHARACTERS ON BLACK BACKGROUND WITH A WHITE UNDERLINE ON THE MONOCHROME DISPLAY

Note: Not all Monitors Recognize the (1) Bit

Table 4. Color/Graphics Modes

	HORIZONTAL	VERTICAL	NO OF COLORS (INCL. BACKGROUND COLOR)
LOW RES	160	100	16 (INCLUDES BLACK AND WHITE)
MED RES	320	200	4 COLORS: 1 OF 16 FOR BACKGROUND PLUS GREEN, RED, YELLOW OR CYAN, MAGENTA, WHITE
HIGH RES	640	200	B & W ONLY

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

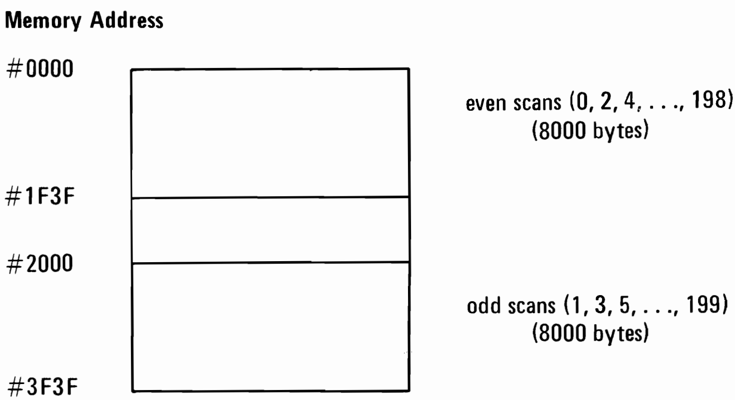
FORMAT:

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



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.

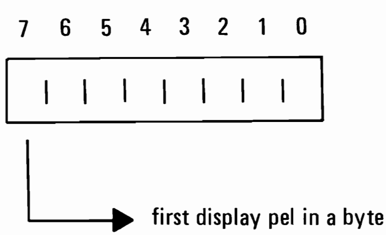
C1	C0	CODE SELECT COLOR FOR DISPLAY POSITION
0	0	DOT TAKES ON COLOR OF 1 OF 16 PRESELECTED BACKGROUND COLORS.
0	1	SELECT 1ST COLOR OF PRESELECT COLOR SET "1" OR "2"
1	0	SELECT 2ND COLOR OF PRESELECT COLOR SET "1" OR "2"
1	1	SELECT 3RD COLOR OF PRESELECT COLOR SET "1" OR "2"

The two color sets are:

SET ONE	SET TWO
COLOR 1 - CYAN	COLOR 1 - GREEN
COLOR 2 - MAGENTA	COLOR 2 - RED
COLOR 3 - WHITE	COLOR 3- BROWN

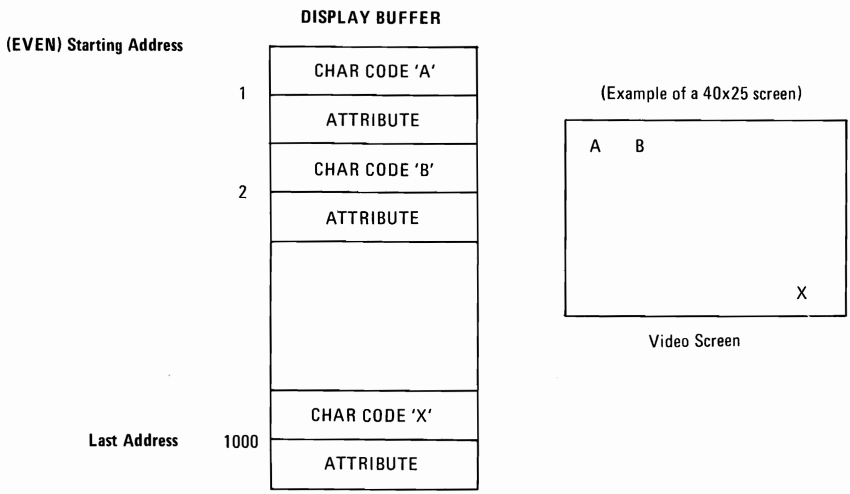
The background colors are the same basic 8 color as defined for low resolution graphic plus 8 alternate intensities defined by the intensity bit for a total of 16 color including black and white.

3. 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**

I	R	G	B	COLOR
0	0	0	0	Black
0	0	0	1	Blue
0	0	1	0	Green
0	0	1	1	Cyan
0	1	0	0	Red
0	1	0	1	Magenta
0	1	1	0	Brown
0	1	1	1	Light Gray
1	0	0	0	Dark Gray
1	0	0	1	Light Blue
1	0	1	0	Light Green
1	0	1	1	Light Cyan
1	1	0	0	Light Red
1	1	0	1	Light Magenta
1	1	1	0	Yellow
1	1	1	1	White

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**

ADDR REG.	REG. #	REGISTER TYPE	UNITS	I/O	40x25 ALPHA	80x25 ALPHA	GRAPHIC MODES
0	R0	Horizontal Total	Char.	Write Only	38	71	38
1	R1	Horizontal Displayed	Char.	Write Only	28	50	28
2	R2	Horiz. Sync Position	Char.	Write Only	2D	5A	2D
3	R3	Horiz. Sync Width	Char.	Write Only	0A	0A	0A
4	R4	Vertical Total	Char. Row	Write Only	1F	1F	7F
5	R5	Vertical Total Adjust	Scan Line	Write Only	06	06	06
6	R6	Vertical Displayed	Char. Row	Write Only	19	19	64
7	R7	Vert. Sync Position	Char. Row	Write Only	1C	1C	70
8	R8	Interlace Mode	–	Write Only	02	02	02
9	R9	Max Scan Line Addr.	Scan Line	Write Only	07	07	01
A	R10	Cursor Start	Scan Line	Write Only	06	06	06
B	R11	Cursor End	Scan Line	Write Only	07	07	07
C	R12	Start Addr. (H)	–	Write Only	00	00	00
D	R13	Start Addr. (L)	–	Write Only	00	00	00
E	R14	Cursor Addr. (H)	–	Read/ Write	XX	XX	XX
F	R15	Cursor Addr. (L)	–	Read/ Write	XX	XX	XX
10	R16	Light Pen (H)	–	Read Only	XX	XX	XX
11	R17	Light Pen (L)	–	Read Only	XX	XX	XX

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.

HEX ADDR.	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	FUNCTION OF REGISTER
X'3D8'	1	1	1	1	0	1	1	0	0	0	DO REG (MODE CONTROL)
X'3D9'	1	1	1	1	0	1	1	0	0	1	DO REG (COLOR SELECT)
X'3DA'	1	1	1	1	0	1	1	0	1	0	DI REG (STATUS)
X'3DB'	1	1	1	1	0	1	1	0	1	1	CLEAR LIGHT PEN LATCH
X'3DC'	1	1	1	1	0	1	1	1	0	0	PRE SET LIGHT PEN LATCH
X'3D0'	1	1	1	1	0	1	0	Z	Z	0	6845 REGISTERS
X'3D1'	1	1	1	1	0	1	0	Z	Z	1	6845 REGISTERS
X'3D0'	1	1	1	1	0	1	0	Z	Z	0	6845 REGISTERS
X'3D1'	1	1	1	1	0	1	0	Z	Z	1	6845 REGISTERS

Z = don't care condition

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	I 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.

HARDWARE

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

Bits

0	1	2	3	4	5	
0	0	1	1	0	1	40 x 25 ALPHA B & W
0	0	0	1	0	1	40 x 25 ALPHA COLOR
1	0	1	1	0	1	80 x 25 ALPHA B & W
1	0	0	1	0	1	80 x 25 ALPHA COLOR
0	1	1	1	0	z	320 x 200 B & W GRAPHICS
0	1	0	1	0	z	320 x 200 COLOR GRAPHICS
0	1	1	1	1	z	640 x 200 B & W GRAPHICS

ENABLE BLINK ATTRIBUTE
640 x 200 B & W
ENABLE VIDEO
SELECT B & W MODE
SELECT 320 x 200 GRAPHICS
80 x 25 ALPHA SELECT

z = don't care condition

* 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.

Bit 0 Display Enable
Bit 1 Light Pen Trigger Set
Bit 2 Light Pen SW Made
Bit 3 Alpha Dots
Bit 4 Not Used
Bit 5 Not Used
Bit 6 Not Used
Bit 7 Not Used

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

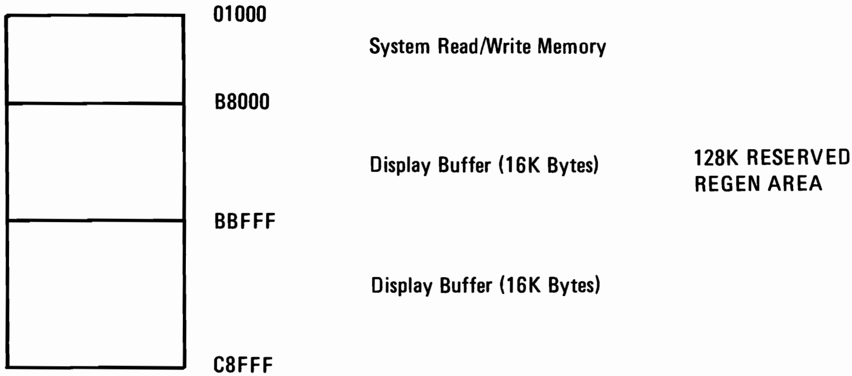
The memory used by this adapter is self-contained. It consists of 16k bytes of memory without parity. This memory is used as both a display buffer for alphanumeric data and as a bit map for graphics data. The Regen Buffers address starts at X'B8000'.

Interrupt Level (Vertical Retrace)

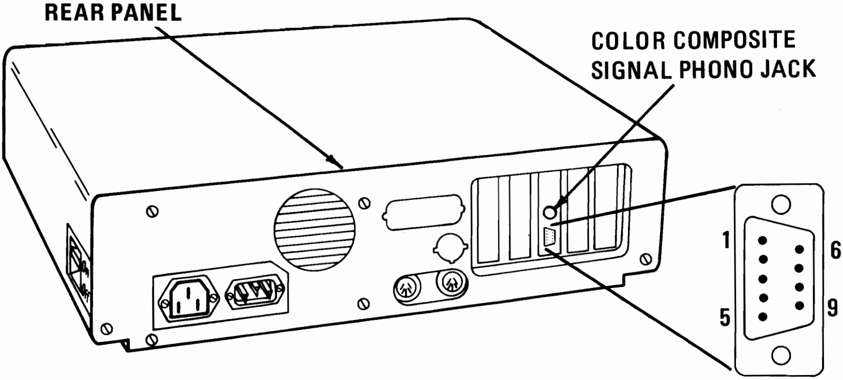
Level 2

I/O Address and Bit Map

Read/Write Memory Address Space

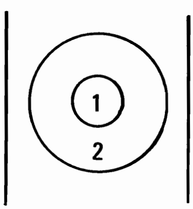


Color/Graphics Monitor Adapter Direct Drive, and Composite Interface Pin Assignment



AT STANDARD TTL LEVELS

Direct Drive Monitor	Signal	Pin	Color/Graphics Direct Drive Adapter
	Ground	1	
	Ground	2	
	Red	3	
	Green	4	
	Blue	5	
	Intensity	6	
	Reserved -	7	
	Horizontal Drive	8	
	Vertical Drive	9	

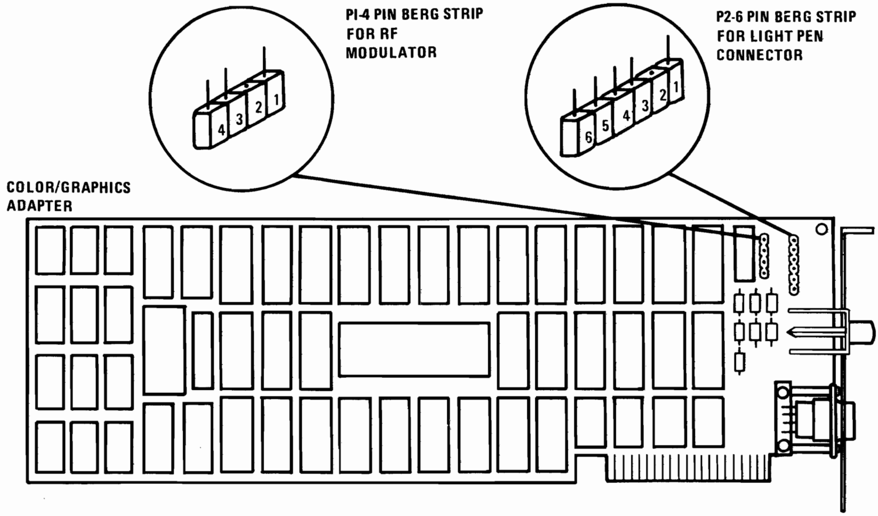


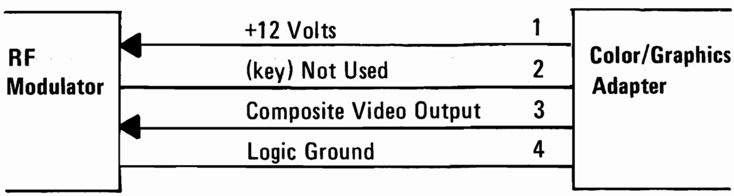
COMPOSITE PHONO JACK

HOOK-UP TO MONITORS

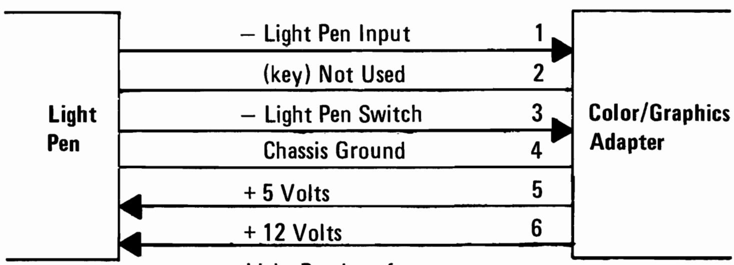
Video Monitor	Signal	Pin	Color/Graphics Composite Jack
	Composite Video Signal of approximately 1.5 Volts Peak to Peak Amplitude	1	
	Chassis Ground	2	

Color/Graphics Monitor Adapter Auxiliary Video Connectors





RF Modulator Interface



Light Pen Interface

NOTES

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

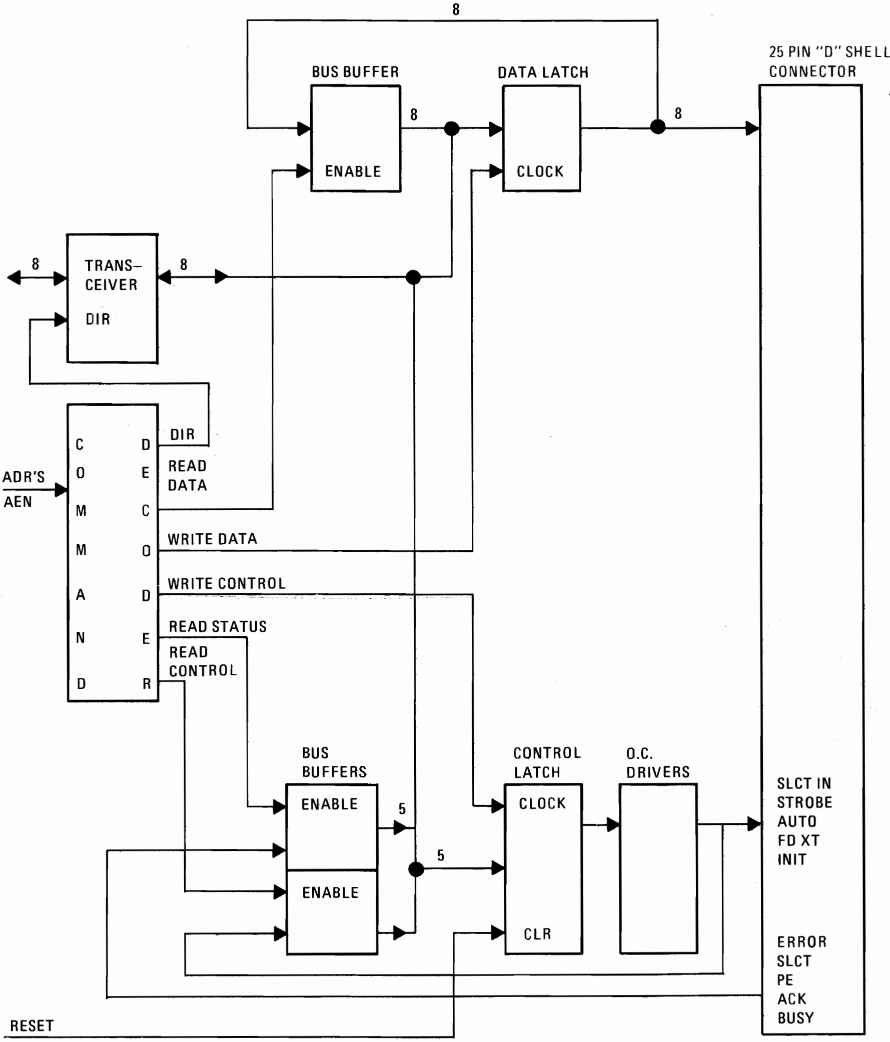


Figure 16. PARALLEL PRINTER ADAPTER BLOCK DIAGRAM

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.

IBM Monochrome Display & Printer Adapter				Parallel Printer Adapter			
Output to address 3BCH				Output to address 378H			
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Pin 9	Pin 8	Pin 7	Pin 6	Pin 5	Pin 4	Pin 3	Pin 2

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.

IBM Monochrome Display & Printer Adapter		Parallel Printer Adapter			
Output to address 3BEH		Output to address 37AH			
	Bit 4	$\overline{\text{Bit 3}}$	Bit 2	$\overline{\text{Bit 1}}$	$\overline{\text{Bit 0}}$
	IRQ Enable	Pin 17	Pin 16	Pin 14	Pin 1

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.

IBM Monochrome Display & Printer Adapter	Parallel Printer Adapter
Input from address x' '3BC'	Input from address 378H

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.

IBM Monochrome Display & Printer Adapter	Parallel Printer Adapter
Input from address 3BDH	Input from address 379H

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	–	–	–

IBM Monochrome Display & Printer Adapter	Parallel Printer Adapter
Input from address 3BEH	Input from address 37AH

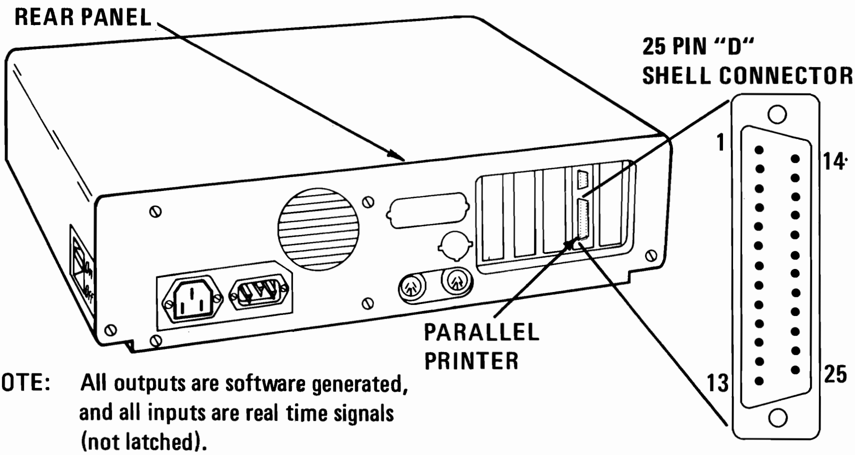
This instruction causes the data present on pins 1, 14, 16, 17 and IRQ bit to be read by the CPU. In the absence of external drive applied to these pins, data read by the CPU will exactly match data last written to x' '3BE' in the same bit positions. Note that data bits 0-2 are not included. If external 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 Enable	$\overline{\text{Pin 17}}$	Pin 16	$\overline{\text{Pin 14}}$	$\overline{\text{Pin 1}}$
			Por=0	Por=1	Por=0	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

Printer ↔ Parallel Printer Adapter

Signal Name	AMP Pin No.
– Strobe	1
+ Data Bit 0	2
+ Data Bit 1	3
+ Data Bit 2	4
+ Data Bit 3	5
+ Data Bit 4	6
+ Data Bit 5	7
+ Data Bit 6	8
+ Data Bit 7	9
– Acknowledge	10
+ Busy	11
+ P. End (out of Paper)	12
+ Select	13
– Auto Feed	14
– Error	15
– Initialize Printer	16
– Select Input	17
Ground	18 - 25

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

(1)	PRINT METHOD:	Serial impact dot matrix	
(2)	PRINT SPEED:	80 CPS	
(3)	PRINT DIRECTION:	Bidirectional with logical seeking	
(4)	NUMBER OF PINS IN HEAD:	9	
(5)	LINE SPACING:	4.23 mm (1/6") or programmable	
(6)	PRINTING CHARACTERISTICS		
	Matrix:	9 x 9	
	Character Set:	Full 96-character ASCII with decoders, plus 9 international characters/symbols	
	Graphic Character:	64 block characters	
(7)	PRINTING SIZES		
		Characters per inch	Maximum characters per line
	Normal:	10	80
	Enlarged:	5	40
	Condensed:	16.5	132
	Condensed Enlarged:	8.25	66
(8)	MEDIA HANDLING		
	Paper Feed:	Adjustable sprocket pin feed	
	Paper Width Range:	101.6 mm (4") to 254 mm (10")	
	Copies:	One original plus two carbon copies (total thickness not to exceed 0.3 mm (0.012")	
	Paper Path:	Rear	
(9)	INTERFACES		
	Standard:	Parallel 8-bit Data & Control Lines	
(10)	INKED RIBBON		
	Color:	Black	
	Type:	Cartridge	
	Life Expectancy:	3 million characters	
(11)	ENVIRONMENTAL CONDITIONS		
	Operating Temperature Range:	5 to 35°C (41 to 95°F)	
	Operating Humidity:	10 to 80% non-condensing	
(12)	POWER REQUIREMENT		
	Voltage:	120VAC, 60 Hz	
	Current:	1 Amp maximum	
	Power Consumption:	100 VA maximum	
(13)	PHYSICAL CHARACTERISTICS		
	Height:	107 mm (4.2")	
	Width:	374 mm (14.7")	
	Depth:	305 mm (12.0")	
	Weight:	5.5 kg (12 lbs.)	

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).

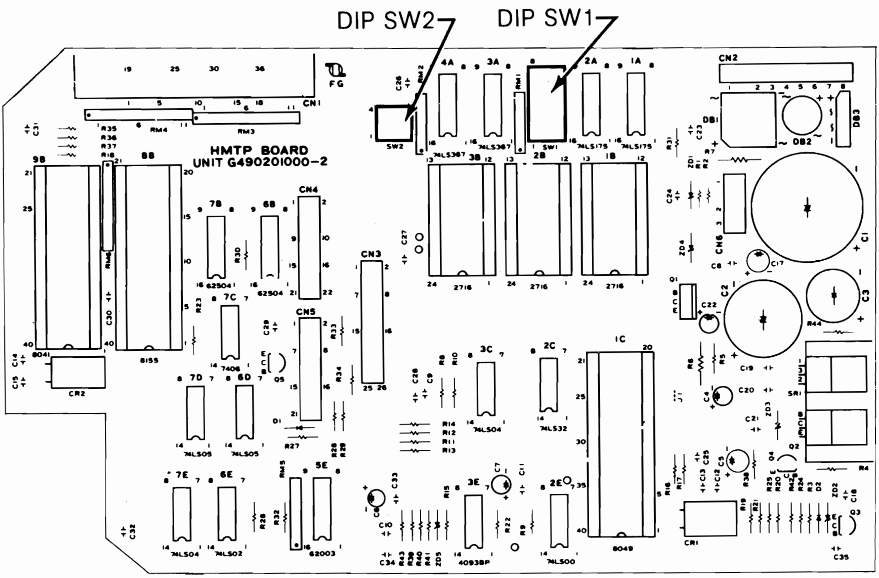


Figure 17. LOCATION OF PRINTER DIP SWITCHES

Table 8. Functions and Conditions of DIP Switch 1

Pin No.	Function	ON	OFF	Factory-set Condition
1	Not applicable	—	—	ON
2	CR { Print & Line Feed / Print only	Print only	Print & line feed	ON
3	Buffer full { Print & Line Feed / Print only	Print only	Print & line feed	ON
4	Cancel code { Valid / Invalid	Invalid	Valid	OFF
5	Delete code { Valid / Invalid	Invalid	Valid	ON
6	Error Buzzer	Sounds	Does not sound	ON
7	Character generator (Graphic pattern select)	N.A.	Graphic patterns select	OFF
8	SLCT IN signal Fixed internally Not fixed internally	Fixed	Not fixed	ON

Table 9. Functions and Conditions of DIP Switch 2

Pin No.	Function	ON	OFF	Factory-set Condition
1	Not applicable	—	—	ON
2		—	—	ON
3	$\overline{\text{AUTO FEED XT}}$ signal { Fixed internally / Not fixed internally	Fixed	Not fixed	OFF
4	Coding table select	N.A.	Standard	OFF

Parallel Interface Description

(1) Specifications
- (a) Data transfer rate: 1000 CPS (max.)
- (b) Synchronization: By externally supplied STROBE pulses.
- (c) Handshaking: $\overline{\text{ACKNLG}}$ 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 10. Connector Pin Assignment and Descriptions of Interface Signals

Signal Pin No.	Return Pin No.	Signal	Direction	Description
1	19	$\overline{\text{STROBE}}$	In	STROBE pulse to read data in. Pulse width must be more than 0.5μs at receiving terminal. The signal level is normally "HIGH"; read-in of data is performed at the "LOW" level of this signal.
2	20	DATA 1	In	These signals represent information of the 1st to 8th bits of parallel data respec– tively. Each signal is at "HIGH" level when data is logical "1" and "LOW" when logical "0".
3	21	DATA 2	In	
4	22	DATA 3	In	
5	23	DATA 4	In	
6	24	DATA 5	In	
7	25	DATA 6	In	
8	26	DATA 7	In	
9	27	DATA 8	In	
10	28	$\overline{\text{ACKNLG}}$	Out	Approx. 5μs pulse. "LOW" indicates that data has been received and that the printer is ready to accept other data.
11	29	BUSY	Out	A "HIGH" signal indicates that the printer cannot receive data. The signal becomes "High" in the following cases: 1. During data entry 2. During printing operation 3. In OFF-LINE state 4. During printer error status.

Table 10. Connector Pin Assignment and Descriptions of Interface Signals (cont.)

Signal Pin No.	Return Pin No.	Signal	Direction	Description
12	30	PE	Out	A "HIGH" signal indicates that the printer is out of paper.
13	–	SLCT	Out	This signal indicates that the printer is in the selected state.
14	–	AUTO FEED XT	In	With this signal being at "LOW" level, the paper is automatically fed one line after printing. (The signal level can be fixed to "LOW" with DIP SW pin 2-3 provided on the control circuit board.)
15	–	NC		Not used.
16	–	0V		Logic GND level.
17	–	CHASSIS-GND	–	Printer chassis GND. In the printer, the chassis GND and the logic GND are isolated from each other.
18	–	NC	–	Not used.
19–30	–	GND	–	TWISTED-PAIR RETURN signal GND level.
31	–	INIT	In	When the level of this signal becomes "LOW" the printer controller is reset to its initial state and the print buffer is cleared. This signal is normally at "HIGH" level, and its pulse width must be more than 50μs at the receiving terminal.

HARDWARE

Table 10. Connector Pin Assignment and Descriptions of Interface Signals (cont.)

Signal Pin No.	Return Pin No.	Signal	Direction	Description
32		$\overline{\text{ERROR}}$	Out	The level of this signal becomes "LOW" when the printer is in— 1. PAPER END state 2. OFF-LINE state 3. Error state
33	–	GND	–	Same as with Pin No. 19 to 30.
34	–	NC	–	Not used.
35				Pulled up to +5V through 4.7KΩ resistance.
36	–	$\overline{\text{SLCT IN}}$	In	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.)

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 $\overline{\text{ACKNLG}}$ or BUSY signal. (Data transfer to this printer can be carried out only after confirming the $\overline{\text{ACKNLG}}$ signal or when the level of the BUSY signal is "LOW".)

(4) Data transfer sequence
Fig. 17 shows the sequence for data transmission.

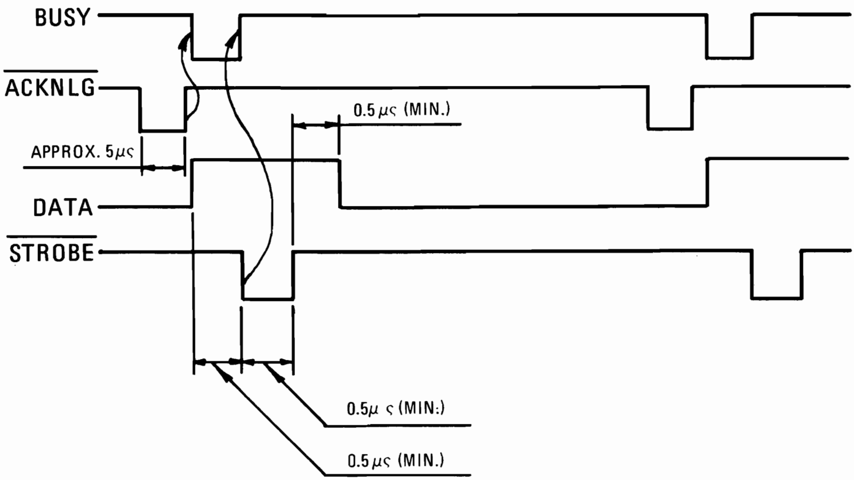


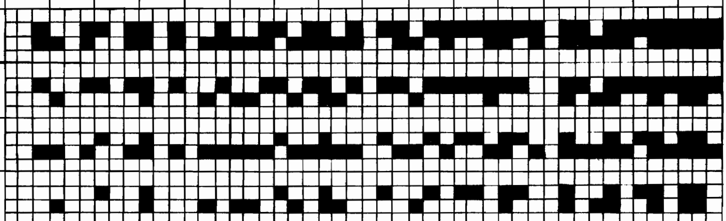
Figure 18. PARALLEL INTERFACE TIMING DIAGRAM

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 11. ASCII Coding Table

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0	0000	NUL		SP	0	@	P	`	p	NUL							
1	0001		DC1	!	1	A	Q	a	q		DC1						
2	0010		DC2	"	2	B	R	b	r		DC2						
3	0011		DC3	#	3	C	S	c	s		DC3						
4	0100		DC4	$	4	D	T	d	t		DC4						
5	0101			%	5	E	U	e	u								
6	0110			&	6	F	V	f	v								
7	0111	BEL			7	G	W	g	w	BEL							
8	1000		CAN	(	8	H	X	h	x		CAN						
9	1001	HT		)	9	I	Y	i	y	HT							
A	1010	LF		*	:	J	Z	j	z	LF							
B	1011	VT	ESC	+	;	K	[	k	{	VT	ESC						
C	1100	FF		,	<	L	\	l	¦	FF							
D	1101	CR		-	=	M	]	m	}	CR							
E	1110	SO		.	>	N	^	n	~	SO							
F	1111	SI		/	?	O	_	o	DEL	SI							



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 $\overline{\text{AUTO FEED XT}}$ (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 $\overline{\text{AUTO FEED XT}}$ 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 $\overline{\text{INIT}}$ signal is applied. If the form length per page is not set by "ESC C+n", it is regarded as 66 or 72 lines.

HARDWARE

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 [DC 4] GHI [CR] [LF]
    [PRINT]  ABCDEFGHI

2.  [DATA    ABCD [SO] EFGH [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] ABCDEFGHIJKL [CR] [LF]
    [PRINT]  ABCDEFGHIJKL

2.  [DATA]   ABC [SI] DEF [SO] GHIJKL [CR] [LF]
    [PRINT]  ABCDEFGHIJKL
```

(7) DC 4 (Device Control 4)

The DC 4 code cancels the SO mode.

```
    [DATA]   [SI] ABCDEF [SO] GHI [DC 4] JKL [CR] [LF]
    [PRINT]  ABCDEFGHIJKL
```

(8) DC 2 (Device Control 2)
The DC 2 code cancels the SI mode.

[DATA] $\boxed{\text{SI}}$ ABCDEF $\boxed{\text{SO}}$ GHI $\boxed{\text{CR}}$ $\boxed{\text{LF}}$ $\boxed{\text{DC 2}}$ JKLMN $\boxed{\text{CR}}$ $\boxed{\text{LF}}$
[PRINT] ABCDEFGHI
JKLMN

(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.

1. [DATA] $\boxed{\text{DC 1}}$ AAAAA $\boxed{\text{DC 3}}$ BBBBB $\boxed{\text{DC 1}}$ CCCCC $\boxed{\text{CR}}$ $\boxed{\text{LF}}$
[PRINT] AAAAACCCCC
2. [DATA] AAAAA $\boxed{\text{DC 1}}$ BBBBB $\boxed{\text{DC 3}}$ CCCCC $\boxed{\text{DC 1}}$ $\boxed{\text{CR}}$ $\boxed{\text{LF}}$
[PRINT] BBBBB

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

ON LINE SWITCH	SLCT IN	DC 1/DC 3	ERROR	BUSY	ACKNLG	SLCT	DATA ENTRY
OFF-LINE	HIGH/LOW	DC 1/DC 3	LOW	HIGH	Not Generated	LOW	Impossible
ON-LINE	HIGH	DC 1	HIGH	LOW/ HIGH	Generated	HIGH	Possible (Normal entry)
		DC 3	HIGH	LOW/ HIGH	Generated	LOW	Possible (See Note 1.)
	LOW	DC 1/DC 3	HIGH	LOW/ HIGH	Generated	HIGH	Possible (Normal entry)

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 DC 1 code for normal entry.

2: The DC 1/DC 3 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 DC 1/DC 3 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 $\overline{\text{ERROR}}$ 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 $1 \leq <n> 10 \leq 85$ (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.

HARDWARE

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.

```
[DATA]   AAAAAAA [CR] [LF] BBBBBBB [CR] [LF] [ESC A+24]
         CCCCCCC [CR] [LF] DDDDDDD [ESC 2] [CR] [LF]
         EEEEEEE [CR] [LF] FFFFFFF [CR] [LF]
[PRINT]  AAAAAAA }
         BBBBBBB }  1/6 inch = 12 steps/72
         CCCCCCC
         DDDDDDD }
                 }  1/3 inch = 24 steps/72
         EEEEEEE }

         FFFFFFF
```

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
($1 \leq <n> 10 \leq 66$, $1 \leq k \leq 64$, $nk \leq nk+1$)
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.

HARDWARE

```
[DATA]   [ESC B]  <4> H   <6> H   <A> H   [NUL]
         AAAAAAA  [VT]  BBBBBBB  [VT]  CCCCCCC  [VT]  DDDDDDD
[PRINT]  AAAAAAA . . . . 1st line
         BBBBBBB . . . . 4th lines
         CCCCCCC . . . . 6th lines
         DDDDDDD . . . . 10th lines
```

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
   CR LF
   [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] ABC DEF GHIJKL
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.

[DATA] $\boxed{\text{ESC G}}$ ABCDEFGHI $\boxed{\text{CR}}$ $\boxed{\text{LF}}$
[PRINT] ABCDEFGHI

8) ESC H
The ESC H code cancels the double printing mode.

HARDWARE

NOTES

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

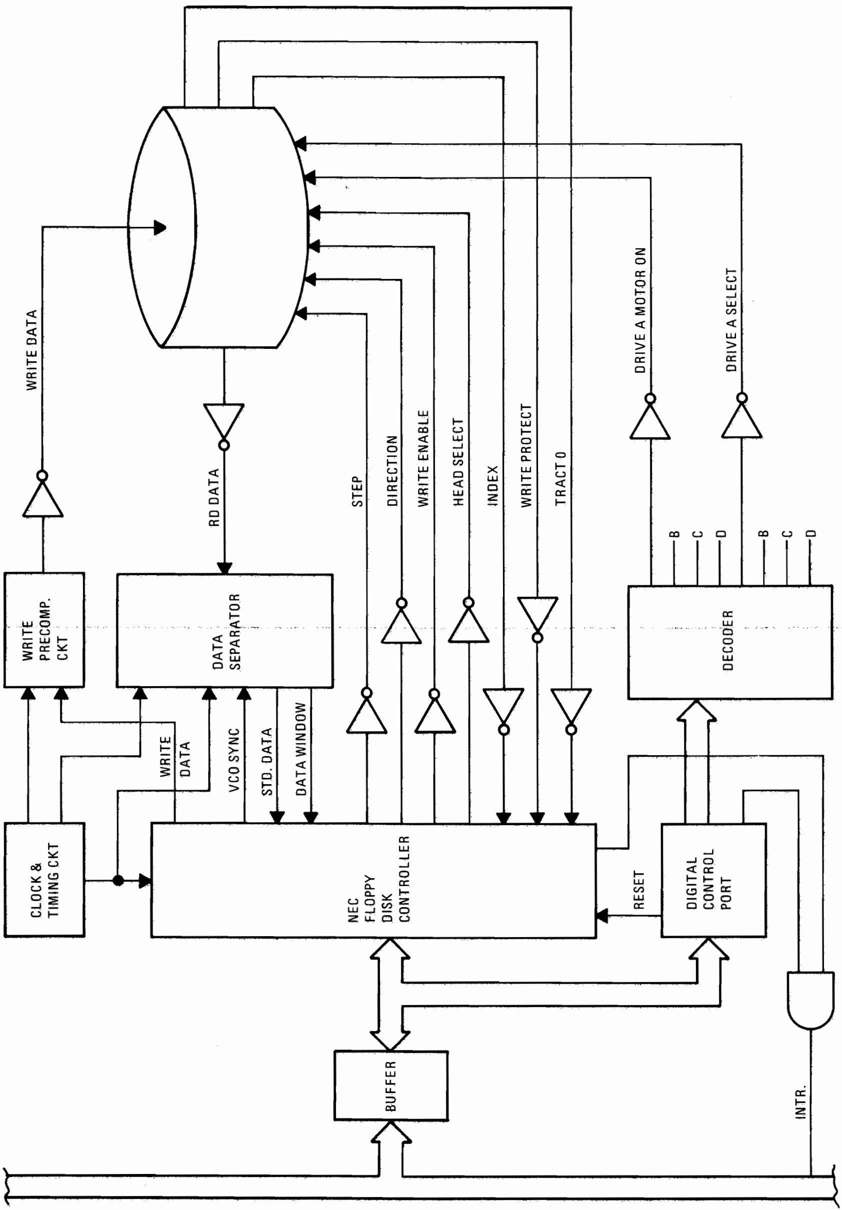


Figure 19. 5¼" DISKETTE DRIVE ADAPTER BLOCK DIAGRAM

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:

Bit 1	Bit 0	Drive	
0	0	0	A
0	1	1	B
1	0	2	C
1	1	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:

Bit Number	Name	Symbol	Description
DB0 FDD	FDD A Busy	DAB	FDD number is in the Seek mode.
DB1	FDD B Busy	DBB	FDD number 1 is in the Seek mode.
DB2	FDD C Busy	DCB	FDD number 2 is in the Seek mode.
DB3	FDD D Busy	DDB	FDD number 3 is in the Seek mode.
DB4	FDC Busy	CB	A read or write command is in process.
DB5	Non-DMA Mode	NDM	The FDC is in the non-DMA mode.
DB6	Data Input/	DIO	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.
DB7	Request for Master	RQM	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.

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.

Table 13. Symbol Descriptions

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

SYMBOL	NAME	DESCRIPTION
A0	Address Line 0	A0 controls selection of Main Status Register (A0 = 0) or Data Register (A0 = 1).
C	Cylinder Number	C stands for the current/selected Cylinder (track) number of the medium.
D	Data	D stands for the data pattern which is going to be written into a Sector.
D7-D0	Data Bus	8-bit Data Bus, where D7 stands for a most significant bit, and D0 stands for a least significant bit.
DTL	Data Length	When N is defined as 00, DTL stands for the data length which users are going to read out or write into the Sector.
EOT	End of Track	EOT stands for the final Sector number on a Cylinder.
GPL	Gap Length	GPL stands for the length of Gap 3 (spacing between Sectors excluding VCO Sync. Field).
H	Head Address	H stands for head number 0 or 1, as specified in ID field.
HD	Head	HD stands for a selected head number 0 or 1. (H = HD in all command words.)
HLT	Head Load Time	HLT stands for the head load time in the FDD (4 to 512 ms in 4 ms increments).
HUT	Head Unload Time	HUT stands for the head unload time after a read or write operation has occurred (0 to 480 ms in 32 ms increments.)
MF	FM or MFM Mode	If MF is low, FM mode is selected, and if it is high, MFM mode is selected only if MFM is implemented.
MT	Multi-Track	If MT is high, a multi-track operation is to be performed. (A cylinder under both HD0 and HD1 will be read or written.)
N	Number	N stands for the number of data bytes written in a Sector.
NCN	New Cylinder Number	NCN stands for a new Cylinder number, which is going to be reached as a result of the Seek operation. Desired position of Head.

Table 13. Symbol Descriptions (continued)

SYMBOL	NAME	DESCRIPTION
ND	Non-DMA Mode	ND stands for operation in the Non-DMA Mode.
PCN	Present Cylinder Number	PCN stands for Cylinder number at the completion of SENSE INTERRUPT STATUS Command, indicating the position of the Head at present time.
R	Record	R stands for the Sector number, which will be read or written.
R/W	Read/Write	R/W stands for either Read (R) or Write (W) signal.
SC	Sector	SC indicates the number of Sectors per Cylinder.
SK	Skip	SK stands for Skip Deleted Data Address Mark.
SRT	Step Rate Time	SRT stands for the Stepping Rate for the FDD. (2 to 32 ms in 2 ms increments.)
ST 0 ST 1 ST 2 ST 3	Status 0 Status 1 Status 2 Status 3	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 A0 = 0). ST 0–3 may be read only after a command has been executed and contain information relevant to that particular command.
STP	Scan Test	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.
US0, US1	Unit Select	US stands for a selected drive number encoded the same as bits 0 and 1 of the digital register (DOR) p 2-91

Command Summary

0 indicates 'logical 0' for that bit, 1 means 'logical 1', X means 'don't care'.

PHASE	R/W	D7	D6	D5	D4	D3	D2	D1	D0	REMARKS
					DATA BUS					
					READ DATA					
Command	W	MT	MF	SK	0	0	1	1	0	Command Codes
	W	X	X	X	X	X	HD	US1	US0	
	W				C					Sector ID
	W				H					information prior
	W				R					to Command execution
	W				N					
	W				EOT					
	W				GPL					
	W				DTL					
Execution										Data-transfer between
										the FDD and main-system
Result	R				ST 0					Status information after
	R				ST 1					Command execution
	R				ST 2					
	R				C					Sector ID information
	R				H					after Command execution
	R				R					
	R				N					
					READ DELETED DATA					
Command	W	MT	MF	SK	0	1	1	0	0	Command Codes
	W	X	X	X	X	X	HD	US1	US0	
	W				C					Sector ID information
	W				H					prior to Command
	W				R					execution
	W				N					
	W				EOT					
	W				GPL					
	W				DTL					
Execution										Data-transfer between
										the FDD and main-system
Result	R				ST 0					Status information after
	R				ST 1					command execution
	R				ST 2					
	R				C					Sector ID information
	R				H					after command execution
	R				R					
	R				N					
					WRITE DATA					
Command	W	MT	MF	0	0	0	1	0	1	Command Codes
	W	X	X	X	X	X	HD	US1	US0	
	W				C					Sector ID information
	W				H					to command execution
	W				R					
	W				N					
	W				EOT					
	W				GPL					
	W				DTL					
Execution										Data-transfer between
										the main-system and FDD
Result	R				ST 0					Status information after
	R				ST 1					command execution
	R				ST 2					
	R				C					Sector ID information
	R				H					after command execution
	R				R					
	R				N					

Command Summary (continued)

PHASE	R/W	D7	D6	D5	D4	D3	D2	D1	D0	REMARKS
					WRITE DELETED DATA					
Command	W	MT	MF	0	0	1	0	0	1	Command Codes
	W	X	X	X	X	X	HD	US1	US0	
	W				C					Sector ID information prior to command execution
	W				H					
	W				R					
	W				N					
	W				EOT					
	W				GPL					
	W				DTL					
Execution										Data-transfer between FDD and main-system
Result	R				ST 0					Status ID information after common execution
	R				ST 1					
	R				ST 2					
	R				C					Sector ID information after command execution
	R				H					
	R				R					
	R				N					
					READ A TRACK					
Command	W	0	MF	SK	0	0	0	1	0	Command Codes
	W	X	X	X	X	X	HD	US1	US0	
	W				C					Sector ID information prior to command execution
	W				H					
	W				R					
	W				N					
	W				EOT					
	W				GPL					
	W				DTL					
Execution										Data-transfer between the FDD and main-system. FDC reads all of cylinders contents from index hole to EOT.
Result	R				ST 0					Status information after command execution
	R				ST 1					
	R				ST 2					
	R				C					Sector ID information after command execution
	R				H					
	R				R					
	R				N					
					READ ID					
Command	W	0	MF	0	0	1	0	1	0	Command Codes
	W	X	X	X	X	X	HD	US1	US0	
Execution										The first correct ID information on the cylinder is stored in data register.
Result	R				ST 0					Status information after command execution
	R				ST 1					
	R				ST 2					
	R				C					Sector ID information during execution phase
	R				H					
	R				R					
	R				N					

HARDWARE

Command Summary (continued)

PHASE	R/W	D7	D6	D5	D4	D3	D2	D1	D0	REMARKS
					FORMAT A TRACK					
Command	W	0	MF	0	0	1	1	0	0	Command Codes
	W	X	X	X	X	X	HD	US1	US0	
	W				N					Bytes/Sector
	W				SC					Sector/Track
	W				GPL					Gap 3
	W				D					filler byte
Execution										FDC formats an entire cylinder
Result	R				ST 0					Status information after command execution
	R				ST 1					
	R				ST 2					
	R				C					In this case, the ID information has no meaning
	R				H					
	R				R					
	R				N					
					SCAN EQUAL					
Command	W	MT	MF	SK	1	0	0	0	1	Command Codes
	W	X	X	X	X	X	HD	US1	US0	
	W				C					Sector ID information prior to command execution
	W				H					
	W				R					
	W				N					
	W				EOT					
	W				GPL					
	W				STP					
Execution										Data compared between the FDD and main-system
Result	R				ST 0					Status information after command execution
	R				ST 1					
	R				ST 2					
	R				C					Sector ID information
	R				H					
	R				R					
	R				N					
					SCAN LOW OR EQUAL					
Command	W	MT	MF	SK	1	1	0	0	1	Command Codes
	W	X	X	X	X	X	HD	US1	US0	
	W				C					Sector ID information prior to command execution
	W				H					
	W				R					
	W				N					
	W				EOT					
	W				GPL					
	W				STP					
Execution										Data compared between the FDD and main-system
Result	R				ST 0					Status information after command execution
	R				ST 1					
	R				ST 2					
	R				C					Sector ID information after command execution
	R				H					
	R				R					
	R				N					

Command Summary (continued)

PHASE	R/W	D7	D6	D5	D4	D3	D2	D1	D0	REMARKS
					DATA BUS					
					SCAN HIGH OR EQUAL					
Command	W	MT	MF	SK	1	1	1	0	1	Command Codes
	W	X	X	X	X	X	HD	US1	US0	
	W				C					Sector ID information prior to command execution
	W				H					
	W				R					
	W				N					
	W				EOT					
	W				GPL					
	W				STP					
Execution										Data compared between the FDD and main-system
Result	R				ST 0					Status information after command execution
	R				ST 1					
	R				ST 2					
	R				C					Sector ID information after command execution
	R				H					
	R				R					
	R				N					
					RECALIBRATE					
Command	W	0	0	0	0	0	1	1	1	Command Codes
	W	X	X	X	X	X	0	US1	US0	
Execution										Head retracted to track 0
No Result Phase										
					SENSE INTERRUPT STATUS					
Command	W	0	0	0	0	1	0	0	0	Command Codes
Result	R				ST 0					Status information at the end of seek operation about the FDC
	R				PCN					
					SPECIFY					
Command	W	0	0	0	0	0	0	1	1	Command Codes
	W	——SRT——				——HUT——				
	W	——HLT——							ND	
No Result Phase										
					SENSE DRIVE STATUS					
Command	W	0	0	0	0	0	1	0	0	Command Codes
	W	X	X	X	X	X	HD	US1	US0	
Result	R				ST 3					Status information about FDD
					SEEK					
Command	W	0	0	0	0	1	1	1	1	Command Codes
	W	X	X	X	X	X	HD	US1	US0	
	W				NCN					
Execution										Head is positioned over proper cylinder on diskette
No Result Phase										
					INVALID					
Command	W				Invalid Codes					Invalid command codes (NoOp – FDC goes into standby state)
Result	R				ST 0					ST 0 = 80

Command Status Registers

Table 14. Status Register 0

BIT			
NO.	NAME	SYMBOL	DESCRIPTION
D7 D6	Interrupt Code	IC	D7 = 0 and D6 = 0 Normal termination of command, (NT), Command was completed and properly executed. D7 = 0 and D6 =1 Abnormal termination of command, (AT). Execution of command was started, but was not successfully completed. D7 =1 and D6 = 0 Invalid command issue (IC). Command which was issued was never started. D7 = 1 and D6 =1 Abnormal termination because during command execution the ready signal from FDD changed state.
D5	Seek End	SE	When the FDC completes the Seek command, this flag is set to 1 (high).
D4	Equipment Check	EC	If a fault signal is received from the FDD, or if the track 0 signal fails to occur after 77 step pulses (recalibrate command) then this flag is set.
D3	Not Ready	NR	When the FDD is in the not-ready state and a read or write command is issued, this flag is set. If a read or write command is issued to side 1 of a single sided drive, then this flag is set.
D2	Head Address	HD	This flag is used to indicate the state of the head at interrupt.
D1 D0	Unit Select 1 Unit Select 0	US 1 US 0	These flags are used to indicate a Drive unit Number at interrupt.

Table 15. Status Register 1

BIT			
NO.	NAME	SYMBOL	DESCRIPTION
D7	End of Cylinder	EN	When the FDC tries to access a sector beyond the final sector of a cylinder, this flag is set.
D6	–	–	Not used. This bit is always 0 (low).
D5	Data Error	DE	When the FDC detects a CRC error in either the ID field or the data field, this flag is set.
D4	Over Run	OR	If the FDC is not serviced by the main-systems during data transfers within a certain time interval, this flag is set.
D3	–	–	Not used. This bit is always 0 (low).
D2	No Data	ND	During Execution of a Read Data, Write Deleted Data, or Scan command, if the FDC cannot find the sector specified in the ID register, this flag is set. During execution of the Read ID command, if the FDC cannot read the ID field without an error, then this flag is set. During the execution of the Read-a-Cylinder command, if the starting sector cannot be found, then this flag is set.
D1	Not Writable	NW	During Execution of a Write Data, Write Deleted Data, or Format a Cylinder command, if the FDC detects a write protect signal from the FDD, then this flag is set.
D0	Missing Address Mark	MA	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.

Table 16. Status Register 2

BIT			DESCRIPTION
NO.	NAME	SYMBOL	
D7	–	–	Not Used. This bit is always 0 (low).
D6	Control Mark	CM	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.
D5	Data Error in Data Field	DD	If the FDC detects a CRC error in the data then this flag is set.
D4	Wrong Cylinder	WC	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.
D3	Scan Equal Hit	SH	During execution of the Scan command, if the condition of "equal" is satisfied, this flag is set.
D2	Scan Not Satisifed	SN	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.
D1	Bad Cylinder	BC	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.
D0	Missing Address Mark in Data Field	MD	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.

Table 17. Status Register 3

BIT			DESCRIPTION
NO.	NAME	SYMBOL	
D7	Fault	FT	This bit is used to indicate the status of the Fault signal from the FDD.
D6	Write Protected	WP	This bit is used to indicate the status of the Write Protected signal from the FDD.
D5	Ready	RY	This bit is used to indicate the status of the Ready signal from the FDD.
D4	Track 0	T0	This bit is used to indicate the status of the Track 0 signal from the FDD.
D3	Two Side	TS	This bit is used to indicate the status of the Two Side signal from the FDD.
D2	Head Address	HD	This bit is used to indicate the status of Side Select signal to the FDD.
D1	Unit Select 1	US 1	This bit is used to indicate the status of the Unit Select 1 signal to the FDD.
D0	Unit Select 0	US 0	This bit is used to indicate the status of the Unit Select 0 signal to the FDD.

HARDWARE

Programming Summary

DPC Registers (Ports)

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

Bit 0	Drive	00: DR #A	10: DR #C
1	Select	01: DR #B	11: DR #D
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: 1 74LS; Driver: 74LS 3-state)

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

+A0-9 (Adapter Input, Load: 1 74LS)

These ten lines form an address bus by which a register is selected to receive or supply the byte transferred 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 completed an operation. It results in an interrupt to a routine which should examine the FDC result bytes to reset the line and determine the ending condition.

Drive A and B Interface

All signals are TTL compatible:

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

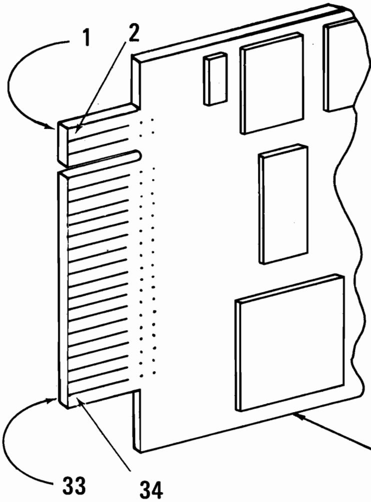
-Drive Select A&B	(Driver: 7438) These two lines are used by drives A&B to degate all drivers to the adapter and receivers from the attachment (except Motor Enable) when the line associated with a drive is not active.
-Motor Enable A&B	(Driver: 7438) The drive associated with each of these lines must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

-Index	The selected drive supplies one pulse per disk revolution on this line.
-Write Protect	The selected drive makes this line active if a write protected diskette is mounted in the drive.
-Track 0	The selected drive makes this line active if the read/write head is over track 0.
-Read Data	The selected drive supplies a pulse on this line for each flux change encountered on the disk.

HARDWARE

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.

COMPONENT
SIDE

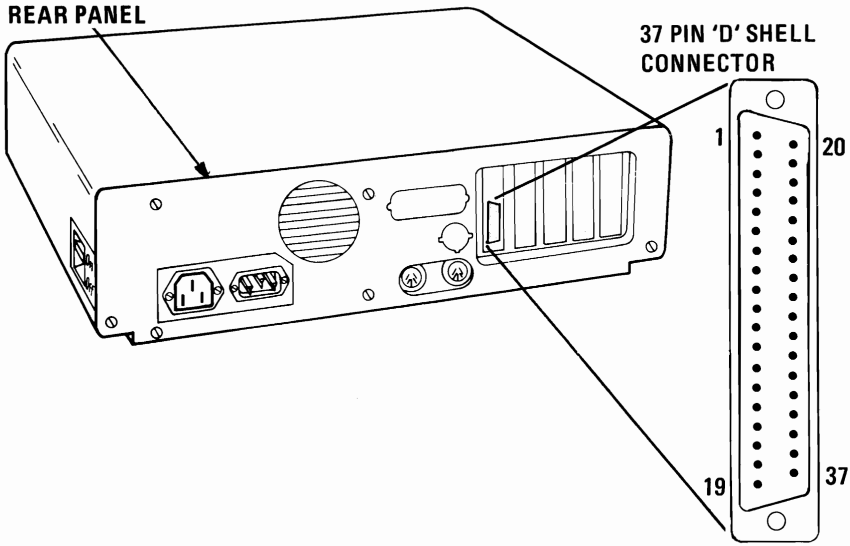
AT STANDARD TTL LEVELS

IBM 5 1/4" Diskette Drives

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

5 1/4" Diskette Drive Adapter

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



AT STANDARD TTL LEVELS

External Drives ↔ 5¼" Diskette Drive Adapter

Signal	Pin no.
Unused	1 - 5
Index	6
Motor Enable C	7
Drive Select D	8
Drive Select C	9
Motor Enable D	10
Direction (Stepper Motor)	11
Step Pulse	12
Select Head 1	13
Write Enable	14
Track 0	15
Write Protect	16
Read Data	17
Write Data	18
Ground	20 - 37

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.

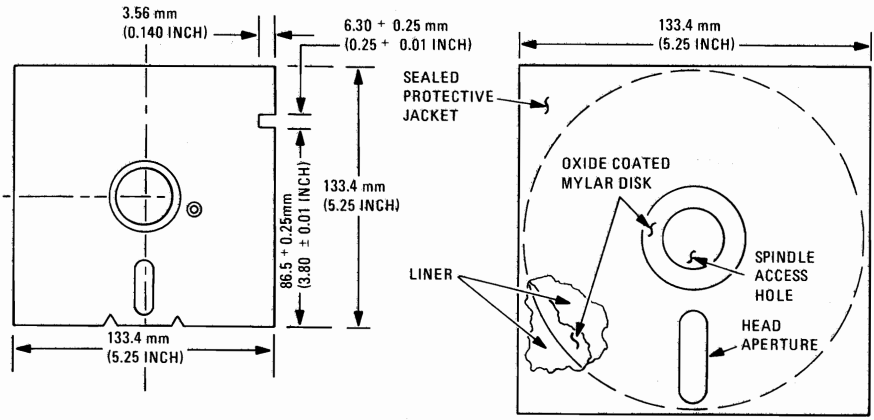
(2) The index sensor, which consists of a LED light source and phototransistor, is positioned such that when an index hole is detected, a sigital signal is generated.

(3) 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.



RECORDING MEDIUM

Table 18. Mechanical and Electrical Specifications

Media	Industry-compatible 5¼-inch diskette
Tracks per inch	48
Number of Tracks	(40)
Dimensions	
Height	85.85 mm (3.38 inches)
Width	149.10 mm (5.87 inches)
Depth	203.2 mm (8.0 inches)
Weight	2.04 Kg (4.5 lbs.)
Temperature (Exclusive of Media)	
Operating	10°C to 44°C (50°F to 112°F)
Non-operating	–40°C to 60°C (–40°F to 140°F)
Relative Humidity (Exclusive of Media)	
Operating	20% to 80% (Non-condensing)
Non-operating	5% to 95% (Non-condensing)
Seek Time	8 msec track to track
Head Setting Time	25 msec (last track addressed)
Error Rate	1 per 10^9 (recoverable) 1 per 10^{12} (non-recoverable) 1 per 10^6 (seeks)
Head Life	20,000 hours (normal use)
Media Life	3.0 x 10^6 passes per track
Disk Speed	300 rpm ± 1.5% (long term)
Instantaneous Speed Variation	± 3.0%
Start/Stop Time	500 msec (maximum)
Transfer Rate	250K bits/sec
Recording Mode	MFM
Power	+12 dc ± 0.6v 900 ma AVE. +5v dc ± 0.25 v, 600 ma AVE.

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

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

* 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

NO.	DESCRIPTION
1	ON: A19=0 ; OFF: A19=1
2	ON: A18=0 ; OFF: A18=1
3	ON: A17=0 ; OFF: A17=1
4	ON: A16=0 ; OFF: A16=1
5	ON: A15=0 ; OFF: A15=1 *
6	Not Used
7	Not Used
8	Used Only In 64KB RAM Card *

* 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.

NOTES

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

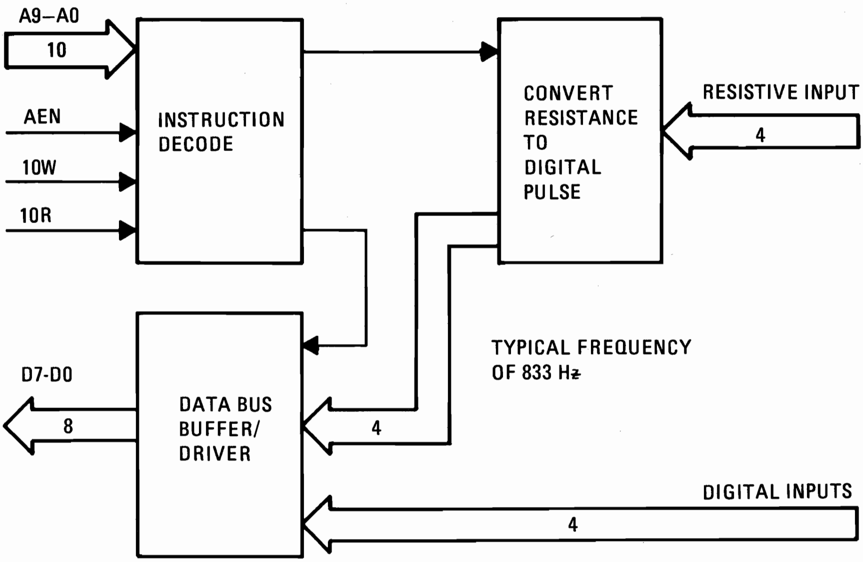


Figure 20. GAME CONTROL ADAPTER BLOCK DIAGRAM

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

HARDWARE

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, acccording to the following equation:

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

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'

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

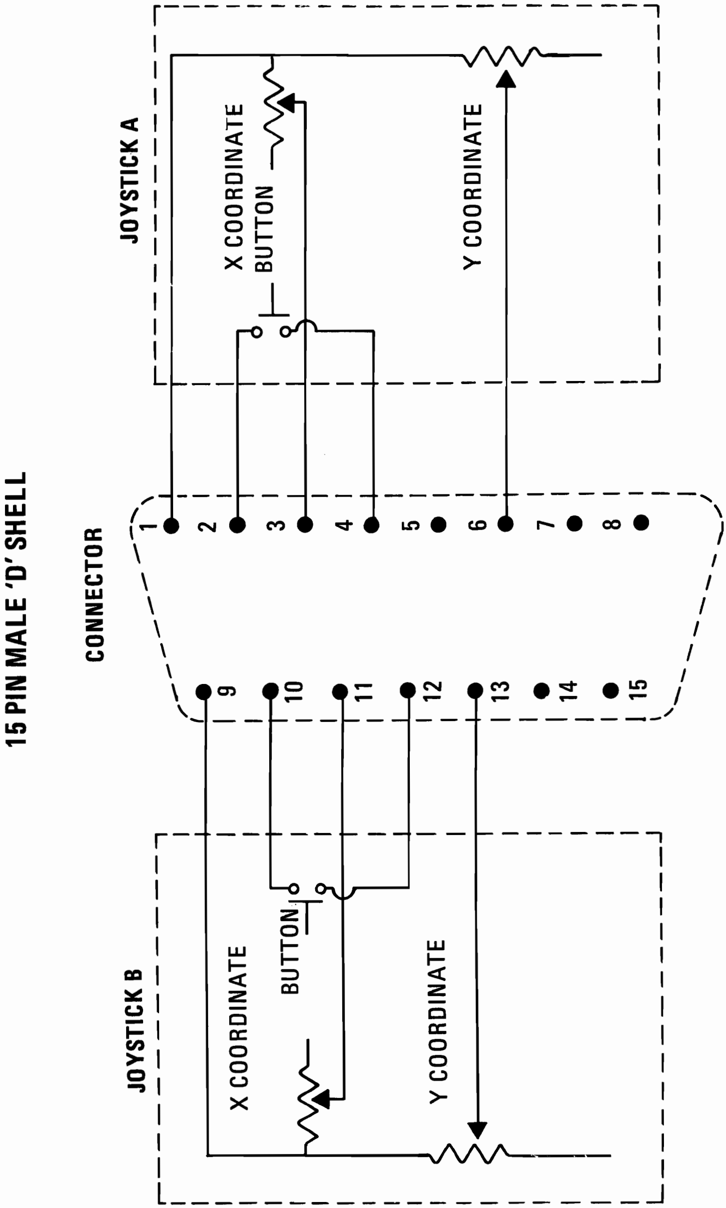
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
B-#2 Button	B-#1 Button	A-#2 Button	A-#1 Button	B-Y Coord.	B-X Coord.	A-Y Coord.	A-X Coord.

The game paddles will have a set of two (A&B) or four (A,B,C, & D) paddles. These will have one button each and one variable resistance each, with a range from 0 to 100 K ohms. This should be attached to give the following input data:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
D Button	C Button	B Button	A Button	D Coord.	C Coord.	B Coord.	A Coord.

A schematic diagram for attaching a set of game controllers is on page 2-121.

Joystick Schematic

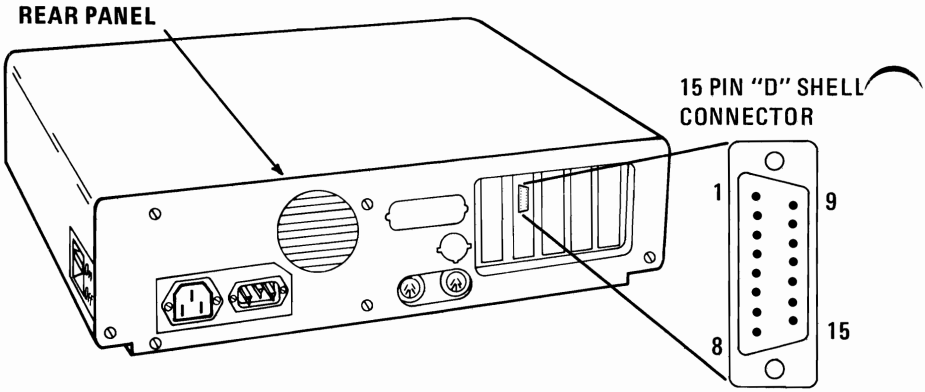


NOTE: POTENTIOMETER FOR X & Y COORDINATES HAS A RANGE OF 0 TO 100KΩ.
BUTTON IS NORMALLY OPEN; CLOSED WHEN DEPRESSED.

Figure 21. JOYSTICK SCHEMATIC

Game Controller Adapter (Analog Input)

Connector Specifications



AT STANDARD TTL LEVELS

External Devices	Voltage	AMP Pin No.	Game Control Adapter
←	+ 5 Volts	1	
	Button 4	2	→
	Position 0	3	→
	Ground	4	
	Ground	5	
	Position 1	6	→
	Button 5	7	→
←	+ 5 Volts	8	
←	+ 5 Volts	9	
	Button 6	10	→
	Position 2	11	→
	Ground	12	
	Position 3	13	→
	Button 7	14	→
←	+ 5 Volts	15	

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

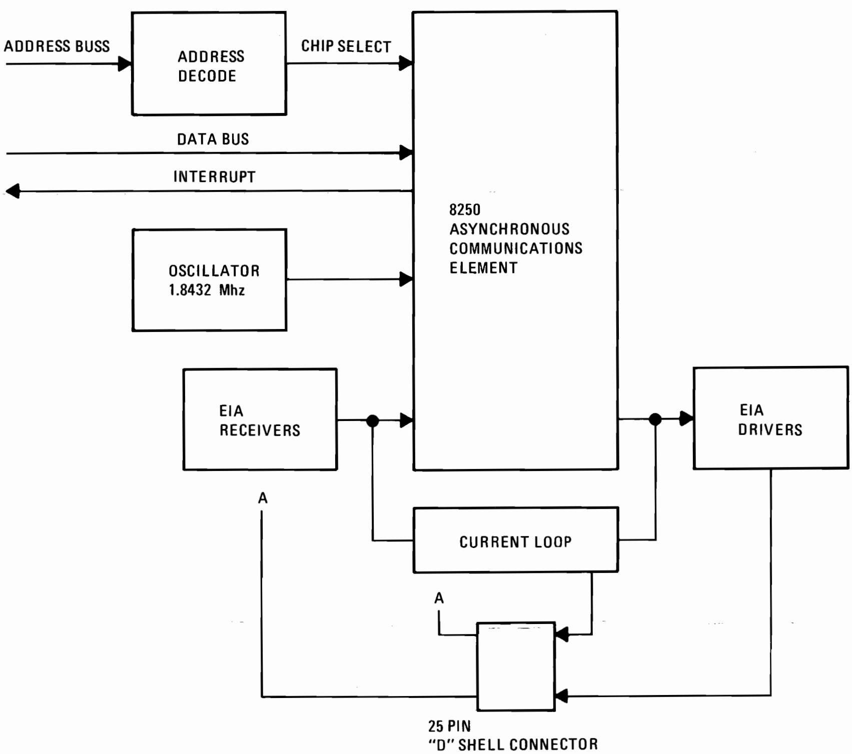


Figure 22. ASYNCHRONOUS COMMUNICATIONS ADAPTER BLOCK DIAGRAM

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)

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

ADDRESS BITS

	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	DLAB	REGISTER
3F8 to 3FF	1	1	1	1	1	1	1	X	X	X		
								0	0	0	0	Receive Buffer (read), Transmit Holding Reg. (write)
								0	0	1	0	Interrupt Enable
								0	1	0	X	Interrupt Identification
								0	1	1	X	Line Control
								1	0	0	X	Modem Control
								1	0	1	X	Line Status
								1	1	0	X	Modem Status
								1	1	1	X	None
								0	0	0	1	Divisor Latch (LSB)
								0	0	1	1	Divisor Latch (MSB)

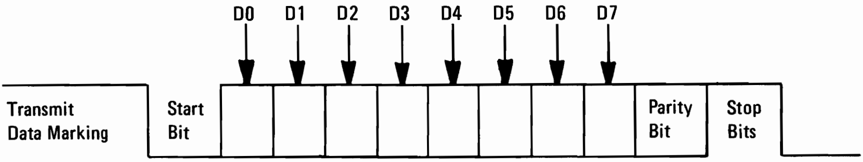
A2, A1 and A0 bits are "Don't Cares" and are used to select the different register of the communications chip.

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



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)

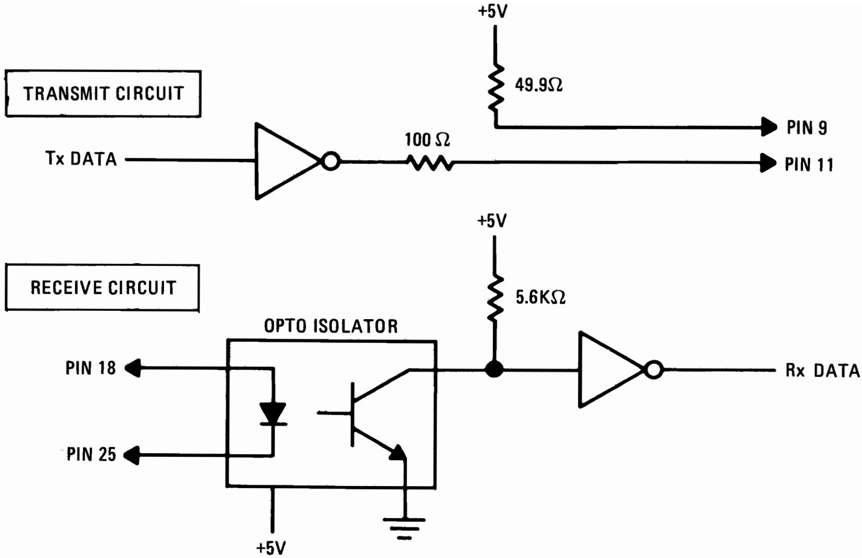


Figure 23. CURRENT LOOP INTERFACE

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

Interchange Voltage	Binary State	Signal Condition	Interface Control Function
Positive Voltage =	Binary (0)	= Spacing	=On
Negative Voltage =	Binary (1)	= Marking	=Off

Invalid Levels
+15V
On Function
+3V
0V Invalid Levels
−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 (SC0, 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 enable comunication between the INS8250 and the CPU.

Data Input Strobe (DISTRDISTR) 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.

HARDWARE

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.

DLAB	A2	A1	A0	Register
0	0	0	0	Receiver Buffer (read), Transmitter Holding Register (write)
0	0	0	1	Interrupt Enable
X	0	1	0	Interrupt Identification (read only)
X	0	1	1	Line Control
X	1	0	0	MODEM Control
X	1	0	1	Line Status
X	1	1	0	MODEM Status
X	1	1	1	None
1	0	0	0	Divisor Latch (least significant byte)
1	0	0	1	Divisor Latch (most significant byte)

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.

<u>Note:</u> 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.

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.

HARDWARE

Programming Considerations

Table 22. Asynchronous Communications Reset Functions

Register/Signal	Reset Control	Reset State
Interrupt Enable Register	Master Reset	All Bits Low (0–3 Forced and 4–7 Permanent)
Interrupt Identification Register	Master Reset	Bit 0 is High, Bits 1 and 2 Low Bits 3–7 are Permanently Low
Line Control Register	Master Reset	All Bits Low
MODEM Control Register	Master Reset	All Bits Low
Line Status Register	Master Reset	Except Bits 5 & 6 are High
MODEM Status Register	Master Reset	Bits 0–3 Low Bits 4–7 – Input Signal
SOUT	Master Reset	High
INTRPT (RCVR Errs)	Read LSR/MR	Low
INTRPT (RCVR Data Ready)	Read RBR/MR	Low
INTRPT (RCVR Data Ready)	Read IIR/Write THR/MR	Low
INTRPT (MODEM Status Changes)	Read MSR/MR	Low
OUT 2	Master Reset	High
RTS	Master Reset	High
DTR	Master Reset	High
OUT 1	Master Reset	High

INS8250 Accessible Registers

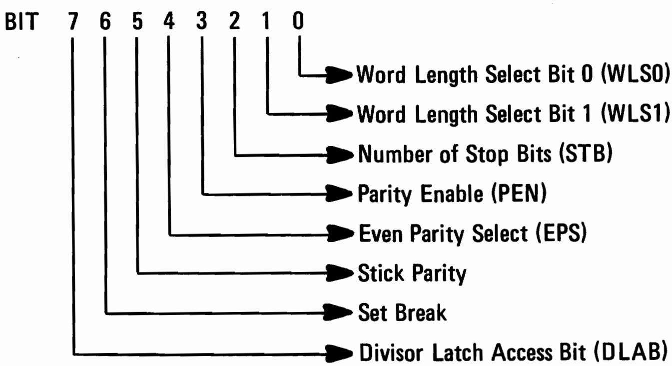
The system programmer may access or control any of the INS8250 registers via the CPU. These registers are used to control INS8250 operations and to transmit and receive data.

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



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:

Bit 1	Bit 0	Word Length
0	0	5 Bits
0	1	6 Bits
1	0	7 Bits
1	1	8 Bits

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.

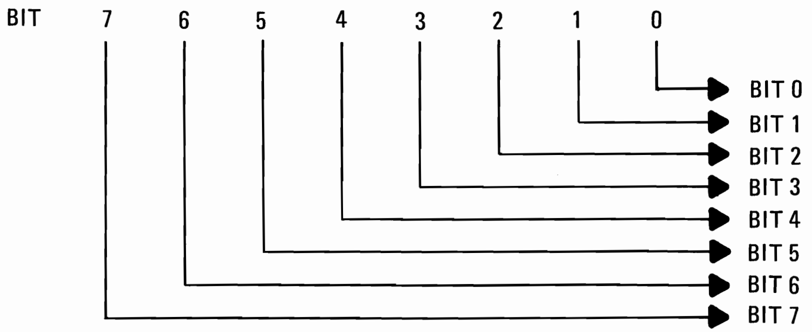
INS8250 Programmable Baud Rate Generator

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 [divisor #= (frequency input) / (baud rate x 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.

HARDWARE

Divisor Latch Least Significant Bit (DLL)

3F8 DLAB=1



Divisor Latch Most Significant Bit (DLM)

3F9 DLAB=1

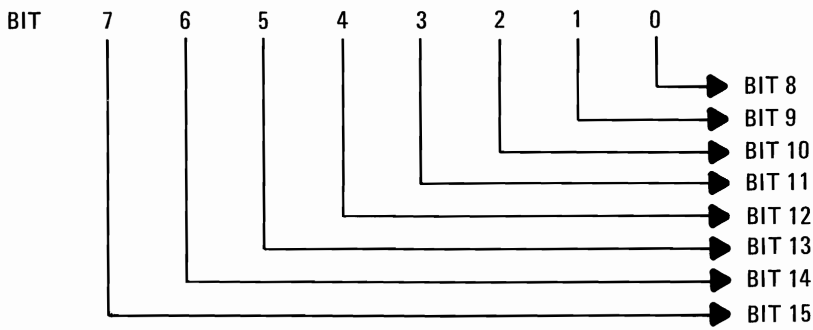


Table 23 illustrates the use of the Baud Rate Generator with a frequency of 1.8432 Mhz. For baud rates of 9600 and below, the error obtained is minimal.

<u>Note:</u> 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

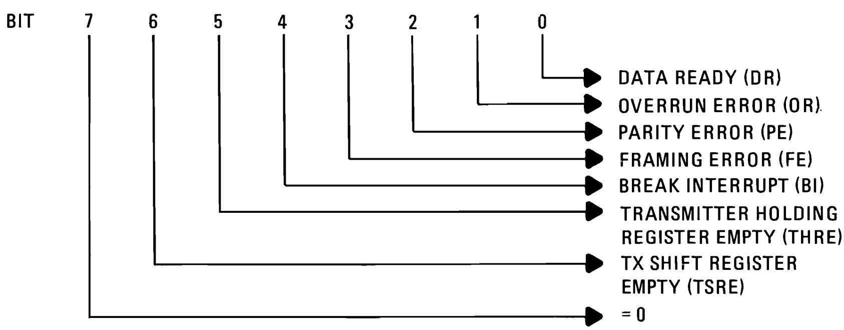
Desired Baud Rate	Divisor Used to Generate 16x Clock		Percent Error Difference Between Desired & Actual
	Decimal	Hex	
50	2304	'900'	--
75	1536	'600'	--
110	1047	'417'	0.026
134.5	857	'359'	0.058
150	768	'300'	--
300	384	'180'	--
600	192	'0C0'	--
1200	96	'060'	--
1800	64	'040'	--
2000	58	'03A'	0.69
2400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

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



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 Tranmitter 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)

3FA

BIT	
0	0 IF INTERRUPT PENDING
1	INTERRUPT ID BIT (0)
2	INTERRUPT ID BIT (1)
3	= 0
4	= 0
5	= 0
6	= 0
7	= 0

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

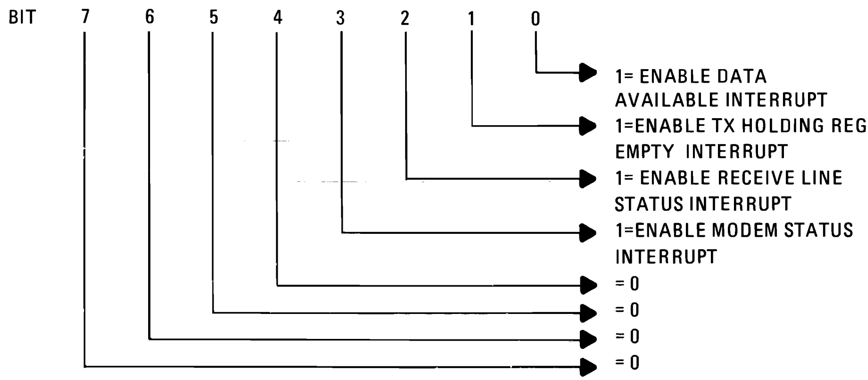
Interrupt ID Register			Interrupt Set and Reset Functions			
Bit 2	**Bit 1**	**Bit 0**	**Priority Level**	**Interrupt Type**	**Interrupt Source**	**Interrupt Reset Control**
0	0	1	–	None	None	–
1	1	0	Highest	Receiver Line Status	Overrun Error or Parity Error or Framing Error or Break Interrupt	Reading the Line Status Register
1	0	0	Second	Received Data Available	Receiver Data Available	Reading the Receiver Buffer Register
0	1	0	Third	Transmitter Holding Register Empty	Transmitter Holding Register Empty	Reading the IIR Register (if source of interrupt) or Writing into the Transmitter Holding Register
0	0	0	Fourth	MODEM Status	Clear to Send or Data Set Ready or Ring Indicator or Received Line Signal Detect	Reading the MODEM Status Register

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



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.

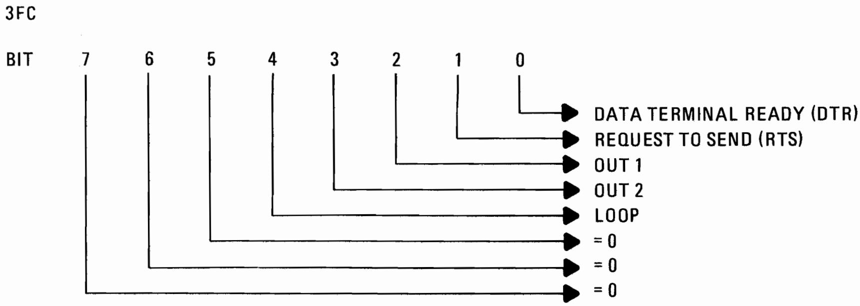
Bit 3: This bit enables the MODEM Status Interrupt when set to logic 1.

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)



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.

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.

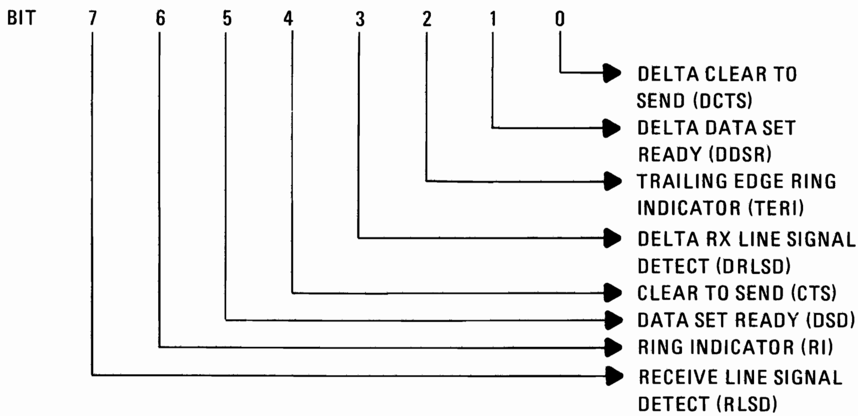
MODEM Status Register

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)

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.

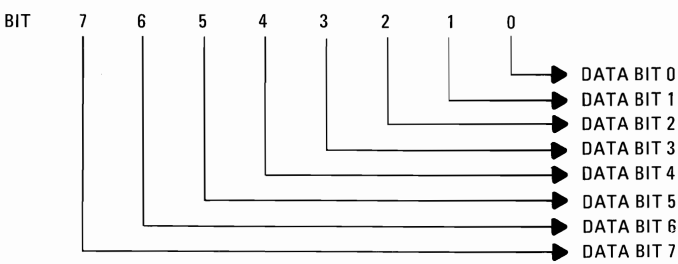
Bit 7: This bit is the complement of the Received Line Signal Detect (RLSD) input. If bit 4 of the MCR is set to a 1, this bit is equivalent to OUT 2 of the MCR.

Receiver Buffer Register

The Receiver Buffer Register contains the received character as defined below.

Receiver Buffer Register (RBR)

3F8 DLAB=0 READ ONLY



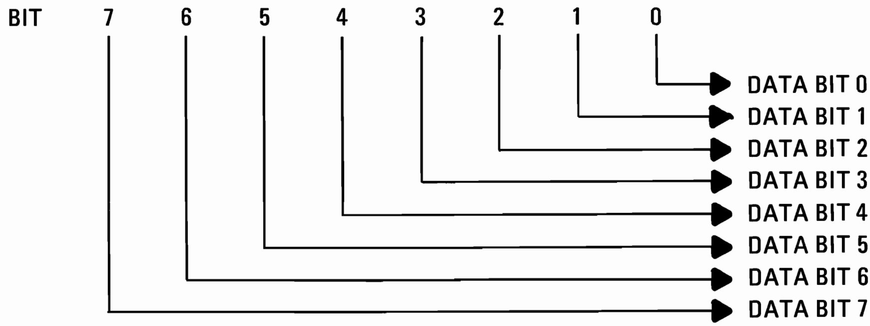
Bit 0 is the least significant bit and is the first bit serially received.

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.

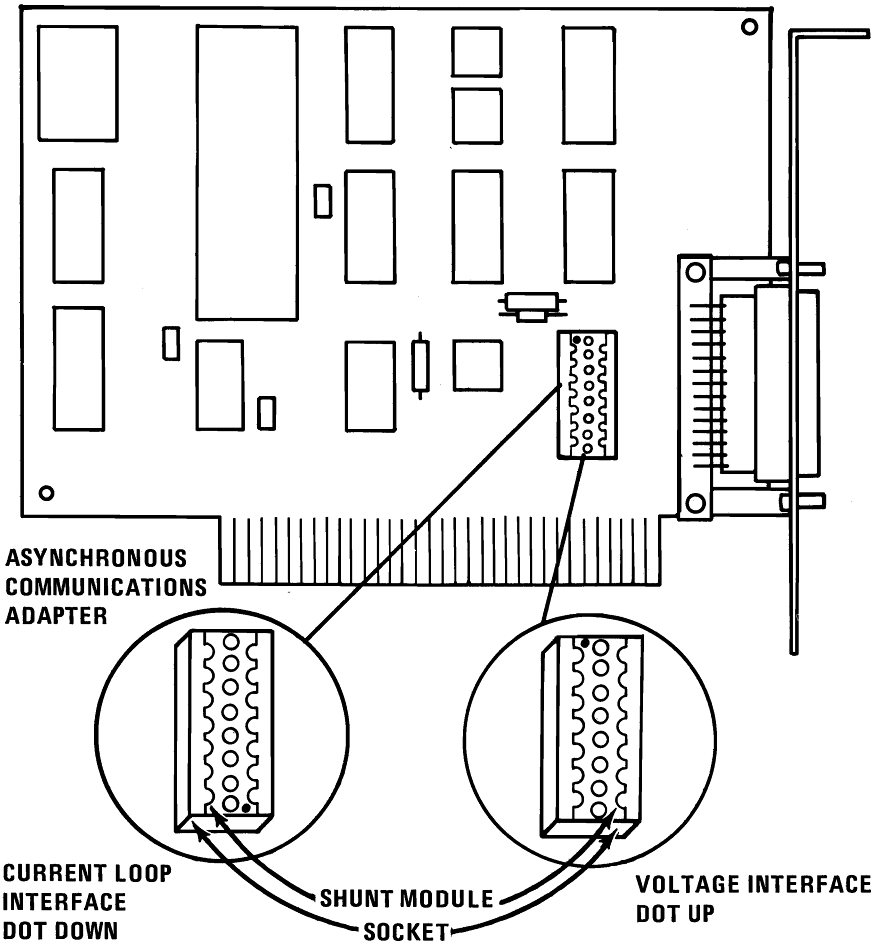
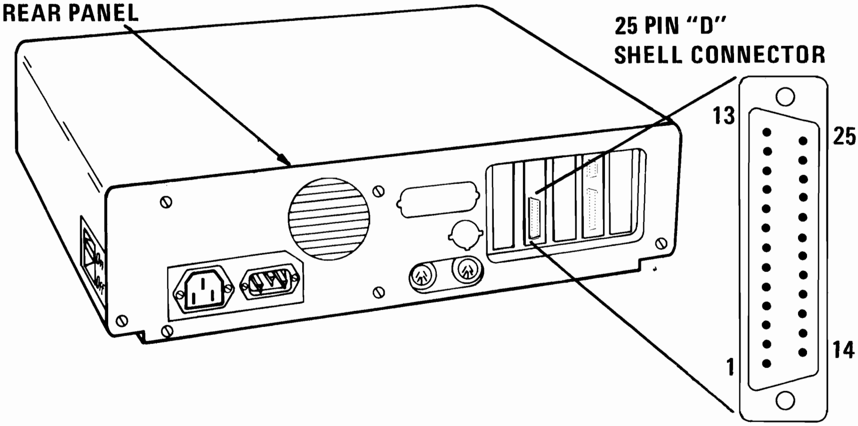


Figure 23. SELECTING THE INTERFACE FORMAT

Asynchronous Communications Adapter Connector Interface Specifications



AT STANDARD TTL LEVELS

External Device ↔ Asynchronous Communications Adapter (RS232C)

Description	Pin
NC	1
Transmit Data	2
Receive Data	3
Request to send	4
Clear to send	5
Data set ready	6
Signal ground	7
Carrier detect	8
+Transmit current loop return (20 ma)	9
NC	10
−Transmit current loop data (20 ma)	11
NC	12
NC	13
NC	14
NC	15
NC	16
NC	17
+Receive current loop data (20 ma)	18
NC	19
Data Terminal Ready	20
NC	21
Ring Indicate	22
NC	23
NC	24
−Receive current loop return (20 ma)	25

NOTE: To avoid inducing voltage surges on interchange circuits, signals from interchange circuits shall not be used to drive inductive devices, such as relay coils.

NOTES

SECTION 3. ROM and SYSTEM USAGE

Contents:

ROM

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

Interrupt Number	Name	BIOS Initialization
0	Divide by Zero	None
1	Single Step	None
2	Non Maskable	NMI_INT (F000:E2C3)
3	Breakpoint	None
4	Overflow	None
5	Print Screen	PRINT_SCREEN (F000:FF54)
6	Unused	
7	Unused	
8	Time of Day	TIMER_INT (F000:FEA5)
9	Keyboard	KB_INT (F000:E987)
A	Unused	
B 8259	Unused	
C Interrupt	Unused (Reserved Communications)	
D Vectors	Unused	
E	Diskette	DISK_INT (F000:EF57)
F	Unused (Reserved Printer)	
10	Video	VIDEO_I O (F000:F065)
11	Equipment Check	EQUIPMENT (F000:F84D)
12	Memory	MEMORY_SIZE_DETERMINE (F000:F841)
13	Diskette	DISKETTE_I O (F000:EC59)
14 BIOS	Communications	RS232_I O (F000:E739)
15 Entry	Cassette	CASSETTE_I O (F000:F859)
16 Points	Keyboard	KEYBOARD_I O (F000:E82E)
17	Printer	PRINTER_I O (F000:EF02)
18	Cassette BASIC	(F600:0000)
19	Bootstrap	BOOT_STRAP (F000:E6F2)
1A	Time of Day	TIME_OF_DAY (F000:FE6E)
1B User Supplied	Keyboard Break	DUMMY_RETURN (F000:FF53)
1C Routines	Timer Tick	DUMMY_RETURN (F000:FF53)
1D BIOS	Video Initialization	VIDEO_PARMS (F000:F0A4)
1E Parameters	Diskette Parameters	DISK_BASE (F000:EFC7)
1F	Video Graphics Chars	None

NOTES

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 mor 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 retrys should be required on reads to ensure the problem is not due to motor start-up.

BIOS Memory Map

STARTING ADDRESS HEX	
00000	BIOS INTERRUPT VECTORS
00080	AVAILABLE INTERRUPT VECTORS
00400	BIOS DATA AREA
00500	USER READ/ WRITE MEMORY
F4000	USER READ ONLY MEMORY
F6000	CASSETTE BASIC INTERPRETER
FE000	BIOS PROGRAM AREA

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.

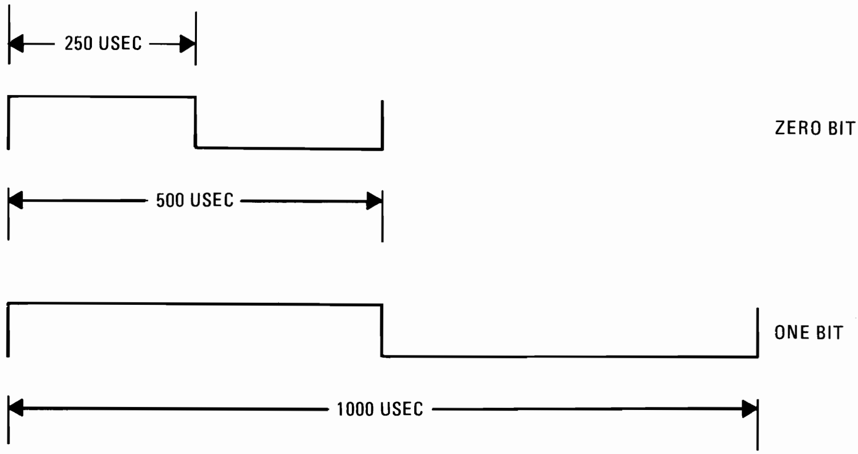
Cassette Write

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

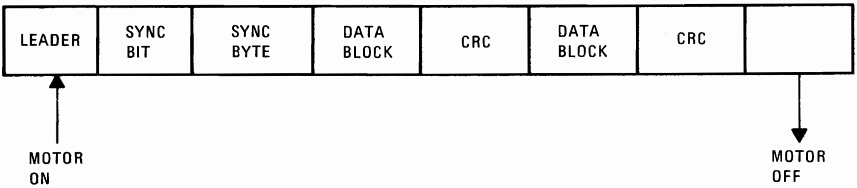
The READ BLOCK routine turns on the cassette motor and then delays for approximately 0.5 secs for it to come up to speed.

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 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:

$G(X) - X^{16}$ « X^{12} « X^{5} X 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

KEY #	BASE CASE	UPPER CASE	CTRL	ALT
1	ESC	ESC	ESC	−1
2	1	½	−1	Note 1
3	2	@	NUL (000) Note 1	Note 1
4	3	#	−1	Note 1
5	4	$	−1	Note 1
6	5	%	−1	Note 1
7	6	∧	RS (030)	Note 1
8	7	&	−1	Note 1
9	8	*	−1	Note 1
10	9	(	−1	Note 1
11	0	)	−1	Note 1
12	−	—	US (031)	Note 1
13	=	+	−1	Note 1
14	Backspace (008)	Backspace (008)	DEL(127)	−1
15	⟶| (009)	|⟵ (Note 1)	−1	−1
16	q	Q	DC1 (017)	Note 1
17	w	W	ETB (023)	Note 1
18	e	E	ENQ (005)	Note 1
19	r	R	DC2 (018)	Note 1
20	t	T	DC4 (020)	Note 1
21	y	Y	EM (025)	Note 1
22	u	U	NAK (021)	Note 1
23	i	I	HT (009)	Note 1
24	o	O	SI (015)	Note 1
25	p	P	DLE (016)	Note 1
26	[	{	ESC (027)	−1
27	]	}	GS (029)	−1

Table 25. Character Codes (cont.)

KEY #	BASE CASE	UPPER CASE	CTRL	ALT
28	CR	CR	LF (010)	–1
29CTRL	–1	–1	–1	–1
30	a	A	SOH (001)	Note 1
31	s	S	DC3 (019)	Note 1
32	d	D	EOT (004)	Note 1
33	f	F	ACK (006)	Note 1
34	g	G	BEL (007)	Note 1
35	h	H	BS (008)	Note 1
36	j	J	LF (010)	Note 1
37	k	K	VT (011)	Note 1
38	l	L	FF (012)	Note 1
39	;	:	–1	–1
40	'	"	–1	–1
41	`	~	–1	–1
42SHIFT	–1	–1	–1	–1
43	\	¦	FS (028)	–1
44	z	Z	SUB (026)	Note 1
45	x	X	CAN (024)	Note 1
46	c	C	ETX (003)	Note 1
47	v	V	SYN (022)	Note 1
48	b	B	STX (002)	Note 1
49	n	N	SO (014)	Note 1
50	m	M	CR (013)	Note 1
51	,	<	–1	–1
52	.	>	–1	–1
53	/	?	–1	–1
54SHIFT	–1	–1	–1	–1
55	*	(Note 2)	(Note 1)	–1
56ALT	–1	–1	–1	–1
57	SP	SP	SP	SP
58CAPS LOCK	–1	–1	–1	–1
59	NUL (Note 1)	NUL (Note 1)	NUL (Note 1)	NUL (Note 1)
60	NUL (Note 1)	NUL (Note 1)	NUL (Note 1)	NUL (Note 1)
61	NUL (Note 1)	NUL (Note 1)	NUL (Note 1)	NUL (Note 1)
62	NUL (Note 1)	NUL (Note 1)	NUL (Note 1)	NUL (Note 1)
63	NUL (Note 1)	NUL (Note 1)	NUL (Note 1)	NUL (Note 1)
64	NUL (Note 1)	NUL (Note 1)	NUL (Note 1)	NUL (Note 1)
65	NUL (Note 1)	NUL (Note 1)	NUL (Note 1)	NUL (Note 1)
66	NUL (Note 1)	NUL (Note 1)	NUL (Note 1)	NUL (Note 1)
67	NUL (Note 1)	NUL (Note 1)	NUL (Note 1)	NUL (Note 1)
68	NUL (Note 1)	NUL (Note 1)	NUL (Note 1)	NUL (Note 1)
69NUM LOCK	–1	–1	Pause (Note 2)	–1
70SCROLL LOCK	–1	–1	Break (Note 2)	–1

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.

KEY #	NUM LOCK	BASE CASE	ALT	CTRL
71	7	Home (Note 1)	Note 1	Clear Screen
72	8	↑ (Note 1)	Note 1	−1
73	9	PageUp (Note 1)	Note 1	Top of Text & Home
74	−	−	−1	−1
75	4	← (Note 1)	Note 1	Reverse Word (Note 1)
76	5	−1	Note 1	−1
77	6	→ (Note 1)	Note 1	Adv Word (Note 1)
78	+	+	−1	−1
79	1	End (Note 1)	Note 1	Erase to EOL (Note 1)
80	2	↓ (Note 1)	Note 1	−1
81	3	PageDown (Note 1)	Note 1	Erase to EOS (Note 1)
82	0	INS	Note 1	−1
83	.	DEL (Notes 1,2)	Note 2	Note 2

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.

ROM

Table 26. **Keyboard Extended Functions**

SECOND CODE	FUNCTION
3	NUL Character
15	←
16-25	ALT Q, W, E, R, T, Y, U, I, O, P
30-38	ALT A, S, D, F, G, H, J, K, L
44-50	ALT Z, X, C, V, B, N, M
59-68	F1-F10 Function Keys Base Case
71	Home
72	↑
73	Page Up & Home Cursor
75	←
77	→
79	End
80	↓
81	Page Down & Home Cursor
82	INS
83	DEL
84-93	F11-F20 (Upper Case F1-F10)
94-103	F21-F30 (CTRL F1-F10)
104-113	F31-F40 (ALT F1-F10)
114	CTRL PRTSC (Start/Stop Echo to Printer) Key 55
115	CTRL ← Reverse Word
116	CTRL → Advance Word
117	CTRL END Erase EOL
118	CTRL PG DN Erase EOS
119	CTRL HOME Clear Screen and home
120-131	ALT 1, 2, 3, 4, 5. 6, 7, 8, 9, 0, -, = (Keys 2-13)
132	CTRL PG UP TOP 25 Lines of Text & Home Cursor

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:

Shift - Temporarily shifts keys 2-13, 15-27, 30-41, 43-53, 55, 59-68 to upper case (lower case if in CAPSLOCK state). Temporarily reverses NUMLOCK/NONUMLOCK state of keys 71-73, 75, 77, 79-83.

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 I.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**

FUNCTION	KEY(S)	COMMENT
Home Cursor	HOME	Editors; word processors
Return to outermost menu	HOME	Menu driven applications
Move cursor up	↑	Full screen editor, word processor
Page up, scroll backwards 25 lines & home	PG UP	Editors; word processors
Move cursor left	← Key 75	Text, command entry
Move cursor right	→	Text, command entry
Scroll to end of text Place cursor at end of line	END	Editors; word processors
Move cursor down	↓	Full screen editor, word processor
Page down, scroll forwards 25 lines & home	PG DN	Editors; word processors
Start/Stop insert text at cursor, shift text right in buffer	INS	Text, command entry
Delete character at cursor	DEL	Text, command entry
Destructive backspace	← Key 14	Text, command entry
Tab forward	⇥	Text entry
Tab reverse	⇤	Text entry
Clear screen and home	CTRL HOME	Command entry
Scroll up	↑	In scroll lock mode
Scroll down	↓	In scroll lock mode
Scroll left	←	In scroll lock mode
Scroll right	→	In scroll lock mode
Delete from cursor to EOL	CTRL END	Text, command entry
Exit/Escape	ESC	Editor, 1 level of menu, etc
Start/Stop Echo screen to printer	PRTSC CTRL K55	Any time
Delete from cursor to EOS	CTRL PG DN	Text, command entry

ROM

Table 27. Keyboard - Commonly Used Functions (cont.)

FUNCTION	KEY(S)	COMMENT
Advance word	CTRL →	Text entry
Reverse word	CTRL ←	Text entry
Window Right	CTRL →	When text is too wide to fit screen
Window Left	CTRL ←	When text is too wide to fit screen
Enter insert mode	INS	Line editor
Exit insert mode	INS	Line editor
Cancel current line	ESC	Command entry, text entry
Suspend system (pause)	CTRL NUMLOCK	Stop list, stop program, etc. Resumes on any key
Break interrupt	CTRL BREAK	Interrupt current process
System reset	ALT CTRL DEL	Reboot
Top of document and home cursor	CTRL PG UP	Editors, word processors
Standard Function Keys	F1–F10	Primary function keys
Secondary function keys	SHIFT F1–F10 CTRL F1–F10 ALT F1–F10	Extra function keys if 10 are not sufficient
Extra function keys	ALT Keys 2–13 (1–9,0,–,=)	Used when stickers are put along top of keyboard
Extra function keys	ALT A–Z	Used when function starts with same letter as one of the alpha keys

Table 28. BASIC Screen Editor Special Functions

FUNCTION	KEY
Carriage return	↵
Line feed	CTRL ↵
Bell	CTRL G
Home	HOME
Cursor up	↑
Cursor down	↓
Cursor left	←
Cursor right	→
Advance one word	CTRL →
Reverse one word	CTRL ←
Insert	INS
Delete	DEL
Clear screen	CTRL HOME
Freeze output	CTRL NUMLOCK
Tab advance	⇥
Stop execution (break)	CTRL BREAK
Delete current line	ESC
Delete to end of line	CTRL END
Position cursor to end of line	END

Table 29. DOS Special Functions

FUNCTION	KEY
Suspend	CTRL NUMLOCK
Echo to printer	CTRL-PRTSC (Key 55 any case)
Stop echo to printer	CTRL-PRTSC (Key 55 any case)
Exit current function (break)	CTRL BREAK
Backspace	← Key 14
Line feed	CTRL ↵
Cancel line	ESC
Copy character	F1 or →
Copy till match	F2
Copy remaining	F3
Skip character	DEL
Skip until match	F4
Enter insert mode	INS
Exit insert mode	INS
Make new line the template	F5
String separator in REPLACE	F6
End of file in keyboard input	F6

NOTES

Low Memory Maps (0-‘0600’x)

Table 30. Interrupt Vectors (0-7F)

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

Notes: (1) Initialized at power up to point to an IRET Instruction.
(2) Initialized at power up to point to tables in ROM.

ROM

Table 31. BASIC and DOS Reserved Interrupts (80-3FF)

ADDRESS HEX	INTERRUPT HEX	FUNCTION
80–83	20	DOS Program Terminate
84–87	21	DOS Function Call
88–8B	22	DOS Terminate Address
8C–8F	23	DOS CTRL–BRK Exit Address
90–93	24	DOS Fatal Error Vector
94–97	25	DOS Absolute Disk read
98–9B	26	DOS Absolute Disk write
9C–9F	27	DOS Terminate, Fix in Storage
A0–FF	28–3F	Reserved for DOS
100–1FF	40–7F	Not Used
200–217	80–85	Reserved By BASIC
218–3C3	86–F0	Used by BASIC Interpreter while BASIC is Running.
3C4–3FF	F1–FF	Not Used

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

ADDRESS HEX	MODE	FUNCTION
400–48F	ROM BIOS	See BIOS Listing
490–4CF	DOS	Used by DOS Mode Command
4D0–4EF		Reserved
4F0–4FF		Reserved as Intra-Application Communication area for any application.
500–5FF		Reserved for DOS and BASIC
500	DOS	Print Screen status flag store. 0—Print screen not active or successful print screen operation. 1—Print screen in progress. 255—Error encountered during print screen operation.
504	DOS	Single drive mode status byte.
510–511	BASIC	BASIC's segment address store.
512–515	BASIC	Clock interrupt vector segment: offset store.
516–519	BASIC	Break key interrupt vector segment: offset store.
51A–51D	BASIC	Disk error interrupt vector segment: offset store.

BASIC Workspace Variables

If you do DEF SEG (Default workspace segment)

	OFFSET	LENGTH
Line number of current line being executed	X '2E'	2
Line number of last error	X '347'	2
Offset into segment of start of program text	X '30'	2
Offset into of start of variables (end of program text 1–1)	X '358'	2
Keyboard buffer contents if 0—no characters in buffer if 1—characters in buffer if you POKE & H6A, 0 you flush any characters in buffer	X '6A'	1

Example:

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

	L	H
100	X '64'	X '00'

NOTES

APPENDICES

Contents:

APPENDIX A
ROM BIOS LISTINGS

CONTENTS

```
LOC OBJ                 LINE   SOURCE

                               $TITLE(ROM BIOS FOR IBM PERSONAL COMPUTER)
                               ;------------------------
                               ; EQUATES
                               ;------------------------
 0060                          PORT_A          EQU     60H             ;8255 PORT A ADDR
 0061                          PORT_B          EQU     61H             ;8255 PORT B ADDR
 0062                          PORT_C          EQU     62H             ;8255 PORT C ADDR
 0063                          CMD_PORT EQU    63H
 0020                          INTA00          EQU     20H             ;8259 PORT
 0021                          INTA01          EQU     21H             ;8259 PORT
 0020                          EOI             EQU     20H
 0040                          TIMER           EQU     40H
 0043                          TIM_CTL EQU     43H             ;8253 TIMER CONTROL PORT ADDR
 0040                          TIMER0  EQU     40H             ;8253 TIMER/CNTER 0 PORT ADDR
 0001                          TMINT   EQU     01              ;TIMER 0 INTR RECVD MASK
 0008                          DMA08   EQU     08              ;DMA STATUS REG PORT ADDR
 0000                          DMA     EQU     00              ;DMA CHANNEL 0 ADDRESS REG PORT ADDR
 0540                          MAX_PERIOD EQU 540H
 0410                          MIN_PERIOD EQU 410H
 0060                          KBD_IN  EQU     60H                     ;KEYBOARD DATA IN ADDR PORT
 0002                          KBDINT  EQU     02                      ;KEYBOARD INTR MASK
 0060                          KB_DATA         EQU     60H     ; KEYBOARD SCAN CODE PORT
 0061                          KB_CTL          EQU     61H     ; CONTROL BITS FOR KEYBOARD SENSE DATA
                               ;-------------------------------------------
                               ; 8088 INTERRUPT LOCATIONS
                               ;-------------------------------------------
0000                           ABS0    SEGMENT AT 0
0000                           STG_LOC0        LABEL   BYTE
0008                                   ORG     2*4
0008                           NMI_PTR         LABEL    WORD
0014                                   ORG     5*4
0014                           INT5_PTR        LABEL   WORD
0020                                   ORG     8*4
0020                           INT_ADDR        LABEL   WORD
0020                           INT_PTR         LABEL   DWORD
0040                                   ORG     10H*4
0040                           VIDEO_INT       LABEL   WORD
0074                                   ORG     1DH*4
0074                           PARM_PTR        LABEL   DWORD           ; POINTER TO VIDEO PARMS
0078                                   ORG     01EH*4                  ; INTERRUPT 1EH
0078                           DISK_POINTER    LABEL   DWORD
007C                                   ORG     01FH*4                  ; LOCATION OF POINTER
007C                           EXT_PTR LABEL   DWORD           ; POINTER TO EXTENSION
7C00                                   ORG     7C00H
7C00                           BOOT_LOCN       LABEL   FAR
                               ABS0    ENDS

                               ;--------------------
                               ; STACK -- USED DURING INITIALIZATION ONLY
                               ;---------------------
                               STACK   SEGMENT AT 30H
0000 (128 ????)                        DW      128 DUP(?)
0100                           TOS     LABEL   WORD
                               STACK   ENDS

                               ;-------------------------------------------
                               ; ROM BIOS DATA AREAS
                               ;-------------------------------------------
0040                           DATA    SEGMENT AT 40H
0000 (4 ????)                  RS232_BASE      DW      4 DUP(?)        ; ADDRESSES OF RS232 ADAPTERS
0008 (4 ????)                  PRINTER_BASE    DW      4 DUP(?)        ; ADDRESSES OF PRINTERS
0010 ????                      EQUIP_FLAG      DW      ?               ; INSTALLED HARDWARE
0012 ??                        MFG_TST         DB      ?               ; INITIALIZATION FLAG
0013 ????                      MEMORY_SIZE     DW      ?               ; MEMORY SIZE IN K BYTES
0015 ????                      IO_RAM_SIZE     DW      ?               ; MEMORY IN I/O CHANNEL
                               ;-------------------------------------------
                               ; KEYBOARD DATA AREAS
                               ;-------------------------------------------
0017 ??                        KB_FLAG         DB      ?

                               ;------ SHIFT FLAG EQUATES WITHIN KB_FLAG

  0080                         INS_STATE       EQU     80H     ; INSERT STATE IS ACTIVE
  0040                         CAPS_STATE      EQU     40H     ; CAPS LOCK STATE HAS BEEN TOGGLED
  0020                         NUM_STATE       EQU     20H     ; NUM LOCK STATE HAS BEEN TOGGLED
  0010                         SCROLL_STATE    EQU     10H     ; SCROLL LOCK STATE HAS BEEN TOGGLED
  0008                         ALT_SHIFT       EQU     08H     ; ALTERNATE SHIFT KEY DEPRESSED
```

```
LOC OBJ                 SOURCE

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

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

 0080                   INS_SHIFT       EQU     80H     ; INSERT KEY IS DEPRESSED
 0040                   CAPS_SHIFT      EQU     40H     ; CAPS LOCK KEY IS DEPRESSED
 0020                   NUM_SHIFT       EQU     20H     ; NUM LOCK KEY IS DEPRESSED
 0010                   SCROLL_SHIFT    EQU     10H     ; SCROLL LOCK KEY IS DEPRESSED
 0008                   HOLD_STATE      EQU     08H     ; SUSPEND KEY HAS BEEN TOGGLED

0019 ??                 ALT_INPUT       DB      ?       ; STORAGE FOR ALTERNATE KEYPAD ENTRY
001A ????               BUFFER_HEAD     DW      ?       ; POINTER TO HEAD OF KEYBOARD BUFFER
001C ????               BUFFER_TAIL     DW      ?       ; POINTER TO TAIL OF KEYBOARD BUFFER
001E (16 ????)          KB_BUFFER       DW      16 DUP(?) ; ROOM FOR 15 ENTRIES
003E                    KB_BUFFER_END   LABEL   WORD

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

 0045                   NUM_KEY         EQU     69      ; SCAN CODE FOR NUMBER LOCK
 0046                   SCROLL_KEY      EQU     70      ; SCROLL LOCK KEY
 0038                   ALT_KEY         EQU     56      ; ALTERNATE SHIFT KEY SCAN CODE
 001D                   CTL_KEY         EQU     29      ; SCAN CODE FOR CONTROL KEY
 003A                   CAPS_KEY        EQU     58      ; SCAN CODE FOR SHIFT LOCK
 002A                   LEFT_KEY        EQU     42      ; SCAN CODE FOR LEFT SHIFT
 0036                   RIGHT_KEY       EQU     54      ; SCAN CODE FOR RIGHT SHIFT
 0052                   INS_KEY         EQU     82      ; SCAN CODE FOR INSERT KEY
 0053                   DEL_KEY         EQU     83      ; SCAN CODE FOR DELETE KEY

                        ;--------------------------------------------
                        ; DISKETTE DATA AREAS
                        ;--------------------------------------------
003E ??                 SEEK_STATUS     DB      ?       ; DRIVE RECALIBRATION STATUS
                        ;                               BIT 3-0 = DRIVE 3-0 NEEDS RECAL BEFORE
                        ;                               NEXT SEEK IF BIT IS = 0
 0080                   INT_FLAG        EQU     080H    ; INTERRUPT OCCURRENCE FLAG
003F ??                 MOTOR_STATUS    DB      ?       ; MOTOR STATUS
                        ;                       BIT 3-0 = DRIVE 3-0 IS CURRENTLY RUNNING
                        ;                       BIT 7   = CURRENT OPERATION IS A WRITE, REQUIRES DELAY
0040 ??                 MOTOR_COUNT     DB      ?       ; TIME OUT COUNTER FOR DRIVE TURN OFF
 0025                   MOTOR_WAIT      EQU     37      ; TWO SECONDS OF COUNTS FOR MOTOR TURN OFF

                        ;
0041 ??                 DISKETTE_STATUS DB      ?       ; SINGLE BYTE OF RETURN CODE INFO FOR STATUS
 0080                   TIME_OUT        EQU     80H     ; ATTACHMENT FAILED TO RESPOND
 0040                   BAD_SEEK        EQU     40H     ; SEEK OPERATION FAILED
 0020                   BAD_NEC         EQU     20H     ; NEC CONTROLLER HAS FAILED
 0010                   BAD_CRC         EQU     10H     ; BAD CRC ON DISKETTE READ
 0009                   DMA_BOUNDARY    EQU     09H     ; ATTEMPT TO DMA ACROSS 64K BOUNDARY
 0008                   BAD_DMA         EQU     08H     ; DMA OVERRUN ON OPERATION
 0004                   RECORD_NOT_FND  EQU     04H     ; REQUESTED SECTOR NOT FOUND
 0003                   WRITE_PROTECT   EQU     03H     ; WRITE ATTEMPTED ON WRITE PROT DISK
 0002                   BAD_ADDR_MARK   EQU     02H     ; ADDRESS MARK NOT FOUND
 0001                   BAD_CMD         EQU     01H     ; BAD COMMAND PASSED TO DISKETTE I/O

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

                        ;--------------------------------------------
                        ; VIDEO DISPLAY DATA AREA
                        ;--------------------------------------------
0049 ??                 CRT_MODE        DB      ?       ; CURRENT CRT MODE
004A ????               CRT_COLS        DW      ?       ; NUMBER OF COLUMNS ON SCREEN
004C ????               CRT_LEN         DW      ?       ; LENGTH OF REGEN IN BYTES
004E ????               CRT_START       DW      ?       ; STARTING ADDRESS IN REGEN BUFFER
0050 (8 ????)           CURSOR_POSN     DW      8 DUP(?)        ; CURSOR FOR EACH OF UP TO 8 PAGES
0060 ????               CURSOR_MODE     DW      ?       ; CURRENT CURSOR MODE SETTING
0062 ??                 ACTIVE_PAGE     DB      ?       ; CURRENT PAGE BEING DISPLAYED
0063 ????               ADDR_6845       DW      ?       ; BASE ADDRESS FOR ACTIVE DISPLAY CARD
0065 ??                 CRT_MODE_SET    DB      ?       ; CURRENT SETTING OF THE 3X8 REGISTER
0066 ??                 CRT_PALLETTE    DB      ?       ; CURRENT PALLETTE SETTING COLOR CARD

                        ;--------------------------------------------
                        ; CASSETTE DATA AREA
                        ;--------------------------------------------
0067 ????               EDGE_CNT        DW      ?               ;TIME COUNT AT DATA EDGE
0069 ????               CRC_REG         DW      ?               ;CRC REGISTER
```

```
LOC OBJ                  LINE    SOURCE

006B ??                          LAST_VAL        DB      ?               ;LAST INPUT VALUE

                                 ;------------------------------------------
                                 ; TIMER DATA AREA
                                 ;------------------------------------------
006C ????                        TIMER_LOW       DW      ?       ; LOW WORD OF TIMER COUNT
006E ????                        TIMER_HIGH      DW      ?       ; HIGH WORD OF TIMER COUNT
0070 ??                          TIMER_OFL       DB      ?       ; TIMER HAS ROLLED OVER SINCE LAST READ
                                 ;COUNTS_SEC     EQU     18
                                 ;COUNTS_MIN     EQU     1092
                                 ;COUNTS_HOUR    EQU     65543
                                 ;COUNTS_DAY     EQU     1573040 = 1800B0H

                                 ;------------------------------------------
                                 ; SYSTEM DATA AREA
                                 ;------------------------------------------
0071 ??                          BIOS_BREAK      DB      ?       ; BIT 7 = 1 IF BREAK KEY HAS BEEN DEPRESSED
0072 ????                        RESET_FLAG      DW      ?       ; WORD = 1234H IF KEYBOARD RESET UNDERWAY
                                 DATA    ENDS

                                 ;------------------------------------------
                                 ; EXTRA DATA AREA
                                 ;------------------------------------------
0050                             XXDATA  SEGMENT AT 50H
0000 ??                          STATUS_BYTE     DB      ?
                                 XXDATA  ENDS

                                 ;------------------------------------------
                                 ; VIDEO DISPLAY BUFFER
                                 ;------------------------------------------
B800                             VIDEO_RAM       SEGMENT AT 0B800H
0000                             REGEN   LABEL   BYTE
0000                             REGENW  LABEL   WORD
0000 (16384 ??)                          DB      16384 DUP(?)
                                 VIDEO_RAM       ENDS
                                 ;------------------------------------------
                                 ; ROM RESIDENT CODE
                                 ;------------------------------------------
F000                             CODE    SEGMENT AT 0F000H
0000 (57344 ??)                          DB      57344 DUP(?)    ; FILL LOWEST 56K

E000 35373030303531                      DB      '5700051 COPR. IBM 1981'       ; COPYRIGHT NOTICE
     20434F50522E20
     49424D20313938
     31

                                 ;------------------------------------------
                                 ; INITIAL RELIABILITY TESTS -- PHASE 1
                                 ;------------------------------------------
                                         ASSUME    CS:CODE,SS:CODE,ES:ABS0,DS:DATA
                                 ;------------------------------------------
                                 ;       DATA DEFINITIONS
                                 ;------------------------------------------
E016 D8E0             R          C1      DW      C11             ; RETURN ADDRESS
E018 EDE1             R          C2      DW      C24             ; RETURN ADDRESS FOR DUMMY STACK
                                 ;------------------------------------------
                                 ;       THIS SUBROUTINE PERFORMS A READ/WRITE STORAGE TEST ON A 16K BLOCK
                                 ;       OF STORAGE.
                                 ;ENTRY REQUIREMENTS:
                                 ;       ES = ADDRESS OF STORAGE SEGMENT BEING TESTED
                                 ;       DS = ADDRESS OF STORAGE SEGMENT BEING TESTED
                                 ;       WHEN ENTERING AT STGTST_CNT, CX MUST BE LOADED WITH THE BYTE COUNT.
                                 ;EXIT PARAMETERS:
                                 ;       ZERO FLAG = 0 IF STORAGE ERROR (DATA COMPARE OR PARITY CHECK.  AL=0
                                 ;               DENOTES A PARITY CHECK. ELSE AL=XOR'ED BIT PATTERN OF THE
                                 ;               EXPECTED DATA PATTERN VS THE ACTUAL DATA READ.
                                 ;       AX,BX,CX,DX,DI, AND SI ARE ALL DESTROYED.
                                 ;------------------------------------------
E01A                             STGTST          PROC    NEAR
E01A B90040                              MOV     CX,4000H                ;SETUP CNT TO TEST A 16K BLK
E01D                             STGTST_CNT:
E01D FC                                  CLD                             ;SET DIR FLAG TO INCREMENT
E01E 8BD9                                MOV     BX,CX                   ;SAVE BYTE CNT (4K FOR VIDEO OR 16K)
E020 B8FFFF                              MOV     AX,0FFFFH               ;GET DATA PATTERN TO WRITE
E023 BA55AA                              MOV     DX,0AA55H               ;SETUP OTHER DATA PATTERNS TO USE
E026 2BFF                                SUB     DI,DI                   ;DI = OFFSET 0 RELATIVE TO ES REG
E028 F3                                  REP     STOSB                   ;WRITE STORAGE LOCATIONS
E029 AA
```

```
LOC OBJ              SOURCE

E02A                 C3:                              ; STG01
E02A 4F                      DEC     DI               ;POINT TO LAST BYTE JUST WRITTEN
E02B FD                      STD                      ;SET DIR FLAG TO GO BACKWARDS
E02C 8BF7            C4:     MOV     SI,DI
E02E 8BCB                    MOV     CX,BX            ;SETUP BYTE CNT
E030 AC              C5:     LODSB                    ;READ CHAR FROM STORAGE
E031 32C4                    XOR     AL,AH            ;DATA READ AS EXPECTED?
E033 7525                    JNE     C7               ;NO - GO TO ERROR ROUTINE
E035 E462                    IN      AL,PORT_C        ;DID A PARITY ERROR OCCUR?
E037 24C0                    AND     AL,0C0H
E039 B000                    MOV     AL,0             ;AL=0 DATA COMPARE OK
E03B 751D                    JNZ     C7
E03D 80FC00                  CMP     AH,0             ;READING ZERO PATTERN?
E040 7403                    JE      C6               ;CONTINUE READING TILL END
E042 8AC2                    MOV     AL,DL            ;GET NEXT DATA PATTERN TO WRITE
E044 AA                      STOSB                    ;WRITE IN BYTE LOC WE JUST READ
E045                 C6:                              ; WRITE_NO_MORE
E045 E2E9                    LOOP    C5               ;CONTINUE TILL 16K/4K BLOCK TESTED
E047 80FC00                  CMP     AH,0             ;ZERO PATTERN WRITTEN TO STG
E04A 740E                    JE      C7               ;YES - RETURN TO CALLER
E04C 8AE0                    MOV     AH,AL            ;SETUP TO NEW VALUE TO COMPARE
E04E 86F2                    XCHG    DH,DL            ;MOVE ZERO DATA PATTERN TO DL
E050 FC                      CLD                      ;SET DIR FLAG TO GO FORWARD
E051 47                      INC     DI               ;SET POINTER TO BEG LOCATION
E052 74D8                    JZ      C4               ;READ/WRITE FORWARD IN STG
E054 4F                      DEC     DI
E055 BA0100                  MOV     DX,1             ;SETUP 01 AND 00 PATTERNS
E058 EBD0                    JMP     SHORT C3         ;READ/WRITE BACKWARD IN STG
E05A                 C7:
E05A C3                      RET
                     STGTST  ENDP
                     ;--------------------------------------------
                     ;TEST.01
                     ;       8088 PROCESSOR TEST
                     ;DESCRIPTION
                     ;       VERIFY 8088 FLAGS, REGISTERS AND CONDITIONAL JUMPS
                     ;--------------------------------------------
E05B                 RESET   LABEL   NEAR
E05B FA              START:  CLI                      ;DISABLE INTERRUPTS
E05C B4D5                    MOV     AH,0D5H          ;SET SF, CF, ZF, AND AF FLAGS ON
E05E 9E                      SAHF
E05F 734E                    JNC     ERR01            ;GO TO ERR ROUTINE IF CF NOT SET
E061 754C                    JNZ     ERR01            ;GO TO ERR ROUTINE IF ZF NOT SET
E063 7B4A                    JNP     ERR01            ;GO TO ERR ROUTINE IF PF NOT SET
E065 7948                    JNS     ERR01            ;GO TO ERR ROUTINE IF SF NOT SET
E067 9F                      LAHF                     ;LOAD FLAG IMAGE TO AH
E068 B105                    MOV     CL,5             ;LOAD CNT REG WITH SHIFT CNT
E06A D2EC                    SHR     AH,CL            ;SHIFT AF INTO CARRY BIT POS
E06C 7341                    JNC     ERR01            ;GO TO ERR ROUTINE IF AF NOT SET
E06E B040                    MOV     AL,40H           ;SET THE OF FLAG ON
E070 D0E0                    SHL     AL,1             ;SETUP FOR TESTING
E072 713B                    JNO     ERR01            ;GO TO ERR ROUTINE IF OF NOT SET
E074 32E4                    XOR     AH,AH            ;SET AH = 0
E076 9E                      SAHF                     ;CLEAR SF, CF, ZF, AND PF
E077 7236                    JC      ERR01            ;GO TO ERR ROUTINE IF CF ON
E079 7434                    JZ      ERR01            ;GO TO ERR ROUTINE IF ZF ON
E07B 7832                    JS      ERR01            ;GO TO ERR ROUTINE IF SF ON
E07D 7A30                    JP      ERR01            ;GO TO ERR ROUTINE IF PF ON
E07F 9F                      LAHF                     ;LOAD FLAG IMAGE TO AH
E080 B105                    MOV     CL,5             ;LOAD CNT REG WITH SHIFT CNT
E082 D2EC                    SHR     AH,CL            ;SHIFT  AF' INTO CARRY BIT POS
E084 7229                    JC      ERR01            ;GO TO ERR ROUTINE IF ON
E086 D0E4                    SHL     AH,1             ;CHECK THAT  OF' IS CLEAR
E088 7025                    JO      ERR01            ;GO TO ERR ROUTINE IF ON

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

E08A B8FFFF                  MOV    AX,0FFFFH                   ;SETUP ONE'S PATTERN IN AX
E08D F9                      STC
E08E 8ED8            C8:     MOV    DS,AX                       ;WRITE PATTERN TO ALL REGS
E090 8CDB                    MOV    BX,DS
E092 8EC3                    MOV    ES,BX
E094 8CC1                    MOV    CX,ES
E096 8ED1                    MOV    SS,CX
E098 8CD2                    MOV    DX,SS
E09A 8BE2                    MOV    SP,DX
E09C 8BEC                    MOV    BP,SP
```

```
LOC OBJ                SOURCE

E09E 8BF5                      MOV     SI,BP
E0A0 8BFE                      MOV     DI,SI
E0A2 7307                      JNC     C9                          ; TST1A
E0A4 33C7                      XOR     AX,DI                       ;PATTERN MAKE IT THRU ALL REGS
E0A6 7507                      JNZ     ERR01                       ;NO - GO TO ERR ROUTINE
E0A8 F8                        CLC
E0A9 73E3                      JNC     C8
E0AB                   C9:                                         ; TST1A
E0AB 0BC7                      OR      AX,DI                       ;ZERO PATTERN MAKE IT THRU?
E0AD 7401                      JZ      C10                         ;YES - GO TO NEXT TEST
E0AF F4                ERR01:  HLT                                 ;HALT SYSTEM
                       ;-------------------------------------------
                       ;TEST.02
                       ;       ROS CHECKSUM TEST I
                       ;DESCRIPTION
                       ;       A CHECKSUM IS DONE FOR THE 8K ROS MODULE CONTAINING POD AND BIOS.
                       ;-------------------------------------------
E0B0                   C10:
E0B0 B000                      MOV     AL,0                        ;DISABLE NMI INTERRUPTS
E0B2 E6A0                      OUT     0A0H,AL
E0B4 E683                      OUT     83H,AL                      ;INITIALZE DMA PAGE REG
E0B6 B099                      MOV     AL,99H                      ;SET 8255 A,C-INPUT,B-OUTPUT
E0B8 E663                      OUT     CMD_PORT,AL                 ;WRITE 8255 CMD/MODE REG
E0BA B0FC                      MOV     AL,0FCH                     ;DISABLE PARITY CHECKERS AND
E0BC E661                      OUT     PORT_B,AL                   ; GATE SNS SWS,CASS MOTOR OFF
E0BE 2AC0                      SUB     AL,AL                       ;
E0C0 BAD803                    MOV     DX,3D8H
E0C3 EE                        OUT     DX,AL                       ;DISABLE COLOR VIDEO
E0C4 FEC0                      INC     AL
E0C6 BAB803                    MOV     DX,3B8H
E0C9 EE                        OUT     DX,AL                       ;DISABLE B/W VIDEO,EN HIGH RES
E0CA B800F0      R             MOV     AX,CODE                     ;SETUP SS SEG REG
E0CD 8ED0                      MOV     SS,AX
E0CF BB00E0                    MOV     BX,0E000H                   ;SETUP STARTING ROS ADDR
E0D2 BC16E0      R             MOV     SP,OFFSET C1                ;SETUP RETURN ADDRESS
E0D5 E93301                    JMP     ROS_CHECKSUM
E0D8 7505              C11:    JNE     ERR01                       ;HALT SYSTEM IF ERROR
                       ;-------------------------------------------
                       ;TEST.03
                       ;       8237 DMA INITIALIZATION CHANNEL REGISTER TEST
                       ;DESCRIPTION
                       ;       DISABLE THE 8237 DMA CONTROLLER.  VERIFY THAT TIMER 1 FUNCTIONS OK.
                       ;       WRITE/READ THE CURRENT ADDRESS AND WORD COUNT REGISTERS FOR ALL
                       ;       CHANNELS.  INITIALIZE AND START DMA FOR MEMORY REFRESH.
                       ;-------------------------------------------
                       ;       DISABLE DMA CONTROLLER

E0DA B004                      MOV     AL,04                       ;DISABLE DMA CONTROLLER
E0DC E608                      OUT     DMA08,AL

                       ;       VERIFY THAT TIMER 1 FUNCTIONS OK

E0DE B054                      MOV     AL,54H                      ;SEL TIMER 1,LSB,MODE 2
E0E0 E643                      OUT     TIMER+3,AL
E0E2 2BC9                      SUB     CX,CX                       ;
E0E4 8AD9                      MOV     BL,CL
E0E6 8AC1                      MOV     AL,CL                       ;SET INITIAL TIMER CNT TO 0
E0E8 E641                      OUT     TIMER+1,AL
E0EA                   C12:                                        ; TIMER1_BITS_ON
E0EA B040                      MOV     AL,40H                      ;LATCH TIMER 1 COUNT
E0EC E643                      OUT     TIMER+3,AL
E0EE E441                      IN      AL,TIMER+1                  ;READ TIMER 1 COUNT
E0F0 0AD8                      OR      BL,AL                       ;ALL BITS ON IN TIMER
E0F2 80FBFF                    CMP     BL,0FFH                     ;YES - SEE IF ALL BITS GO OFF
E0F5 7404                      JE      C13                         ; TIMER1_BITS_OFF
E0F7 E2F1                      LOOP    C12                         ; TIMER1_BITS_ON
E0F9 EBB4                      JMP     SHORT ERR01                 ;TIMER 1 FAILURE, HALT SYS
E0FB                   C13:                                        ; TIMER1_BITS_OFF
E0FB 8AC3                      MOV     AL,BL                       ;SET TIMER 1 CNT
E0FD 2BC9                      SUB     CX,CX
E0FF E641                      OUT     TIMER+1,AL
E101                   C14:                                        ; TIMER_LOOP
E101 B040                      MOV     AL,40H                      ;LATCH TIMER 1 COUNT
E103 E643                      OUT     TIMER+3,AL
E105 E441                      IN      AL,TIMER+1                  ;READ TIMER 1 COUNT
E107 22D8                      AND     BL,AL
E109 7404                      JZ      C15                         ; WRAP_DMA_REG
E10B E2F4                      LOOP    C14                         ; TIMER_LOOP
```

```
LOC OBJ              LINE    SOURCE

E10D EBA0                            JMP     SHORT ERR01

                             ;       INITIALIZE TIMER 1 TO REFRESH MEMORY

E10F                         C15:                                    ; WRAP_DMA_REG
E10F B054                            MOV     AL,54H                  ;SEL TIM 1, LSB, MODE 2
E111 E643                            OUT     TIMER+3,AL              ;WRITE TIMER MODE REG
E113 B012                            MOV     AL,18                   ;SETUP DIVISOR FOR REFRESH
E115 E641                            OUT     TIMER+1,AL              ;WRITE TIMER 1 CNT REG
E117 E60D                            OUT     DMA+0DH,AL              ;SEND MASTER CLEAR TO DMA

                             ;       WRAP DMA CHANNELS ADDRESS AND COUNT REGISTERS

E119 B0FF                            MOV     AL,0FFH                 ;WRITE PATTERN FFH TO ALL REGS
E11B 8AD8                    C16:    MOV     BL,AL                   ;SAVE PATTERN FOR COMPARE
E11D 8AF8                            MOV     BH,AL
E11F B90800                          MOV     CX,8                    ;SETUP LOOP CNT
E122 BA0000                          MOV     DX,DMA                  ;SETUP I/O PORT ADDR OF REG
E125 EE                      C17:    OUT     DX,AL                   ;WRITE PATTERN TO REG, LSB
E126 EE                              OUT     DX,AL                   ;MSB OF 16 BIT REG
E127 B80101                          MOV     AX,0101H                ;AX TO ANOTHER PAT BEFORE RD
E12A EC                              IN      AL,DX                   ;READ 16-BIT DMA CH REG, LSB
E12B 8AE0                            MOV     AH,AL                   ;SAVE LSB OF 16-BIT REG
E12D EC                              IN      AL,DX                   ;READ MSB OF DMA CH REG
E12E 3BD8                            CMP     BX,AX                   ;PATTERN READ AS WRITTEN?
E130 7403                            JE      C18                     ;YES - CHECK NEXT REG
E132 E97AFF                          JMP     ERR01                   ;NO - HALT THE SYSTEM
E135                         C18:                                    ; NXT_DMA_CH
E135 42                              INC     DX                      ;SET I/O PORT TO NEXT CH REG
E136 E2ED                            LOOP    C17                     ;WRITE PATTERN TO NEXT REG
E138 F6D0                            NOT     AL                      ;SET PATTERN TO ZERO
E13A 74DF                            JZ      C16                     ;WRITE TO CHANNEL REGS

                             ;       INITIALIZE AND START DMA FOR MEMORY REFRESH.

E13C B0FF                            MOV     AL,0FFH                 ;SET CNT OF 64K FOR RAM REFRESH
E13E E601                            OUT     DMA+1,AL
E140 E601                            OUT     DMA+1,AL
E142 B058                            MOV     AL,058H                 ;SET DMA MODE,CH 0,READ,AUOTINT
E144 E60B                            OUT     DMA+0BH,AL              ;WRITE DMA MODE REG
E146 B000                            MOV     AL,0                    ;ENABLE DMA CONTROLLER
E148 E608                            OUT     DMA+8,AL                ;SETUP DMA COMMAND REG
E14A E60A                            OUT     DMA+10,AL               ;ENABLE CHANNEL 0 FOR REFRESH
E14C B041                            MOV     AL,41H                  ;SET MODE FOR CHANNEL 1
E14E E60B                            OUT     DMA+0BH,AL
E150 B042                            MOV     AL,42H                  ;SET MODE FOR CHANNEL 2
E152 E60B                            OUT     DMA+0BH,AL              ;
E154 B043                            MOV     AL,43H                  ;SET MODE FOR CHANNEL 3
E156 E60B                            OUT     DMA+0BH,AL
                             ;-------------------------------------------
                             ;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 ADDRESSABILITY.
                             ;       INITIALIZE THE 8259 INTERRUPT CONTROLLER CHIP FOR CHECKING
                             ;       MANUFACTURING TEST 2 MODE.
                             ;-------------------------------------------
                             ;       DETERMINE MEMORY SIZE AND FILL MEMORY WITH DATA

E158 B84000          R               MOV     AX,DATA                 ;POINT DS TO DATA SEG
E15B 8ED8                            MOV     DS,AX                   ;
E15D 8B1E7200        R               MOV     BX,RESET_FLAG           ;SAVE  RESET_FLAG' IN BX
E161 2BC0                            SUB     AX,AX                   ;SET ES AND DS TO 0
E163 8EC0                            MOV     ES,AX                   ;SETUP ES SEGMENT REG
E165 8ED8                            MOV     DS,AX
E167 2BFF                            SUB     DI,DI
E169 E460                            IN      AL,PORT_A               ;DETERMINE BASE RAM SIZE
E16B 240C                            AND     AL,0CH                  ;ISOLATE RAM SIZE SWS
E16D 0404                            ADD     AL, 4                   ; CALCULATE MEMORY SIZE
E16F B10C                            MOV     CL, 12
E171 D3E0                            SHL     AX, CL
E173 8BC8                            MOV     CX, AX
E175 8AE0                            MOV     AH, AL
E177 FC                              CLD                             ;SET DIR FLAG TO INCR
E178 AA                      C19:    STOSB                           ;FILL BASE RAM WITH DATA
```

```
LOC OBJ              LINE    SOURCE

E179 E2FD             462          LOOP    C19                      ; LOOP TIL ALL ZERO
                      463
                      464    ;     DETERMINE IO CHANNEL RAM SIZE
                      465
E17B E462             466          IN      AL,PORT_C
E17D 240F             467          AND     AL,0FH
E17F 7418             468          JZ      C21
E181 BA0010           469          MOV     DX,1000H                 ; SEGMENT FOR I/O RAM
E184 8AE0             470          MOV     AH,AL
E186 B000             471          MOV     AL,0
E188                  472    C20:                                   ; FILL_IO:
E188 8EC2             473          MOV     ES,DX
E18A B90080           474          MOV     CX,8000H                 ; FILL 32K BYTES
E18D 2BFF             475          SUB     DI,DI
E18F F3               476          REP     STOSB
E190 AA
E191 81C20008         477          ADD     DX,800H                  ; NEXT SEGMENT VALUE
E195 FECC             478          DEC     AH
E197 75EF             479          JNZ     C20                      ; FILL_IO
                      480    ;------------------------------------------
                      481    ;     INITIALIZE THE 8259 INTERRUPT CONTROLLER CHIP
                      482    ;------------------------------------------
E199                  483    C21:
E199 B013             484          MOV     AL,13H                   ;ICW1 - EDGE, SNGL, ICW4
E19B E620             485          OUT     INTA00,AL
E19D B008             486          MOV     AL,8                     ;SETUP ICW2 - INT TYPE 8 (8-F)
E19F E621             487          OUT     INTA01,AL
E1A1 B009             488          MOV     AL,9                     ;SETUP ICW4 - BUFFRD,8086 MODE
E1A3 E621             489          OUT     INTA01,AL
E1A5 2BC0             490          SUB     AX,AX                    ;POINT DS AND ES TO BEGIN
E1A7 8EC0             491          MOV     ES,AX                    ;  OF R/W STORAGE
E1A9 BE4000      R    492          MOV     SI,DATA                  ;POINT DS TO DATA SEG
E1AC 8EDE             493          MOV     DS,SI                    ;
E1AE 891E7200    R    494          MOV     RESET_FLAG,BX            ;RESTORE RESET_FLAG
E1B2 813E72003412 R   495          CMP     RESET_FLAG,1234H         ;RESET_FLAG SET?
E1B8 7438             496          JE      C25                      ;YES - SKIP STG TEST
E1BA 8ED8             497          MOV     DS,AX                    ;POINT DS TO 1ST 16K OF STG
                      498    ;------------------------------------------
                      499    ;     CHECK FOR MANUFACTURING TEST 2 TO LOAD TEST PROGRAMS FROM KEYBOARD.
                      500    ;------------------------------------------
E1BC BCF03F           501          MOV     SP, 3FF0H                ; ESTABLISH TEMPORARY STACK
E1BF 8ED0             502          MOV     SS, AX
E1C1 8BF8             503          MOV     DI, AX
E1C3 BB2400           504          MOV     BX, 24H
E1C6 C707B6E2    R    505          MOV     WORD PTR [BX],OFFSET D11 ;SET UP KB INTERRUPT
E1CA 43               506          INC     BX
E1CB 43               507          INC     BX
E1CC 8C0F             508          MOV     [BX],CS
E1CE E8B704           509          CALL    KBD_RESET        ; READ IN KB RESET CODE TO BL
E1D1 80FB65           510          CMP     BL,065H          ; IS THIS MANUFACTURING TEST 2?
E1D4 750E             511          JNZ     C23              ; JUMP IF NOT MAN. TEST
E1D6 B2FF             512          MOV     DL,255           ; READ IN TEST PROGRAM
E1D8 E8BA04           513    C22:  CALL    SP_TEST
E1DB 8AC3             514          MOV     AL,BL
E1DD AA               515          STOSB
E1DE FECA             516          DEC     DL
E1E0 75F6             517          JNZ     C22              ; JUMP IF NOT DONE YET
E1E2 CD3E             518          INT     3EH              ;SET INTERUPT TYPE 62 ADDRESS F8H
E1E4                  519    C23:                           ;CONTINUE IN NORMAL MODE
E1E4 0E               520          PUSH    CS               ; PUT SS BACK
E1E5 17               521          POP     SS
E1E6 FA               522          CLI                      ;
E1E7 BC18E0      R    523          MOV     SP,OFFSET C2             ;SETUP RETURN ADDRESS
E1EA E920FE           524          JMP     STGTST                   ;GO TO RD/WRT STG SUBROUTINE
E1ED 7403             525    C24:  JE      C25                      ;GO TO NEXT TEST IF OK
E1EF E9BDFE           526          JMP     ERR01
                      527
                      528    ;     SETUP STACK SEG AND SP
                      529
E1F2                  530    C25:
E1F2 B83000           531          MOV     AX,STACK                 ; GET STACK VALUE
E1F5 8ED0             532          MOV     SS,AX                    ; SET THE STACK UP
E1F7 BC0001           533          MOV     SP,OFFSET TOS            ; STACK IS READY TO GO
                      534
                      535    ;     SETUP THE NMI INTERRUPT VECTOR POINTER
                      536
```

```
LOC OBJ                  LINE    SOURCE

E1FA 26C7060800C3E2 R     537            MOV     ES:NMI_PTR,OFFSET NMI_INT
E201 26C7060A0000F0 R     538            MOV     ES:NMI_PTR+2,CODE
E208 E92A00               539            JMP     TST6                         ;GO TO NEXT TEST
                          540
E20B                      541    ROS_CHECKSUM    PROC    NEAR                 ; NEXT_ROS_MODULE
E20B B90020               542            MOV     CX,8192                      ;NUMBER OF BYTES TO ADD
E20E 32C0                 543            XOR     AL,AL
E210                      544    C26:
E210 2E0207               545            ADD     AL,CS:[BX]                   ;
E213 43                   546            INC     BX                           ;POINT TO NEXT BYTE
E214 E2FA                 547            LOOP    C26                          ;ADD ALL BYTES IN ROS MODULE
E216 0AC0                 548            OR      AL,AL                        ;SUM = 0?
E218 C3                   549            RET
                          550    ROS_CHECKSUM    ENDP
                          551    ;--------------------------------------
                          552    ; INITIAL RELIABILITY TEST -- PHASE 2
                          553    ;--------------------------------------
                          554            ASSUME  CS:CODE,ES:ABS0
                          555
E219 50415249545920       556    D1      DB      'PARITY CHECK 2'
     434845434B2032
  000E                    557    D1L  EQU        $-D1
E227 50415249545920       558    D2      DB      'PARITY CHECK 1'
     434845434B2031
  000E                    559    D2L     EQU     $-D2
                          560    ;------------------------------------------
                          561    ;TEST.06
                          562    ;       8259 INTERRUPT CONTROLLER TEST
                          563    ;DESCRIPTION
                          564    ;       READ/WRITE THE INTERRUPT MASK REGISTER (IMR) WITH ALL ONES AND ZEROES.
                          565    ;       ENABLE SYSTEM INTERRUPTS.  MASK DEVICE INTERRUPTS OFF.  CHECK FOR
                          566    ;       HOT INTERRUPTS (UNEXPECTED).
                          567    ;------------------------------------------
E235                      568    TST6:
E235 2BC0                 569            SUB     AX,AX                        ;SET UP ES REG
E237 8EC0                 570            MOV     ES,AX
                          571
                          572    ;------ SET UP THE INTERRUPT 5 POINTER TO A DUMMY
                          573
E239 26C706140054FF R     574            MOV     ES:INT5_PTR,OFFSET PRINT_SCREEN         ;PRINT SCREEN
E240 26C706160000F0 R     575            MOV     ES:INT5_PTR+2,CODE                      ;
                          576
                          577    ;       TEST THE IMR REGISTER
                          578
E247 FA                   579            CLI                                  ;DIABLE INTERRUPTS
E248 B000                 580            MOV     AL,0                         ;SET IMR TO ZERO
E24A E621                 581            OUT     INTA01,AL
E24C E421                 582            IN      AL,INTA01                    ;READ IMR
E24E 0AC0                 583            OR      AL,AL                        ;IMR = 0?
E250 752B                 584            JNZ     D6                           ;GO TO ERR ROUTINE IF NOT 0
E252 B0FF                 585            MOV     AL,0FFH                      ;DISABLE DEVICE INTERRUPTS
E254 E621                 586            OUT     INTA01,AL                    ;WRITE TO IMR
E256 E421                 587            IN      AL,INTA01                    ;READ IMR
E258 0401                 588            ADD     AL,1                         ;ALL IMR BIT ON?
E25A 7521                 589            JNZ     D6                           ;NO - GO TO ERR ROUTINE
                          590
                          591    ;       CHECK FOR HOT INTERRUPTS
                          592
E25C FC                   593            CLD                                  ;SET DIR FLAG TO GO FORWARD
E25D B90800               594            MOV     CX,8                         ;SETUP TEMP INT RTNE IN PRT TBL
E260 BF2000               595            MOV     DI,OFFSET INT_PTR            ;GET ADDRESS OF INT PROC TABLE
E263                      596    D3:                                          ; VECTBL0:
E263 B8B6E2          R    597            MOV     AX,OFFSET D11                ;MOVE ADDR OF INTR PROC TO TBL
E266 AB                   598            STOSW
E267 B800F0          R    599            MOV     AX,CODE                      ;GET ADDR OF INTR PROC SEG
E26A AB                   600            STOSW
E26B 83C304               601            ADD     BX,4                         ;SET BX TO POINT TO NEXT VAL
E26E E2F3                 602            LOOP    D3                           ; VECTBL0
                          603
                          604    ;       INTERRUPTS ARE MASKED OFF.  CHECK THAT NO INTERRUPTS OCCUR.
                          605
E270 32E4                 606            XOR     AH,AH                        ;CLEAR AH REG
E272 FB                   607            STI                                  ;ENABLE EXTERNAL INTERRUPTS
E273 2BC9                 608            SUB     CX,CX                        ;WAIT 1 SEC FOR ANY INTRS THAT
E275 E2FE                 609    D4:     LOOP    D4                           ;MIGHT OCCUR
E277 E2FE                 610    D5:     LOOP    D5
E279 0AE4                 611            OR      AH,AH                        ;DID ANY INTERRUPTS OCCUR?
```

```
LOC OBJ              LINE    SOURCE

E27B 7408                            JZ      D7                        ;NO - GO TO NEXT TEST
E27D BA0101                  D6:     MOV     DX,101H                   ;BEEP SPEAKER IF ERROR
E280 E8AD03                          CALL    ERR_BEEP                  ;GO TO BEEP SUBROUTINE
E283 FA                              CLI
E284 F4                              HLT                               ;HALT THE SYSTEM
                             ;-------------------------------------------
                             ;TEST.7
                             ;       8253 TIMER CHECKOUT
                             ;DESCRIPTION
                             ;       VERIFY THAT THE SYSTEM TIMER (0) DOESN'T COUNT TOO FAST NOR TOO
                             ;       SLOW.
                             ;-------------------------------------------
E285                         D7:
E285 B400                            MOV     AH,0              ;RESET TIMER INTR RECVD FLAG
E287 32ED                            XOR     CH,CH             ;CLEAR THE CH REG
E289 B0FE                            MOV     AL,0FEH           ;MASK ALL INTRS EXCEPT LVL 0
E28B E621                            OUT     INTA01,AL         ;WRITE THE 8259 IMR
E28D B010                            MOV     AL,00010000B      ;SEL TIM 0, LSB, MODE 0, BINARY
E28F E643                            OUT     TIM_CTL,AL        ;WRITE TIMER CONTROL MODE REG
E291 B116                            MOV     CL,16H            ;SET PGM LOOP CNT
E293 8AC1                            MOV     AL,CL             ;SET TIMER 0 CNT REG
E295 E640                            OUT     TIMER0,AL         ;WRITE TIMER 0 CNT REG
E297 F6C4FF                  D8:     TEST    AH,0FFH           ;DID TIMER 0 INTERRUPT OCCUR?
E29A 7504                            JNZ     D9                ; YES - CHECK TIMER OP FOR SLOW TIME
E29C E2F9                            LOOP    D8                ;WAIT FOR INTR FOR SPECIFIED TIME
E29E EBDD                            JMP     D6                ;TIMER 0 INTR DIDN T OCCUR - ERR
E2A0 B112                    D9:     MOV     CL,18             ;SET PGM LOOP CNT
E2A2 B0FF                            MOV     AL,0FFH           ;WRITE TIMER 0 CNT REG
E2A4 E640                            OUT     TIMER0,AL
E2A6 B400                            MOV     AH,0              ;RESET INTR RECEIVED FLAG
E2A8 B0FE                            MOV     AL,0FEH           ;REENABLE TIMER 0 INTERRUTS
E2AA E621                            OUT     INTA01,AL
E2AC F6C4FF                  D10:    TEST    AH,0FFH           ;DID TIMER 0 INTERRUPT OCCUR?
E2AF 75CC                            JNZ     D6                ;YES - TIMER CNTING TOO FAST, ERR
E2B1 E2F9                            LOOP    D10               ;WAIT FOR INTR FOR SPECIFIED TIME
E2B3 E93600                          JMP     TST8              ;GO TO NEXT TEST ROUTINE
                             ;-------------------------------------------
                             ;       TEMPORARY INTERRUPT SERVICE ROUTINE
                             ;-------------------------------------------
E2B6                         D11     PROC    NEAR
E2B6 B401                            MOV     AH,1
E2B8 50                              PUSH    AX                        ;SAVE REG AX CONTENTS
E2B9 B0FF                            MOV     AL,0FFH                   ;MASK ALL INTERRUPTS OFF
E2BB E621                            OUT     INTA01,AL
E2BD B020                            MOV     AL,EOI
E2BF E620                            OUT     INTA00,AL
E2C1 58                              POP     AX                        ;RESTORE REG AX CONTENTS
E2C2 CF                              IRET
                             D11     ENDP

E2C3                         NMI_INT PROC    NEAR
E2C3 50                              PUSH    AX                        ;SAVE ORIG CONTENTS OF AX
E2C4 E462                            IN      AL,PORT_C
E2C6 A840                            TEST    AL,40H                    ;IO CH PARITY CHECK?
E2C8 7408                            JZ      D12                       ;YES - FLAG IS SET TO 0
E2CA BE19E2          R               MOV     SI,OFFSET D1              ;ADDR OF ERROR MSG
E2CD B90E00                          MOV     CX,D1L                    ;MSG LENGTH
E2D0 EB0A                            JMP     SHORT D13                 ;DISPLAY ERROR MSG
E2D2                         D12:
E2D2 A880                            TEST    AL,80H                    ;PLANAR RAM P-CHECK?
E2D4 7410                            JZ      D14                       ;NO - AUX INT
E2D6 BE27E2          R               MOV     SI,OFFSET D2              ;ADDR OF ERROR MSG
E2D9 B90E00                          MOV     CX,D2L                    ;MSG LENGTH
E2DC                         D13:
E2DC B80000                          MOV     AX,0                      ;INIT AND SET MODE FOR VIDEO
E2DF CD10                            INT     10H                       ;CALL VIDEO_IO PROCEDURE
E2E1 E8E603                          CALL    P_MSG                     ;PRINT ERROR MSG
E2E4 FA                              CLI
E2E5 F4                              HLT                               ;HALT SYSTEM
E2E6                         D14:
E2E6 58                              POP     AX                        ;RESTORE ORIG CONTENTS OF AX
E2E7 CF                              IRET
                             NMI_INT ENDP
```

```
LOC OBJ              LINE    SOURCE

                             ;-------------------------------------
                             ; INITIAL RELIABILITY TEST -- PHASE 3
                             ;--------------------------------------
                                     ASSUME  CS:CODE,DS:DATA

E2E8 20323031                E1       DB      ' 201'
 0004                        E1L     EQU     $-E1

                             ;       ESTABLISH BIOS SUBROUTINE CALL INTERRUPT VECTORS

E2EC                         TST8:
E2EC FC                              CLD                                     ;SET DIR FLAG TO GO FORWARD
E2ED BF4000                          MOV     DI,OFFSET VIDEO_INT             ;SETUP ADDR TO INTR AREA
E2F0 0E                              PUSH    CS
E2F1 1F                              POP     DS                              ;SETUP ADDR OF VECTOR TABLE
E2F2 BE13FF                          MOV     SI,0FF13H                       ; OFFSET VECTOR_TABLE+32
E2F5 B92000                          MOV     CX,20H
E2F8 F3                              REP     MOVSW                           ;MOVE VECTOR TABLE TO RAM
E2F9 A5

                             ;       SETUP TIMER 0 TO MODE 3

E2FA B0FF                            MOV     AL,0FFH                         ;DISABLE ALL DEVICE INTERRUPTS
E2FC E621                            OUT     INTA01,AL
E2FE B036                            MOV     AL,36H                          ;SEL TIM 0,LSB,MSB,MODE 3
E300 E643                            OUT     TIMER+3,AL                      ;WRITE TIMER MODE REG
E302 B000                            MOV     AL,0
E304 E640                            OUT     TIMER,AL                        ;WRITE LSB TO TIMER 0 REG
E306 E640                            OUT     TIMER,AL                        ;WRITE MSB TO TIMER 0 REG

                             ;       SETUP TIMER 0 TO BLINK LED IF MANUFACTURING TEST MODE

                                     ASSUME  DS:DATA
E308 B84000          R               MOV     AX,DATA                         ;POINT DS TO DATA SEG
E30B 8ED8                            MOV     DS,AX
E30D E87803                          CALL    KBD_RESET                       ;SEND SOFTWARE RESET TO KEYBRD
E310 80FBAA                          CMP     BL,0AAH                         ;SCAN CODE  AA' RETURNED?
E313 7426                            JE      E3                              ;YES - CONTINUE (NON MFG MODE)
E315 B03C                            MOV     AL,3CH                          ;EN KBD, SET KBD CLK LINE LOW
E317 E661                            OUT     PORT_B,AL                       ;WRITE 8255 PORT B
E319 90                              NOP
E31A 90                              NOP
E31B E460                            IN      AL,PORT_A                       ;WAS A BIT CLOCKED IN?
E31D 24FF                            AND     AL,0FFH
E31F 7516                            JNZ     E2                              ;YES - CONTINUE (NON MFG MODE)
E321 FE061200        R               INC     MFG_TST                         ;ELSE SET SW FOR MFG TEST MODE
E325 26C7062000B2E6 R                MOV     ES:INT_ADDR,OFFSET BLINK_INT    ;SETUP TIMER INTR TO BLINK LED
E32C 26C706220000F0 R                MOV     ES:INT_ADDR+2,CODE
E333 B0FE                            MOV     AL,0FEH                         ;ENABLE TIMER INTERRUPT
E335 E621                            OUT     INTA01,AL
E337                         E2:                                             ; JUMPER_NOT_IN:
E337 B0CC                            MOV     AL,0CCH                         ;RESET THE KEYBOARD
E339 E661                            OUT     PORT_B,AL
                             ;--------------------------------------------
                             ;TEST.05
                             ;       ROS CHECKSUM II
                             ;DESCRIPTION
                             ;       A CHECKSUM IS DONE FOR THE 4 ROS MODULES CONTAINING BASIC CODE
                             ;--------------------------------------------
E33B                         E3:
E33B B204                            MOV     DL,4                            ;NO. OF ROS MODULES TO CHECK
E33D BB0060                          MOV     BX,6000H                        ;SETUP STARTING ROS ADDR
E340                         E4:                                             ; CHECK_ROS:
E340 E8C8FE                          CALL    ROS_CHECKSUM
E343 7507                            JNE     E5                              ;BEEP SPEAKER IF ERROR
E345 FECA                            DEC     DL                              ;ANY MORE TO DO?
E347 75F7                            JNZ     E4                              ;YES - CONTINUE
E349 EB0790                          JMP     E6                              ;NO - GO TO NEXT TEST
E34C                         E5:                                             ; ROS_ERROR:
E34C BA0101                          MOV     DX,101H
E34F E8DE02                          CALL    ERR_BEEP                        ;BEEP SPEAKER
```

```
LOC OBJ                SOURCE

                       ;-----------------------------------------
                       ;TEST.08
                       ;       INITIALIZE AND START CRT CONTROLLER (6845)
                       ;       TEST VIDEO READ/WRITE STORAGE.
                       ;DESCRIPTION
                       ;       RESET THE VIDEO ENABLE SIGNAL.
                       ;       SELECT ALPHANUMERIC MODE, 40 * 25, B & W.
                       ;       READ/WRITE DATA PATTERNS TO STG. CHECK STG ADDRESSABILITY.
                       ;-----------------------------------------
E352                   E6:
E352 E460                      IN      AL,PORT_A               ;READ SENSE SWITCHES
E354 B400                      MOV     AH,0
E356 A31000          R         MOV     EQUIP_FLAG,AX           ;STORE SENSE SW INFO
E359 2430                      AND     AL,30H                  ;ISOLATE VIDEO SWS
E35B 7503                      JNZ     E7                      ;VIDEO SWS SET TO 0?
E35D E99800                    JMP     E19                     ;SKIP VIDEO TESTS FOR BURN-IN
E360                   E7:                                     ; TEST_VIDEO:
E360 86E0                      XCHG    AH,AL
E362 80FC30                    CMP     AH,30H                  ;B/W CARD ATTACHED?
E365 7409                      JE      E8                      ;YES - SET MODE FOR B/W CARD
E367 FEC0                      INC     AL                      ;SET COLOR MODE FOR COLOR CD
E369 80FC20                    CMP     AH,20H                  ;80X25 MODE SELECTED?
E36C 7502                      JNE     E8                      ;NO - SET MODE FOR 40X25
E36E B003                      MOV     AL,3                    ;SET MODE FOR 80X25
E370                   E8:                                     ; SET_MODE:
E370 50                        PUSH    AX                      ;SAVE VIDEO MODE ON STACK
E371 2AE4                      SUB     AH,AH                   ;INITIALIZE TO ALPHANUMERIC MD
E373 CD10                      INT     10H                     ;CALL VIDEO_IO
E375 58                        POP     AX                      ;RESTORE VIDEO SENSE SWS IN AH
E376 50                        PUSH    AX                      ; RESAVE VALUE
E377 BB00B0                    MOV     BX,0B000H               ;BEG VIDEO RAM ADDR B/W CD
E37A BAB803                    MOV     DX,3B8H                 ;MODE REG FOR B/W
E37D B90010                    MOV     CX,4096                 ;RAM BYTE CNT FOR B/W CD
E380 B001                      MOV     AL,1                    ; SET MODE FOR BW CARD
E382 80FC30                    CMP     AH,30H                  ;B/W VIDEO CARD ATTACHED?
E385 740B                      JE      E9                      ;YES - GO TEST VIDEO STG
E387 BB00B8                    MOV     BX,0B800H               ;BEG VIDEO RAM ADDR COLOR CD
E38A BAD803                    MOV     DX,3D8H                 ;MODE REG FOR COLOR CD
E38D B90040                    MOV     CX,4000H                ;RAM BYTE CNT FOR COLOR CD
E390 FEC8                      DEC     AL                      ; SET MODE TO 0 FOR COLOR CD
E392                   E9:                                     ; TEST_VIDEO_STG:
E392 EE                        OUT     DX,AL                   ;DISABLE VIDEO FOR COLOR CD
E393 8EC3                      MOV     ES,BX                   ;POINT ES TO VIDEO RAM STG
E395 B84000          R         MOV     AX,DATA                 ;POINT DS TO DATA SEGMENT
E398 8ED8                      MOV     DS,AX
E39A 813E72003412    R         CMP     RESET_FLAG,1234H        ;POD INITIATED BY KBD RESET?
E3A0 740D                      JE      E10                     ;YES - SKIP VIDEO RAM TEST
E3A2 8EDB                      MOV     DS,BX                   ;POINT DS TO VIDEO RAM STG
E3A4 E876FC                    CALL    STGTST_CNT              ;GO TEST VIDEO R/W STG
E3A7 7406                      JE      E10                     ;STG OK - CONTINUE TESTING
E3A9 BA0201                    MOV     DX,102H                 ;SETUP # OF BEEPS
E3AC E88102                    CALL    ERR_BEEP                ;GO BEEP SPEAKER
                       ;-----------------------------------------
                       ;TEST.09
                       ;       SETUP VIDEO DATA ON SCREEN FOR VIDEO LINE TEST.
                       ;DESCRIPTION
                       ;       ENABLE VIDEO SIGNAL AND SET MODE.
                       ;       DISPLAY A HORIZONTAL BAR ON SCREEN.
                       ;-----------------------------------------
E3AF                   E10:
E3AF 58                        POP     AX                      ;GET VIDEO SENSE SWS (AH)
E3B0 50                        PUSH    AX                      ;SAVE IT
E3B1 B400                      MOV     AH,0                    ;ENABLE VIDEO AND SET MODE
E3B3 CD10                      INT     10H                     ; VIDEO
E3B5 B82070                    MOV     AX,7020H                ;WRT BLANKS IN REVERSE VIDEO
E3B8 2BFF                      SUB     DI,DI                   ;SETUP STARTING LOC
E3BA B92800                    MOV     CX,40                   ;NO. OF BLANKS TO DISPLAY
E3BD FC                        CLD                             ;SET DIR FLAG TO INCREMENT
E3BE F3                        REP     STOSW                   ;WRITE VIDEO STORAGE
E3BF AB
```

```
LOC OBJ                SOURCE

                       ;--------------------------------------------
                       ;TEST.10
                       ;       CRT INTERFACE LINES TEST
                       ;DESCRIPTION
                       ;       SENSE ON/OFF TRANSITION OF THE VIDEO ENABLE AND HORIZONTAL
                       ;       SYNC LINES.
                       ;--------------------------------------------
E3C0 58                        POP     AX                              ;GET VIDEO SENSE SW INFO
E3C1 50                        PUSH    AX                              ;SAVE IT
E3C2 80FC30                    CMP     AH,30H                          ;B/W CARD ATTACHED?
E3C5 BABA03                    MOV     DX,03BAH                        ;SETUP ADDR OF BW STATUS PORT
E3C8 7403                      JE      E11                             ;YES - GO TEST LINES
E3CA BADA03                    MOV     DX,03DAH                        ;COLOR CARD IS ATTACHED
E3CD                   E11:                                            ; LINE_TST:
E3CD B408                      MOV     AH,8
E3CF                   E12:                                            ; OFLOOP_CNT:
E3CF 2BC9                      SUB     CX,CX
E3D1 EC                E13:    IN      AL,DX                           ;READ CRT STATUS PORT
E3D2 22C4                      AND     AL,AH                           ;CHECK VIDEO/HORZ LINE
E3D4 7504                      JNZ     E14                             ;ITS ON - CHECK IF IT GOES OFF
E3D6 E2F9                      LOOP    E13                             ;LOOP TILL ON OR TIMEOUT
E3D8 EB13                      JMP     SHORT E17                       ;GO PRINT ERROR MSG
E3DA 2BC9              E14:    SUB     CX,CX
E3DC EC                E15:    IN      AL,DX                           ;READ CRT STATUS PORT
E3DD 22C4                      AND     AL,AH                           ;CHECK VIDEO/HORZ LINE
E3DF 7404                      JZ      E16                             ;ITS ON - CHECK NEXT LINE
E3E1 E2F9                      LOOP    E15                             ;LOOP IF OFF TILL IT GOES ON
E3E3 EB08                      JMP     SHORT E17
E3E5                   E16:                                            ; NXT_LINE:
E3E5 B103                      MOV     CL,3                            ;GET NEXT BIT TO CHECK
E3E7 D2EC                      SHR     AH,CL                           ;
E3E9 75E4                      JNZ     E12                             ;GO CHECK HORIZONTAL LINE
E3EB EB06                      JMP     SHORT E18                       ;DISPLAY CURSOR ON SCREEN
E3ED                   E17:                                            ; CRT_ERR:
E3ED BA0201                    MOV     DX,102H
E3F0 E83D02                    CALL    ERR_BEEP                        ;GO BEEP SPEAKER
E3F3                   E18:                                            ; DISPLAY_CURSOR:
E3F3 58                        POP     AX                              ;GET VIDEO SENSE SWS (AH)
E3F4 B400                      MOV     AH,0                            ;SET MODE AND DISPLAY CURSOR
E3F6 CD10                      INT     10H                             ;CALL VIDEO I/O PROCEDURE
                       ;--------------------------------------------
                       ;TEST.11
                       ;       ADDITIONAL READ/WRITE STORAGE TEST
                       ;DESCRIPTION
                       ;       WRITE/READ DATA PATTERNS TO ANY READ/WRITE STORAGE AFTER THE BASIC
                       ;       16K.  STORAGE ADDRESSABILITY IS CHECKED.
                       ;--------------------------------------------
                               ASSUME  DS:DATA
E3F8                   E19:
E3F8 B84000        R           MOV     AX,DATA
E3FB 8ED8                      MOV     DS,AX

                       ;       DETERMINE RAM SIZE ON PLANAR BOARD

E3FD 8A261000      R           MOV     AH,BYTE PTR EQUIP_FLAG          ;GET SENSE SWS INFO
E401 80E40C                    AND     AH,0CH                          ;ISOLATE RAM SIZE SWS
E404 B004                      MOV     AL,4
E406 F6E4                      MUL     AH
E408 0410                      ADD     AL,16                           ;ADD BASIC 16K
E40A 8BD0                      MOV     DX,AX                           ;SAVE PLANAR RAM SIZE IN DX
E40C 8BD8                      MOV     BX,AX                           ;  AND IN BX

                       ;       DETERMINE IO CHANNEL RAM SIZE

E40E E462                      IN      AL,PORT_C                       ;READ IO CH RAM SIZE SWS
E410 240F                      AND     AL,0FH                          ;ISOLATE FROM OTHER BITS
E412 B420                      MOV     AH,32
E414 F6E4                      MUL     AH
E416 A31500        R           MOV     IO_RAM_SIZE,AX                  ;SAVE IO CHANNEL RAM SIZE
E419 83FB40                    CMP     BX,40H                          ;PLANAR RAM SIZE = 64K?
E41C 7402                      JE      E20                             ;YES - ADD IO CHN RAM SIZE
E41E 2BC0                      SUB     AX,AX                           ;NO - DON T ADD ANY IO RAM
E420                   E20:                                            ; ADD_IO_SIZE:
E420 03C3                      ADD     AX,BX                           ;SUM TOTAL RAM SIZE
E422 A31300        R           MOV     MEMORY_SIZE,AX                  ;SETUP MEMORY SIZE PARM
E425 813E72003412  R           CMP     RESET_FLAG,1234H                ;POD INITIATED BY KBD RESET?
```

```
LOC OBJ                LINE    SOURCE

E42B 744D                              JE      E22                     ;YES - SKIP MEMORY TEST

                               ;       TEST ANY OTHER READ/WRITE STORAGE AVAILABLE

E42D BB0004                            MOV     BX,400H
E430 B91000                            MOV     CX,16
E433                           E21:
E433 3BD1                              CMP     DX,CX                   ;ANY MORE STG TO BE TESTED?
E435 7646                              JBE     E23                     ;NO - GO TO NEXT TEST
E437 8EDB                              MOV     DS,BX                   ;SETUP STG ADDR IN DS AND ES
E439 8EC3                              MOV     ES,BX
E43B 83C110                            ADD     CX,16                   ;INCREMENT STG BYTE COUNTER
E43E 81C30004                          ADD     BX,400H                 ;SET POINTER TO NEXT 16K BLK
E442 51                                PUSH    CX                      ;SAVE REGS
E443 53                                PUSH    BX
E444 52                                PUSH    DX
E445 E8D2FB                            CALL    STGTST                  ;GO TEST A 16K BLK OF STG
E448 5A                                POP     DX
E449 5B                                POP     BX                      ;RESTORE REGS
E44A 59                                POP     CX
E44B 74E6                              JE      E21                     ;CHECK IF MORE STG TO TEST

                               ;       PRINT FAILING ADDRESS AND XOR'ED PATTERN IF DATA COMPARE ERROR

E44D 8CDA                              MOV     DX,DS                   ;CONVERT FAILING HIGH-ORDER
E44F 8AE8                              MOV     CH,AL                   ;SAVE FAILING BIT PATTERN
E451 8AC6                              MOV     AL,DH                   ;GET FAILING ADDR (HIGH BYTE)
E453 B104                              MOV     CL,4
E455 D2E8                              SHR     AL,CL                   ;RIGHT-JUSTIFY HIGH BYTE
E457 E83E00                            CALL    XLAT_PRINT_CODE         ;CONVERT AND PRINT CODE
E45A 8AC6                              MOV     AL,DH
E45C 240F                              AND     AL,0FH
E45E E83700                            CALL    XLAT_PRINT_CODE         ;CONVERT AND PRINT CODE
E461 8AC5                              MOV     AL,CH                   ;GET FAILING BIT PATTERN
E463 B104                              MOV     CL,4                    ; AND ISOLATE LEFTMOST NIBBLE
E465 D2E8                              SHR     AL,CL
E467 E82E00                            CALL    XLAT_PRINT_CODE         ;CONVERT AND PRINT CODE
E46A 8AC5                              MOV     AL,CH                   ;GET FAILING BIT PATTERN AND
E46C 240F                              AND     AL,0FH                  ;  ISOLATE RIGHTMOST NIBBLE
E46E E82700                            CALL    XLAT_PRINT_CODE         ;CONVERT AND PRINT CODE
E471 BEE8E2         R                  MOV     SI,OFFSET E1            ;SETUP ADDRESS OF ERROR MSG
E474 B90400                            MOV     CX,E1L                  ;GET MSG BYTE COUNT
E477 E85002                            CALL    P_MSG                   ;PRINT ERROR MSG
E47A                           E22:                                    ; GO_TST12:
E47A E94A00                            JMP     TST12                   ;GO TO NEXT TEST
E47D                           E23:                                    ; STG_TEST_DONE:
E47D B84000         R                  MOV     AX,DATA                 ;POINT DS TO DATA SEGMENT
E480 8ED8                              MOV     DS,AX                   ; CHG MADE 3/27/81
E482 8B161500       R                  MOV     DX,IO_RAM_SIZE          ;GET IO CHANNEL RAM SIZE
E486 0BD2                              OR      DX,DX                   ;SET FLAG RESULT
E488 74F0                              JZ      E22                     ;NO IO RAM, GO TO NEXT TEST
E48A B90000                            MOV     CX,0
E48D 81FB0010                          CMP     BX,1000H                ;HAS IO RAM BEEN TESTED
E491 77E7                              JA      E22                     ;YES - GO TO NEXT TEST
E493 BB0010                            MOV     BX,1000H                ;SETUP BEG LOC FOR IO RAM
E496 EB9B                              JMP     SHORT E21               ;GO TEST IO CHANNEL RAM
                               ;------------------------------------------
                               ;       CONVERT AND PRINT ASCII CODE
                               ;
                               ;       AL MUST CONTAIN NUMBER TO BE CONVERTED.
                               ;       AX AND BX DESTROYED.
                               ;------------------------------------------
E498                           XLAT_PRINT_CODE  PROC    NEAR
E498 1E                                PUSH    DS                      ;SAVE DS VALUE
E499 0E                                PUSH    CS                      ;POINT DS TO CODE SEG
E49A 1F                                POP     DS
E49B BBB7E4                            MOV     BX,0E4B7H               ; OFFSET ASCII_TBL-XLAT TABLE
E49E D7                                XLATB
E49F B40E                              MOV     AH,14
E4A1 B700                              MOV     BH,0
E4A3 CD10                              INT     10H                     ;CALL VIDEO_IO
E4A5 1F                                POP     DS                      ;RESTORE ORIG VALUE IN DS
E4A6 C3                                RET
                               XLAT_PRINT_CODE  ENDP
```

```
LOC OBJ                          SOURCE

                                 ;---------------------------------------
                                 ; INITIAL RELIABILITY TEST -- PHASE 4
                                 ;---------------------------------------
                                         ASSUME  CS:CODE,DS:DATA
E4A7 20333031                    F1      DB      ' 301'
  0004                           F1L     EQU     $-F1            ; KEYBOARD MESSAGE
E4AB 313331                      F2      DB      '131'
  0003                           F2L     EQU     $-F2            ; CASSETTE MESSAGE
E4AE 363031                      F3      DB      '601'
  0003                           F3L     EQU     $-F3            ; DISKETTE MESSAGE

E4B1                             F4      LABEL   WORD            ; PRINTER SOURCE TABLE
E4B1 BC03                                DW      3BCH
E4B3 7803                                DW      378H
E4B5 7802                                DW      278H
E4B7                             F4E     LABEL   WORD
E4B7 30313233343536              ASCII_TBL       DB      '0123456789ABCDEF'
     37383941424344
     4546
                                 ;-------------------------------------------
                                 ;TEST.12
                                 ;       KEYBOARD TEST
                                 ;DESCRIPTION
                                 ;       RESET THE KEYBOARD AND CHECK THAT SCAN CODE  AA' IS RETURNED
                                 ;       TO THE CPU.  CHECK FOR STUCK KEYS.
                                 ;-------------------------------------------
E4C7                             TST12:
E4C7 B84000           R                  MOV     AX,DATA                 ;POINT DS TO DATA SEG
E4CA 8ED8                                MOV     DS,AX
E4CC 803E120001       R                  CMP     MFG_TST,1               ;MANUFACTURING TEST MODE?
E4D1 7439                                JE      F7                      ;YES - SKIP KEYBOARD TEST
E4D3 E8B201                              CALL    KBD_RESET               ;ISSUE SOFTWARE RESET TO KEYBRD
E4D6 E32B                                JCXZ    F6                      ;PRINT ERR MSG IF NO INTERRUPT
E4D8 B04D                                MOV     AL,4DH                  ;ENABLE KEYBOARD
E4DA E661                                OUT     PORT_B,AL
E4DC 80FBAA                              CMP     BL,0AAH                 ;SCAN CODE AS EXPECTED?
E4DF 7522                                JNE     F6                      ;NO - DISPLAY ERROR MSG

                                 ;       CHECK FOR STUCK KEYS

E4E1 B0CC                                MOV     AL,0CCH                 ;CLR KBD, SET CLK LINE HIGH
E4E3 E661                                OUT     PORT_B,AL
E4E5 B04C                                MOV     AL,4CH                  ;ENABLE KBD,CLK IN NEXT BYTE
E4E7 E661                                OUT     PORT_B,AL
E4E9 2BC9                                SUB     CX,CX
E4EB                             F5:                                     ; KBD_WAIT:
E4EB E2FE                                LOOP    F5                      ;DELAY FOR A WHILE
E4ED E460                                IN      AL,KBD_IN               ;CHECK FOR STUCK KEYS
E4EF 3C00                                CMP     AL,0                    ;SCAN CODE = 0?
E4F1 7419                                JE      F7                      ;YES - CONTINUE TESTING
E4F3 8AE8                                MOV     CH,AL                   ;SAVE SCAN CODE
E4F5 B104                                MOV     CL,4
E4F7 D2E8                                SHR     AL,CL                   ;RIGHT-JUSTIFY HIGH BYTE
E4F9 E89CFF                              CALL    XLAT_PRINT_CODE         ;CONVERT AND PRINT
E4FC 8AC5                                MOV     AL,CH                   ;RECOVER SCAN CODE
E4FE 240F                                AND     AL,0FH                  ;ISOLATE LOW ORDER BYTE
E500 E895FF                              CALL    XLAT_PRINT_CODE         ;CONVERT AND PRINT
E503 BEA7E4           R          F6:     MOV     SI,OFFSET F1            ;GET MSG ADDR
E506 B90400                              MOV     CX,F1L                  ;GET MSG BYTE COUNT
E509 E8BE01                              CALL    P_MSG                   ;PRINT MSG ON SCREEN

                                 ;       SETUP INTERRUPT VECTOR TABLE

E50C                             F7:                                     ; SETUP_INT_TABLE:
E50C 2BC0                                SUB     AX,AX
E50E 8EC0                                MOV     ES,AX
E510 B93000                              MOV     CX,24*2                 ;GET VECTOR CNT
E513 0E                                  PUSH    CS                      ;SETUP DS SEG REG
E514 1F                                  POP     DS
E515 BEF3FE                              MOV     SI,0FEF3H               ; OFFSET VECTOR_TABLE
E518 BF2000                              MOV     DI,OFFSET INT_PTR
E51B FC                                  CLD
E51C F3                                  REP     MOVSW
E51D A5
```

```
LOC OBJ              LINE    SOURCE

                             ;-------------------------------------------
                             ;TEST.13
                             ;       CASSETTE DATA WRAP TEST
                             ;DESCRIPTION
                             ;       TURN CASSETTE MOTOR OFF. WRITE A BIT OUT TO THE CASSETTE DATA BUS.
                             ;       VERIFY THAT CASSETTE DATA READ IS WITHIN A VALID RANGE.
                             ;-------------------------------------------

                             ;       TURN THE CASSETTE MOTOR OFF

E51E B84000        R                 MOV     AX,DATA                         ;POINT DS REG TO DATA SEG
E521 8ED8                            MOV     DS,AX
E523 B04D                            MOV     AL,04DH                         ;SET TIMER 2 SPK OUT, AND CASST
E525 E661                            OUT     PORT_B,AL                       ;OUT BITS ON, CASSETTE MOT OFF

                             ;       WRITE A BIT

E527 B0FF                            MOV     AL,0FFH                         ;DISABLE TIMER INTERRUPTS
E529 E621                            OUT     INTA01,AL
E52B B0B6                            MOV     AL,0B6H                         ;SEL TIM 2, LSB, MSB, MD 3
E52D E643                            OUT     TIMER+3,AL                      ;WRITE 8253 CMD/MODE REG
E52F B8D304                          MOV     AX,1235                         ;SET TIMER 2 CNT FOR 1000 USEC
E532 E642                            OUT     TIMER+2,AL                      ;WRITE TIMER 2 COUNTER REG
E534 8AC4                            MOV     AL,AH                           ;WRITE MSB
E536 E642                            OUT     TIMER+2,AL

                             ;       READ CASSETTE INPUT

E538 E462                            IN      AL,PORT_C                       ;READ VALUE OF CASS IN BIT
E53A 2410                            AND     AL,10H                          ;ISOLATE FROM OTHER BITS
E53C A26B00        R                 MOV     LAST_VAL,AL
E53F E83E14                          CALL    READ_HALF_BIT
E542 E83B14                          CALL    READ_HALF_BIT
E545 E30C                            JCXZ    F8                              ; CAS_ERR
E547 81FB4005                        CMP     BX,MAX_PERIOD
E54B 7306                            JNC     F8                              ; CAS_ERR
E54D 81FB1004                        CMP     BX,MIN_PERIOD
E551 7309                            JNC     F9                              ;GO TO NEXT TEST IF OK
E553                         F8:                                             ; CAS_ERR:
E553 BEABE4        R                 MOV     SI,OFFSET F2                    ;CASSETTE WRAP FAILED
E556 B90300                          MOV     CX,F2L
E559 E86E01                          CALL    P_MSG                           ;GO PRINT ERROR MSG
                             ;-------------------------------------------
                             ;TEST.14
                             ;       DISKETTE ATTACHMENT TEST
                             ;DESCRIPTION
                             ;       CHECK IF IPL DISKETTE DRIVE IS ATTACHED TO SYSTEM.  IF ATTACHED,
                             ;       VERIFY STATUS OF NEC FDC AFTER A RESET.  ISSUE A RECAL AND SEEK
                             ;       CMD TO FDC AND CHECK STATUS. COMPLETE SYSTEM INITIALIZATION THEN
                             ;       PASS CONTROL TO THE BOOT LOADER PROGRAM.
                             ;-------------------------------------------
E55C                         F9:
E55C B0FC                            MOV     AL,0FCH                         ;ENABLE TIMER AND KBD INTS
E55E E621                            OUT     INTA01,AL
E560 A01000        R                 MOV     AL,BYTE PTR EQUIP_FLAG          ;GET SENSE SWS INFO
E563 A801                            TEST    AL,01H                          ;IPL DISKETTE DRIVE ATTCH?
E565 7503                            JNZ     F10                             ;YES - TEST DISKETTE CONTR
E567 E9B900                          JMP     F22                             ;NO - SKIP THIS TEST
E56A                         F10:                                            ; DISK_TEST:
E56A B0BC                            MOV     AL,0BCH                         ;ENABLE DISKETTE, KEYBOARD,
E56C E621                            OUT     INTA01,AL                       ; AND TIMER INTERRUPTS
E56E B400                            MOV     AH,0                            ;RESET NEC FDC
E570 CD13                            INT     13H                             ;VERIFY STATUS AFTER RESET
E572 F6C4FF                          TEST    AH,0FFH                         ;STATUS OK?
E575 7520                            JNZ     F13                             ;NO - FDC FAILED

                             ;       TURN DRIVE 0 MOTOR ON

E577 BAF203                          MOV     DX,03F2H                        ;GET ADDR OF FDC CARD
E57A B01C                            MOV     AL,1CH                          ;TURN MOTOR ON, EN DMA/INT
E57C EE                              OUT     DX,AL                           ;WRITE FDC CONTROL REG
E57D 2BC9                            SUB     CX,CX
E57F                         F11:                                            ; MOTOR_WAIT:
E57F E2FE                            LOOP    F11                             ;WAIT FOR 1 SECOND
E581                         F12:                                            ; MOTOR_WAIT1:
E581 E2FE                            LOOP    F12
E583 33D2                            XOR     DX,DX                           ;SELECT DRIVE 0
```

```
LOC OBJ                 LINE    SOURCE

E585 B501                               MOV     CH,1                            ;SELECT TRACK 1
E587 88163E00       R                   MOV     SEEK_STATUS,DL
E58B E8F308                             CALL    SEEK                            ;RECALIBRATE DISKETTE
E58E 7207                               JC      F13                             ;GO TO ERR SUBROUTINE IF ERR
E590 B522                               MOV     CH,34                           ;SELECT TRACK 34
E592 E8EC08                             CALL    SEEK                            ;SEEK TO TRACK 34
E595 7309                               JNC     F14                             ;OK, TURN MOTOR OFF
E597                            F13:                                            ; DSK_ERR:
E597 BEAEE4         R                   MOV     SI,OFFSET F3                    ;GET ADDR OF MSG
E59A B90300                             MOV     CX,F3L                          ;GET MSG BYTE COUNT
E59D E82A01                             CALL    P_MSG                           ;GO PRINT ERROR MSG

                                ;       TURN DRIVE 0 MOTOR OFF

E5A0                            F14:                                            ; DRO_OFF:
E5A0 B00C                               MOV     AL,0CH                          ;TURN DRIVE 0 MOTOR OFF
E5A2 BAF203                             MOV     DX,03F2H                        ; FDC CTL ADDRESS
E5A5 EE                                 OUT     DX,AL

                                ;       SETUP PRINTER AND RS232 BASE ADDRESSES IF DEVICE ATTACHED

E5A6                            F15:                                            ; JMP_BOOT:
E5A6 C7061A001E00   R                   MOV     BUFFER_HEAD,OFFSET KB_BUFFER    ;SETUP KEYBOARD PARAMETERS
E5AC C7061C001E00   R                   MOV     BUFFER_TAIL,OFFSET KB_BUFFER
E5B2 BDB1E4         R                   MOV     BP,OFFSET F4                    ; PRT_SRC_TBL
E5B5 BE0000                             MOV     SI,0
E5B8                            F16:                                            ; PRT_BASE:
E5B8 2E8B5600                           MOV     DX,CS:[BP]                      ;GET PRINTER BASE ADDR
E5BC B0AA                               MOV     AL,0AAH                         ;WRITE DATA TO PORT A
E5BE EE                                 OUT     DX,AL
E5BF 2AC0                               SUB     AL,AL
E5C1 EC                                 IN      AL,DX                           ;READ PORT A
E5C2 3CAA                               CMP     AL,0AAH                         ;DATA PATTERN SAME
E5C4 7506                               JNE     F17                             ;NO - CHECK NEXT PRT CD
E5C6 89940800       R                   MOV     PRINTER_BASE[SI],DX             ;YES - STORE PRT BASE ADDR
E5CA 46                                 INC     SI                              ;INCREMENT TO NEXT WORD
E5CB 46                                 INC     SI
E5CC                            F17:                                            ; NO_STORE:
E5CC 45                                 INC     BP                              ;POINT TO NEXT BASE ADDR
E5CD 45                                 INC     BP
E5CE 81FDB7E4       R                   CMP     BP,OFFSET F4E                   ;ALL POSSIBLE ADDRS CHECKED?
E5D2 75E4                               JNE     F16                             ;PRT_BASE
E5D4 BB0000                             MOV     BX,0                            ;POINTER TO RS232 TABLE
E5D7 BAFA03                             MOV     DX,3FAH                         ;CHECK IF RS232 CD 1 ATTCH?
E5DA EC                                 IN      AL,DX                           ;READ INTR ID REG
E5DB A8F8                               TEST    AL,0F8H
E5DD 7508                               JNZ     F18
E5DF C7870000F803   R                   MOV     RS232_BASE[BX],3F8H             ;SETUP RS232 CD #1 ADDR
E5E5 43                                 INC     BX
E5E6 43                                 INC     BX
E5E7 BAFA02                     F18:    MOV     DX,2FAH                         ;CHECK IF RS232 CD 2 ATTCH
E5EA EC                                 IN      AL,DX                           ;READ INTERRUPT ID REG
E5EB A8F8                               TEST    AL,0F8H
E5ED 7508                               JNZ     F19                             ;BASE_END
E5EF C7870000F802   R                   MOV     RS232_BASE[BX],2F8H             ;SETUP RS232 CD #2
E5F5 43                                 INC     BX
E5F6 43                                 INC     BX

                                ;------ SET UP EQUIP FLAG TO INDICATE NUMBER OF PRINTERS AND RS232 CARDS

E5F7                            F19:                                    ; BASE_END:
E5F7 8BC6                               MOV     AX,SI                   ; SI HAS 2* NUMBER OF RS232
E5F9 B103                               MOV     CL,3                    ; SHIFT COUNT
E5FB D2C8                               ROR     AL,CL                   ; ROTATE RIGHT 3 POSITIONS
E5FD 0AC3                               OR      AL,BL                   ; OR IN THE PRINTER COUNT
E5FF A21100         R                   MOV     BYTE PTR EQUIP_FLAG+1,AL        ; STORE AS SECOND BYTE
E602 BA0102                             MOV     DX,201H
E605 EC                                 IN      AL,DX
E606 A80F                               TEST    AL,0FH
E608 7505                               JNZ     F20                             ; NO_GAME_CARD
E60A 800E110010     R                   OR      BYTE PTR EQUIP_FLAG+1,16
E60F                            F20:                                            ; NO_GAME_CARD:

                                ;       ENABLF NMI INTERRUPTS

E60F B080                               MOV     AL,80H                          ;ENABLE NMI INTERRUPTS
E611 E6A0                               OUT     0A0H,AL
```

```
LOC OBJ                LINE    SOURCE

E613 803E120001   R            CMP     MFG_TST,1                 ;MFG MODE?
E618 7406                      JE      F21                       ; LOAD_BOOT_STRAP
E61A BA0100                    MOV     DX,1                      ;
E61D E81000                    CALL    ERR_BEEP                  ;BEEP 1 SHORT TONE
E620                   F21:                                      ; LOAD_BOOT_STRAP:
E620 E9CF00                    JMP     BOOT_STRAP                ;GO TO THE BOOT LOADER
E623                   F22:                                      ; LOOP_POD:
E623 803E120001   R            CMP     MFG_TST,1                 ;MANUFACTURING TEST MODE?
E628 7503                      JNE     F23                       ;NO - GO TO BOOT LOADER
E62A E92EFA                    JMP     START                     ;YES - LOOP POWER-ON-DIAGS
E62D                   F23:                                      ; GO_TO_BOOT:
E62D E976FF                    JMP     F15                       ; JMP_BOOT
                       ;-------------------------------------------
                       ; INITIAL RELIABILITY TEST -- SUBROUTINES
                       ;-------------------------------------------
                               ASSUME  CS:CODE,DS:DATA
                       ;-------------------------------------------
                       ;       SUBROUTINES FOR POWER ON DIAGNOSTICS
                       ;-------------------------------------------
                       ;       THIS PROCEDURE WILL ISSUE ONE LONG TONE (3 SECS) AND ONE OR
                       ;       MORE SHORT TONES (1 SEC) TO INDICATE A FAILURE ON THE PLANAR
                       ;       BOARD, A BAD RAM MODULE, OR A PROBLEM WITH THE CRT.
                       ;ENTRY PARAMETERS:
                       ;       DH = NUMBER OF LONG TONES TO BEEP
                       ;       DL = NUMBER OF SHORT TONES TO BEEP.
                       ;-------------------------------------------
E630                   ERR_BEEP  PROC  NEAR
E630 9C                        PUSHF                             ;SAVE FLAGS
E631 FA                        CLI                               ;DISABLE SYSTEM INTERRUPTS
E632 1E                        PUSH    DS                        ;SAVE DS REG CONTENTS
E633 B84000       R            MOV     AX,DATA                   ;POINT DS TO DATA SEG
E636 8ED8                      MOV     DS,AX
E638 0AF6                      OR      DH,DH           ; ANY LONG ONES TO BEEP
E63A 7418                      JZ      G3              ; NO, DO THE SHORT ONES
E63C                   G1:                             ; LONG_BEEP:
E63C B306                      MOV     BL,6            ; COUNTER FOR BEEPS
E63E E82500                    CALL    BEEP            ; DO THE BEEP
E641 E2FE              G2:     LOOP    G2              ; DELAY BETWEEN BEEPS
E643 FECE                      DEC     DH              ; ANY MORE TO DO
E645 75F5                      JNZ     G1              ; DO IT
E647 803E120001   R            CMP     MFG_TST,1       ; MFG TEST MODE?
E64C 7506                      JNE     G3              ; YES - CONTINUE BEEPING SPEAKER
E64E B0CD                      MOV     AL,0CDH         ; STOP BLINKING LED
E650 E661                      OUT     PORT_B,AL
E652 EBE8                      JMP     SHORT G1
E654                   G3:                             ; SHORT_BEEP:
E654 B301                      MOV     BL,1            ; COUNTER FOR A SHORT BEEP
E656 E80D00                    CALL    BEEP            ; DO THE SOUND
E659 E2FE              G4:     LOOP    G4              ; DELAY BETWEEN BEEPS
E65B FECA                      DEC     DL              ; DONE WITH SHORTS
E65D 75F5                      JNZ     G3              ; DO SOME MORE
E65F E2FE              G5:     LOOP    G5              ; LONG DELAY BEFORE RETURN
E661 E2FE              G6:     LOOP    G6
E663 1F                        POP     DS              ;RESTORE ORIG CONTENTS OF DS
E664 9D                        POPF                    ;RESTORE FLAGS TO ORIG SETTINGS
E665 C3                        RET                     ; RETURN TO CALLER
                       ERR_BEEP        ENDP

                       ;       ROUTINE TO SOUND BEEPER

E666                   BEEP    PROC    NEAR
E666 B0B6                      MOV     AL,10110110B              ;SEL TIM 2,LSB,MSB,BINARY
E668 E643                      OUT     TIMER+3,AL                ;WRITE THE TIMER MODE REG
E66A B83305                    MOV     AX,533H                   ;DIVISOR FOR 1000 HZ
E66D E642                      OUT     TIMER+2,AL                ;WRITE TIMER 2 CNT - LSB
E66F 8AC4                      MOV     AL,AH
E671 E642                      OUT     TIMER+2,AL                ;WRITE TIMER 2 CNT - MSB
E673 E461                      IN      AL,PORT_B                 ;GET CURRENT SETTING OF PORT
E675 8AE0                      MOV     AH,AL                     ; SAVE THAT SETTINGH
E677 0C03                      OR      AL,03                     ;TURN SPEAKER ON
E679 E661                      OUT     PORT_B,AL
E67B 2BC9                      SUB     CX,CX                     ;SET CNT TO WAIT 500 MS
E67D E2FE              G7:     LOOP    G7                        ;DELAY BEFORE TURNING OFF
E67F FECB                      DEC     BL                        ;DELAY CNT EXPIRED?
E681 75FA                      JNZ     G7                        ;NO - CONTINUE BEEPING SPK
E683 8AC4                      MOV     AL,AH                     ; RECOVER VALUE OF PORT
```

```
LOC OBJ                 LINE    SOURCE

E685 E661                               OUT     PORT_B,AL
E687 C3                                 RET                             ;RETURN TO CALLER
                                BEEP    ENDP
                                ;-------------------------------------------
                                ;       THIS PROCEDURE WILL SEND A SOFTWARE RESET TO THE KEYBOARD.
                                ;       SCAN CODE  AA' SHOULD BE RETURNED TO THE CPU.
                                ;-------------------------------------------
E688                            KBD_RESET       PROC    NEAR
E688 B00C                               MOV     AL,0CH                  ;SET KBD CLK LINE LOW
E68A E661                               OUT     PORT_B,AL               ;WRITE 8255 PORT B
E68C B95629                             MOV     CX,10582                ;HOLD KBD CLK LOW FOR 20 MS
E68F E2FE                       G8:     LOOP    G8                      ;LOOP FOR 20 MS
E691 B0CC                               MOV     AL,0CCH                 ;SET CLK, ENABLE LINES HIGH
E693 E661                               OUT     PORT_B,AL
E695                            SP_TEST:                ; ENTRY FOR MANUFACTURING TEST 2
E695 B04C                               MOV     AL,4CH                  ;SET KBD CLK HIBH, ENABLE LOW
E697 E661                               OUT     PORT_B,AL
E699 B0FD                               MOV     AL,0FDH                 ;ENABLE KEYBOARD INTERRUPTS
E69B E621                               OUT     INTA01,AL               ;WRITE 8259 IMR
E69D FB                                 STI                             ;ENABLE SYSTEM INTERRUPTS
E69E B400                               MOV     AH,0                    ;RESET INTERRUPT INDICATOR
E6A0 2BC9                               SUB     CX,CX                   ;SETUP INTERRUPT TIMEOUT CNT
E6A2 F6C4FF                     G9:     TEST    AH,0FFH                 ;DID A KEYBOARD INTR OCCUR?
E6A5 7502                               JNZ     G10                     ;YES - READ SCAN CODE RETURNED
E6A7 E2F9                               LOOP    G9                      ;NO - LOOP TILL TIMEOUT
E6A9 E460                       G10:    IN      AL,PORT_A               ;READ KEYBOARD SCAN CODE
E6AB 8AD8                               MOV     BL,AL                   ;SAVE SCAN CODE JUST READ
E6AD B0CC                               MOV     AL,0CCH                 ;CLEAR KEYBOARD
E6AF E661                               OUT     PORT_B,AL
E6B1 C3                                 RET                             ;RETURN TO CALLER
                                KBD_RESET       ENDP
                                ;-------------------------------------------
                                ;       BLINK LED PROCEDURE FOR MFG BURN-IN AND RUN-IN TESTS
                                ;       (LED WILL BLINK APPROXIMATELY .25 SECOND)
                                ;-------------------------------------------
E6B2                            BLINK_INT       PROC    NEAR
E6B2 FB                                 STI
E6B3 51                                 PUSH    CX                      ;SAVE CX REG CONTENTS
E6B4 50                                 PUSH    AX                      ;SAVE AX REG CONTENTS
E6B5 E461                               IN      AL,PORT_B               ;READ CURRENT VAL OF PORT B
E6B7 24BF                               AND     AL,0BFH
E6B9 E661                               OUT     PORT_B,AL               ;BLINK LED
E6BB 2BC9                               SUB     CX,CX
E6BD E2FE                       G11:    LOOP    G11
E6BF 0C40                               OR      AL,40H                  ;STOP BLINKING LED
E6C1 E661                               OUT     PORT_B,AL
E6C3 B020                               MOV     AL,EOI
E6C5 E620                               OUT     INTA00,AL
E6C7 58                                 POP     AX                      ;RESTORE AX REG
E6C8 59                                 POP     CX                      ;RESTORE CX REG
E6C9 CF                                 IRET
                                BLINK_INT       ENDP
                                ;-------------------------------------------
                                ;       THIS SUBROUTINE WILL PRINT A MESSAGE ON THE DISPLAY
                                ;
                                ;ENTRY REQUIREMENTS:
                                ;       SI = OFFSET(ADDRESS) OF MESSAGE BUFFER
                                ;       CX = MESSAGE BYTE COUNT
                                ;       MAXIMUM MESSAGE LENGTH IS 36 CHARACTERS
                                ;-------------------------------------------
E6CA                            P_MSG   PROC    NEAR
E6CA B84000          R                  MOV     AX,DATA                 ;POINT DS TO DATA SEG
E6CD 8ED8                               MOV     DS,AX
E6CF 803E120001      R                  CMP     MFG_TST,1               ;MFG TEST MODE?
E6D4 7505                               JNE     G12                     ;NO - DISPLAY ERROR MSG
E6D6 B601                               MOV     DH,1                    ;YES - SETUP TO BEEP SPEAKER
E6D8 E955FF                             JMP     ERR_BEEP                ;YES - BEEP SPEAKER
E6DB                            G12:                                    ; WRITE_MSG:
E6DB 2E8A04                             MOV     AL,CS:[SI]              ;PUT CHAR IN AL
E6DE 46                                 INC     SI                      ;POINT TO NEXT CHAR
E6DF B700                               MOV     BH,0                    ;SET PAGE # TO ZERO
E6E1 B40E                               MOV     AH,14                   ;WRITE CHAR (TTY-INTERFACE)
E6E3 CD10                               INT     10H                     ;CALL VIDEO_IO
E6E5 E2F4                               LOOP    G12                     ;CONTINUE TILL MSG WRITTEN
E6E7 B80D0E                             MOV     AX,0E0DH                ;POSITION CURSOR TO NEXT LINE
E6EA CD10                               INT     10H                     ;SEND CARRIAGE RETURN AND
```

```
LOC OBJ                 LINE    SOURCE

E6EC B80A0E                             MOV     AX,0E0AH                        ;LINE FEED CHARS
E6EF CD10                               INT     10H
E6F1 C3                                 RET
                                P_MSG   ENDP
                                ;--- INT 19 -----------------------------
                                ;BOOT STRAP LOADER
                                ;       IF A 5 1/4" DISKETTE DRIVE IS AVAILABLE
                                ;       ON THE SYSTEM, TRACK 0, SECTOR 1 IS READ INTO THE
                                ;       BOOT LOCATION (SEGMENT 0, OFFSET 7C00)
                                ;       AND CONTROL IS TRANSFERRED THERE.
                                ;
                                ;       IF THERE IS NO DISKETTE DRIVE, OR IF THERE IS
                                ;       IS A HARDWARE ERROR CONTROL IS TRANSFERRED
                                ;       TO THE CASSETTE BASIC ENTRY POINT.
                                ;
                                ; IPL ASSUMPTIONS
                                ;       8255 PORT 60H BIT 0
                                ;       = 1 IF IPL FROM DISKETTE
                                ;----------------------------------------
                                        ASSUME  CS:CODE,DS:DATA
E6F2                            BOOT_STRAP      PROC    NEAR

E6F2 FB                                 STI                     ; ENABLE INTERRUPTS
E6F3 B84000             R               MOV     AX,DATA         ; ESTABLISH ADDRESSING
E6F6 8ED8                               MOV     DS,AX
E6F8 A11000             R               MOV     AX,EQUIP_FLAG   ; GET THE EQUIPMENT SWITCHES
E6FB A801                               TEST    AL,1            ; ISOLATE IPL SENSE SWITCH
E6FD 7423                               JZ      H3              ; GO TO CASSETTE BASIC ENTRY POINT

                                ;------ MUST LOAD SYSTEM FROM DISKETTE -- CX HAS RETRY COUNT

E6FF B90400                             MOV     CX,4            ; SET RETRY COUNT
E702                            H1:                             ; IPL_SYSTEM
E702 51                                 PUSH    CX              ; SAVE RETRY COUNT
E703 B400                               MOV     AH,0            ; RESET THE DISKETTE SYSTEM
E705 CD13                               INT     13H             ; DISKETTE_IO
E707 7214                               JC      H2              ; IF ERROR, TRY AGAIN
E709 B402                               MOV     AH,2            ; READ IN THE SINGLE SECTOR
E70B BB0000                             MOV     BX,0            ; TO THE BOOT LOCATION
E70E 8EC3                               MOV     ES,BX
E710 BB007C                             MOV     BX,OFFSET BOOT_LOCN
E713 BA0000                             MOV     DX,0            ; DRIVE 0, HEAD 0
E716 B90100                             MOV     CX,1            ; SECTOR 1, TRACK 0
E719 B001                               MOV     AL,1            ; READ ONE SECTOR
E71B CD13                               INT     13H             ; DISKETTE_IO
E71D 59                         H2:     POP     CX              ; RECOVER RETRY COUNT
E71E 7304                               JNC     H4              ; CF SET BY UNSUCCESSFUL READ
E720 E2E0                               LOOP    H1              ; DO IT FOR RETRY TIMES

                                ;------ UNABLE TO IPL FROM THE DISKETTE

E722                            H3:                             ; CASSETTE_JUMP:
E722 CD18                               INT     18H             ; USE INTERRUPT VECTOR TO GET TO BASIC

                                ;------ IPL WAS SUCCESSFUL

E724                            H4:
E724 EA007C0000                         JMP     BOOT_LOCN
                                BOOT_STRAP      ENDP
                                ;-----INT 14--------------------------------
                                ;RS232_IO
                                ;       THIS ROUTINE PROVIDES BYTE STREAM I/O TO THE COMMUNICATIONS
                                ;       PORT ACCORDING TO THE PARAMETERS:
                                ;       (AH)=0  INITIALIZE THE COMMUNICATIONS PORT
                                ;               (AL) HAS PARMS FOR INITIALIZATION
                                ;
                                ;               7       6       5       4       3       2       1       0
                                ;               ----- BAUD RATE --      -PARITY--     STOPBIT  --WORD LENGTH--

                                ;               000 - 110               X0 - NONE       0 - 1   10 - 7 BITS
                                ;               001 - 150               01 - ODD        1 - 2   11 - 8 BITS
                                ;               010 - 300               11 - EVEN
                                ;               011 - 600
                                ;               100 - 1200
                                ;               101 - 2400
                                ;               110 - 4800
                                ;               111 - 9600
```

```
LOC OBJ                  LINE    SOURCE

                                 ;               ON RETURN, CONDITIONS SET AS IN CALL TO COMMO STATUS (AH=3)
                                 ;       (AH)=1  SEND THE CHARACTER IN (AL) OVER THE COMMO LINE
                                 ;               (AL) REGISTER IS PRESERVED
                                 ;               ON EXIT, BIT 7 OF AH IS SET IF THE ROUTINE WAS UNABLE TO
                                 ;                       TO TRANSMIT THE BYTE OF DATA OVER THE LINE.  THE
                                 ;                       REMAINDER OF AH IS SET AS IN A STATUS REQUEST,
                                 ;                       REFELECTING THE CURRENT STATUS OF THE LINE.
                                 ;       (AH)=2  RECEIVE A CHARACTER IN (AL) FROM COMMO LINE BEFORE
                                 ;                       RETURNING TO CALLER
                                 ;               ON EXIT, AH HAS THE CURRENT LINE STATUS, AS SET BY THE
                                 ;                       THE STATUS ROUTINE, EXCEPT THAT THE ONLY BITS
                                 ;                       LEFT ON ARE THE ERROR BITS (7,4,3,2,1)
                                 ;                       IN THIS CASE, THE TIME OUT BIT INDICATES DATA SET
                                 ;                       READY WAS NOT RECEIVED.
                                 ;                       THUS, AH IS NON ZERO ONLY WHEN AN ERROR OCCURRED.
                                 ;       (AH)=3  RETURN THE COMMO PORT STATUS IN (AX)
                                 ;               AH CONTAINS THE LINE CONTROL STATUS
                                 ;               BIT 7 = TIME OUT
                                 ;               BIT 6 = TRANS SHIFT REGISTER EMPTY
                                 ;               BIT 5 = TRAN HOLDING REGISTER EMPTY
                                 ;               BIT 4 = BREAK DETECT
                                 ;               BIT 3 = FRAMING ERROR
                                 ;               BIT 2 = PARITY ERROR
                                 ;               BIT 1 = OVERRUN ERROR
                                 ;               BIT 0 = DATA READY
                                 ;               AL CONTAINS THE MODEM STATUS
                                 ;               BIT 7 = RECEVED LINE SIGNAL DETECT
                                 ;               BIT 6 = RING INDICATOR
                                 ;               BIT 5 = DATA SET READY
                                 ;               BIT 4 = CLEAR TO SEND
                                 ;               BIT 3 = DELTA RECEIVE LINE SIGNAL DETECT
                                 ;               BIT 2 = TRAILING EDGE RING DETECTOR
                                 ;               BIT 1 = DELTA DATA SET READY
                                 ;               BIT 0 = DELTA CLEAR TO SEND

                                 ;       (DX) = PARAMETER INDICATING WHICH RS232 CARD (0,1 ALLOWED)
                                 ; DATA AREA RS232_BASE CONTAINS THE BASE ADDRESS OF THE 8250 ON THE CARD
                                 ;       LOCATION 400H CONTAINS UP TO 4 RS232 ADDRESSES POSSIBLE
                                 ;OUTPUT
                                 ;               AX      MODIFIED ACCORDING TO PARMS OF CALL
                                 ;               ALL OTHERS UNCHANGED
                                 ;------------------------------------------
                                         ASSUME  CS:CODE,DS:DATA
E729                             A1      LABEL   WORD
E729 1704                                        DW      1047    ; 110 BAUD      ; TABLE OF INIT VALUE
E72B 0003                                        DW      768     ; 150
E72D 8001                                        DW      384     ; 300
E72F C000                                        DW      192     ; 600
E731 6000                                        DW      96      ; 1200
E733 3000                                        DW      48      ; 2400
E735 1800                                        DW      24      ; 4800
E737 0C00                                        DW      12      ; 9600

E739                             RS232_IO        PROC    FAR

                                 ;------ VECTOR TO APPROPRIATE ROUTINE

E739 FB                                  STI                     ; INTERRUPTS BACK ON
E73A 1E                                  PUSH    DS              ; SAVE SEGMENT
E73B 52                                  PUSH    DX
E73C 56                                  PUSH    SI
E73D 57                                  PUSH    DI
E73E 51                                  PUSH    CX
E73F 8BF2                                MOV     SI,DX           ; RS232 VALUE TO SI
E741 D1E6                                SHL     SI,1            ; WORD OFFSET
E743 BA4000           R                  MOV     DX,DATA
E746 8EDA                                MOV     DS,DX           ; SET UP OUR SEGMENT
E748 8B940000         R                  MOV     DX,RS232_BASE[SI]       ; GET BASE ADDRESS
E74C 0BD2                                OR      DX,DX           ; TEST FOR 0 BASE ADDRESS
E74E 7416                                JZ      A3              ; RETURN
E750 0AE4                                OR      AH,AH           ; TEST FOR (AH)=0
E752 7418                                JZ      A4              ; COMMUN INIT
E754 FECC                                DEC     AH              ; TEST FOR (AH)=1
E756 744E                                JZ      A5              ; SEND AL
E758 FECC                                DEC     AH              ; TEST FOR (AH)=2
E75A 7503                                JNZ     A2
E75C E98900                              JMP     A12             ; RECEIVE INTO AL
```

```
LOC OBJ                 SOURCE

E75F                    A2:
E75F FECC                       DEC     AH              ; TEST FOR (AH)=3
E761 7503                       JNZ     A3
E763 E9B900                     JMP     A18             ; COMMUNICATION STATUS
E766                    A3:                             ; RETURN FROM RS232
E766 59                         POP     CX
E767 5F                         POP     DI
E768 5E                         POP     SI
E769 5A                         POP     DX
E76A 1F                         POP     DS
E76B CF                         IRET                    ; RETURN TO CALLER, NO ACTION

                        ;------ INITIALIZE THE COMMUNICATIONS PORT

E76C                    A4:
E76C 8AE0                       MOV     AH,AL           ; SAVE INIT PARMS IN AH
E76E 83C203                     ADD     DX,3            ; POINT TO 8250 CONTROL REGISTER
E771 B080                       MOV     AL,80H
E773 EE                         OUT     DX,AL           ; SET DLAB=1

                        ;------ DETERMINE BAUD RATE DIVISOR

E774 8AD4                       MOV     DL,AH           ; GET PARMS TO DL
E776 D0C2                       ROL     DL,1
E778 D0C2                       ROL     DL,1            ; GET BAUD RATE TERM TO LOW BITS
E77A D0C2                       ROL     DL,1
E77C D0C2                       ROL     DL,1            ; *2 FOR WORD TABLE ACCESS
E77E 81E20E00                   AND     DX,0EH          ; ISOLATE THEM
E782 BF29E7          R          MOV     DI,OFFSET A1    ; BASE OF TABLE
E785 03FA                       ADD     DI,DX           ; PUT INTO INDEX REGISTER
E787 8B940000        R          MOV     DX,RS232_BASE[SI]       ; POINT TO HIGH ORDER OF DIVISOR
E78B 42                         INC     DX
E78C 2E8A4501                   MOV     AL,CS:[DI]+1    ; GET HIGH ORDER OF DIVISOR
E790 EE                         OUT     DX,AL           ; SET MS OF DIV TO 0
E791 4A                         DEC     DX
E792 2E8A05                     MOV     AL,CS:[DI]      ; GET LOW ORDER OF DIVISOR
E795 EE                         OUT     DX,AL           ; SET LOW OF DIVISOR
E796 83C203                     ADD     DX,3
E799 8AC4                       MOV     AL,AH           ; GET PARMS BACK
E79B 241F                       AND     AL,01FH         ; STRIP OFF THE BAUD BITS
E79D EE                         OUT     DX,AL           ; LINE CONTROL TO 8 BITS
E79E 83EA02                     SUB     DX,2
E7A1 B000                       MOV     AL,0
E7A3 EE                         OUT     DX,AL           ; INTERRUPT ENABLES ALL OFF
E7A4 EB79                       JMP     SHORT A18       ; COM_STATUS

                        ;------ SEND CHARACTER IN (AL) OVER COMMO LINE

E7A6                    A5:
E7A6 50                         PUSH    AX              ; SAVE CHAR TO SEND
E7A7 83C204                     ADD     DX,4            ; MODEM CONTROL REGISTER
E7AA B003                       MOV     AL,3            ; DTR AND RTS
E7AC EE                         OUT     DX,AL           ; DATA TERMINAL READY, REQUEST TO SEND
E7AD 33C9                       XOR     CX,CX           ; INITIALIZE TIME OUT COUNT
E7AF 83C202                     ADD     DX,2            ; MODEM STATUS REGISTER
E7B2                    A6:                             ; WAIT_DATA_SET_READY
E7B2 EC                         IN      AL,DX           ; GET MODEM STATUS
E7B3 A820                       TEST    AL,20H          ; DATA SET READY
E7B5 7508                       JNZ     A7              ; TEST_CLEAR_TO_SEND
E7B7 E2F9                       LOOP    A6              ; WAIT_DATA_SET_READY
E7B9 58                         POP     AX
E7BA 80CC50                     OR      AH,80           ; INDICATE TIME OUT
E7BD EBA7                       JMP     A3              ; RETURN
E7BF                    A7:                             ; TEST_CLEAR_TO_SEND
E7BF 2BC9                       SUB     CX,CX
E7C1                    A8:                             ; WAIT_CLEAR_TO_SEND
E7C1 EC                         IN      AL,DX           ; GET MODEM STATUS
E7C2 A810                       TEST    AL,10H          ; TEST CLEAR TO SEND
E7C4 7508                       JNZ     A9              ; CLEAR_TO_SEND
E7C6 E2F9                       LOOP    A8              ; WAIT_CLEAR_TO_SEND
E7C8 58                         POP     AX              ; TIME OUT HAS OCCURRED
E7C9 80CC80                     OR      AH,80H
E7CC EB98                       JMP     A3              ; RETURN
E7CE                    A9:                             ; CLEAR_TO_SEND
E7CE 4A                         DEC     DX              ; LINE STATUS REGISTER
E7CF 2BC9                       SUB     CX,CX           ; INITIALIZE WAIT COUNT
E7D1                    A10:                            ; WAIT_SEND
```

```
LOC OBJ                  SOURCE

E7D1 EC                         IN    AL,DX           ; GET STATUS
E7D2 A820                       TEST  AL,20H          ; IS TRANSMITTER READY
E7D4 7508                       JNZ   A11             ; OUT_CHAR
E7D6 E2F9                       LOOP  A10             ; GO BACK FOR MORE, AND TEST FOR TIME OUT
E7D8 58                         POP   AX              ; RECOVER ORIGINAL INPUT
E7D9 80CC80                     OR    AH,80H          ; SET THE TIME OUT BIT
E7DC EB88                       JMP   A3              ; RETURN
E7DE                     A11:                         ; OUT_CHAR
E7DE 83EA05                     SUB   DX,5            ; DATA PORT
E7E1 59                         POP   CX              ; RECOVER IN CX TEMPORARILY
E7E2 8AC1                       MOV   AL,CL           ; GET OUT CHAR TO AL FOR OUT, STATUS IN AH
E7E4 EE                         OUT   DX,AL           ; OUTPUT CHARACTER
E7E5 E97EFF                     JMP   A3              ; RETURN

                         ;------ RECEIVE CHARACTER FROM COMMO LINE

E7E8                     A12:
E7E8 802671007F      R          AND   BIOS_BREAK,07FH         ; TURN OFF BREAK BIT IN BYTE
E7ED 83C204                     ADD   DX,4            ; MODEM CONTROL REGISTER
E7F0 B001                       MOV   AL,1            ; DATA TERMINAL READY
E7F2 EE                         OUT   DX,AL
E7F3 83C202                     ADD   DX,2            ; MODEM STATUS REGISTER
E7F6 2BC9                       SUB   CX,CX           ; ESTABLISH TIME OUT COUNT
E7F8                     A13:                         ; WAIT_DSR
E7F8 EC                         IN    AL,DX           ; MODEM STATUS
E7F9 A820                       TEST  AL,20H          ; DATA SET READY
E7FB 7507                       JNZ   A15             ; IS IT READY YET
E7FD E2F9                       LOOP  A13             ; WAIT UNTIL IT IS
E7FF                     A14:                         ; TIME_OUT_ERR
E7FF B480                       MOV   AH,80H          ; SET TIME OUT ERROR
E801 E962FF                     JMP   A3              ; RETURN WITH ERROR
E804                     A15:                         ; WAIT_DSR_END
E804 4A                         DEC   DX              ; LINE STATUS REGISTER
E805                     A16:                         ; WAIT_RECV
E805 EC                         IN    AL,DX           ; GET STATUS
E806 A801                       TEST  AL,1            ; RECEIVE BUFFER FULL
E808 7509                       JNZ   A17             ; GET CHAR
E80A F606710080      R          TEST  BIOS_BREAK,80H  ; TEST FOR BREAK KEY
E80F 74F4                       JZ    A16             ; LOOP IF NOT
E811 EBEC                       JMP   A14             ; SET TIME OUT ERROR
E813                     A17:                         ; GET_CHAR
E813 241E                       AND   AL,00011110B    ; TEST FOR ERROR CONDITIONS ON RECV CHAR
E815 8AE0                       MOV   AH,AL           ; SAVE THIS PART OF STATUS FOR LATER OPERATION
E817 8B940000        R          MOV   DX,RS232_BASE[SI]       ; DATA PORT
E81B EC                         IN    AL,DX           ; GET CHARACTER FROM LINE
E81C E947FF                     JMP   A3              ; RETURN

                         ;------ COMMO PORT STATUS ROUTINE

E81F                     A18:
E81F 8B940000        R          MOV   DX,RS232_BASE[SI]
E823 83C205                     ADD   DX,5            ; CONTROL PORT
E826 EC                         IN    AL,DX           ; GET LINE CONTROL STATUS
E827 8AE0                       MOV   AH,AL           ; PUT IN AH FOR RETURN
E829 42                         INC   DX              ; POINT TO MODEM STATUS REGISTER
E82A EC                         IN    AL,DX           ; GET MODEM CONTROL STATUS
E82B E938FF                     JMP   A3              ; RETURN
                         RS232_IO        ENDP
                         ;---- INT 16 --------------------------------
                         ; KEYBOARD I/O
                         ;       THESE ROUTINES PROVIDE KEYBOARD SUPPORT
                         ; INPUT
                         ;       (AH)=0  READ THE NEXT ASCII CHARACTER STRUCK FROM THE KEYBOARD
                         ;               RETURN THE RESULT IN (AL), SCAN CODE IN (AH)
                         ;       (AH)=1  SET THE Z FLAG TO INDICATE IF AN ASCII CHARACTER IS AVAILABLE
                         ;               TO BE READ.
                         ;               (ZF)=1 -- NO CODE AVAILABLE
                         ;               (ZF)=0 -- CODE IS AVAILABLE
                         ;               IF ZF = 0, THE NEXT CHARACTER IN THE BUFFER TO BE READ IS
                         ;               IN AX, AND THE ENTRY REMAINS IN THE BUFFER
                         ;       (AH)=2  RETURN THE CURRENT SHIFT STATUS IN AL REGISTER
                         ;               THE BIT SETTINGS FOR THIS CODE ARE INDICATED IN THE
                         ;               THE EQUATES FOR KB_FLAG
                         ; OUTPUT
                         ;       AS NOTED ABOVE, ONLY AX AND FLAGS CHANGED
                         ;       ALL REGISTERS RETAINED
                         ;----------------------------------------
```

```
LOC OBJ                 SOURCE

                                ASSUME  CS:CODE,DS:DATA
E82E                    KEYBOARD_IO     PROC    FAR
E82E FB                         STI                     ; INTERRUPTS BACK ON
E82F 1E                         PUSH    DS              ; SAVE CURRENT DS
E830 53                         PUSH    BX              ; SAVE BX TEMPORARILY
E831 BB4000          R          MOV     BX,DATA         ;
E834 8EDB                       MOV     DS,BX           ; ESTABLISH POINTER TO DATA REGION
E836 0AE4                       OR      AH,AH           ; AH=0
E838 740B                       JZ      K1              ; ASCII_READ
E83A FECC                       DEC     AH              ; AH=1
E83C 7420                       JZ      K2              ; ASCII_STATUS
E83E FECC                       DEC     AH              ; AH=2
E840 742D                       JZ      K3              ; SHIFT_STATUS
E842 5B                         POP     BX              ; RECOVER REGISTER
E843 1F                         POP     DS
E844 CF                         IRET                    ; INVALID COMMAND

                        ;------ READ THE KEY TO FIGURE OUT WHAT TO DO

E845                    K1:                             ; ASCII READ
E845 FB                         STI                     ; INTERRUPTS BACK ON DURING LOOP
E846 90                         NOP                     ; ALLOW AN INTERRUPT TO OCCUR
E847 FA                         CLI                     ; INTERRUPTS BACK OFF
E848 8B1E1A00        R          MOV     BX,BUFFER_HEAD  ; GET POINTER TO HEAD OF BUFFER
E84C 3B1E1C00        R          CMP     BX,BUFFER_TAIL  ; TEST END OF BUFFER
E850 74F3                       JZ      K1              ; LOOP UNTIL SOMETHING IN BUFFER
E852 8B07                       MOV     AX,[BX]         ; GET SCAN CODE AND ASCII CODE
E854 E81E00                     CALL    K4              ; MOVE POINTER TO NEXT POSITION
E857 891E1A00        R          MOV     BUFFER_HEAD,BX  ; STORE VALUE IN VARIABLE
E85B 5B                         POP     BX              ; RECOVER REGISTER
E85C 1F                         POP     DS              ; RECOVER SEGMENT
E85D CF                         IRET                    ; RETURN TO CALLER

                        ;------ ASCII STATUS

E85E                    K2:
E85E FA                         CLI                     ; INTERRUPTS OFF
E85F 8B1E1A00        R          MOV     BX,BUFFER_HEAD  ; GET HEAD POINTER
E863 3B1E1C00        R          CMP     BX,BUFFER_TAIL  ; IF EQUAL (Z=1) THEN NOTHING THERE
E867 8B07                       MOV     AX,[BX]
E869 FB                         STI                     ; INTERRUPTS BACK ON
E86A 5B                         POP     BX              ; RECOVER REGISTER
E86B 1F                         POP     DS              ; RECOVER SEGMENT
E86C CA0200                     RET     2               ; THROW AWAY FLAGS

                        ;------ SHIFT STATUS

E86F                    K3:
E86F A01700          R          MOV     AL,KB_FLAG      ; GET THE SHIFT STATUS FLAGS
E872 5B                         POP     BX              ; RECOVER REGISTER
E873 1F                         POP     DS              ; RECOVER REGISTERS
E874 CF                         IRET                    ; RETURN TO CALLER
                        KEYBOARD_IO     ENDP

                        ;------ INCREMENT A BUFFER POINTER

E875                    K4      PROC    NEAR
E875 83C302                     ADD     BX,2                    ; MOVE TO NEXT WORD IN LIST
E878 81FB3E00        R          CMP     BX,OFFSET KB_BUFFER_END ; AT END OF BUFFER?
E87C 7503                       JNE     K5                      ; NO, CONTINUE
E87E BB1E00          R          MOV     BX,OFFSET KB_BUFFER     ; YES, RESET TO BUFFER BEGINNING
E881                    K5:
E881 C3                         RET
                        K4      ENDP

                        ;------ TABLE OF SHIFT KEYS AND MASK VALUES

E882                    K6      LABEL   BYTE
E882 52                         DB      INS_KEY                 ; INSERT KEY
E883 3A4546381D                 DB      CAPS_KEY,NUM_KEY,SCROLL_KEY,ALT_KEY,CTL_KEY
E888 2A36                       DB      LEFT_KEY,RIGHT_KEY
  0008                  K6L     EQU     $-K6

                        ;------ SHIFT_MASK_TABLE

E88A                    K7      LABEL   BYTE
E88A 80                         DB      INS_SHIFT               ; INSERT MODE SHIFT
```

```
LOC OBJ                 LINE     SOURCE

E88B 4020100804                           DB       CAPS_SHIFT,NUM_SHIFT,SCROLL_SHIFT,ALT_SHIFT,CTL_SHIFT
E890 0201                                 DB       LEFT_SHIFT,RIGHT_SHIFT

                                 ;------ SCAN CODE TABLES

E892 1BFF00FFFFFF1EFF            K8                DB       27,-1,0,-1,-1,-1,30,-1
E89A FFFFFF1FFF7FFF11                              DB       -1,-1,-1,31,-1,127,-1,17
E8A2 170512141915090F                              DB       23,5,18,20,25,21,9,15
E8AA 101B1D0AFF0113                                DB       16,27,29,10,-1,1,19
E8B1 040607080A0B0CFFFF                            DB       4,6,7,8,10,11,12,-1,-1
E8BA FFFF1C1A18031602                              DB       -1,-1,28,26,24,3,22,2
E8C2 0E0DFFFFFFFFFFFF                              DB       14,13,-1,-1,-1,-1,-1,-1
E8CA 20FF                                          DB       ' ',-1
                                 ;-------- CTL TABLE SCAN
E8CC                             K9       LABEL    BYTE
E8CC 5E5F606162636465                              DB       94,95,96,97,98,99,100,101
E8D4 6667FFFF77FF84FF                              DB       102,103,-1,-1,119,-1,132,-1
E8DC 73FF74FF75FF76FF                              DB       115,-1,116,-1,117,-1,118,-1
E8E4 FF                                            DB       -1
                                 ;------- LC TABLE
E8E5                             K10      LABEL    BYTE
E8E5 1B                                            DB       01BH,'1234567890-=',08H,09H
E8E6 31323334353637
     3839302D3D0809
E8F4 71776572747975                                DB       'qwertyuiop[]',0DH,-1,'asdfghjkl;',027H
     696F705B5D0DFF
E902 6173646667686A
     6B6C3B27
E90D 60FF5C                                        DB       60H,-1,5CH,'zxcvbnm,./',-1,'*',-1,' '
E910 7A786376626E6D
     2C2E2FFF2AFF20
E91E FF                                            DB       -1

                                 ;------ UC TABLE
E91F                             K11      LABEL    BYTE
E91F 1B                                            DB       27,'!@#$',37,05EH,'&*()_+',08H,0
E920 21402324255E
E926 262A28295F2B0800
E92E 51574552545955                                DB       'QWERTYUIOP  ',0DH,-1,'ASDFGHJKL:"'
     494F507B7D0DFF
E93C 4153444647484A
     4B4C3A22
E947 7EFF                                          DB       07EH,-1,' ZXCVBNM<>?',-1,0,-1,' ',-1
E949 7C5A584356424E
     4D3C3E3FFF00FF20FF
                                 ;------ UC TABLE SCAN
E959                             K12      LABEL    BYTE
E959 5455565758595A                                DB       84,85,86,87,88,89,90
E960 5B5C5D                                        DB       91,92,93
                                 ;------ ALT TABLE SCAN
E963                             K13      LABEL    BYTE
E963 68696A6B6C                                    DB       104,105,106,107,108
E968 6D6E6F7071                                    DB       109,110,111,112,113

                                 ;------ NUM STATE TABLE
E96D                             K14      LABEL    BYTE
E96D 3738392D343536                                DB       '789-456+1230.'
     2B313233302E
                                 ;------ BASE CASE TABLE
E97A                             K15      LABEL    BYTE
E97A 474849FF4BFF4D                                DB       71,72,73,-1,75,-1,77
E981 FF4F50515253                                  DB       -1,79,80,81,82,83

                                 ;------ KEYBOARD INTERRUPT ROUTINE

E987                             KB_INT   PROC     FAR
E987 FB                                   STI                        ; ALLOW FURTHER INTERRUPTS
E988 50                                   PUSH     AX
E989 53                                   PUSH     BX
E98A 51                                   PUSH     CX
E98B 52                                   PUSH     DX
E98C 56                                   PUSH     SI
E98D 57                                   PUSH     DI
E98E 1E                                   PUSH     DS
E98F 06                                   PUSH     ES
E990 FC                                   CLD                        ; FORWARD DIRECTION
E991 B84000             R                 MOV      AX,DATA
```

```
LOC OBJ                          SOURCE

E994 8ED8                                 MOV     DS,AX           ; SET UP ADDRESSING
E996 E460                                 IN      AL,KB_DATA      ; READ IN THE CHARACTER
E998 50                                   PUSH    AX              ; SAVE IT
E999 E461                                 IN      AL,KB_CTL       ; GET THE CONTROL PORT
E99B 8AE0                                 MOV     AH,AL           ; SAVE VALUE
E99D 0C80                                 OR      AL,80H          ; RESET BIT FOR KEYBOARD
E99F E661                                 OUT     KB_CTL,AL
E9A1 86E0                                 XCHG    AH,AL           ; GET BACK ORIGINAL CONTROL
E9A3 E661                                 OUT     KB_CTL,AL       ; KB HAS BEEN RESET
E9A5 58                                   POP     AX              ; RECOVER SCAN CODE
E9A6 8AE0                                 MOV     AH,AL           ; SAVE SCAN CODE IN AH ALSO

                                  ;------ TEST FOR OVERRUN SCAN CODE FROM KEYBOARD

E9A8 3CFF                                 CMP     AL,0FFH         ; IS THIS AN OVERRUN CHAR
E9AA 7503                                 JNZ     K16             ; NO, TEST FOR SHIFT KEY
E9AC E97502                               JMP     K62             ; BUFFER_FULL_BEEP

                                  ;------ TEST FOR SHIFT KEYS

E9AF                              K16:                            ; TEST_SHIFT
E9AF 247F                                 AND     AL,07FH         ; TURN OFF THE BREAK BIT
E9B1 0E                                   PUSH    CS
E9B2 07                                   POP     ES              ; ESTABLISH ADDRESS OF SHIFT TABLE
E9B3 BF82E8          R                    MOV     DI,OFFSET K6    ; SHIFT KEY TABLE
E9B6 B90800                               MOV     CX,K6L          ; LENGTH
E9B9 F2                                   REPNE   SCASB           ; LOOK THROUGH THE TABLE FOR A MATCH
E9BA AE
E9BB 8AC4                                 MOV     AL,AH           ; RECOVER SCAN CODE
E9BD 7403                                 JE      K17             ; JUMP IF MATCH FOUND
E9BF E98800                               JMP     K25             ; IF NO MATCH, THEN SHIFT NOT FOUND

                                  ;------ SHIFT KEY FOUND

E9C2 81EF83E8        R            K17:    SUB     DI,OFFSET K6+1  ; ADJUST PTR TO SCAN CODE MTCH
E9C6 2E8AA58AE8      R                    MOV     AH,CS:K7[DI]    ; GET MASK INTO AH
E9CB A880                                 TEST    AL,80H          ; TEST FOR BREAK KEY
E9CD 7554                                 JNZ     K23             ; BREAK_SHIFT_FOUND

                                  ;------ SHIFT MAKE FOUND, DETERMINE SET OR TOGGLE

E9CF 80FC10                               CMP     AH,SCROLL_SHIFT
E9D2 7307                                 JAE     K18                     ; IF SCROLL SHIFT OR ABOVE, TOGGLE KEY

                                  ;------ PLAIN SHIFT KEY, SET SHIFT ON

E9D4 08261700        R                    OR      KB_FLAG,AH              ; TURN ON SHIFT BIT
E9D8 E98300                               JMP     K26                     ; INTERRUPT_RETURN

                                  ;------ TOGGLED SHIFT KEY, TEST FOR 1ST MAKE OR NOT

E9DB                              K18:                                    ; SHIFT-TOGGLE
E9DB F606170004      R                    TEST    KB_FLAG, CTL_SHIFT      ; CHECK CTL SHIFT STATE
E9E0 7568                                 JNZ     K25                     ; JUMP IF CTL STATE
E9E2 3C52                                 CMP     AL, INS_KEY             ; CHECK FOR INSERT KEY
E9E4 7525                                 JNZ     K22                     ; JUMP IF NOT INSERT KEY
E9E6 F606170008      R                    TEST    KB_FLAG, ALT_SHIFT      ; CHECK FOR ALTERNATE SHIFT
E9EB 7403                                 JZ      K19                     ; JUMP IF NOT ALTERNATE SHIFT
E9ED EB5B90                               JMP     K25                     ; JUMP IF ALTERNATE SHIFT
E9F0 F606170020      R            K19:    TEST    KB_FLAG, NUM_STATE      ; CHECK FOR BASE STATE
E9F5 750D                                 JNZ     K21                     ; JUMP IF NUM LOCK IS ON
E9F7 F606170003      R                    TEST    KB_FLAG, LEFT_SHIFT+ RIGHT_SHIFT    ;
E9FC 740D                                 JZ      K22                     ; JUMP IF BASE STATE

E9FE                              K20:                                    ; NUMERIC ZERO, NOT INSERT KEY
E9FE B83052                               MOV     AX, 5230H               ; PUT OUT AN ASCII ZERO
EA01 E9D801                               JMP     K57                     ; BUFFER_FILL
EA04                              K21:                                    ; MIGHT BE NUMERIC
EA04 F606170003      R                    TEST    KB_FLAG, LEFT_SHIFT+ RIGHT_SHIFT    ;
EA09 74F3                                 JZ      K20                     ; JUMP NUMERIC, NOT INSERT

EA0B                              K22:                                    ; SHIFT TOGGLE KEY HIT; PROCESS IT
EA0B 84261800        R                    TEST    AH,KB_FLAG_1            ; IS KEY ALREADY DEPRESSED
EA0F 754D                                 JNZ     K26                     ; JUMP IF KEY ALREADY DEPRESSED
EA11 08261800        R                    OR      KB_FLAG_1,AH            ; INDICATE THAT THE KEY IS DEPRESSED
EA15 30261700        R                    XOR     KB_FLAG,AH              ; TOGGLE THE SHIFT STATE
EA19 3C52                                 CMP     AL,INS_KEY              ; TEST FOR 1ST MAKE OF INSERT KEY
EA1B 7541                                 JNE     K26                     ; JUMP IF NOT INSERT KEY
```

```
LOC OBJ                    SOURCE

EA1D B80052                        MOV   AX,INS_KEY*256         ; SET SCAN CODE INTO AH, 0 INTO AL
EA20 E9B901                        JMP   K57                    ; PUT INTO OUTPUT BUFFER
                           ;------ BREAK SHIFT FOUND
EA23                       K23:                                 ; BREAK-SHIFT-FOUND
EA23 80FC10                        CMP   AH,SCROLL_SHIFT        ; IS THIS A TOGGLE KEY
EA26 731A                          JAE   K24                    ; YES, HANDLE BREAK TOGGLE
EA28 F6D4                          NOT   AH                     ; INVERT MASK
EA2A 20261700         R            AND   KB_FLAG,AH             ; TURN OFF SHIFT BIT
EA2E 3CB8                          CMP   AL,ALT_KEY+80H         ; IS THIS ALTERNATE SHIFT RELEASE
EA30 752C                          JNE   K26                    ; INTERRUPT_RETURN
                           ;------ ALTERNATE SHIFT KEY RELEASED, GET THE VALUE INTO BUFFER
EA32 A01900           R            MOV   AL,ALT_INPUT
EA35 B400                          MOV   AH,0                   ; SCAN CODE OF 0
EA37 88261900         R            MOV   ALT_INPUT,AH           ; ZERO OUT THE FIELD
EA3B 3C00                          CMP   AL,0                   ; WAS THE INPUT=0
EA3D 741F                          JE    K26                    ; INTERRUPT_RETURN
EA3F E9A301                        JMP   K58                    ; IT WASN'T, SO PUT IN BUFFER
EA42                       K24:                                 ; BREAK-TOGGLE
EA42 F6D4                          NOT   AH                     ; INVERT MASK
EA44 20261800         R            AND   KB_FLAG_1,AH           ; INDICATE NO LONGER DEPRESSED
EA48 EB14                          JMP   SHORT K26              ; INTERRUPT_RETURN
                           ;------ TEST FOR HOLD STATE
EA4A                       K25:                                 ; NO-SHIFT-FOUND
EA4A 3C80                          CMP   AL,80H                 ; TEST FOR BREAK KEY
EA4C 7310                          JAE   K26                    ; NOTHING FOR BREAK CHARS FROM HERE ON
EA4E F606180008       R            TEST  KB_FLAG_1,HOLD_STATE   ; ARE WE IN HOLD STATE
EA53 7417                          JZ    K28                    ; BRANCH AROUND TEST IF NOT
EA55 3C45                          CMP   AL,NUM_KEY
EA57 7405                          JE    K26                    ; CAN'T END HOLD ON NUM_LOCK
EA59 80261800F7       R            AND   KB_FLAG_1,NOT HOLD_STATE       ; TURN OFF THE HOLD STATE BIT
EA5E                       K26:                                 ; INTERRUPT-RETURN
EA5E FA                            CLI                  ; TURN OFF INTERRUPTS
EA5F B020                          MOV   AL,EOI         ; END OF INTERRUPT COMMAND
EA61 E620                          OUT   020H,AL        ; SEND COMMAND TO INTERRUPT CONTROL PORT
EA63                       K27:                         ; INTERRUPT-RETURN-NO-EOI
EA63 07                            POP   ES
EA64 1F                            POP   DS
EA65 5F                            POP   DI
EA66 5E                            .POP  SI
EA67 5A                            POP   DX
EA68 59                            POP   CX
EA69 5B                            POP   BX
EA6A 58                            POP   AX             ; RESTORE STATE
EA6B CF                            IRET                 ; RETURN, INTERRUPTS BACK ON WITH FLAG CHANGE
                           ;------ NOT IN  HOLD STATE, TEST FOR SPECIAL CHARS
EA6C                       K28:                         ; NO-HOLD-STATE
EA6C F606170008       R            TEST  KB_FLAG,ALT_SHIFT      ; ARE WE IN ALTERNATE SHIFT
EA71 7503                          JNZ   K29                    ; JUMP IF ALTERNATE SHIFT
EA73 E98F00                        JMP   K38                    ; JUMP IF NOT ALTERNATE
                           ;------ TEST FOR RESET KEY SEQUENCE (CTL ALT DEL)
EA76                       K29:                                 ; TEST-RESET
EA76 F606170004       R            TEST  KB_FLAG,CTL_SHIFT      ; ARE WE IN CONTROL SHIFT ALSO
EA7B 7431                          JZ    K31                    ; NO_RESET
EA7D 3C53                          CMP   AL,DEL_KEY             ; SHIFT STATE IS THERE, TEST KEY
EA7F 752D                          JNE   K31                    ; NO_RESET
                           ;------ CTL-ALT-DEL HAS BEEN FOUND, DO I/O CLEANUP
EA81 C70672003412     R            MOV   RESET_FLAG, 1234H      ; SET FLAG FOR RESET FUNCTION
EA87 E9D1F5                        JMP   RESET                  ; JUMP TO POWER ON DIAGNOSTICS
                           ;------ ALT-INPUT-TABLE
EA8A                       K30     LABEL BYTE
EA8A 524F50514B4C4D                DB    82,79,80,81,75,76,77
```

```
LOC  OBJ                    SOURCE

EA91 474849                        DB      71,72,73              ; 10 NUMBERS ON KEYPAD
                            ;------ SUPER-SHIFT-TABLE
EA94 1011121314151617              DB      16,17,18,19,20,21,22,23 ; A-Z TYPEWRITER CHARS
EA9C 18191E1F20212223              DB      24,25,30,31,32,33,34,35
EAA4 2425262C2D2E2F30              DB      36,37,38,44,45,46,47,48
EAAC 3132                          DB      49,50

                            ;------ IN ALTERNATE SHIFT, RESET NOT FOUND

EAAE                        K31:                                 ; NO-RESET
EAAE 3C39                          CMP     AL,57                 ; TEST FOR SPACE KEY
EAB0 7505                          JNE     K32                   ; NOT THERE
EAB2 B020                          MOV     AL,' '                ; SET SPACE CHAR
EAB4 E92501                        JMP     K57                   ; BUFFER_FILL

                            ;------ LOOK FOR KEY PAD ENTRY

EAB7                        K32:                                 ; ALT-KEY-PAD
EAB7 BF8AEA          R             MOV     DI,OFFSET K30         ; ALT-INPUT-TABLE
EABA B90A00                        MOV     CX,10                 ; LOOK FOR ENTRY USING KEYPAD
EABD F2                            REPNE   SCASB                 ; LOOK FOR MATCH
EABE AE
EABF 7512                          JNE     K33                   ; NO_ALT_KEYPAD
EAC1 81EF8BEA        R             SUB     DI,OFFSET K30+1       ; DI NOW HAS ENTRY VALUE
EAC5 A01900          R             MOV     AL,ALT_INPUT          ; GET THE CURRENT BYTE
EAC8 B40A                          MOV     AH,10                 ; MULTIPLY BY 10
EACA F6E4                          MUL     AH
EACC 03C7                          ADD     AX,DI                 ; ADD IN THE LATEST ENTRY
EACE A21900          R             MOV     ALT_INPUT,AL          ; STORE IT AWAY
EAD1 EB8B                          JMP     K26                   ; THROW AWAY THAT KEYSTROKE

                            ;------ LOOK FOR SUPERSHIFT ENTRY

EAD3                        K33:                                 ; NO-ALT-KEYPAD
EAD3 C606190000      R             MOV     ALT_INPUT,0           ; ZERO ANY PREVIOUS ENTRY INTO INPUT
EAD8 B91A00                        MOV     CX,26                 ; DI,ES ALREADY POINTING
EADB F2                            REPNE   SCASB                 ; LOOK FOR MATCH IN ALPHABET
EADC AE
EADD 7505                          JNE     K34                   ; NOT FOUND, FUNCTION KEY OR OTHER
EADF B000                          MOV     AL,0                  ; ASCII CODE OF ZERO
EAE1 E9F800                        JMP     K57                   ; PUT IT IN THE BUFFER

                            ;------ LOOK FOR TOP ROW OF ALTERNATE SHIFT

EAE4                        K34:                                 ; ALT-TOP-ROW
EAE4 3C02                          CMP     AL,2                  ; KEY WITH '1' ON IT
EAE6 720C                          JB      K35                   ; NOT ONE OF INTERESTING KEYS
EAE8 3C0E                          CMP     AL,14                 ; IS IT IN THE REGION
EAEA 7308                          JAE     K35                   ; ALT-FUNCTION
EAEC 80C476                        ADD     AH,118                ; CONVERT PSUEDO SCAN CODE TO RANGE
EAEF B000                          MOV     AL,0                  ; INDICATE AS SUCH
EAF1 E9E800                        JMP     K57                   ; BUFFER_FILL

                            ;------ TRANSLATE ALTERNATE SHIFT PSEUDO SCAN CODES

EAF4                        K35:                                 ; ALT-FUNCTION
EAF4 3C3B                          CMP     AL,59                 ; TEST FOR IN TABLE
EAF6 7303                          JAE     K37                   ; ALT-CONTINUE
EAF8                        K36:                                 ; CLOSE-RETURN
EAF8 E963FF                        JMP     K26                   ; IGNORE THE KEY
EAFB                        K37:                                 ; ALT-CONTINUE
EAFB 3C47                          CMP     AL,71                 ; IN KEYPAD REGION
EAFD 73F9                          JAE     K36                   ; IF SO, IGNORE
EAFF BB63E9          R             MOV     BX,OFFSET K13         ; ALT SHIFT PSEUDO SCAN TABLE
EB02 E92501                        JMP     K63                   ; TRANSLATE THAT

                            ;------ NOT IN ALTERNATE SHIFT

EB05                        K38:                                 ; NOT-ALT-SHIFT
EB05 F606170004      R             TEST    KB_FLAG,CTL_SHIFT     ; ARE WE IN CONTROL SHIFT
EB0A 745B                          JZ      K44                   ; NOT-CTL-SHIFT

                            ;------ CONTROL SHIFT, TEST SPECIAL CHARACTERS
                            ;------ TEST FOR BREAK AND PAUSE KEYS

EB0C 3C46                          CMP     AL,SCROLL_KEY         ; TEST FOR BREAK
EB0E 7518                          JNE     K39                   ; NO-BREAK
```

```
LOC  OBJ                 SOURCE

EB10 BB1E00          R          MOV   BX,OFFSET KB_BUFFER      ; RESET BUFFER TO EMPTY
EB13 891E1A00        R          MOV   BUFFER_HEAD,BX           ;
EB17 891E1C00        R          MOV   BUFFER_TAIL,BX           ;
EB1B C606710080      R          MOV   BIOS_BREAK,80H           ; TURN ON BIOS_BREAK BIT
EB20 CD1B                       INT   1BH                      ; BREAK INTERRUPT VECTOR
EB22 B80000                     MOV   AX, 0                    ; PUT OUT DUMMY CHARACTER
EB25 E9B400                     JMP   K57                      ; BUFFER_FILL

EB28                     K39:                                  ; NO-BREAK
EB28 3C45                       CMP   AL,NUM_KEY               ; LOOK FOR PAUSE KEY
EB2A 7521                       JNE   K41                      ; NO-PAUSE
EB2C 800E180008      R          OR    KB_FLAG_1,HOLD_STATE     ; TURN ON THE HOLD FLAG
EB31 B020                       MOV   AL,EOI                   ; END OF INTERRUPT TO CONTROL PORT
EB33 E620                       OUT   020H,AL                  ; ALLOW FURTHER KEYSTROKE INTS

                         ;------ DURING PAUSE INTERVAL, TURN CRT BACK ON

EB35 803E490007      R          CMP   CRT_MODE,7               ; IS THIS BLACK AND WHITE CARD
EB3A 7407                       JE    K40                      ; YES, NOTHING TO DO
EB3C BAD803                     MOV   DX,03D8H                 ; PORT FOR COLOR CARD
EB3F A06500          R          MOV   AL,CRT_MODE_SET          ; GET THE VALUE OF THE CURRENT MODE
EB42 EE                         OUT   DX,AL                    ; SET THE CRT MODE, SO THAT CRT IS ON
EB43                     K40:                                  ; PAUSE-LOOP
EB43 F606180008      R          TEST  KB_FLAG_1,HOLD_STATE
EB48 75F9                       JNZ   K40                      ; LOOP UNTIL FLAG TURNED OFF
EB4A E916FF                     JMP   K27                      ; INTERRUPT_RETURN_NO_EOI
EB4D                     K41:                                  ; NO-PAUSE

                         ;------ TEST SPECIAL CASE KEY 55

EB4D 3C37                       CMP   AL,55
EB4F 7506                       JNE   K42                      ; NOT-KEY-55
EB51 B80072                     MOV   AX,114*256               ; START/STOP PRINTING SWITCH
EB54 E98500                     JMP   K57                      ; BUFFER_FILL

                         ;------ SET UP TO TRANSLATE CONTROL SHIFT

EB57                     K42:                                  ; NOT-KEY-55
EB57 BB92E8          R          MOV   BX,OFFSET K8             ; SET UP TO TRANSLATE CTL
EB5A 3C3B                       CMP   AL,59                    ; IS IT IN TABLE
EB5C 7303                       JAE   K43                      ; CTL-TABLE-TRANSLATE
EB5E EB7890                     JMP   K56                      ; YES, GO TRANSLATE CHAR
EB61                     K43:                                  ; CTL-TABLE-TRANSLATE
EB61 BBCCE8          R          MOV   BX,OFFSET K9             ; CTL TABLE SCAN
EB64 E9C300                     JMP   K63                      ; TRANSLATE_SCAN

                         ;------ NOT IN CONTROL SHIFT

EB67                     K44:                                  ; NOT-CTL-SHIFT

EB67 3C47                       CMP   AL,71                    ; TEST FOR KEYPAD REGION
EB69 732D                       JAE   K48                      ; HANDLE KEYPAD REGION
EB6B F606170003      R          TEST  KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT
EB70 745B                       JZ    K54                      ; TEST FOR SHIFT STATE

                         ;------ UPPER CASE, HANDLE SPECIAL CASES

EB72 3C0F                       CMP   AL,15                    ; BACK TAB KEY
EB74 7506                       JNE   K45                      ; NOT-BACK-TAB
EB76 B8000F                     MOV   AX,15*256                ; SET PSEUDO SCAN CODE
EB79 EB6190                     JMP   K57                      ; BUFFER_FILL

EB7C                     K45:                                  ; NOT-BACK-TAB
EB7C 3C37                       CMP   AL,55                    ; PRINT SCREEN KEY
EB7E 7509                       JNE   K46                      ; NOT-PRINT-SCREEN

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

EB80 B020                       MOV   AL,EOI                   ; END OF CURRENT INTERRUPT
EB82 E620                       OUT   020H,AL                  ;  SO FURTHER THINGS CAN HAPPEN
EB84 CD05                       INT   5H                       ; ISSUE PRINT SCREEN INTERRUPT
EB86 E9DAFE                     JMP   K27                      ; GO BACK WITHOUT EOI OCCURRING

EB89                     K46:                                  ; NOT-PRINT-SCREEN
EB89 3C3B                       CMP   AL,59                    ; FUNCTION KEYS
EB8B 7206                       JB    K47                      ; NOT-UPPER-FUNCTION
EB8D BB59E9          R          MOV   BX,OFFSET K12            ; UPPER CASE PSEUDO SCAN CODES
```

```
LOC OBJ              LINE    SOURCE

EB90 E99700                          JMP     K63                     ; TRANSLATE_SCAN

EB93                         K47:                                    ; NOT-UPPER-FUNCTION
EB93 BB1FE9          R               MOV     BX,OFFSET K11           ; POINT TO UPPER CASE TABLE
EB96 EB40                            JMP     SHORT K56               ; OK, TRANSLATE THE CHAR

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

EB98                         K48:                                    ; KEYPAD-REGION
EB98 F606170020      R               TEST    KB_FLAG,NUM_STATE       ; ARE WE IN NUM_LOCK
EB9D 7520                            JNZ     K52                     ; TEST FOR SURE
EB9F F606170003      R               TEST    KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT  ; ARE WE IN SHIFT STATE
EBA4 7520                            JNZ     K53                     ; IF SHIFTED, REALLY NUM STATE

                             ;------ BASE CASE FOR KEYPAD

EBA6                         K49:                                    ; BASE-CASE
                                                                              ;
EBA6 3C4A                            CMP     AL,74                   ; SPECIAL CASE FOR A COUPLE OF KEYS
EBA8 740B                            JE      K50                     ; MINUS
EBAA 3C4E                            CMP     AL,78
EBAC 740C                            JE      K51
EBAE 2C47                            SUB     AL,71                   ; CONVERT ORIGIN
EBB0 BB7AE9          R               MOV     BX,OFFSET K15           ; BASE CASE TABLE
EBB3 EB77                            JMP     SHORT K64               ; CONVERT TO PSEUDO SCAN

EBB5 B82D4A                  K50:    MOV     AX,74*256+'-'           ; MINUS
EBB8 EB22                            JMP     SHORT K57               ; BUFFER_FILL

EBBA B82B4E                  K51:    MOV     AX,78*256+'+'           ; PLUS
EBBD EB1D                            JMP     SHORT K57               ; BUFFER_FILL

                             ;------ MIGHT BE NUM LOCK, TEST SHIFT STATUS

EBBF                         K52:                                    ; ALMOST-NUM-STATE
EBBF F606170003      R               TEST    KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT
EBC4 75E0                            JNZ     K49                     ; SHIFTED TEMP OUT OF NUM STATE

EBC6                         K53:                                    ; REALLY_NUM_STATE
EBC6 2C46                            SUB     AL,70                   ; CONVERT ORIGIN
EBC8 BB6DE9          R               MOV     BX,OFFSET K14           ; NUM STATE TABLE
EBCB EB0B                            JMP     SHORT K56               ; TRANSLATE_CHAR

                             ;------ PLAIN OLD LOWER CASE

EBCD                         K54:                                    ; NOT-SHIFT
EBCD 3C3B                            CMP     AL,59                   ; TEST FOR FUNCTION KEYS
EBCF 7204                            JB      K55                     ; NOT-LOWER-FUNCTION
EBD1 B000                            MOV     AL,0                    ; SCAN CODE IN AH ALREADY
EBD3 EB07                            JMP     SHORT K57               ; BUFFER_FILL

EBD5                         K55:                                    ; NOT-LOWER-FUNCTION
EBD5 BBE5E8          R               MOV     BX,OFFSET K10           ; LC TABLE

                             ;------ TRANSLATE THE CHARACTER

EBD8                         K56:                                    ; TRANSLATE-CHAR
EBD8 FEC8                            DEC     AL                      ; CONVERT ORIGIN
EBDA 2ED7                            XLAT    CS:K11                  ; CONVERT THE SCAN CODE TO ASCII

                             ;------ PUT CHARACTER INTO BUFFER

EBDC                         K57:                                    ; BUFFER-FILL
EBDC 3CFF                            CMP     AL,-1                   ; IS THIS AN IGNORE CHAR
EBDE 741F                            JE      K59                     ; YES, DO NOTHING WITH IT
EBE0 80FCFF                          CMP     AH,-1                   ; LOOK FOR -1 PSEUDO SCAN
EBE3 741A                            JE      K59                     ; NEAR_INTERRUPT_RETURN

                             ;------ HANDLE THE CAPS LOCK PROBLEM

EBE5                         K58:                                    ; BUFFER-FILL-NOTEST
EBE5 F606170040      R               TEST    KB_FLAG,CAPS_STATE      ; ARE WE IN CAPS LOCK STATE
EBEA 7420                            JZ      K61                     ; SKIP IF NOT

                             ;------ IN CAPS LOCK STATE

EBEC F606170003      R               TEST    KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT  ; TEST FOR SHIFT STATE
```

```
LOC  OBJ                SOURCE

EBF1 740F                       JZ      K60                 ; IF NOT SHIFT, CONVERT LOWER TO UPPER

                        ;------ CONVERT ANY UPPER CASE TO LOWER CASE

EBF3 3C41                       CMP     AL,'A'              ; FIND OUT IF ALPHABETIC
EBF5 7215                       JB      K61                 ; NOT_CAPS_STATE
EBF7 3C5A                       CMP     AL,'Z'
EBF9 7711                       JA      K61                 ; NOT_CAPS_STATE
EBFB 0420                       ADD     AL,'a'-'A'          ; CONVERT TO LOWER CASE
EBFD EB0D                       JMP     SHORT K61           ; NOT_CAPS_STATE

EBFF                    K59:                                ; NEAR-INTERRUPT-RETURN
EBFF E95CFE                     JMP     K26                 ; INTERRUPT_RETURN

                        ;------ CONVERT ANY LOWER CASE TO UPPER CASE

EC02                    K60:                                ; LOWER-TO-UPPER
EC02 3C61                       CMP     AL,'a'              ; FIND OUT IF ALPHABETIC
EC04 7206                       JB      K61                 ; NOT_CAPS_STATE
EC06 3C7A                       CMP     AL,'z'
EC08 7702                       JA      K61                 ; NOT_CAPS_STATE
EC0A 2C20                       SUB     AL,'a'-'A'          ; CONVERT TO UPPER CASE

EC0C                    K61:                                ; NOT-CAPS-STATE
EC0C 8B1E1C00      R            MOV     BX,BUFFER_TAIL      ; GET THE END POINTER TO THE BUFFER
EC10 8BF3                       MOV     SI,BX               ; SAVE THE VALUE
EC12 E860FC                     CALL    K4                  ; ADVANCE THE TAIL
EC15 3B1E1A00      R            CMP     BX,BUFFER_HEAD      ; HAS THE BUFFER WRAPPED AROUND
EC19 7409                       JE      K62                 ; BUFFER_FULL_BEEP
EC1B 8904                       MOV     [SI],AX             ; STORE THE VALUE
EC1D 891E1C00      R            MOV     BUFFER_TAIL,BX      ; MOVE THE POINTER UP
EC21 E93AFE                     JMP     K26                 ; INTERRUPT_RETURN

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

EC24                    K62:                                ; BUFFER-FULL-BEEP
EC24 E80D00                     CALL    ERROR_BEEP
EC27 E934FE                     JMP     K26                 ; INTERRUPT_RETURN

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

EC2A                    K63:                                ; TRANSLATE-SCAN
EC2A 2C3B                       SUB     AL,59               ; CONVERT ORIGIN TO FUNCTION KEYS
EC2C                    K64:                                ; TRANSLATE-SCAN-ORGD
EC2C 2ED7                       XLAT    CS:K9               ; CTL TABLE SCAN
EC2E 8AE0                       MOV     AH,AL               ; PUT VALUE INTO AH
EC30 B000                       MOV     AL,0                ; ZERO ASCII CODE
EC32 EBA8                       JMP     K57                 ; PUT IT INTO THE BUFFER

                        KB_INT  ENDP
EC34                    ERROR_BEEP      PROC    NEAR
EC34 50                         PUSH    AX          ; SAVE REGISTERS
EC35 53                         PUSH    BX
EC36 51                         PUSH    CX
EC37 BBC000                     MOV     BX,0C0H     ; NUMBER OF CYCLES FOR 1/8 SECOND TONE
EC3A E461                       IN      AL,KB_CTL   ; GET CONTROL INFORMATION
EC3C 50                         PUSH    AX          ; SAVE
EC3D                    K65:                        ; BEEP-CYCLE
EC3D 24FC                       AND     AL,0FCH     ; TURN OFF TIMER GATE AND SPEAKER DATA
EC3F E661                       OUT     KB_CTL,AL   ; OUTPUT TO CONTROL
EC41 B94800                     MOV     CX,48H      ; HALF CYCLE TIME FOR TONE
EC44 E2FE               K66:    LOOP    K66         ; SPEAKER OFF
EC46 0C02                       OR      AL,2        ; TURN ON SPEAKER BIT
EC48 E661                       OUT     KB_CTL,AL   ; OUTPUT TO CONTROL
EC4A B94800                     MOV     CX,48H      ; SET UP COUNT
EC4D E2FE               K67:    LOOP    K67         ; ANOTHER HALF CYCLE
EC4F 4B                         DEC     BX          ; TOTAL TIME COUNT
EC50 75EB                       JNZ     K65         ; DO ANOTHER CYCLE
EC52 58                         POP     AX          ; RECOVER CONTROL
EC53 E661                       OUT     KB_CTL,AL   ; OUTPUT THE CONTROL
EC55 59                         POP     CX          ; RECOVER REGISTERS
EC56 5B                         POP     BX
EC57 58                         POP     AX
EC58 C3                         RET
                        ERROR_BEEP      ENDP
```

```
LOC OBJ                 LINE    SOURCE

                                ;-- INT 13 ---------------------------------
                                ;DISKETTE I/O
                                ;       THIS INTERFACE PROVIDES ACCESS TO THE 5 1/4" DISKETTE DRIVES
                                ;INPUT
                                ;       (AH)=0  RESET DISKETTE SYSTEM
                                ;               HARD RESET TO NEC, PREPARE COMMAND, RECAL REQD ON ALL DRIVES
                                ;       (AH)=1  READ THE STATUS OF THE SYSTEM INTO (AL)
                                ;               DISKETTE_STATUS FROM LAST OP'N IS USED
                                ;       REGISTERS FOR READ/WRITE/VERIFY/FORMAT
                                ;       (DL) - DRIVE NUMBER (0-3 ALLOWED, VALUE CHECKED)
                                ;       (DH) - HEAD NUMBER (0-1 ALLOWED, NOT VALUE CHECKED)
                                ;       (CH) - TRACK NUMBER (0-39, NOT VALUE CHECKED)
                                ;       (CL) - SECTOR NUMBER (1-8, NOT VALUE CHECKED)
                                ;       (AL) - NUMBER OF SECTORS ( MAX = 8, NOT VALUE CHECKED)
                                ;
                                ;       (ES:BX) - ADDRESS OF BUFFER ( NOT REQUIRED FOR VERIFY)
                                ;
                                ;       (AH)=2  READ THE DESIRED SECTORS INTO MEMORY
                                ;       (AH)=3  WRITE THE DESIRED SECTORS FROM MEMORY
                                ;       (AH)=4  VERIFY THE DESIRED SECTORS
                                ;       (AH)=5  FORMAT THE DESIRED TRACK
                                ;               FOR THE FORMAT OPERATION, THE BUFFER POINTER (ES,BX) MUST
                                ;               POINT TO THE COLLECTION OF DESIRED ADDRESS FIELDS FOR THE
                                ;               TRACK.  EACH FIELD IS COMPOSED OF 4 BYTES, (C,H,R,N), WHERE
                                ;               C = TRACK NUMBER, H=HEAD NUMBER, R = SECTOR NUMBER, N= NUMBER
                                ;               OF BYTES PER SECTOR (00=128, 01=256, 02=512, 03=1024,)
                                ;               THERE MUST BE ONE ENTRY FOR EVERY SECTOR ON THE TRACK.
                                ;               THIS INFORMATION IS USED TO FIND THE REQUESTED SECTOR DURING
                                ;               READ/WRITE ACCESS.
                                ; DATA VARIABLE -- DISK_POINTER
                                ;       DOUBLE WORD POINTER TO THE CURRENT SET OF DISKETTE PARAMETERS
                                ; OUTPUT
                                ;       AH = STATUS OF OPERATION
                                ;               STATUS BITS ARE DEFINED IN THE EQUATES FOR DISKETTE_STATUS
                                ;               VARIABLE IN THE DATA SEGMENT OF THIS MODULE
                                ;       CY = 0  SUCCESSFUL OPERATION (AH=0 ON RETURN)
                                ;       CY = 1  FAILED OPERATION (AH HAS ERROR REASON)
                                ;       FOR READ/WRITE/VERIFY
                                ;               DS,BX,DX,CH,CL PRESERVED
                                ;               AL = NUMBER OF SECTORS ACTUALLY READ
                                ;               ***** AL MAY NOT BE CORRECT IF TIME OUT ERROR OCCURS
                                ;       NOTE: IF AN ERROR IS REPORTED BY THE DISKETTE CODE, THE APPROPRIATE
                                ;               ACTION IS TO RESET THE DISKETTE, THEN RETRY THE OPERATION.
                                ;               ON READ ACCESSES, NO MOTOR START DELAY IS TAKEN, SO THAT
                                ;               THREE RETRIES ARE REQUIRED ON READS TO ENSURE THAT THE
                                ;               PROBLEM IS NOT DUE TO MOTOR START-UP.
                                ;-------------------------------------------
                                        ASSUME  CS:CODE,DS:DATA,ES:DATA
EC59                            DISKETTE_IO     PROC    FAR
EC59 FB                                 STI                     ; INTERRUPTS BACK ON
EC5A 53                                 PUSH    BX              ; SAVE ADDRESS
EC5B 51                                 PUSH    CX
EC5C 1E                                 PUSH    DS              ; SAVE SEGMENT REGISTER VALUE
EC5D 56                                 PUSH    SI              ; SAVE ALL REGISTERS DURING OPERATION
EC5E 57                                 PUSH    DI
EC5F 55                                 PUSH    BP
EC60 52                                 PUSH    DX
EC61 8BEC                               MOV     BP,SP           ; SET UP POINTER TO HEAD PARM
EC63 BE4000          R                  MOV     SI,DATA
EC66 8EDE                               MOV     DS,SI           ; SET DATA REGION
EC68 E81C00                             CALL    J1              ; CALL THE REST TO ENSURE DS RESTORED
EC6B BB0400                             MOV     BX,4            ; GET THE MOTOR WAIT PARAMETER
EC6E E8FF01                             CALL    GET_PARM
EC71 88264000        R                  MOV     MOTOR_COUNT,AH  ; SET THE TIMER COUNT FOR THE MOTOR
EC75 8A264100        R                  MOV     AH,DISKETTE_STATUS      ; GET STATUS OF OPERATION
EC79 80FC01                             CMP     AH,1            ; SET THE CARRY FLAG TO INDICATE
EC7C F5                                 CMC                     ;  SUCCESS OR FAILURE
EC7D 5A                                 POP     DX              ; RESTORE ALL REGISTERS
EC7E 5D                                 POP     BP
EC7F 5F                                 POP     DI
EC80 5E                                 POP     SI
EC81 1F                                 POP     DS
EC82 59                                 POP     CX
EC83 5B                                 POP     BX              ; RECOVER ADDRESS
EC84 CA0200                             RET     2               ; THROW AWAY SAVED FLAGS
                                DISKETTE_IO     ENDP
EC87                            J1      PROC    NEAR
```

```
LOC  OBJ                 SOURCE

EC87 8AF0                        MOV    DH,AL            ; SAVE # SECTORS IN DH
EC89 80263F007F     R            AND    MOTOR_STATUS,07FH      ; INDICATE A READ OPERATION
EC8E 0AE4                        OR     AH,AH            ; AH=0
EC90 7427                        JZ     DISK_RESET
EC92 FECC                        DEC    AH               ; AH=1
EC94 7474                        JZ     DISK_STATUS
EC96 C606410000     R            MOV    DISKETTE_STATUS,0      ; RESET THE STATUS INDICATOR
EC9B 80FA04                      CMP    DL,4             ; TEST FOR DRIVE IN 0-3 RANGE
EC9E 7313                        JAE    J3               ; ERROR IF ABOVE
ECA0 FECC                        DEC    AH               ; AH=2
ECA2 746A                        JZ     DISK_READ
ECA4 FECC                        DEC    AH               ; AH=3
ECA6 7503                        JNZ    J2               ; TEST_DISK_VERF

ECA8 E99600                      JMP    DISK_WRITE
ECAB                     J2:                             ; TEST_DISK_VERF

ECAB FECC                        DEC    AH               ; AH=4
ECAD 7468                        JZ     DISK_VERF
ECAF FECC                        DEC    AH               ; AH=5
ECB1 7468                        JZ     DISK_FORMAT
ECB3                     J3:                             ; BAD_COMMAND
ECB3 C606410001     R            MOV    DISKETTE_STATUS,BAD_CMD ; ERROR CODE, NO SECTORS TRANSFERRED
ECB8 C3                          RET                     ; UNDEFINED OPERATION
                         J1      ENDP

                         ;------ RESET THE DISKETTE SYSTEM

ECB9                     DISK_RESET     PROC    NEAR
ECB9 BAF203                      MOV    DX,03F2H         ; ADAPTER CONTROL PORT
ECBC FA                          CLI                     ; NO INTERRUPTS
ECBD A03F00         R            MOV    AL,MOTOR_STATUS  ; WHICH MOTOR IS ON
ECC0 B104                        MOV    CL,4             ; SHIFT COUNT
ECC2 D2E0                        SAL    AL,CL            ; MOVE MOTOR VALUE TO HIGH NYBBLE
ECC4 A820                        TEST   AL, 20H          ; SELECT CORRESPONDING DRIVE
ECC6 750C                        JNZ    J5               ; JUMP IF MOTOR ONE IS ON
ECC8 A840                        TEST   AL, 40H
ECCA 7506                        JNZ    J4               ; JUMP IF MOTOR TWO IS ON
ECCC A880                        TEST   AL, 80H
ECCE 7406                        JZ     J6               ; JUMP IF MOTOR ZERO IS ON
ECD0 FEC0                        INC    AL
ECD2 FEC0                J4:     INC    AL
ECD4 FEC0                J5:     INC    AL
ECD6 0C08                J6:     OR     AL,8             ; TURN ON INTERRUPT ENABLE
ECD8 EE                          OUT    DX,AL            ; RESET THE ADAPTER
ECD9 C6063E0000     R            MOV    SEEK_STATUS,0    ; SET RECAL REQUIRED ON ALL DRIVES
ECDE C606410000     R            MOV    DISKETTE_STATUS,0      ; SET OK STATUS FOR DISKETTE
ECE3 0C04                        OR     AL,4             ; TURN OFF RESET
ECE5 EE                          OUT    DX,AL            ; TURN OFF THE RESET
ECE6 FB                          STI                     ; REENABLE THE INTERRUPTS
ECE7 E82802                      CALL   CHK_STAT_2       ; DO SENSE INTERRUPT STATUS FOLLOWING RESET
ECEA A04200         R            MOV    AL,NEC_STATUS    ; IGNORE ERROR RETURN AND DO OWN TEST
ECED 3CC0                        CMP    AL,0C0H          ; TEST FOR DRIVE READY TRANSITION
ECEF 7407                        JZ     J7               ; EVERYTHING OK
ECF1 800E410020     R            OR     DISKETTE_STATUS,BAD_NEC ; SET ERROR CODE
ECF6 EB11                        JMP    SHORT J8               ; RESET_RET

                         ;------ SEND SPECIFY COMMAND TO NEC

ECF8                     J7:                             ; DRIVE_READY
ECF8 B403                        MOV    AH,03H           ; SPECIFY COMMAND
ECFA E84701                      CALL   NEC_OUTPUT       ; OUTPUT THE COMMAND
ECFD BB0100                      MOV    BX,1             ; FIRST BYTE PARM IN BLOCK
ED00 E86D01                      CALL   GET_PARM         ;  TO THE NEC CONTROLLER
ED03 BB0300                      MOV    BX,3             ; SECOND BYTE PARM IN BLOCK
ED06 E86701                      CALL   GET_PARM         ;  TO THE NEC CONTROLLER
ED09                     J8:                             ; RESET_RET
ED09 C3                          RET                     ; RETURN TO CALLER
                         DISK_RESET     ENDP

                         ;------ DISKETTE STATUS ROUTINE

ED0A                     DISK_STATUS    PROC    NEAR
ED0A A04100         R            MOV    AL,DISKETTE_STATUS
ED0D C3                          RET
                         DISK_STATUS    ENDP
```

```
LOC OBJ                 SOURCE

                        ;------ DISKETTE READ

ED0E                    DISK_READ       PROC    NEAR
ED0E B046                       MOV     AL,046H         ; READ COMMAND FOR DMA
ED10                    J9:                             ; DISK_READ_CONT
ED10 E8B901                     CALL    DMA_SETUP       ; SET UP THE DMA
ED13 B466                       MOV     AH,066H         ; SET UP READ COMMAND FOR NEC CONTROLLER
ED15 EB36                       JMP     SHORT RW_OPN    ; GO DO THE OPERATION
                        DISK_READ       ENDP

                        ;------ DISKETTE VERIFY

ED17                    DISK_VERF       PROC    NEAR
ED17 B042                       MOV     AL,042H                 ; VERIFY COMMAND FOR DMA
ED19 EBF5                       JMP     J9                      ; DO AS IF DISK READ
                        DISK_VERF       ENDP

                        ;------ DISKETTE FORMAT

ED1B                    DISK_FORMAT     PROC    NEAR
ED1B 800E3F0080      R          OR      MOTOR_STATUS,80H        ; INDICATE WRITE OPERATION
ED20 B04A                       MOV     AL,04AH                 ; WILL WRITE TO THE DISKETTE
ED22 E8A701                     CALL    DMA_SETUP               ; SET UP THE DMA
ED25 B44D                       MOV     AH,04DH                 ; ESTABLISH THE FORMAT COMMAND
ED27 EB24                       JMP     SHORT RW_OPN            ; DO THE OPERATION
ED29                    J10:                                    ; CONTINUATION OF RW_OPN FOR FMT
ED29 BB0700                     MOV     BX,7                    ; GET THE
ED2C E84101                     CALL    GET_PARM                ;  BYTES/SECTOR VALUE TO NEC
ED2F BB0900                     MOV     BX,9                    ; GET THE
ED32 E83B01                     CALL    GET_PARM                ;  SECTORS/TRACK VALUE TO NEC
ED35 BB0F00                     MOV     BX,15                   ; GET THE
ED38 E83501                     CALL    GET_PARM                ;  GAP LENGTH VALUE TO NEC
ED3B BB1100                     MOV     BX,17                   ; GET THE FILLER BYTE
ED3E E9AB00                     JMP     J16                     ;  TO THE CONTROLLER
                        DISK_FORMAT     ENDP

                        ;------ DISKETTE WRITE ROUTINE

ED41                    DISK_WRITE      PROC    NEAR
ED41 800E3F0080      R          OR      MOTOR_STATUS,80H        ; INDICATE WRITE OPERATION
ED46 B04A                       MOV     AL,04AH                 ; DMA WRITE COMMAND
ED48 E88101                     CALL    DMA_SETUP
ED4B B445                       MOV     AH,045H                 ; NEC COMMAND TO WRITE TO DISKETTE
                        DISK_WRITE      ENDP
                        ;----- ALLOW WRITE ROUTINE TO FALL INTO RW_OPN
                        ;--------------------------------------
                        ; RW_OPN
                        ;       THIS ROUTINE PERFORMS THE READ/WRITE/VERIFY OPERATION
                        ;--------------------------------------
ED4D                    RW_OPN  PROC    NEAR
ED4D 7308                       JNC     J11                     ; TEST FOR DMA ERROR
ED4F C606410009      R          MOV     DISKETTE_STATUS,DMA_BOUNDARY    ; SET ERROR
ED54 B000                       MOV     AL,0                    ; NO SECTORS TRANSFERRED
ED56 C3                         RET                             ; RETURN TO MAIN ROUTINE
ED57                    J11:                                    ; DO_RW_OPN
ED57 50                         PUSH    AX              ; SAVE THE COMMAND

                        ;------ TURN ON THE MOTOR AND SELECT THE DRIVE

ED58 51                         PUSH    CX              ; SAVE THE T/S PARMS
ED59 8ACA                       MOV     CL,DL           ; GET DRIVE NUMBER AS SHIFT COUNT
ED5B B001                       MOV     AL,1            ; MASK FOR DETERMINING MOTOR BIT
ED5D D2E0                       SAL     AL,CL           ; SHIFT THE MASK BIT
ED5F FA                         CLI                     ; NO INTERRUPTS WHILE DETERMINING MOTOR STATUS
ED60 C6064000FF      R          MOV     MOTOR_COUNT,0FFH        ; SET LARGE COUNT DURING OPERATION
ED65 84063F00        R          TEST    AL,MOTOR_STATUS ; TEST THAT MOTOR FOR OPERATING
ED69 7531                       JNZ     J14             ; IF RUNNING, SKIP THE WAIT
ED6B 80263F00F0      R          AND     MOTOR_STATUS,0F0H       ; TURN OFF ALL MOTOR BITS
ED70 08063F00        R          OR      MOTOR_STATUS,AL         ; TURN ON THE CURRENT MOTOR
ED74 FB                         STI                     ; INTERRUPTS BACK ON
ED75 B010                       MOV     AL,10H          ; MASK BIT
ED77 D2E0                       SAL     AL,CL           ; DEVELOP BIT MASK FOR MOTOR ENABLE
ED79 0AC2                       OR      AL,DL           ; GET DRIVE SELECT BITS IN
ED7B 0C0C                       OR      AL,0CH          ; NO RESET, ENABLE DMA/INT
ED7D 52                         PUSH    DX              ; SAVE REG
ED7E BAF203                     MOV     DX,03F2H        ; CONTROL PORT ADDRESS
ED81 EE                         OUT     DX,AL
```

```
LOC OBJ                  SOURCE

ED82 5A                        POP     DX                ; RECOVER REGISTERS

                         ;------ WAIT FOR MOTOR IF WRITE OPERATION

ED83 F6063F0080     R          TEST    MOTOR_STATUS,80H          ; IS THIS A WRITE
ED88 7412                      JZ      J14                       ; NO, CONTINUE WITHOUT WAIT
ED8A BB1400                    MOV     BX,20                     ; GET THE MOTOR WAIT
ED8D E8E000                    CALL    GET_PARM                  ;  PARAMETER
ED90 0AE4                      OR      AH,AH                     ; TEST FOR NO WAIT
ED92                     J12:                                    ; TEST_WAIT_TIME
ED92 7408                      JZ      J14                       ; EXIT WITH TIME EXPIRED
ED94 2BC9                      SUB     CX,CX                     ; SET UP 1/8 SECOND LOOP TIME
ED96 E2FE                J13:  LOOP    J13                       ; WAIT FOR THE REQUIRED TIME
ED98 FECC                      DEC     AH                        ; DECREMENT TIME VALUE
ED9A EBF6                      JMP     J12                       ; ARE WE DONE YET

ED9C                     J14:                                    ; MOTOR_RUNNING
ED9C FB                        STI                       ; INTERRUPTS BACK ON FOR BYPASS WAIT
ED9D 59                        POP     CX

                         ;------ DO THE SEEK OPERATION

ED9E E8E000                    CALL    SEEK              ; MOVE TO CORRECT TRACK
EDA1 58                        POP     AX                ; RECOVER COMMAND
EDA2 8AFC                      MOV     BH,AH             ; SAVE COMMAND IN BH
EDA4 B600                      MOV     DH,0              ; SET NO SECTORS READ IN CASE OF ERROR
EDA6 724B                      JC      J17               ; IF ERROR, THEN EXIT AFTER MOTOR OFF
EDA8 BEF3ED90       R          MOV     SI,OFFSET J17     ; DUMMY RETURN ON STACK FOR NEC_OUTPUT
EDAC 56                        PUSH    SI                ;  SO THAT IT WILL RETURN TO MOTOR OFF LOCATION

                         ;------ SEND OUT THE PARAMETERS TO THE CONTROLLER

EDAD E89400                    CALL    NEC_OUTPUT        ; OUTPUT THE OPERATION COMMAND
EDB0 8A6601                    MOV     AH,[BP+1]         ; GET THE CURRENT HEAD NUMBER
EDB3 D0E4                      SAL     AH,1              ; MOVE IT TO BIT 2
EDB5 D0E4                      SAL     AH,1
EDB7 80E404                    AND     AH,4              ; ISOLATE THAT BIT
EDBA 0AE2                      OR      AH,DL             ; OR IN THE DRIVE NUMBER
EDBC E88500                    CALL    NEC_OUTPUT

                         ;------ TEST FOR FORMAT COMMAND

EDBF 80FF4D                    CMP     BH,04DH           ; IS THIS A FORMAT OPERATION
EDC2 7503                      JNE     J15               ; NO. CONTINUE WITH R/W/V
EDC4 E962FF                    JMP     J10               ; IF SO, HANDLE SPECIAL

EDC7 8AE5                J15:  MOV     AH,CH             ; CYLINDER NUMBER
EDC9 E87800                    CALL    NEC_OUTPUT
EDCC 8A6601                    MOV     AH,[BP+1]         ; HEAD NUMBER FROM STACK
EDCF E87200                    CALL    NEC_OUTPUT
EDD2 8AE1                      MOV     AH,CL             ; SECTOR NUMBER
EDD4 E86D00                    CALL    NEC_OUTPUT
EDD7 BB0700                    MOV     BX,7              ; BYTES/SECTOR PARM FROM BLOCK
EDDA E89300                    CALL    GET_PARM          ;  TO THE NEC
EDDD BB0900                    MOV     BX,9              ; EOT PARM FROM BLOCK
EDE0 E88D00                    CALL    GET_PARM          ;  TO THE NEC
EDE3 BB0B00                    MOV     BX,11             ; GAP LENGTH PARM FROM BLOCK
EDE6 E88700                    CALL    GET_PARM          ;  TO THE NEC
EDE9 BB0D00                    MOV     BX,13             ; DTL PARM FROM BLOCK
EDEC                     J16:                            ; RW_OPN_FINISH
EDEC E88100                    CALL    GET_PARM          ;  TO THE NEC
EDEF 5E                        POP     SI                ; CAN NOW DISCARD THAT DUMMY RETURN ADDRESS

                         ;------ LET THE OPERATION HAPPEN

EDF0 E84001                    CALL    WAIT_INT          ; WAIT FOR THE INTERRUPT
EDF3                     J17:                            ; MOTOR_OFF
EDF3 7245                      JC      J21               ; LOOK FOR ERROR
EDF5 E87301                    CALL    RESULTS           ; GET THE NEC STATUS
EDF8 723F                      JC      J20               ; LOOK FOR ERROR

                         ;------ CHECK THE RESULTS RETURNED BY THE CONTROLLER

EDFA FC                        CLD                       ; SET THE CORRECT DIRECTION
EDFB BE4200         R          MOV     SI,OFFSET NEC_STATUS      ; POINT TO STATUS FIELD
EDFE AC                        LODS    NEC_STATUS                ; GET ST0
EDFF 24C0                      AND     AL,0C0H                   ; TEST FOR NORMAL TERMINATION
```

```
LOC OBJ               SOURCE

EE01 7438                      JZ      J22                     ; OPN_OK
EE03 3C40                      CMP     AL,040H                 ; TEST FOR ABNORMAL TERMINATION
EE05 7529                      JNZ     J18                     ; NOT ABNORMAL, BAD NEC

                      ;------ ABNORMAL TERMINATION, FIND OUT WHY

EE07 AC                        LODS    NEC_STATUS              ; GET ST1
EE08 D0E0                      SAL     AL,1                    ; TEST FOR EOT FOUND
EE0A B404                      MOV     AH,RECORD_NOT_FND
EE0C 7224                      JC      J19                     ; RW_FAIL
EE0E D0E0                      SAL     AL,1
EE10 D0E0                      SAL     AL,1                    ; TEST FOR CRC ERROR
EE12 B410                      MOV     AH,BAD_CRC
EE14 721C                      JC      J19                     ; RW_FAIL
EE16 D0E0                      SAL     AL,1                    ; TEST FOR DMA OVERRUN
EE18 B408                      MOV     AH,BAD_DMA
EE1A 7216                      JC      J19                     ; RW_FAIL
EE1C D0E0                      SAL     AL,1
EE1E D0E0                      SAL     AL,1                    ; TEST FOR RECORD NOT FOUND
EE20 B404                      MOV     AH,RECORD_NOT_FND
EE22 720E                      JC      J19                     ; RW_FAIL
EE24 D0E0                      SAL     AL,1
EE26 B403                      MOV     AH,WRITE_PROTECT        ; TEST FOR WRITE_PROTECT
EE28 7208                      JC      J19                     ; RW_FAIL
EE2A D0E0                      SAL     AL,1                    ; TEST MISSING ADDRESS MARK
EE2C B402                      MOV     AH,BAD_ADDR_MARK
EE2E 7202                      JC      J19                     ; RW_FAIL

                      ;------ NEC MUST HAVE FAILED

EE30                  J18:                                     ; RW-NEC-FAIL
EE30 B420                      MOV     AH,BAD_NEC
EE32                  J19:                                     ; RW-FAIL
EE32 08264100      R           OR      DISKETTE_STATUS,AH
EE36 E87701                    CALL    NUM_TRANS               ; HOW MANY WERE REALLY TRANSFERRED
EE39                  J20:                                     ; RW_ERR
EE39 C3                        RET                     ; RETURN TO CALLER

EE3A                  J21:                             ; RW_ERR_RES
EE3A E82E01                    CALL    RESULTS         ; FLUSH THE RESULTS BUFFER
EE3D C3                        RET

                      ;------ OPERATION WAS SUCCESSFUL

EE3E                  J22:                                     ; OPN_OK
EE3E E86F01                    CALL    NUM_TRANS               ; HOW MANY GOT MOVED
EE41 32E4                      XOR     AH,AH                   ; NO ERRORS
EE43 C3                        RET
                      RW_OPN   ENDP
                      ;--------------------------------------------
                      ; NEC_OUTPUT
                      ;        THIS ROUTINE SENDS A BYTE TO THE NEC CONTROLLER
                      ;        AFTER TESTING FOR CORRECT DIRECTION AND CONTROLLER READY
                      ;        THIS ROUTINE WILL TIME OUT IF THE BYTE IS NOT ACCEPTED
                      ;        WITHIN A REASONABLE AMOUNT OF TIME, SETTING THE DISKETTE STATUS
                      ;        ON COMPLETION
                      ; INPUT
                      ;        (AH)    BYTE TO BE OUTPUT
                      ; OUTPUT
                      ;        CY = 0  SUCCESS
                      ;        CY = 1  FAILURE -- DISKETTE STATUS UPDATED
                      ;                IF A FAILURE HAS OCCURRED, THE RETURN IS MADE ONE LEVEL
                      ;                HIGHER THAN THE CALLER OF NEC_OUTPUT
                      ;                THIS REMOVES THE REQUIREMENT OF TESTING AFTER EVERY CALL
                      ;                OF NEC_OUTPUT
                      ;        (AL) DESTROYED
                      ;--------------------------------------------
EE44                  NEC_OUTPUT       PROC    NEAR
EE44 52                        PUSH    DX              ; SAVE REGISTERS
EE45 51                        PUSH    CX
EE46 BAF403                    MOV     DX,03F4H        ; STATUS PORT
EE49 33C9                      XOR     CX,CX           ; COUNT FOR TIME OUT
EE4B                  J23:
EE4B EC                        IN      AL,DX           ; GET STATUS
EE4C A840                      TEST    AL,040H         ; TEST DIRECTION BIT
EE4E 740C                      JZ      J25             ; DIRECTION OK
EE50 E2F9                      LOOP    J23
EE52                  J24:                             ; TIME_ERROR
```

```
LOC OBJ                  SOURCE

EE52 800E410080     R            OR      DISKETTE_STATUS,TIME_OUT
EE57 59                          POP     CX
EE58 5A                          POP     DX                ; SET ERROR CODE AND RESTORE REGS
EE59 58                          POP     AX                ; DISCARD THE RETURN ADDRESS
EE5A F9                          STC                       ; INDICATE ERROR TO CALLER
EE5B C3                          RET

EE5C                     J25:
EE5C 33C9                        XOR     CX,CX             ; RESET THE COUNT
EE5E                     J26:
EE5E EC                          IN      AL,DX             ; GET THE STATUS
EE5F A880                        TEST    AL,080H           ; IS IT READY
EE61 7504                        JNZ     J27               ; YES, GO OUTPUT
EE63 E2F9                        LOOP    J26               ; COUNT DOWN AND TRY AGAIN
EE65 EBEB                        JMP     J24               ; ERROR CONDITION
EE67                     J27:                              ; OUTPUT
EE67 8AC4                        MOV     AL,AH             ; GET BYTE TO OUTPUT
EE69 BAF503                      MOV     DX,03F5H          ; DATA PORT
EE6C EE                          OUT     DX,AL             ; OUTPUT THE BYTE
EE6D 59                          POP     CX                ; RECOVER REGISTERS
EE6E 5A                          POP     DX
EE6F C3                          RET                       ; CY = 0 FROM TEST INSTRUCTION
                         NEC_OUTPUT      ENDP
                         ;-----------------------------------------
                         ; GET_PARM
                         ;  THIS ROUTINE FETCHES THE INDEXED POINTER FROM
                         ;  THE DISK_BASE BLOCK POINTED AT BY THE DATA
                         ;  VARIABLE DISK_POINTER
                         ; A BYTE FROM THAT TABLE IS THEN MOVED INTO AH,
                         ;  THE INDEX OF THAT BYTE BEING THE PARM IN BX
                         ; ENTRY --
                         ;   BX = INDEX OF BYTE TO BE FETCHED * 2
                         ;        IF THE LOW BIT OF BX IS ON, THE BYTE IS IMMEDIATELY
                         ;        OUTPUT TO THE NEC CONTROLLER
                         ; EXIT --
                         ;   AH = THAT BYTE FROM BLOCK
                         ;-------------------------------------------
EE70                     GET_PARM        PROC    NEAR
EE70 1E                          PUSH    DS                ; SAVE SEGMENT
EE71 2BC0                        SUB     AX,AX             ; ZERO TO AX
EE73 8ED8                        MOV     DS,AX
                                 ASSUME  DS:ABS0
EE75 C5367800                    LDS     SI,DISK_POINTER ; POINT TO BLOCK
EE79 D1EB                        SHR     BX,1              ; DIVIDE BX BY 2, AND SET FLAG FOR EXIT
EE7B 8A20                        MOV     AH,[SI+BX]        ; GET THE WORD
EE7D 1F                          POP     DS                ; RESTORE SEGMENT
                                 ASSUME  DS:DATA
EE7E 72C4                        JC      NEC_OUTPUT        ; IF FLAG SET, OUTPUT TO CONTROLLER
EE80 C3                          RET                       ; RETURN TO CALLER
                         GET_PARM        ENDP
                         ;-------------------------------------------
                         ; SEEK
                         ;       THIS ROUTINE WILL MOVE THE HEAD ON THE NAMED DRIVE
                         ;       TO THE NAMED TRACK.  IF THE DRIVE HAS NOT BEEN ACCESSED
                         ;       SINCE THE DRIVE RESET COMMAND WAS ISSUED, THE DRIVE WILL BE
                         ;       RECALIBRATED.
                         ; INPUT
                         ;       (DL) = DRIVE TO SEEK ON
                         ;       (CH) = TRACK TO SEEK TO
                         ; OUTPUT
                         ;       CY = 0 SUCCESS
                         ;       CY = 1 FAILURE -- DISKETTE_STATUS SET ACCORDINGLY
                         ;       (AX) DESTROYED
                         ;-------------------------------------------
EE81                     SEEK    PROC    NEAR
EE81 B001                        MOV     AL,1              ; ESTABLISH MASK FOR RECAL TEST
EE83 51                          PUSH    CX                ; SAVE INPUT VALUES
EE84 8ACA                        MOV     CL,DL             ; GET DRIVE VALUE INTO CL
EE86 D2C0                        ROL     AL,CL             ; SHIFT IT BY THE DRIVE VALUE
EE88 59                          POP     CX                ; RECOVER TRACK VALUE
EE89 84063E00       R            TEST    AL,SEEK_STATUS    ; TEST FOR RECAL REQUIRED
EE8D 7513                        JNZ     J28               ; NO_RECAL
EE8F 08063E00       R            OR      SEEK_STATUS,AL    ; TURN ON THE NO RECAL BIT IN FLAG
EE93 B407                        MOV     AH,07H            ; RECALIBRATE COMMAND
EE95 E8ACFF                      CALL    NEC_OUTPUT
EE98 8AE2                        MOV     AH,DL
EE9A E8A7FF                      CALL    NEC_OUTPUT        ; OUTPUT THE DRIVE NUMBER
```

```
LOC  OBJ          SOURCE

EE9D E87200               CALL  CHK_STAT_2     ; GET THE INTERUPT AND SENSE INT STATUS
EEA0 7229                 JC    J32            ; SEEK_ERROR

                   ;----- DRIVE IS IN SYNCH WITH CONTROLLER, SEEK TO TRACK

EEA2               J28:
EEA2 B40F                 MOV   AH,0FH         ; SEEK COMMAND TO NEC
EEA4 E89DFF               CALL  NEC_OUTPUT
EEA7 8AE2                 MOV   AH,DL          ; DRIVE NUMBER
EEA9 E898FF               CALL  NEC_OUTPUT
EEAC 8AE5                 MOV   AH,CH          ; TRACK NUMBER
EEAE E893FF               CALL  NEC_OUTPUT
EEB1 E85E00               CALL  CHK_STAT_2     ; GET ENDING INTERRUPT AND SENSE STATUS

                   ;----- WAIT FOR HEAD SETTLE

EEB4 9C                   PUSHF                ; SAVE STATUS FLAGS
EEB5 BB1200               MOV   BX,18          ; GET HEAD SETTLE PARAMETER
EEB8 E8B5FF               CALL  GET_PARM
EEBB 51                   PUSH  CX             ; SAVE REGISTER
EEBC               J29:                        ; HEAD_SETTLE
EEBC B92602               MOV   CX,550         ; 1 MS LOOP
EEBF 0AE4                 OR    AH,AH          ; TEST FOR TIME EXPIRED
EEC1 7406                 JZ    J31
EEC3 E2FE          J30:   LOOP  J30            ; DELAY FOR 1 MS
EEC5 FECC                 DEC   AH             ; DECREMENT THE COUNT
EEC7 EBF3                 JMP   J29            ; DO IT SOME MORE
EEC9               J31:
EEC9 59                   POP   CX             ; RECOVER STATE
EECA 9D                   POPF
EECB               J32:                        ; SEEK_ERROR
EECB C3                   RET                  ; RETURN TO CALLER
                   SEEK   ENDP
                   ;--------------------------------------------
                   ; DMA_SETUP
                   ;      THIS ROUTINE SETS UP THE DMA FOR READ/WRITE/VERIFY
                   ;      OPERATIONS.
                   ; INPUT
                   ;      (AL) = MODE BYTE FOR THE DMA
                   ;      (ES:BX) - ADDRESS TO READ/WRITE THE DATA
                   ; OUTPUT
                   ;      (AX) DESTROYED
                   ;--------------------------------------------
EECC               DMA_SETUP      PROC    NEAR
EECC 51                   PUSH  CX             ; SAVE THE REGISTER
EECD E60C                 OUT   DMA+12,AL      ; SET THE FIRST/LAST F/F
EECF E60B                 OUT   DMA+11,AL      ; OUTPUT THE MODE BYTE
EED1 8CC0                 MOV   AX,ES          ; GET THE ES VALUE
EED3 B104                 MOV   CL,4           ; SHIFT COUNT
EED5 D3C0                 ROL   AX,CL          ; ROTATE LEFT
EED7 8AE8                 MOV   CH,AL          ; GET HIGHEST NYBLE OF ES TO CH
EED9 24F0                 AND   AL,0F0H        ; ZERO THE LOW NYBBLE FROM SEGMENT
EEDB 03C3                 ADD   AX,BX          ; TEST FOR CARRY FROM ADDITION
EEDD 7302                 JNC   J33
EEDF FEC5                 INC   CH             ; CARRY MEANS HIGH 4 BITS MUST BE INC
EEE1               J33:
EEE1 50                   PUSH  AX             ; SAVE START ADDRESS
EEE2 E604                 OUT   DMA+4,AL       ; OUTPUT LOW ADDRESS
EEE4 8AC4                 MOV   AL,AH
EEE6 E604                 OUT   DMA+4,AL       ; OUTPUT HIGH ADDRESS
EEE8 8AC5                 MOV   AL,CH          ; GET HIGH 4 BITS
EEEA 240F                 AND   AL,0FH
EEEC E681                 OUT   081H,AL        ; OUTPUT THE HIGH 4 BITS TO PAGE REGISTER

                   ;------ DETERMINE COUNT

EEEE 8AE6                 MOV   AH,DH          ; NUMBER OF SECTORS
EEF0 2AC0                 SUB   AL,AL          ;  TIMES 256 INTO AX
EEF2 D1E8                 SHR   AX,1           ; SECTORS * 128 INTO AX
EEF4 50                   PUSH  AX
EEF5 BB0600               MOV   BX,6           ; GET THE BYTES/SECTOR PARM
EEF8 E875FF               CALL  GET_PARM
EEFB 8ACC                 MOV   CL,AH          ; USE AS SHIFT COUNT (0=128, 1=256 ETC)
EEFD 58                   POP   AX
EEFE D3E0                 SHL   AX,CL          ; MULTIPLY BY CORRECT AMOUNT
EF00 48                   DEC   AX             ; -1 FOR DMA VALUE
EF01 50                   PUSH  AX             ; SAVE COUNT VALUE
```

```
LOC  OBJ                   SOURCE

EF02 E605                          OUT     DMA+5,AL         ; LOW BYTE OF COUNT
EF04 8AC4                          MOV     AL,AH
EF06 E605                          OUT     DMA+5,AL         ; HIGH BYTE OF COUNT
EF08 59                            POP     CX               ; RECOVER COUNT VALUE
EF09 58                            POP     AX               ; RECOVER ADDRESS VALUE
EF0A 03C1                          ADD     AX,CX            ; ADD, TEST FOR 64K OVERFLOW
EF0C 59                            POP     CX               ; RECOVER REGISTER
EF0D B002                          MOV     AL,2             ; MODE FOR 8237
EF0F E60A                          OUT     DMA+10,AL        ; INITIALIZE THE DISKETTE CHANNEL
EF11 C3                            RET                      ; RETURN TO CALLER, CFL SET BY ABOVE IF ERROR
                           DMA_SETUP       ENDP
                           ;--------------------------------------------
                           ; CHK_STAT_2
                           ;       THIS ROUTINE HANDLES THE INTERRUPT RECEIVED AFTER
                           ;       A RECALIBRATE, SEEK, OR RESET TO THE ADAPTER.
                           ;       THE INTERRUPT IS WAITED FOR, THE INTERRUPT STATUS SENSED,
                           ;       AND THE RESULT RETURNED TO THE CALLER.
                           ; INPUT
                           ;       NONE
                           ; OUTPUT
                           ;       CY = 0 SUCCESS
                           ;       CY = 1 FAILURE -- ERROR IS IN DISKETTE_STATUS
                           ;       (AX) DESTROYED
                           ;--------------------------------------------
EF12                       CHK_STAT_2      PROC    NEAR
EF12 E81E00                        CALL    WAIT_INT                 ; WAIT FOR THE INTERRUPT
EF15 7214                          JC      J34                      ; IF ERROR, RETURN IT
EF17 B408                          MOV     AH,08H                   ; SENSE INTERRUPT STATUS COMMAND
EF19 E828FF                        CALL    NEC_OUTPUT
EF1C E84C00                        CALL    RESULTS                  ; READ IN THE RESULTS
EF1F 720A                          JC      J34                      ; CHK2_RETURN
EF21 A04200          R             MOV     AL,NEC_STATUS            ; GET THE FIRST STATUS BYTE
EF24 2460                          AND     AL,060H                  ; ISOLATE THE BITS
EF26 3C60                          CMP     AL,060H                  ; TEST FOR CORRECT VALUE
EF28 7402                          JZ      J35                      ; IF ERROR, GO MARK IT
EF2A F8                            CLC                              ; GOOD RETURN
EF2B                       J34:
EF2B C3                            RET                              ; RETURN TO CALLER
EF2C                       J35:                                     ; CHK2_ERROR
EF2C 800E410040      R             OR      DISKETTE_STATUS,BAD_SEEK
EF31 F9                            STC                              ; ERROR RETURN CODE
EF32 C3                            RET
                           CHK_STAT_2      ENDP
                           ;--------------------------------------------
                           ; WAIT_INT
                           ;       THIS ROUTINE WAITS FOR AN INTERRUPT TO OCCUR
                           ;       A TIME OUT ROUTINE TAKES PLACE DURING THE WAIT, SO
                           ;       THAT AN ERROR MAY BE RETURNED IF THE DRIVE IS NOT READY
                           ; INPUT
                           ;       NONE
                           ; OUTPUT
                           ;       CY = 0 SUCCESS
                           ;       CY = 1 FAILURE -- DISKETTE_STATUS IS SET ACCORDINGLY
                           ;       (AX) DESTROYED
                           ;--------------------------------------------
EF33                       WAIT_INT        PROC    NEAR
EF33 FB                            STI             ; TURN ON INTERRUPTS, JUST IN CASE
EF34 53                            PUSH    BX
EF35 51                            PUSH    CX                       ; SAVE REGISTERS
EF36 B302                          MOV     BL,2                     ; CLEAR THE COUNTERS
EF38 33C9                          XOR     CX,CX                    ; FOR 2 SECOND WAIT
EF3A                       J36:
EF3A F6063E0080      R             TEST    SEEK_STATUS,INT_FLAG     ; TEST FOR INTERRUPT OCCURRING
EF3F 750C                          JNZ     J37
EF41 E2F7                          LOOP    J36                      ; COUNT DOWN WHILE WAITING
EF43 FECB                          DEC     BL                       ; SECOND LEVEL COUNTER
EF45 75F3                          JNZ     J36
EF47 800E410080      R             OR      DISKETTE_STATUS,TIME_OUT         ; NOTHING HAPPENED
EF4C F9                            STC                              ; ERROR RETURN
EF4D                       J37:
EF4D 9C                            PUSHF                            ; SAVE CURRENT CARRY
EF4E 80263E007F      R             AND     SEEK_STATUS,NOT INT_FLAG         ; TURN OFF INTERRUPT FLAG
EF53 9D                            POPF                             ; RECOVER CARRY
EF54 59                            POP     CX
EF55 5B                            POP     BX              ; RECOVER REGISTERS
EF56 C3                            RET                     ; GOOD RETURN CODE COMES FROM TEST INST
                           WAIT_INT        ENDP
```

```
LOC OBJ                         SOURCE

                                ;--------------------------------------------
                                ; DISK_INT
                                ;       THIS ROUTINE HANDLES THE DISKETTE INTERRUPT
                                ;INPUT
                                ;       NONE
                                ; OUTPUT
                                ;       THE INTERRUPT FLAG IS SET IS SEEK_STATUS
                                ;--------------------------------------------
EF57                            DISK_INT        PROC    FAR
EF57 FB                                 STI                     ; RE ENABLE INTERRUPTS
EF58 1E                                 PUSH    DS
EF59 50                                 PUSH    AX
EF5A B84000          R                  MOV     AX,DATA
EF5D 8ED8                               MOV     DS,AX
EF5F 800E3E0080      R                  OR      SEEK_STATUS,INT_FLAG
EF64 B020                               MOV     AL,20H                  ; END OF INTERRUPT MARKER
EF66 E620                               OUT     20H,AL                  ; INTERRUPT CONTROL PORT
EF68 58                                 POP     AX
EF69 1F                                 POP     DS                      ; RECOVER SYSTEM
EF6A CF                                 IRET                            ; RETURN FROM INTERRUPT
                                DISK_INT        ENDP
                                ;--------------------------------------------
                                ; RESULTS
                                ;       THIS ROUTINE WILL READ ANYTHING THAT THE NEC CONTROLLER
                                ;       HAS TO SAY FOLLOWING AN INTERRUPT.
                                ; INPUT
                                ;       NONE
                                ; OUTPUT
                                ;       CY = 0  SUCCESSFUL TRANSFER
                                ;       CY = 1  FAILURE -- TIME OUT IN WAITING FOR STATUS
                                ;       NEC_STATUS AREA HAS STATUS BYTE LOADED INTO IT
                                ;       (AH) DESTROYED
                                ;--------------------------------------------
EF6B                            RESULTS PROC    NEAR
EF6B FC                                 CLD
EF6C BF4200          R                  MOV     DI,OFFSET NEC_STATUS    ; POINTER TO DATA AREA
EF6F 51                                 PUSH    CX                      ; SAVE COUNTER
EF70 52                                 PUSH    DX
EF71 53                                 PUSH    BX
EF72 B307                               MOV     BL,7                    ; MAX STATUS BYTES

                                ;------ WAIT FOR REQUEST FOR MASTER

EF74                            J38:                                    ; INPUT_LOOP
EF74 33C9                               XOR     CX,CX                   ; COUNTER
EF76 BAF403                             MOV     DX,03F4H                ; STATUS PORT
EF79                            J39:                                    ; WAIT FOR MASTER
EF79 EC                                 IN      AL,DX                   ; GET STATUS
EF7A A880                               TEST    AL,080H                 ; MASTER READY
EF7C 750C                               JNZ     J40A                    ; TEST_DIR
EF7E E2F9                               LOOP    J39                     ; WAIT_MASTER
EF80 800E410080      R                  OR      DISKETTE_STATUS,TIME_OUT
EF85                            J40:                                    ; RESULTS_ERROR
EF85 F9                                 STC                             ; SET ERROR RETURN
EF86 5B                                 POP     BX
EF87 5A                                 POP     DX
EF88 59                                 POP     CX
EF89 C3                                 RET

                                ;------ TEST THE DIRECTION BIT

EF8A EC                         J40A:   IN      AL,DX           ; GET STATUS REG AGAIN
EF8B A840                               TEST    AL,040H         ; TEST DIRECTION BIT
EF8D 7507                               JNZ     J42             ; OK TO READ STATUS
EF8F                            J41:                            ; NEC_FAIL
EF8F 800E410020      R                  OR      DISKETTE_STATUS,BAD_NEC
EF94 EBEF                               JMP     J40             ; RESULTS_ERROR

                                ;------ READ IN THE STATUS

EF96                            J42:                            ; INPUT_STAT
EF96 42                                 INC     DX              ; POINT AT DATA PORT
EF97 EC                                 IN      AL,DX           ; GET THE DATA
EF98 8805                               MOV     [DI],AL         ; STORE THE BYTE
EF9A 47                                 INC     DI              ; INCREMENT THE POINTER
EF9B B90A00                             MOV     CX,10           ; LOOP TO KILL TIME FOR NEC
```

```
LOC OBJ                 LINE    SOURCE

EF9E E2FE                       J43:    LOOP    J43
EFA0 4A                                 DEC     DX                  ; POINT AT STATUS PORT
EFA1 EC                                 IN      AL,DX               ; GET STATUS
EFA2 A810                               TEST    AL,010H             ; TEST FOR NEC STILL BUSY
EFA4 7406                               JZ      J44                 ; RESULTS DONE
EFA6 FECB                               DEC     BL                  ; DECREMENT THE STATUS COUNTER
EFA8 75CA                               JNZ     J38                 ; GO BACK FOR MORE
EFAA EBE3                               JMP     J41                 ; CHIP HAS FAILED

                                ;------ RESULT OPERATION IS DONE

EFAC                            J44:
EFAC 5B                                 POP     BX
EFAD 5A                                 POP     DX
EFAE 59                                 POP     CX                  ; RECOVER REGISTERS
EFAF C3                                 RET                         ; GOOD RETURN CODE FROM TEST INST
                                ;-------------------------------------------
                                ; NUM_TRANS
                                ;       THIS ROUTINE CALCULATES THE NUMBER OF SECTORS THAT
                                ;       WERE ACTUALLY TRANSFERRED TO/FROM THE DISKETTE
                                ; INPUT
                                ;       (CH) = CYLINDER OF OPERATION
                                ;       (CL) = START SECTOR OF OPERATION
                                ; OUTPUT
                                ;       (AL) = NUMBER ACTUALLY TRANSFERRED
                                ;       NO OTHER REGISTERS MODIFIED
                                ;-------------------------------------------
EFB0                            NUM_TRANS       PROC    NEAR
EFB0 A04500           R                 MOV     AL,NEC_STATUS+3         ; GET CYLINDER ENDED UP ON
EFB3 3AC5                               CMP     AL,CH                   ; SAME AS WE STARTED
EFB5 A04700           R                 MOV     AL,NEC_STATUS+5         ; GET ENDING SECTOR
EFB8 740A                               JZ      J45                     ; IF ON SAME CYL, THEN NO ADJUST
EFBA BB0800                             MOV     BX,8
EFBD E8B0FE                             CALL    GET_PARM                ; GET EOT VALUE
EFC0 8AC4                               MOV     AL,AH                   ;  INTO AL
EFC2 FEC0                               INC     AL                      ; USE EOT+1 FOR CALCULATION
EFC4 2AC1                       J45:    SUB     AL,CL                   ; SUBTRACT START FROM END
EFC6 C3                                 RET
                                NUM_TRANS       ENDP
                                RESULTS ENDP

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

EFC7                            DISK_BASE       LABEL   BYTE
EFC7 CF                                 DB      11001111B       ; SRT=C, HD UNLOAD=0F - 1ST SPECIFY BYTE
EFC8 02                                 DB      2               ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
EFC9 25                                 DB      MOTOR_WAIT      ; WAIT AFTER OPN TIL MOTOR OFF
EFCA 02                                 DB      2               ; 512 BYTES/SECTOR
EFCB 08                                 DB      8               ; EOT ( LAST SECTOR ON TRACK)
EFCC 2A                                 DB      02AH            ; GAP LENGTH
EFCD FF                                 DB      0FFH            ; DTL
EFCE 50                                 DB      050H            ; GAP LENGTH FOR FORMAT
EFCF F6                                 DB      0F6H            ; FILL BYTE FOR FORMAT
EFD0 19                                 DB      25              ; HEAD SETTLE TIME (MILLISECONDS)
EFD1 04                                 DB      4               ; MOTOR START TIME (1/8 SECONDS)
```

```
LOC OBJ                 SOURCE

                        ;--- INT 17 --------------------------------
                        ;PRINTER_IO
                        ;        THIS ROUTINE PROVIDES COMMUNICATION WITH THE PRINTER
                        ;        (AH)=0  PRINT THE CHARACTER IN (AL)
                        ;                ON RETURN, AH=1 IF CHARACTER COULD NOT BE PRINTED (TIME OUT)
                        ;                OTHER BITS SET AS ON NORMAL STATUS CALL
                        ;        (AH)=1  INITIALIZE THE PRINTER PORT
                        ;                RETURNS WITH (AH) SET WITH PRINTER STATUS
                        ;        (AH)=2  READ THE PRINTER STATUS INTO (AH)
                        ;                7       6       5       4       3       2-1     0
                        ;                |       |       |       |       |        |      _ TIME OUT
                        ;                |       |       |       |       |        _ UNUSED
                        ;                |       |       |       |       |_ 1 = I/O ERROR
                        ;                |       |       |       |_ 1 = SELECTED
                        ;                |       |       |_ 1 = OUT OF PAPER
                        ;                |       |_ 1 = ACKNOWLEDGE
                        ;                |_ 1 = BUSY
                        ;
                        ;        (DX) = PRINTER TO BE USED (0,1,2) CORRESPONDING TO ACTUAL VALUES
                        ;                IN PRINTER_BASE AREA
                        ; DATA AREA PRINTER_BASE CONTAINS THE BASE ADDRESS OF THE PRINTER CARD(S)
                        ;   AVAILABLE (LOCATED AT BEGINNING OF DATA SEGMENT, 408H ABSOLUTE, 3 WORDS)
                        ;REGISTERS       AH IS MODIFIED
                        ;                ALL OTHERS UNCHANGED
                        ;-------------------------------------------
                                ASSUME  CS:CODE,DS:DATA
EFD2                    PRINTER_IO      PROC    FAR
EFD2 FB                         STI                     ; INTERRUPTS BACK ON
EFD3 1E                         PUSH    DS              ; SAVE SEGMENT
EFD4 52                         PUSH    DX
EFD5 56                         PUSH    SI
EFD6 51                         PUSH    CX
EFD7 53                         PUSH    BX
EFD8 BE4000          R          MOV     SI,DATA
EFDB 8EDE                       MOV     DS,SI           ; ESTABLISH PRINTER SEGMENT
EFDD 8BF2                       MOV     SI,DX           ; GET PRINTER PARM
EFDF D1E6                       SHL     SI,1            ; WORD OFFSET INTO TABLE
EFE1 8B940800        R          MOV     DX,PRINTER_BASE[SI]     ; GET BASE ADDRESS FOR PRINTER CARD
EFE5 0BD2                       OR      DX,DX           ; TEST DX FOR ZERO, INDICATING NO PRINTER
EFE7 740C                       JZ      B1              ; RETURN
EFE9 0AE4                       OR      AH,AH           ; TEST FOR (AH)=0
EFEB 740E                       JZ      B2              ; PRINT_AL
EFED FECC                       DEC     AH              ; TEST FOR (AH)=1
EFEF 7442                       JZ      B8              ; INIT_PRT
EFF1 FECC                       DEC     AH              ; TEST FOR (AH)=2
EFF3 742A                       JZ      B5              ; PRINTER STATUS
EFF5                    B1:                             ; RETURN
EFF5 5B                         POP     BX
EFF6 59                         POP     CX
EFF7 5E                         POP     SI              ; RECOVER REGISTERS
EFF8 5A                         POP     DX              ; RECOVER REGISTERS
EFF9 1F                         POP     DS
EFFA CF                         IRET

                        ;------ PRINT THE CHARACTER IN (AL)

EFFB                    B2:
EFFB 50                         PUSH    AX              ; SAVE VALUE TO PRINT
EFFC B30A                       MOV     BL,10           ; TIME OUT VALUE
EFFE 33C9                       XOR     CX,CX           ; ESTABLISH SHIFT COUNT
F000 EE                         OUT     DX,AL           ; OUTPUT CHAR TO PORT
F001 42                         INC     DX              ; POINT TO STATUS PORT
F002                    B3:                             ; WAIT_BUSY
F002 EC                         IN      AL,DX           ; GET STATUS
F003 8AE0                       MOV     AH,AL           ; STATUS TO AH ALSO
F005 A880                       TEST    AL,80H          ; IS THE PRINTER CURRENTLY BUSY
F007 750E                       JNZ     B4              ; OUT_STROBE
F009 E2F7                       LOOP    B3              ; DECREMENT COUNT ON TIME OUT
F00B FECB                       DEC     BL
F00D 75F3                       JNZ     B3              ; WAIT FOR NOT BUSY
F00F 80CC01                     OR      AH,1            ; SET ERROR FLAG
F012 80E4F9                     AND     AH,0F9H         ; TURN OFF THE OTHER BITS
F015 EB14                       JMP     SHORT B7        ; RETURN WITH ERROR FLAG SET
F017                    B4:                             ; OUT_STROBE
F017 B00D                       MOV     AL,0DH          ; SET THE STROBE HIGH
F019 42                         INC     DX              ; STROBE IS BIT 0 OF PORT C OF 8255
```

```
F01A EE                        OUT     DX,AL
F01B B00C                      MOV     AL,0CH          ; SET THE STROBE LOW
F01D EE                        OUT     DX,AL
F01E 58                        POP     AX              ; RECOVER THE OUTPUT CHAR

                  ;------ PRINTER STATUS

F01F              B5:
F01F 50                        PUSH    AX              ; SAVE AL REG
F020              B6:
F020 8B940800   R              MOV     DX,PRINTER_BASE[SI]
F024 42                        INC     DX
F025 EC                        IN      AL,DX           ; GET PRINTER STATUS
F026 8AE0                      MOV     AH,AL
F028 80E4F8                    AND     AH,0F8H         ; TURN OFF UNUSED BITS
F02B              B7:                                  ; STATUS_SET
F02B 5A                        POP     DX              ; RECOVER AL REG
F02C 8AC2                      MOV     AL,DL           ; GET CHARACTER INTO AL
F02E 80F448                    XOR     AH,48H          ; FLIP A COUPLE OF BITS
F031 EBC2                      JMP     B1              ; RETURN FROM ROUTINE

                  ;------ INITIALIZE THE PRINTER PORT

F033              B8:
F033 50                        PUSH    AX              ; SAVE AL
F034 83C202                    ADD     DX,2            ; POINT TO OUTPUT PORT
F037 B008                      MOV     AL,8            ; SET INIT LINE LOW
F039 EE                        OUT     DX,AL
F03A B8E803                    MOV     AX,1000
F03D              B9:                                  ; INIT_LOOP
F03D 48                        DEC     AX              ; LOOP FOR RESET TO TAKE
F03E 75FD                      JNZ     B9              ; INIT_LOOP
F040 B00C                      MOV     AL,0CH          ; NO INTERRUPTS, NON AUTO LF, INIT HIGH
F042 EE                        OUT     DX,AL
F043 EBDB                      JMP     B6              ; PRT_STATUS_1
                  PRINTER_IO      ENDP
                  ;--- INT 10 --------------------------------
                  ; VIDEO_IO
                  ;        THESE ROUTINES PROVIDE THE CRT INTERFACE
                  ;        THE FOLLOWING FUNCTIONS ARE PROVIDED:
                  ;        (AH)=0  SET MODE (AL) CONTAINS MODE VALUE
                  ;                (AL)=0 40X25 BW (POWER ON DEFAULT)
                  ;                (AL)=1 40X25 COLOR
                  ;                (AL)=2  80X25 BW
                  ;                (AL)=3  80X25 COLOR
                  ;                GRAPHICS MODES
                  ;                (AL)=4  320X200 COLOR
                  ;                (AL)=5  320X200 BW
                  ;                (AL)=6  640X200 BW
                  ;                CRT MODE = 7 80X25 B&W CARD (USED INTERNAL TO VIDEO ONLY)
                  ;                *** NOTE BW MODES OPERATE SAME AS COLOR MODES, BUT COLOR
                  ;                        BURST IS NOT ENABLED
                  ;        (AH)=1  SET CURSOR TYPE
                  ;                (CH) =  BITS 4-0 = START LINE FOR CURSOR
                  ;                        ** HARDWARE WILL ALWAYS CAUSE BLINK
                  ;                        ** SETTING BIT 5 OR 6 WILL CAUSE ERRATIC BLINKING
                  ;                           OR NO CURSOR AT ALL
                  ;                (CL) =  BITS 4-0 = END LINE FOR CURSOR
                  ;        (AH)=2  SET CURSOR POSITION
                  ;                (DH,DL) = ROW,COLUMN  (0,0) IS UPPER LEFT
                  ;                (BH) = PAGE NUMBER (MUST BE 0 FOR GRAPHICS MODES)
                  ;        (AH)=3  READ CURSOR POSITION
                  ;                (BH) = PAGE NUMBER (MUST BE 0 FOR GRAPHICS MODES)
                  ;                ON EXIT (DH,DL) = ROW,COLUMN OF CURRENT CURSOR
                  ;                        (CH,CL) = CURSOR MODE CURRENTLY SET
                  ;        (AH)=4  READ LIGHT PEN POSITION
                  ;                ON EXIT:
                  ;                (AH) = 0 -- LIGHT PEN SWITCH NOT DOWN/NOT TRIGGERED
                  ;                (AH) = 1 -- VALID LIGHT PEN VALUE IN REGISTERS
                  ;                        (DH,DL) = ROW,COLUMN OF CHARACTER LP POSN
                  ;                        (CH) = RASTER LINE (0-199)
                  ;                        (BX) = PIXEL COLUMN (0-319,639)
                  ;        (AH)=5  SELECT ACTIVE DISPLAY PAGE (VALID ONLY FOR ALPHA MODES)
                  ;                (AL)=NEW PAGE VALUE (0-7 FOR MODES 0&1, 0-3 FOR MODES 2&3)
```

```
                ;       (AH)=6  SCROLL ACTIVE PAGE UP
                ;               (AL) = NUMBER OF LINES, INPUT LINES BLANKED AT BOTTOM OF WINDOW
                ;                       AL = 0 MEANS BLANK ENTIRE WINDOW
                ;               (CH,CL) = ROW,COLUMN OF UPPER LEFT CORNER OF SCROLL
                ;               (DH,DL) = ROW,COLUMN OF LOWER RIGHT CORNER OF SCROLL
                ;               (BH) = ATTRIBUTE TO BE USED ON BLANK LINE
                ;       (AH)=7  SCROLL ACTIVE PAGE DOWN
                ;               (AL) = NUMBER OF LINES, INPUT LINES BLANKED AT TOP OF WINDOW
                ;                       AL = 0 MEANS BLANK ENTIRE WINDOW
                ;               (CH,CL) = ROW,COLUMN OF UPPER LEFT CORNER OF SCROLL
                ;               (DH,DL) = ROW,COLUMN OF LOWER RIGHT CORNER OF SCROLL
                ;               (BH) = ATTRIBUTE TO BE USED ON BLANK LINE
                ;
                ;       CHARACTER HANDLING ROUTINES
                ;
                ;       (AH) = 8 READ ATTRIBUTE/CHARACTER AT CURRENT CURSOR POSITION
                ;               (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY)
                ;               ON EXIT:
                ;               (AL) = CHAR READ
                ;               (AH) = ATTRIBUTE OF CHARACTER READ (ALPHA MODES ONLY)
                ;       (AH) = 9 WRITE ATTRIBUTE/CHARACTER AT CURRENT CURSOR POSITION
                ;               (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY)
                ;               (CX) = COUNT OF CHARACTERS TO WRITE
                ;               (AL) = CHAR TO WRITE
                ;               (BL) = ATTRIBUTE OF CHARACTER (ALPHA)/COLOR OF CHAR (GRAPHICS)
                ;                       SEE NOTE ON WRITE DOT FOR BIT 7 OF BL = 1.
                ;       (AH) = 10 WRITE CHARACTER ONLY AT CURRENT CURSOR POSITION
                ;               (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY)
                ;               (CX) = COUNT OF CHARACTERS TO WRITE
                ;               (AL) = CHAR TO WRITE
                ;       FOR READ/WRITE CHARACTER INTERFACE WHILE IN GRAPHICS MODE, THE
                ;               CHARACTERS ARE FORMED FROM A CHARACTER GENERATOR IMAGE
                ;               MAINTAINED IN THE SYSTEM ROM.  ONLY THE 1ST 128 CHARS
                ;               ARE CONTAINED THERE.  TO READ/WRITE THE SECOND 128 CHARS,
                ;               THE USER MUST INITIALIZE THE POINTER AT INTERRUPT 1FH
                ;               (LOCATION 0007CH) TO POINT TO THE 1K BYTE TABLE CONTAINING
                ;               THE CODE POINTS FOR THE SECOND 128 CHARS (128-255).
                ;       FOR WRITE CHARACTER INTERFACE IN GRAPHICS MODE, THE REPLICATION FACTOR
                ;               CONTAINED IN (CX) ON ENTRY WILL PRODUCE VALID RESULTS ONLY
                ;               FOR CHARACTERS CONTAINED ON THE SAME ROW.  CONTINUATION TO
                ;               SUCCEEDING LINES WILL NOT PRODUCE CORRECTLY.
                ;
                ;       GRAPHICS INTERFACE
                ;       (AH) = 11 SET COLOR PALETTE
                ;               (BH) = PALLETTE COLOR ID BEING SET (0-127)
                ;               (BL) = COLOR VALUE TO BE USED WITH THAT COLOR ID
                ;                  NOTE: FOR THE CURRENT COLOR CARD, THIS ENTRY POINT HAS
                ;                               MEANING ONLY FOR 320X200 GRAPHICS.
                ;                       COLOR ID = 0 SELECTS THE BACKGROUND COLOR (0-15)
                ;                       COLOR ID = 1 SELECTS THE PALLETTE TO BE USED:
                ;                               0 = GREEN(1)/RED(2)/YELLOW(3)
                ;                               1 = CYAN(1)/MAGENTA(2)/WHITE(3)
                ;                       IN 40X25 OR 80X25 ALPHA MODES, THE VALUE SET FOR
                ;                               PALLETTE COLOR 0 INDICATES THE BORDER COLOR
                ;                               TO BE USED (VALUES 0-31, WHERE 16-31 SELECT THE
                ;                               HIGH INTENSITY BACKGROUND SET.
                ;       (AH) = 12 WRITE DOT
                ;               (DX) = ROW NUMBER
                ;               (CX) = COLUMN NUMBER
                ;               (AL) = COLOR VALUE
                ;                       IF BIT 7 OF AL = 1, THEN THE COLOR VALUE IS EXCLUSIVE
                ;                       OR'D WITH THE CURRENT CONTENTS OF THE DOT
                ;       (AH) = 13 READ DOT
                ;               (DX) = ROW NUMBER
                ;               (CX) = COLUMN NUMBER
                ;               (AL) RETURNS THE DOT READ
                ;
                ; ASCII TELETYPE ROUTINE FOR OUTPUT
                ;
                ;       (AH) = 14 WRITE TELETYPE
                ;               (AL) = CHAR TO WRITE
                ;               (BL) = FOREGROUND COLOR IN GRAPHICS MODE
                ;               (BH) = DISPLAY PAGE IN ALPHA MODE
                ;               NOTE -- SCREEN WIDTH IS CONTROLLED BY PREVIOUS MODE SET
                ;
```

```
LOC OBJ                 SOURCE

                        ;       (AH) = 15 CURRENT VIDEO STATE
                        ;               RETURNS THE CURRENT VIDEO STATE
                        ;               (AL) = MODE CURRENTLY SET ( SEE AH=0 FOR EXPLANATION)
                        ;               (AH) = NUMBER OF CHARACTER COLUMNS ON SCREEN
                        ;               (BH) = CURRENT ACTIVE DISPLAY PAGE
                        ;
                        ;       CS,SS,DS,ES,BX,CX,DX PRESERVED DURING CALL
                        ;       ALL OTHERS DESTROYED
                        ;------------------------------------------
                                ASSUME  CS:CODE,DS:DATA,ES:VIDEO_RAM

F045                    M1      LABEL   WORD    ; TABLE OF ROUTINES WITHIN VIDEO I/O
F045 FCF0          R            DW      OFFSET  SET_MODE
F047 CFF1          R            DW      OFFSET  SET_CTYPE
F049 F0F1          R            DW      OFFSET  SET_CPOS
F04B 1AF2          R            DW      OFFSET  READ_CURSOR
F04D A9F7          R            DW      OFFSET  READ_LPEN
F04F 30F2          R            DW      OFFSET  ACT_DISP_PAGE
F051 9CF2          R            DW      OFFSET  SCROLL_UP
F053 41F3          R            DW      OFFSET  SCROLL_DOWN
F055 70F3          R            DW      OFFSET  READ_AC_CURRENT
F057 C3F3          R            DW      OFFSET  WRITE_AC_CURRENT
F059 F6F3          R            DW      OFFSET  WRITE_C_CURRENT
F05B 54F2          R            DW      OFFSET  SET_COLOR
F05D 38F4          R            DW      OFFSET  WRITE_DOT
F05F 27F4          R            DW      OFFSET  READ_DOT
F061 22F7          R            DW      OFFSET  WRITE_TTY
F063 7AF2          R            DW      OFFSET  VIDEO_STATE
  0020                  M1L     EQU     $-M1

F065                    VIDEO_IO        PROC    NEAR
F065 FB                         STI             ; INTERRUPTS BACK ON
F066 FC                         CLD             ; SET DIRECTION FORWARD
F067 06                         PUSH    ES
F068 1E                         PUSH    DS              ; SAVE SEGMENT REGISTERS
F069 52                         PUSH    DX
F06A 51                         PUSH    CX
F06B 53                         PUSH    BX
F06C 56                         PUSH    SI
F06D 57                         PUSH    DI
F06E 50                         PUSH    AX              ; SAVE AX VALUE
F06F 8AC4                       MOV     AL,AH           ; GET INTO LOW BYTE
F071 32E4                       XOR     AH,AH           ; ZERO TO HIGH BYTE
F073 D1E0                       SAL     AX,1            ; *2 FOR TABLE LOOKUP
F075 8BF0                       MOV     SI,AX           ; PUT INTO SI FOR BRANCH
F077 3D2000                     CMP     AX,M1L          ; TEST FOR WITHIN RANGE
F07A 7204                       JB      M2              ; BRANCH AROUND BRANCH
F07C 58                         POP     AX              ; THROW AWAY THE PARAMETER
F07D E94701                     JMP     VIDEO_RETURN            ; DO NOTHING IF NOT IN RANGE
F080 B84000        R    M2:     MOV     AX,DATA
F083 8ED8                       MOV     DS,AX
F085 B800B8                     MOV     AX,0B800H       ; SEGMENT FOR COLOR CARD
F088 8B3E1000      R            MOV     DI,EQUIP_FLAG   ; GET EQUIPMENT SETTING
F08C 81E73000                   AND     DI,30H          ; ISOLATE CRT SWITCHES
F090 83FF30                     CMP     DI,30H          ; IS SETTING FOR BW CARD?
F093 7503                       JNE     M3
F095 B800B0                     MOV     AX,0B000H       ; SEGMENT FOR BW CARD
F098 8EC0               M3:     MOV     ES,AX           ; SET UP TO POINT AT VIDEO RAM AREAS
F09A 58                         POP     AX              ; RECOVER VALUE
F09B 8A264900      R            MOV     AH,CRT_MODE     ; GET CURRENT MODE INTO AH
F09F 2EFFA445F0    R            JMP     WORD PTR CS:[SI+OFFSET M1]
                        VIDEO_IO        ENDP
                        ;------------------------------------------
                        ; SET_MODE
                        ;       THIS ROUTINE INITIALIZES THE ATTACHMENT TO
                        ;       THE SELECTED MODE.  THE SCREEN IS BLANKED.
                        ; INPUT
                        ;       (AL) = MODE SELECTED (RANGE 0-9)
                        ; OUTPUT
                        ;       NONE
                        ;------------------------------------------

                        ;------ TABLES FOR USE IN SETTING OF MODE

F0A4                    VIDEO_PARMS     LABEL   BYTE
                        ;------ INIT_TABLE
```

```
LOC OBJ                  SOURCE

F0A4 38282D0A1F0619              DB      38H,28H,2DH,0AH,1FH,6,19H      ; SET UP FOR 40X25
F0AB 1C02070607                  DB      1CH,2,7,6,7
F0B0 00000000                    DB      0,0,0,0
 0010                    M4      EQU     $-VIDEO_PARMS

F0B4 71505A0A1F0619              DB      71H,50H,5AH,0AH,1FH,6,19H      ; SET UP FOR 80X25
F0BB 1C02070607                  DB      1CH,2,7,6,7
F0C0 00000000                    DB      0,0,0,0

F0C4 38282D0A7F0664              DB      38H,28H,2DH,0AH,7FH,6,64H      ; SET UP FOR GRAPHICS
F0CB 7002010607                  DB      70H,2,1,6,7
F0D0 00000000                    DB      0,0,0,0

F0D4 6150520F190619              DB      61H,50H,52H,0FH,19H,6,19H      ; SET UP FOR 80X25 B&W CARD
F0DB 19020D0B0C                  DB      19H,2,0DH,0BH,0CH
F0E0 00000000                    DB      0,0,0,0

F0E4                     M5      LABEL   WORD                   ; TABLE OF REGEN LENGTHS
F0E4 0008                        DW      2048           ; 40X25
F0E6 0010                        DW      4096           ; 80X25
F0E8 0040                        DW      16384          ; GRAPHICS
F0EA 0040                        DW      16384          ;

                         ;------ COLUMNS
F0EC                     M6      LABEL   BYTE
F0EC 2828505028285050            DB      40,40,80,80,40,40,80,80

                         ;------ C_REG_TAB
F0F4                     M7      LABEL   BYTE                   ; TABLE OF MODE SETS
F0F4 2C282D292A2E1E29            DB      2CH,28H,2DH,29H,2AH,2EH,1EH,29H ;

F0FC                     SET_MODE        PROC    NEAR
F0FC BAD403                      MOV     DX,03D4H               ; ADDRESS OF COLOR CARD
F0FF B300                        MOV     BL,0                   ; MODE SET FOR COLOR CARD
F101 83FF30                      CMP     DI,30H                 ; IS BW CARD INSTALLED
F104 7507                        JNE     M8                     ; OK WITH COLOR
F106 B007                        MOV     AL,7                   ; INDICATE BW CARD MODE
F108 BAB403                      MOV     DX,03B4H               ; ADDRESS OF BW CARD
F10B FEC3                        INC     BL                     ; MODE SET FOR BW CARD
F10D 8AE0                M8:     MOV     AH,AL          ; SAVE MODE IN AH
F10F A24900         R            MOV     CRT_MODE,AL    ; SAVE IN GLOBAL VARIABLE
F112 89166300       R            MOV     ADDR_6845,DX   ; SAVE ADDRESS OF BASE
F116 1E                          PUSH    DS             ; SAVE POINTER TO DATA SEGMENT
F117 50                          PUSH    AX             ; SAVE MODE
F118 52                          PUSH    DX             ; SAVE OUTPUT PORT VALUE
F119 83C204                      ADD     DX,4           ; POINT TO CONTROL REGISTER
F11C 8AC3                        MOV     AL,BL          ; GET MODE SET FOR CARD
F11E EE                          OUT     DX,AL          ; RESET VIDEO
F11F 5A                          POP     DX             ; BACK TO BASE REGISTER
F120 2BC0                        SUB     AX,AX          ; SET UP FOR ABS0 SEGMENT
F122 8ED8                        MOV     DS,AX          ; ESTABLISH VECTOR TABLE ADDRESSING
                                 ASSUME  DS:ABS0
F124 C51E7400                    LDS     BX,PARM_PTR    ; GET POINTER TO VIDEO PARMS
F128 58                          POP     AX             ; RECOVER PARMS
                                 ASSUME  DS:CODE
F129 B91000                      MOV     CX,M4          ; LENGTH OF EACH ROW OF TABLE
F12C 80FC02                      CMP     AH,2           ; DETERMINE WHICH ONE TO USE
F12F 7210                        JC      M9             ; MODE IS 0 OR 1
F131 03D9                        ADD     BX,CX          ; MOVE TO NEXT ROW OF INIT TABLE
F133 80FC04                      CMP     AH,4
F136 7209                        JC      M9             ; MODE IS 2 OR 3
F138 03D9                        ADD     BX,CX          ; MOVE TO GRAPHICS ROW OF INIT_TABLE
F13A 80FC07                      CMP     AH,7
F13D 7202                        JC      M9             ; MODE IS 4,5, OR 6
F13F 03D9                        ADD     BX,CX          ; MOVE TO BW CARD ROW OF INIT_TABLE

                         ;------ BX POINTS TO CORRECT ROW OF INITIALIZATION TABLE

F141                     M9:                            ; OUT_INIT
F141 50                          PUSH    AX             ; SAVE MODE IN AH
F142 32E4                        XOR     AH,AH          ; AH WILL SERVE AS REGISTER NUMBER DURING LOOP

                         ;------ LOOP THROUGH TABLE, OUTPUTTTING REG ADDRESS, THEN VALUE FROM TABLE

F144                     M10:                           ; INIT LOOP
F144 8AC4                        MOV     AL,AH          ; GET 6845 REGISTER NUMBER
```

```
LOC OBJ                        SOURCE

F146 EE                                OUT     DX,AL
F147 42                                INC     DX                  ; POINT TO DATA PORT
F148 FEC4                              INC     AH                  ; NEXT REGISTER VALUE
F14A 8A07                              MOV     AL,[BX]             ; GET TABLE VALUE
F14C EE                                OUT     DX,AL               ; OUT TO CHIP
F14D 43                                INC     BX                  ; NEXT IN TABLE
F14E 4A                                DEC     DX                  ; BACK TO POINTER REGISTER
F14F E2F3                              LOOP    M10                 ; DO THE WHOLE TABLE
F151 58                                POP     AX                  ; GET MODE BACK
F152 1F                                POP     DS                  ; RECOVER SEGMENT VALUE
                                       ASSUME  DS:DATA

                               ;------ FILL REGEN AREA WITH BLANK

F153 33FF                              XOR     DI,DI               ; SET UP POINTER FOR REGEN
F155 893E4E00          R               MOV     CRT_START,DI        ; START ADDRESS SAVED IN GLOBAL
F159 C606620000        R               MOV     ACTIVE_PAGE,0       ; SET PAGE VALUE
F15E B90020                            MOV     CX,8192             ; NUMBER OF WORDS IN COLOR CARD
F161 80FC04                            CMP     AH,4                ; TEST FOR GRAPHICS
F164 720C                              JC      M12                 ; NO_GRAPHICS_INIT
F166 80FC07                            CMP     AH,7                ; TEST FOR BW CARD
F169 7404                              JE      M11                 ; BW_CARD_INIT
F16B 33C0                              XOR     AX,AX               ; FILL FOR GRAPHICS MODE
F16D EB06                              JMP     SHORT M13           ; CLEAR_BUFFER
F16F                           M11:                                ; BW_CARD_INIT
F16F B90008                            MOV     CX,2048             ; BUFFER SIZE ON BW CARD
F172                           M12:                                ; NO_GRAPHICS_INIT
F172 B82007                            MOV     AX,' '+7*256        ; FILL CHAR FOR ALPHA
F175                           M13:                                ; CLEAR_BUFFER
F175 F3                                REP     STOSW               ; FILL THE REGEN BUFFER WITH BLANKS
F176 AB

                               ;------ ENABLE VIDEO AND CORRECT PORT SETTING

F177 C70660006700      R               MOV     CURSOR_MODE,67H     ; SET CURRENT CURSOR MODE
F17D A04900            R               MOV     AL,CRT_MODE         ; GET THE MODE
F180 32E4                              XOR     AH,AH               ;  INTO AX REGISTER
F182 8BF0                              MOV     SI,AX               ; TABLE POINTER, INDEXED BY MODE
F184 8B166300          R               MOV     DX,ADDR_6845        ; PREPARE TO OUTPUT TO VIDEO ENABLE PORT
F188 83C204                            ADD     DX,4
F18B 2E8A84F4F0        R               MOV     AL,CS:[SI+OFFSET M7]
F190 EE                                OUT     DX,AL               ; SET VIDEO ENABLE PORT
F191 A26500            R               MOV     CRT_MODE_SET,AL     ; SAVE THAT VALUE

                               ;------ DETERMINE NUMBER OF COLUMNS, BOTH FOR ENTIRE DISPLAY
                               ;------ AND THE NUMBER TO BE USED FOR TTY INTERFACE

F194 2E8A84ECF0        R               MOV     AL,CS:[SI + OFFSET M6]
F199 32E4                              XOR     AH,AH
F19B A34A00            R               MOV     CRT_COLS,AX         ; NUMBER OF COLUMNS IN THIS SCREEN

                               ;------ SET CURSOR POSITIONS

F19E 81E60E00                          AND     SI,0EH              ; WORD OFFSET INTO CLEAR LENGTH TABLE
F1A2 2E8B8CE4F0        R               MOV     CX,CS:[SI + OFFSET M5]  ; LENGTH TO CLEAR
F1A7 890E4C00          R               MOV     CRT_LEN,CX          ; SAVE LENGTH OF CRT -- NOT USED FOR BW
F1AB B90800                            MOV     CX,8                ; CLEAR ALL CURSOR POSITIONS
F1AE BF5000            R               MOV     DI,OFFSET CURSOR_POSN
F1B1 1E                                PUSH    DS                  ; ESTABLISH SEGMENT
F1B2 07                                POP     ES                  ;   ADDRESSING
F1B3 33C0                              XOR     AX,AX
F1B5 F3                                REP     STOSW               ; FILL WITH ZEROES
F1B6 AB

                               ;------ SET UP OVERSCAN REGISTER

F1B7 42                                INC     DX                  ; SET OVERSCAN PORT TO A DEFAULT
F1B8 B030                              MOV     AL,30H              ; VALUE OF 30H FOR ALL MODES EXCEPT 640X200
F1BA 803E490006        R               CMP     CRT_MODE,6          ; SEE IF THE MODE IS 640X200 BW
F1BF 7502                              JNZ     M14                 ; IF IT ISNT 640X200, THEN GOTO REGULAR
F1C1 B03F                              MOV     AL,3FH              ; IF IT IS 640X200, THEN PUT IN 3FH
F1C3 EE                        M14:    OUT     DX,AL               ; OUTPUT THE CORRECT VALUE TO 3D9 PORT
F1C4 A26600            R               MOV     CRT_PALLETTE,AL     ; SAVE THE VALUE FOR FUTURE USE

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

```

```
LOC OBJ                SOURCE

F1C7                   VIDEO_RETURN:
F1C7 5F                        POP     DI
F1C8 5E                        POP     SI
F1C9 5B                        POP     BX
F1CA                   M15:                            ; VIDEO_RETURN_C
F1CA 59                        POP     CX
F1CB 5A                        POP     DX
F1CC 1F                        POP     DS
F1CD 07                        POP     ES              ; RECOVER SEGMENTS
F1CE CF                        IRET                    ; ALL DONE
                       SET_MODE        ENDP
                       ;--------------------------------------------
                       ; SET_CTYPE
                       ;       THIS ROUTINE SETS THE CURSOR VALUE
                       ; INPUT
                       ;       (CX) HAS CURSOR VALUE CH-START LINE, CL-STOP LINE
                       ; OUTPUT
                       ;       NONE
                       ;--------------------------------------------
F1CF                   SET_CTYPE       PROC    NEAR
F1CF B40A                      MOV     AH,10           ; 6845 REGISTER FOR CURSOR SET
F1D1 890E6000      R           MOV     CURSOR_MODE,CX  ; SAVE IN DATA AREA
F1D5 E80200                    CALL    M16             ; OUTPUT CX REG
F1D8 EBED                      JMP     VIDEO_RETURN

                       ;------ THIS ROUTINE OUTPUTS THE CX REGISTER TO THE 6845 REGS NAMED IN AH

F1DA                   M16:
F1DA 8B166300      R           MOV     DX,ADDR_6845    ; ADDRESS REGISTER
F1DE 8AC4                      MOV     AL,AH           ; GET VALUE
F1E0 EE                        OUT     DX,AL           ; REGISTER SET
F1E1 42                        INC     DX              ; DATA REGISTER
F1E2 8AC5                      MOV     AL,CH           ; DATA
F1E4 EE                        OUT     DX,AL
F1E5 4A                        DEC     DX
F1E6 8AC4                      MOV     AL,AH
F1E8 FEC0                      INC     AL              ; POINT TO OTHER DATA REGISTER
F1EA EE                        OUT     DX,AL           ; SET FOR SECOND REGISTER
F1EB 42                        INC     DX
F1EC 8AC1                      MOV     AL,CL           ; SECOND DATA VALUE
F1EE EE                        OUT     DX,AL
F1EF C3                        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
                       ;--------------------------------------------
F1F0                   SET_CPOS        PROC    NEAR
F1F0 8ACF                      MOV     CL,BH
F1F2 32ED                      XOR     CH,CH           ; ESTABLISH LOOP COUNT
F1F4 D1E1                      SAL     CX,1            ; WORD OFFSET
F1F6 8BF1                      MOV     SI,CX           ; USE INDEX REGISTER
F1F8 89945000      R           MOV     [SI+OFFSET CURSOR_POSN],DX      ; SAVE THE POINTER
F1FC 383E6200      R           CMP     ACTIVE_PAGE,BH
F200 7505                      JNZ     M17             ; SET_CPOS_RETURN
F202 8BC2                      MOV     AX,DX           ; GET ROW/COLUMN TO AX
F204 E80200                    CALL    M18             ; CURSOR_SET
F207                   M17:                            ; SET_CPOS_RETURN
F207 EBBE                      JMP     VIDEO_RETURN
                       SET_CPOS        ENDP

                       ;------ SET CURSOR POSITION, AX HAS ROW/COLUMN FOR CURSOR

F209                   M18     PROC    NEAR
F209 E87F00                    CALL    POSITION        ; DETERMINE LOCATION IN REGEN BUFFER
F20C 8BC8                      MOV     CX,AX
F20E 030E4E00      R           ADD     CX,CRT_START    ; ADD IN THE START ADDRESS FOR THIS PAGE
F212 D1F9                      SAR     CX,1            ; DIVIDE BY 2 FOR CHAR ONLY COUNT
F214 B40E                      MOV     AH,14           ; REGISTER NUMBER FOR CURSOR
```

```
LOC OBJ                    SOURCE

F216 E8C1FF                       CALL    M16             ; OUTPUT THE VALUE TO THE 6845
F219 C3                           RET
                           M18    ENDP
                           ;-----------------------------------------
                           ; READ_CURSOR
                           ;      THIS ROUTINE READS THE CURRENT CURSOR VALUE FROM THE
                           ;      6845, FORMATS IT, AND SENDS IT BACK TO THE CALLER
                           ; INPUT
                           ;      BH - PAGE OF CURSOR
                           ; OUTPUT
                           ;      DX - ROW, COLUMN OF THE CURRENT CURSOR POSITION
                           ;      CX - CURRENT CURSOR MODE
                           ;-----------------------------------------
F21A                       READ_CURSOR    PROC    NEAR
F21A 8ADF                         MOV     BL,BH
F21C 32FF                         XOR     BH,BH
F21E D1E3                         SAL     BX,1            ; WORD OFFSET
F220 8B975000        R            MOV     DX,[BX+OFFSET CURSOR_POSN]
F224 8B0E6000        R            MOV     CX,CURSOR_MODE
F228 5F                           POP     DI
F229 5E                           POP     SI
F22A 5B                           POP     BX
F22B 58                           POP     AX      ; DISCARD SAVED CX AND DX
F22C 58                           POP     AX
F22D 1F                           POP     DS
F22E 07                           POP     ES
F22F CF                           IRET
                           READ_CURSOR    ENDP
                           ;-----------------------------------------
                           ; ACT_DISP_PAGE
                           ;      THIS ROUTINE SETS THE ACTIVE DISPLAY PAGE, ALLOWING
                           ;      THE FULL USE OF THE RAM SET ASIDE FOR THE VIDEO ATTACHMENT
                           ; INPUT
                           ;      AL HAS THE NEW ACTIVE DISPLAY PAGE
                           ; OUTPUT
                           ;      THE 6845 IS RESET TO DISPLAY THAT PAGE
                           ;-----------------------------------------
F230                       ACT_DISP_PAGE  PROC    NEAR
F230 A26200          R            MOV     ACTIVE_PAGE,AL  ; SAVE ACTIVE PAGE VALUE
F233 8B0E4C00        R            MOV     CX,CRT_LEN      ; GET SAVED LENGTH OF REGEN BUFFER
F237 98                           CBW                     ; CONVERT AL TO WORD
F238 50                           PUSH    AX              ; SAVE PAGE VALUE
F239 F7E1                         MUL     CX              ; DISPLAY PAGE TIMES REGEN LENGTH
F23B A34E00          R            MOV     CRT_START,AX    ; SAVE START ADDRESS FOR LATER REQUIREMENTS
F23E 8BC8                         MOV     CX,AX           ; START ADDRESS TO CX
F240 D1F9                         SAR     CX,1            ; DIVIDE BY 2 FOR 6845 HANDLING
F242 B40C                         MOV     AH,12           ; 6845 REGISTER FOR START ADDRESS
F244 E893FF                       CALL    M16
F247 5B                           POP     BX              ; RECOVER PAGE VALUE
F248 D1E3                         SAL     BX,1            ; *2 FOR WORD OFFSET
F24A 8B875000        R            MOV     AX,[BX + OFFSET CURSOR_POSN]    ; GET CURSOR FOR THIS PAGE
F24E E8B8FF                       CALL    M18             ; SET THE CURSOR POSITION
F251 E973FF                       JMP     VIDEO_RETURN
                           ACT_DISP_PAGE  ENDP
                           ;-----------------------------------------
                           ; SET COLOR
                           ;      THIS ROUTINE WILL ESTABLISH THE BACKGROUND COLOR, THE OVERSCAN COLOR,
                           ;      AND THE FOREGROUND COLOR SET FOR MEDIUM RESOLUTION GRAPHICS
                           ; INPUT
                           ;      (BH) HAS COLOR ID
                           ;              IF BH=0, THE BACKGROUND COLOR VALUE IS SET
                           ;                      FROM THE LOW BITS OF BL (0-31)
                           ;              IF BH=1, THE PALLETTE SELECTION IS MADE
                           ;                      BASED ON THE LOW BIT OF BL:
                           ;                              0 = GREEN, RED, YELLOW FOR COLORS 1,2,3
                           ;                              1 = BLUE, CYAN, MAGENTA FOR COLORS 1,2,3
                           ;      (BL) HAS THE COLOR VALUE TO BE USED
                           ; OUTPUT
                           ;      THE COLOR SELECTION IS UPDATED
                           ;-----------------------------------------
F254                       SET_COLOR      PROC    NEAR
F254 8B166300        R            MOV     DX,ADDR_6845    ; I/O PORT FOR PALETTE
F258 83C205                       ADD     DX,5            ; OVERSCAN PORT
F25B A06600          R            MOV     AL,CRT_PALLETTE ; GET THE CURRENT PALLETTE VALUE
F25E 0AFF                         OR      BH,BH           ; IS THIS COLOR 0?
```

```
LOC OBJ              LINE    SOURCE

F260 750E                            JNZ     M20              ; OUTPUT COLOR 1
                             ;------ HANDLE COLOR 0 BY SETTING THE BACKGROUND COLOR
F262 24E0                            AND     AL,0E0H          ; TURN OFF LOW 5 BITS OF CURRENT
F264 80E31F                          AND     BL,01FH          ; TURN OFF HIGH 3 BITS OF INPUT VALUE
F267 0AC3                            OR      AL,BL            ; PUT VALUE INTO REGISTER
F269                         M19:                             ; OUTPUT THE PALLETTE
F269 EE                              OUT     DX,AL            ;output color selection to 3d9 port
F26A A26600          R               MOV     CRT_PALLETTE,AL  ; SAVE THE COLOR VALUE
F26D E957FF                          JMP     VIDEO_RETURN
                             ;------ HANDLE COLOR 1 BY SELECTING THE PALLETTE TO BE USED
F270                         M20:
F270 24DF                            AND     AL,0DFH          ; TURN OFF PALLETTE SELECT BIT
F272 D0EB                            SHR     BL,1             ; TEST THE LOW ORDER BIT OF BL
F274 73F3                            JNC     M19              ; ALREADY DONE
F276 0C20                            OR      AL,20H           ; TURN ON PALLETTE SELECT BIT
F278 EBEF                            JMP     M19              ; GO DO IT
                             SET_COLOR       ENDP
                             ;--------------------------------------------
                             ;VIDEO STATE
                             ;  RETURNS THE CURRENT VIDEO STATE IN AX
                             ;  AH = NUMBER OF COLUMNS ON THE SCREEN
                             ;  AL = CURRENT VIDEO MODE
                             ;  BH = CURRENT ACTIVE PAGE
                             ;--------------------------------------------
F27A                         VIDEO_STATE     PROC    NEAR
F27A 8A264A00        R               MOV     AH,BYTE PTR CRT_COLS    ; GET NUMBER OF COLUMNS
F27E A04900          R               MOV     AL,CRT_MODE             ; CURRENT MODE
F281 8A3E6200        R               MOV     BH,ACTIVE_PAGE          ; GET CURRENT ACTIVE PAGE
F285 5F                              POP     DI                      ; RECOVER REGISTERS
F286 5E                              POP     SI                      ;
F287 59                              POP     CX                      ; DISCARD SAVED BX
F288 E93FFF                          JMP     M15                     ; RETURN TO CALLER
                             VIDEO_STATE     ENDP
                             ;------------------------------------
                             ; POSITION
                             ;       THIS SERVICE ROUTINE CALCULATES THE REGEN BUFFER ADDRESS
                             ;       OF A CHARACTER IN THE ALPHA MODE
                             ;INPUT
                             ;       AX = ROW, COLUMN POSITION
                             ; OUTPUT
                             ;       AX = OFFSET OF CHAR POSITION IN REGEN BUFFER
                             ;------------------------------------
F28B                         POSITION        PROC    NEAR
F28B 53                              PUSH    BX               ; SAVE REGISTER
F28C 8BD8                            MOV     BX,AX
F28E 8AC4                            MOV     AL,AH            ; ROWS TO AL
F290 F6264A00        R               MUL     BYTE PTR CRT_COLS       ; DETERMINE BYTES TO ROW
F294 32FF                            XOR     BH,BH
F296 03C3                            ADD     AX,BX            ; ADD IN COLUMN VALUE
F298 D1E0                            SAL     AX,1             ; * 2 FOR ATTRIBUTE BYTES
F29A 5B                              POP     BX
F29B C3                              RET
                             POSITION        ENDP
                             ;------------------------------------------
                             ; SCROLL UP
                             ;       THIS ROUTINE MOVES A BLOCK OF CHARACTERS UP
                             ;       ON THE SCREEN
                             ;INPUT
                             ;       (AH) = CURRENT CRT MODE
                             ;       (AL) = NUMBER OF ROWS TO SCROLL
                             ;       (CX) = ROW/COLUMN OF UPPER LEFT CORNER
                             ;       (DX) = ROW/COLUMN OF LOWER RIGHT CORNER
                             ;       (BH) = ATTRIBUTE TO BE USED ON BLANKED LINE
                             ;       (DS) = DATA SEGMENT
                             ;       (ES) = REGEN BUFFER SEGMENT
                             ;OUTPUT
                             ;       NONE -- THE REGEN BUFFER IS MODIFIED
                             ;------------------------------------------
                                     ASSUME  CS:CODE,DS:DATA,ES:DATA
F29C                         SCROLL_UP       PROC    NEAR
F29C 8AD8                            MOV     BL,AL            ; SAVE LINE COUNT IN BL
F29E 80FC04                          CMP     AH,4             ; TEST FOR GRAPHICS MODE
```

```
LOC OBJ                 SOURCE

F2A1 7208                       JC      N1              ; HANDLE SEPARATELY
F2A3 80FC07                     CMP     AH,7            ; TEST FOR BW CARD
F2A6 7403                       JE      N1
F2A8 E9F301                     JMP     GRAPHICS_UP
F2AB                    N1:                             ; UP_CONTINUE
F2AB 53                         PUSH    BX              ; SAVE FILL ATTRIBUTE IN BH
F2AC 8BC1                       MOV     AX,CX           ; UPPER LEFT POSITION
F2AE E83900                     CALL    SCROLL_POSITION ; DO SETUP FOR SCROLL
F2B1 7433                       JZ      N7              ; BLANK_FIELD
F2B3 03F0                       ADD     SI,AX           ; FROM ADDRESS
F2B5 8AE6                       MOV     AH,DH           ; # ROWS IN BLOCK
F2B7 2AE3                       SUB     AH,BL           ; # ROWS TO BE MOVED
F2B9                    N2:                             ; ROW_LOOP
F2B9 E87500                     CALL    N10             ; MOVE ONE ROW
F2BC 03F5                       ADD     SI,BP
F2BE 03FD                       ADD     DI,BP           ; POINT TO NEXT LINE IN BLOCK
F2C0 FECC                       DEC     AH              ; COUNT OF LINES TO MOVE
F2C2 75F5                       JNZ     N2              ; ROW_LOOP
F2C4                    N3:                             ; CLEAR_ENTRY
F2C4 58                         POP     AX              ; RECOVER ATTRIBUTE IN AH
F2C5 B020                       MOV     AL,' '          ; FILL WITH BLANKS
F2C7                    N4:                             ; CLEAR_LOOP
F2C7 E87000                     CALL    N11             ; CLEAR THE ROW
F2CA 03FD                       ADD     DI,BP           ; POINT TO NEXT LINE
F2CC FECB                       DEC     BL              ; COUNTER OF LINES TO SCROLL
F2CE 75F7                       JNZ     N4              ; CLEAR_LOOP
F2D0                    N5:                             ; SCROLL_END
F2D0 B84000        R            MOV     AX,DATA         ; GET LOCATION
F2D3 8ED8                       MOV     DS,AX
F2D5 803E490007    R            CMP     CRT_MODE,7      ; IS THIS THE BLACK AND WHITE CARD
F2DA 7407                       JE      N6              ; IF SO, SKIP THE MODE RESET
F2DC A06500        R            MOV     AL,CRT_MODE_SET ; GET THE VALUE OF THE MODE SET
F2DF BAD803                     MOV     DX,03D8H        ; ALWAYS SET COLOR CARD PORT
F2E2 EE                         OUT     DX,AL
F2E3                    N6:                             ; VIDEO_RET_HERE
F2E3 E9E1FE                     JMP     VIDEO_RETURN
F2E6                    N7:                             ; BLANK_FIELD
F2E6 8ADE                       MOV     BL,DH           ; GET ROW COUNT
F2E8 EBDA                       JMP     N3              ; GO CLEAR THAT AREA
                        SCROLL_UP       ENDP

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

F2EA                    SCROLL_POSITION PROC    NEAR
F2EA 803E490002    R            CMP     CRT_MODE,2      ; TEST FOR SPECIAL CASE HERE
F2EF 7219                       JB      N9              ; HAVE TO HANDLE 80X25 SEPARATELY
F2F1 803E490003    R            CMP     CRT_MODE,3
F2F6 7712                       JA      N9

                        ;------ 80X25 COLOR CARD SCROLL

F2F8 52                         PUSH    DX
F2F9 BADA03                     MOV     DX,3DAH         ; GUARANTEED TO BE COLOR CARD HERE
F2FC 50                         PUSH    AX
F2FD                    N8:                             ; WAIT_DISP_ENABLE
F2FD EC                         IN      AL,DX           ; GET PORT
F2FE A808                       TEST    AL,8            ; WAIT FOR VERTICAL RETRACE
F300 74FB                       JZ      N8              ; WAIT_DISP_ENABLE
F302 B025                       MOV     AL,25H
F304 BAD803                     MOV     DX,03D8H
F307 EE                         OUT     DX,AL           ; TURN OFF VIDEO
F308 58                         POP     AX              ; DURING VERTICAL RETRACE
F309 5A                         POP     DX
F30A E87EFF             N9:     CALL    POSITION        ; CONVERT TO REGEN POINTER
F30D 03064E00      R            ADD     AX,CRT_START    ; OFFSET OF ACTIVE PAGE
F311 8BF8                       MOV     DI,AX           ; TO ADDRESS FOR SCROLL
F313 8BF0                       MOV     SI,AX           ; FROM ADDRESS FOR SCROLL
F315 2BD1                       SUB     DX,CX           ; DX = #ROWS, #COLS IN BLOCK
F317 FEC6                       INC     DH
F319 FEC2                       INC     DL              ; INCREMENT FOR 0 ORIGIN
F31B 32ED                       XOR     CH,CH           ; SET HIGH BYTE OF COUNT TO ZERO
F31D 8B2E4A00      R            MOV     BP,CRT_COLS     ; GET NUMBER OF COLUMNS IN DISPLAY
F321 03ED                       ADD     BP,BP           ; TIMES 2 FOR ATTRIBUTE BYTE
F323 8AC3                       MOV     AL,BL           ; GET LINE COUNT
F325 F6264A00      R            MUL     BYTE PTR CRT_COLS       ; DETERMINE OFFSET TO FROM ADDRESS
F329 03C0                       ADD     AX,AX           ; *2 FOR ATTRIBUTE BYTE
```

```
LOC OBJ              SOURCE

F32B 06                       PUSH    ES              ; ESTABLISH ADDRESSING TO REGEN BUFFER
F32C 1F                       POP     DS              ;  FOR BOTH POINTERS
F32D 80FB00                   CMP     BL,0            ; 0 SCROLL MEANS BLANK FIELD
F330 C3                       RET                     ; RETURN WITH FLAGS SET
                     SCROLL_POSITION ENDP

                     ;------ MOVE_ROW
F331                 N10      PROC    NEAR
F331 8ACA                     MOV     CL,DL           ; GET # OF COLS TO MOVE
F333 56                       PUSH    SI
F334 57                       PUSH    DI              ; SAVE START ADDRESS
F335 F3                       REP     MOVSW           ; MOVE THAT LINE ON SCREEN
F336 A5
F337 5F                       POP     DI
F338 5E                       POP     SI              ; RECOVER ADDRESSES
F339 C3                       RET
                     N10      ENDP

                     ;------ CLEAR_ROW
F33A                 N11      PROC    NEAR
F33A 8ACA                     MOV     CL,DL           ; GET # COLUMNS TO CLEAR
F33C 57                       PUSH    DI
F33D F3                       REP     STOSW           ; STORE THE FILL CHARACTER
F33E AB
F33F 5F                       POP     DI
F340 C3                       RET
                     N11      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
                     ;        (DX) = LOWER RIGHT CORNER OF REGION
                     ;        (BH) = FILL CHARACTER
                     ;        (DS) = DATA SEGMENT
                     ;        (ES) = REGEN SEGMENT
                     ;OUPUT
                     ;        NONE -- SCREEN IS SCROLLED
                     ;-------------------------------------
F341                 SCROLL_DOWN      PROC    NEAR
F341 FD                       STD                     ; DIRECTION FOR SCROLL DOWN
F342 8AD8                     MOV     BL,AL           ; LINE COUNT TO BL
F344 80FC04                   CMP     AH,4            ; TEST FOR GRAPHICS
F347 7208                     JC      N12
F349 80FC07                   CMP     AH,7            ; TEST FOR BW CARD
F34C 7403                     JE      N12
F34E E9A601                   JMP     GRAPHICS_DOWN
F351                 N12:                             ; CONTINUE_DOWN
F351 53                       PUSH    BX              ; SAVE ATTRIBUTE IN BH
F352 8BC2                     MOV     AX,DX           ; LOWER RIGHT CORNER
F354 E893FF                   CALL    SCROLL_POSITION ; GET REGEN LOCATION
F357 7420                     JZ      N16
F359 2BF0                     SUB     SI,AX           ; SI IS FROM ADDRESS
F35B 8AE6                     MOV     AH,DH           ; GET TOTAL # ROWS
F35D 2AE3                     SUB     AH,BL           ; COUNT TO MOVE IN SCROLL
F35F                 N13:
F35F E8CFFF                   CALL    N10             ; MOVE ONE ROW
F362 2BF5                     SUB     SI,BP
F364 2BFD                     SUB     DI,BP
F366 FECC                     DEC     AH
F368 75F5                     JNZ     N13
F36A                 N14:
F36A 58                       POP     AX              ; RECOVER ATTRIBUTE IN AH
F36B B020                     MOV     AL,' '
F36D                 N15:
F36D E8CAFF                   CALL    N11             ; CLEAR ONE ROW
F370 2BFD                     SUB     DI,BP           ; GO TO NEXT ROW
F372 FECB                     DEC     BL
F374 75F7                     JNZ     N15
F376 E957FF                   JMP     N5                      ; SCROLL_END
F379                 N16:
F379 8ADE                     MOV     BL,DH
```

```
LOC OBJ                   SOURCE

F37B EBED                         JMP     N14
                          SCROLL_DOWN     ENDP
                          ;----------------------------------------
                          ; READ_AC_CURRENT
                          ;       THIS ROUTINE READS THE ATTRIBUTE AND CHARACTER AT THE CURRENT
                          ;       CURSOR POSITION AND RETURNS THEM TO THE CALLER
                          ;INPUT
                          ;       (AH) = CURRENT CRT MODE
                          ;       (BH) = DISPLAY PAGE ( ALPHA MODES ONLY )
                          ;       (DS) = DATA SEGMENT
                          ;       (ES) = REGEN SEGMENT
                          ;OUTPUT
                          ;       (AL) = CHAR READ
                          ;       (AH) = ATTRIBUTE READ
                          ;----------------------------------------
                                  ASSUME  CS:CODE,DS:DATA,ES:DATA
F37D                      READ_AC_CURRENT PROC    NEAR
F37D 80FC04                       CMP     AH,4                  ; IS THIS GRAPHICS
F380 7208                         JC      P1
F382 80FC07                       CMP     AH,7                  ; IS THIS BW CARD
F385 7403                         JE      P1
F387 E9A902                       JMP     GRAPHICS_READ
F38A                      P1:                                   ; READ_AC_CONTINUE
F38A E81A00                       CALL    FIND_POSITION
F38D 8BF3                         MOV     SI,BX                 ; ESTABLISH ADDRESSING IN SI

                          ;------ WAIT FOR HORIZONTAL RETRACE

F38F 8B166300         R           MOV     DX,ADDR_6845          ; GET BASE ADDRESS
F393 83C206                       ADD     DX,6                  ; POINT AT STATUS PORT
F396 06                           PUSH    ES                    ;
F397 1F                           POP     DS                    ; GET SEGMENT FOR QUICK ACCESS
F398                      P2:                                   ; WAIT FOR RETRACE LOW
F398 EC                           IN      AL,DX                 ; GET STATUS
F399 A801                         TEST    AL,1                  ; IS HORZ RETRACE LOW
F39B 75FB                         JNZ     P2                    ; WAIT UNTIL IT IS
F39D FA                           CLI                           ; NO MORE INTERRUPTS
F39E                      P3:                                   ; WAIT FOR RETRACE HIGH
F39E EC                           IN      AL,DX                 ; GET STATUS
F39F A801                         TEST    AL,1                  ; IS IT HIGH
F3A1 74FB                         JZ      P3                    ; WAIT UNTIL IT IS
F3A3 AD                           LODSW                         ; GET THE CHAR/ATTR
F3A4 E920FE                       JMP     VIDEO_RETURN
                          READ_AC_CURRENT ENDP

F3A7                      FIND_POSITION   PROC    NEAR
F3A7 8ACF                         MOV     CL,BH                 ; DISPLAY PAGE TO CX
F3A9 32ED                         XOR     CH,CH
F3AB 8BF1                         MOV     SI,CX                 ; MOVE TO SI FOR INDEX
F3AD D1E6                         SAL     SI,1                  ; * 2 FOR WORD OFFSET
F3AF 8B845000         R           MOV     AX,[SI+ OFFSET CURSOR_POSN]     ; GET ROW/COLUMN OF THAT PAGE
F3B3 33DB                         XOR     BX,BX                 ; SET START ADDRESS TO ZERO
F3B5 E306                         JCXZ    P5                    ; NO_PAGE
F3B7                      P4:                                   ; PAGE_LOOP
F3B7 031E4C00         R           ADD     BX,CRT_LEN            ; LENGTH OF BUFFER
F3BB E2FA                         LOOP    P4
F3BD                      P5:                                   ; NO_PAGE
F3BD E8CBFE                       CALL    POSITION              ; DETERMINE LOCATION IN REGEN
F3C0 03D8                         ADD     BX,AX                 ; ADD TO START OF REGEN
F3C2 C3                           RET
                          FIND_POSITION   ENDP
                          ;---------------------------------------
                          ;WRITE_AC_CURRENT
                          ;       THIS ROUTINE WRITES THE ATTRIBUTE AND CHARACTER AT
                          ;       THE CURRENT CURSOR POSITION
                          ;INPUT
                          ;       (AH) = CURRENT CRT MODE
                          ;       (BH) = DISPLAY PAGE
                          ;       (CX) = COUNT OF CHARACTERS TO WRITE
                          ;       (AL) = CHAR TO WRITE
                          ;       (BL) = ATTRIBUTE OF CHAR TO WRITE
                          ;       (DS) = DATA SEGMENT
                          ;       (ES) = REGEN SEGMENT
                          ;OUTPUT
                          ;       NONE
                          ;------------------------------------------
```

```
LOC OBJ                 SOURCE

F3C3                    WRITE_AC_CURRENT        PROC    NEAR
F3C3 80FC04                     CMP     AH,4                    ; IS THIS GRAPHICS
F3C6 7208                       JC      P6
F3C8 80FC07                     CMP     AH,7                    ; IS THIS BW CARD
F3CB 7403                       JE      P6
F3CD E9B101                     JMP     GRAPHICS_WRITE
F3D0                    P6:                                     ; WRITE_AC_CONTINUE
F3D0 8AE3                       MOV     AH,BL                   ; GET ATTRIBUTE TO AH
F3D2 50                         PUSH    AX                      ; SAVE ON STACK
F3D3 51                         PUSH    CX                      ; SAVE WRITE COUNT
F3D4 E8D0FF                     CALL    FIND_POSITION
F3D7 8BFB                       MOV     DI,BX                   ; ADDRESS TO DI REGISTER
F3D9 59                         POP     CX                      ; WRITE COUNT
F3DA 5B                         POP     BX                      ; CHARACTER IN BX REG
F3DB                    P7:                                     ; WRITE_LOOP

                        ;------ WAIT FOR HORIZONTAL RETRACE

F3DB 8B166300      R            MOV     DX,ADDR_6845            ; GET BASE ADDRESS
F3DF 83C206                     ADD     DX,6                    ; POINT AT STATUS PORT
F3E2                    P8:
F3E2 EC                         IN      AL,DX                   ; GET STATUS
F3E3 A801                       TEST    AL,1                    ; IS IT LOW
F3E5 75FB                       JNZ     P8                      ; WAIT UNTIL IT IS
F3E7 FA                         CLI                             ; NO MORE INTERRUPTS
F3E8                    P9:
F3E8 EC                         IN      AL,DX                   ; GET STATUS
F3E9 A801                       TEST    AL,1                    ; IS IT HIGH
F3EB 74FB                       JZ      P9                      ; WAIT UNTIL IT IS
F3ED 8BC3                       MOV     AX,BX                   ; RECOVER THE CHAR/ATTR
F3EF AB                         STOSW                           ; PUT THE CHAR/ATTR
F3F0 FB                         STI                             ; INTERRUPTS BACK ON
F3F1 E2E8                       LOOP    P7                      ;  AS MANY TIMES AS REQUESTED
F3F3 E9D1FD                     JMP     VIDEO_RETURN
                        WRITE_AC_CURRENT        ENDP
                        ;----------------------------------------
                        ;WRITE_C_CURRENT
                        ;       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
                        ;       (AL) = CHAR TO WRITE
                        ;       (DS) = DATA SEGMENT
                        ;       (ES) = REGEN SEGMENT
                        ;OUTPUT
                        ;       NONE
                        ;--------------------------------------------
F3F6                    WRITE_C_CURRENT PROC    NEAR
F3F6 80FC04                     CMP     AH,4                    ; IS THIS GRAPHICS
F3F9 7208                       JC      P10
F3FB 80FC07                     CMP     AH,7                    ; IS THIS BW CARD
F3FE 7403                       JE      P10
F400 E97E01                     JMP     GRAPHICS_WRITE
F403                    P10:
F403 50                         PUSH    AX                      ; SAVE ON STACK
F404 51                         PUSH    CX                      ; SAVE WRITE COUNT
F405 E89FFF                     CALL    FIND_POSITION
F408 8BFB                       MOV     DI,BX                   ; ADDRESS TO DI
F40A 59                         POP     CX                      ; WRITE COUNT
F40B 5B                         POP     BX                      ; BL HAS CHAR TO WRITE
F40C                    P11:                                    ; WRITE_LOOP

                        ;------ WAIT FOR HORIZONTAL RETRACE

F40C 8B166300      R            MOV     DX,ADDR_6845            ; GET BASE ADDRESS
F410 83C206                     ADD     DX,6                    ; POINT AT STATUS PORT
F413                    P12:
F413 EC                         IN      AL,DX                   ; GET STATUS
F414 A801                       TEST    AL,1                    ; IS IT LOW
F416 75FB                       JNZ     P12                     ; WAIT UNTIL IT IS
F418 FA                         CLI                             ; NO MORE INTERRUPTS
F419                    P13:
F419 EC                         IN      AL,DX                   ; GET STATUS
F41A A801                       TEST    AL,1                    ; IS IT HIGH
```

```
LOC OBJ                SOURCE

F41C 74FB                      JZ      P13                  ; WAIT UNTIL IT IS
F41E 8AC3                      MOV     AL,BL                ; RECOVER CHAR
F420 AA                        STOSB                        ; PUT THE CHAR/ATTR
F421 47                        INC     DI                   ; BUMP POINTER PAST ATTRIBUTE
F422 E2E8                      LOOP    P11                  ;  AS MANY TIMES AS REQUESTED
F424 E9A0FD                    JMP     VIDEO_RETURN
                       WRITE_C_CURRENT ENDP
                       ;------------------------------------------
                       ; READ DOT  -- WRITE DOT
                       ; THESE ROUTINES WILL WRITE A DOT, OR READ THE
                       ;  DOT AT THE INDICATED LOCATION
                       ; ENTRY --
                       ;   DX = 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
                       ;------------------------------------------
                               ASSUME  CS:CODE,DS:DATA,ES:DATA
F427                   READ_DOT        PROC    NEAR
F427 E83100                    CALL    R3              ; DETERMINE BYTE POSITION OF DOT
F42A 268A04                    MOV     AL,ES:[SI]              ; GET THE BYTE
F42D 22C4                      AND     AL,AH           ; MASK OFF THE OTHER BITS IN THE BYTE
F42F D2E0                      SHL     AL,CL           ; LEFT JUSTIFY THE VALUE
F431 8ACE                      MOV     CL,DH           ; GET NUMBER OF BITS IN RESULT
F433 D2C0                      ROL     AL,CL           ; RIGHT JUSTIFY THE RESULT
F435 E98FFD                    JMP     VIDEO_RETURN    ; RETURN FROM VIDEO IO
                       READ_DOT        ENDP

F438                   WRITE_DOT       PROC    NEAR
F438 50                        PUSH    AX              ; SAVE DOT VALUE
F439 50                        PUSH    AX              ;  TWICE
F43A E81E00                    CALL    R3              ; DETERMINE BYTE POSITION OF THE DOT
F43D D2E8                      SHR     AL,CL           ; SHIFT TO SET UP THE BITS FOR OUTPUT
F43F 22C4                      AND     AL,AH           ; STRIP OFF THE OTHER BITS
F441 268A0C                    MOV     CL,ES:[SI]      ; GET THE CURRENT BYTE
F444 5B                        POP     BX              ; RECOVER XOR FLAG
F445 F6C380                    TEST    BL,80H          ; IS IT ON
F448 750D                      JNZ     R2              ; YES, XOR THE DOT
F44A F6D4                      NOT     AH              ; SET THE MASK TO REMOVE THE INDICATED BITS
F44C 22CC                      AND     CL,AH
F44E 0AC1                      OR      AL,CL           ; OR IN THE NEW VALUE OF THOSE BITS
F450                   R1:                             ; FINISH_DOT
F450 268804                    MOV     ES:[SI],AL      ; RESTORE THE BYTE IN MEMORY
F453 58                        POP     AX
F454 E970FD                    JMP     VIDEO_RETURN    ; RETURN FROM VIDEO IO
F457                   R2:                             ; XOR_DOT
F457 32C1                      XOR     AL,CL           ; EXCLUSIVE OR THE DOTS
F459 EBF5                      JMP     R1              ; FINISH UP THE WRITING
                       WRITE_DOT       ENDP
                       ;------------------------------------------
                       ; THIS SUBROUTINE DETERMINES 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                   R3      PROC    NEAR
F45B 53                        PUSH    BX              ; SAVE BX DURING OPERATION
F45C 50                        PUSH    AX              ; WILL SAVE AL DURING OPERATION

                       ;------ DETERMINE 1ST BYTE IN IDICATED ROW BY MULTIPLYING ROW VALUE BY 40
                       ;------ ( LOW BIT OF ROW DETERMINES EVEN/ODD, 80 BYTES/ROW

F45D B028                      MOV     AL,40
F45F 52                        PUSH    DX              ; SAVE ROW VALUE
```

```
LOC  OBJ              LINE  SOURCE

F460 80E2FE                        AND   DL,0FEH          ; STRIP OFF ODD/EVEN BIT
F463 F6E2                          MUL   DL               ; AX HAS ADDRESS OF 1ST BYTE OF INDICATED ROW
F465 5A                            POP   DX               ; RECOVER IT
F466 F6C201                        TEST  DL,1             ; TEST FOR EVEN/ODD
F469 7403                          JZ    R4               ; JUMP IF EVEN ROW
F46B 050020                        ADD   AX,2000H         ; OFFSET TO LOCATION OF ODD ROWS
F46E                         R4:                          ; EVEN_ROW
F46E 8BF0                          MOV   SI,AX            ; MOVE POINTER TO SI
F470 58                            POP   AX               ; RECOVER AL VALUE
F471 8BD1                          MOV   DX,CX            ; COLUMN VALUE TO DX
                             ;------ DETERMINE GRAPHICS MODE CURRENTLY IN EFFECT
                             ; SET UP THE REGISTERS ACCORDING TO THE MODE
                             ;  CH = MASK FOR LOW OF COLUMN ADDRESS ( 7/3 FOR HIGH/MED RES)
                             ;  CL = # OF ADDRESS BITS IN COLUMN VALUE ( 3/2 FOR H/M)
                             ;  BL = MASK TO SELECT BITS FROM POINTED BYTE (80H/C0H FOR H/M)
                             ;  BH = NUMBER OF VALID BITS IN POINTED BYTE ( 1/2 FOR H/M)
F473 BBC002                        MOV   BX,2C0H
F476 B90203                        MOV   CX,302H          ; SET PARMS FOR MED RES
F479 803E490006      R             CMP   CRT_MODE,6
F47E 7206                          JC    R5               ; HANDLE IF MED ARES
F480 BB8001                        MOV   BX,180H
F483 B90307                        MOV   CX,703H          ; SET PARMS FOR HIGH RES
                             ;------ DETERMINE BIT OFFSET IN BYTE FROM COLUMN MASK
F486                         R5:
F486 22EA                          AND   CH,DL            ; ADDRESS OF PEL WITHIN BYTE TO CH
                             ;------ DETERMINE BYTE OFFSET FOR THIS LOCATION IN COLUMN
F488 D3EA                          SHR   DX,CL            ; SHIFT BY CORRECT AMOUNT
F48A 03F2                          ADD   SI,DX            ; INCREMENT THE POINTER
F48C 8AF7                          MOV   DH,BH            ; GET THE # OF BITS IN RESULT TO DH
                             ;------ MULTIPLY BH (VALID BITS IN BYTE) BY CH (BIT OFFSET)
F48E 2AC9                          SUB   CL,CL            ; ZERO INTO STORAGE LOCATION
F490                         R6:
F490 D0C8                          ROR   AL,1             ; LEFT JUSTIFY THE VALUE IN AL (FOR WRITE)
F492 02CD                          ADD   CL,CH            ; ADD IN THE BIT OFFSET VALUE
F494 FECF                          DEC   BH               ; LOOP CONTROL
F496 75F8                          JNZ   R6               ; ON EXIT, CL HAS SHIFT COUNT TO RESTORE BITS
F498 8AE3                          MOV   AH,BL            ; GET MASK TO AH
F49A D2EC                          SHR   AH,CL            ; MOVE THE MASK TO CORRECT LOCATION
F49C 5B                            POP   BX               ; RECOVER REG
F49D C3                            RET                    ; RETURN WITH EVERYTHING SET UP
                             R3    ENDP
                             ;------------------------------------------
                             ; SCROLL UP
                             ;  THIS ROUTINE SCROLLS UP THE INFORMATION ON THE CRT
                             ; ENTRY --
                             ;  CH,CL = UPPER LEFT CORNER OF REGION TO SCROLL
                             ;  DH,DL = LOWER RIGHT CORNER OF REGION TO SCROLL
                             ;   BOTH OF THE ABOVE ARE IN CHARACTER POSITIONS
                             ;  BH = FILL VALUE FOR BLANKED LINES
                             ;  AL = # LINES TO SCROLL (AL=0 MEANS BLANK THE ENTIRE FIELD)
                             ;  DS = DATA SEGMENT
                             ;  ES = REGEN SEGMENT
                             ; EXIT --
                             ;  NOTHING, THE SCREEN IS SCROLLED
                             ;------------------------------------------
F49E                         GRAPHICS_UP     PROC    NEAR
F49E 8AD8                          MOV   BL,AL  ; SAVE LINE COUNT IN BL
F4A0 8BC1                          MOV   AX,CX  ; GET UPPER LEFT POSITION INTO AX REG
                             ;------ USE CHARACTER SUBROUTINE FOR POSITIONING
                             ;------ ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE
F4A2 E86A02                        CALL  GRAPH_POSN
F4A5 8BF8                          MOV   DI,AX            ; SAVE RESULT AS DESTINATION ADDRESS
                             ;------ DETERMINE SIZE OF WINDOW
```

```
LOC OBJ                LINE    SOURCE

F4A7 2BD1                              SUB     DX,CX
F4A9 81C20101                          ADD     DX,101H          ; ADJUST VALUES
F4AD D0E6                              SAL     DH,1             ; MULTIPLY # ROWS BY 4 SINCE 8 VERT DOTS/CHAR
F4AF D0E6                              SAL     DH,1             ;  AND EVEN/ODD ROWS

                               ;------ DETERMINE CRT MODE

F4B1 803E490006      R                 CMP     CRT_MODE,6       ; TEST FOR MEDIUM RES
F4B6 7304                              JNC     R7               ; FIND_SOURCE

                               ;------ MEDIUM RES UP
F4B8 D0E2                              SAL     DL,1             ; # COLUMNS * 2, SINCE 2 BYTES/CHAR
F4BA D1E7                              SAL     DI,1             ; OFFSET *2 SINCE 2 BYTES/CHAR

                               ;------ DETERMINE THE SOURCE ADDRESS IN THE BUFFER
F4BC                           R7:                              ; FIND_SOURCE
F4BC 06                                PUSH    ES               ; GET SEGMENTS BOTH POINTING TO REGEN
F4BD 1F                                POP     DS
F4BE 2AED                              SUB     CH,CH            ; ZERO TO HIGH OF COUNT REG
F4C0 D0E3                              SAL     BL,1             ; MULTIPLY NUMBER OF LINES BY 4
F4C2 D0E3                              SAL     BL,1
F4C4 742D                              JZ      R11              ; IF ZERO, THEN BLANK ENTIRE FIELD
F4C6 8AC3                              MOV     AL,BL            ; GET NUMBER OF LINES IN AL
F4C8 B450                              MOV     AH,80            ; 80 BYTES/ROW
F4CA F6E4                              MUL     AH               ; DETERMINE OFFSET TO SOURCE
F4CC 8BF7                              MOV     SI,DI            ; SET UP SOURCE
F4CE 03F0                              ADD     SI,AX            ;  ADD IN OFFSET TO IT
F4D0 8AE6                              MOV     AH,DH            ; NUMBER OF ROWS IN FIELD
F4D2 2AE3                              SUB     AH,BL            ; DETERMINE NUMBER TO MOVE

                               ;------ LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD FIELDS
F4D4                           R8:                              ; ROW_LOOP
F4D4 E88000                            CALL    R17              ; MOVE ONE ROW
F4D7 81EEB01F                          SUB     SI,2000H-80      ; MOVE TO NEXT ROW
F4DB 81EFB01F                          SUB     DI,2000H-80
F4DF FECC                              DEC     AH               ; NUMBER OF ROWS TO MOVE
F4E1 75F1                              JNZ     R8               ; CONTINUE TILL ALL MOVED

                               ;------ FILL IN THE VACATED LINE(S)
F4E3                           R9:                              ; CLEAR_ENTRY
F4E3 8AC7                              MOV     AL,BH            ; ATTRIBUTE TO FILL WITH
F4E5                           R10:
F4E5 E88800                            CALL    R18              ; CLEAR THAT ROW
F4E8 81EFB01F                          SUB     DI,2000H-80      ; POINT TO NEXT LINE
F4EC FECB                              DEC     BL               ; NUMBER OF LINES TO FILL
F4EE 75F5                              JNZ     R10              ; CLEAR_LOOP
F4F0 E9D4FC                            JMP     VIDEO_RETURN     ; EVERYTHING DONE

F4F3                           R11:                             ; BLANK_FIELD
F4F3 8ADE                              MOV     BL,DH            ; SET BLANK COUNT TO EVERYTHING IN FIELD
F4F5 EBEC                              JMP     R9               ; CLEAR THE FIELD
                               GRAPHICS_UP     ENDP
                               ;--------------------------------------------
                               ; SCROLL DOWN
                               ;  THIS ROUTINE SCROLLS DOWN THE INFORMATION ON THE CRT
                               ; ENTRY --
                               ;  CH,CL = UPPER LEFT CORNER OF REGION TO SCROLL
                               ;  DH,DL = LOWER RIGHT CORNER OF REGION TO SCROLL
                               ;   BOTH OF THE ABOVE ARE IN CHARACTER POSITIONS
                               ;  BH = FILL VALUE FOR BLANKED LINES
                               ;  AL = # LINES TO SCROLL (AL=0 MEANS BLANK THE ENTIRE FIELD)
                               ;  DS = DATA SEGMENT
                               ;  ES = REGEN SEGMENT
                               ; EXIT --
                               ;  NOTHING, THE SCREEN IS SCROLLED
                               ;--------------------------------------------

F4F7                           GRAPHICS_DOWN   PROC    NEAR
F4F7 FD                                STD             ; SET DIRECTION
F4F8 8AD8                              MOV     BL,AL   ; SAVE LINE COUNT IN BL
F4FA 8BC2                              MOV     AX,DX   ; GET LOWER RIGHT POSITION INTO AX REG

                               ;------ USE CHARACTER SUBROUTINE FOR POSITIONING
                               ;------ ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE

F4FC E81002                            CALL    GRAPH_POSN
```

```
LOC OBJ              LINE    SOURCE

F4FF 8BF8                            MOV     DI,AX           ; SAVE RESULT AS DESTINATION ADDRESS

                             ;------ DETERMINE SIZE OF WINDOW

F501 2BD1                            SUB     DX,CX
F503 81C20101                        ADD     DX,101H         ; ADJUST VALUES
F507 D0E6                            SAL     DH,1            ; MULTIPLY # ROWS BY 4 SINCE 8 VERT DOTS/CHAR
F509 D0E6                            SAL     DH,1            ;  AND EVEN/ODD ROWS

                             ;------ DETERMINE CRT MODE

F50B 803E490006      R               CMP     CRT_MODE,6      ; TEST FOR MEDIUM RES
F510 7305                            JNC     R12             ; FIND_SOURCE_DOWN

                             ;------ MEDIUM RES DOWN
F512 D0E2                            SAL     DL,1            ; # COLUMNS * 2, SINCE 2 BYTES/CHAR (OFFSET OK)
F514 D1E7                            SAL     DI,1            ; OFFSET *2 SINCE 2 BYTES/CHAR
F516 47                              INC     DI              ; POINT TO LAST BYTE

                             ;------ DETERMINE THE SOURCE ADDRESS IN THE BUFFER
F517                         R12:                            ; FIND_SOURCE_DOWN
F517 06                              PUSH    ES              ; BOTH SEGMENTS TO REGEN
F518 1F                              POP     DS
F519 2AED                            SUB     CH,CH           ; ZERO TO HIGH OF COUNT REG
F51B 81C7F000                        ADD     DI,240          ; POINT TO LAST ROW OF PIXELS
F51F D0E3                            SAL     BL,1            ; MULTIPLY NUMBER OF LINES BY 4
F521 D0E3                            SAL     BL,1
F523 742E                            JZ      R16             ; IF ZERO, THEN BLANK ENTIRE FIELD
F525 8AC3                            MOV     AL,BL           ; GET NUMBER OF LINES IN AL
F527 B450                            MOV     AH,80           ; 80 BYTES/ROW
F529 F6E4                            MUL     AH              ; DETERMINE OFFSET TO SOURCE
F52B 8BF7                            MOV     SI,DI           ; SET UP SOURCE
F52D 2BF0                            SUB     SI,AX           ;  SUBTRACT THE OFFSET
F52F 8AE6                            MOV     AH,DH           ; NUMBER OF ROWS IN FIELD
F531 2AE3                            SUB     AH,BL           ; DETERMINE NUMBER TO MOVE

                             ;------ LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD FIELDS
F533                         R13:                            ; ROW_LOOP_DOWN
F533 E82100                          CALL    R17             ; MOVE ONE ROW
F536 81EE5020                        SUB     SI,2000H+80     ; MOVE TO NEXT ROW
F53A 81EF5020                        SUB     DI,2000H+80
F53E FECC                            DEC     AH              ; NUMBER OF ROWS TO MOVE
F540 75F1                            JNZ     R13             ; CONTINUE TILL ALL MOVED

                             ;------ FILL IN THE VACATED LINE(S)
F542                         R14:                            ; CLEAR_ENTRY_DOWN
F542 8AC7                            MOV     AL,BH           ; ATTRIBUTE TO FILL WITH
F544                         R15:                            ; CLEAR_LOOP_DOWN
F544 E82900                          CALL    R18             ; CLEAR A ROW
F547 81EF5020                        SUB     DI,2000H+80     ; POINT TO NEXT LINE
F54B FECB                            DEC     BL              ; NUMBER OF LINES TO FILL
F54D 75F5                            JNZ     R15             ; CLEAR_LOOP_DOWN
F54F FC                              CLD                     ; RESET THE DIRECTION FLAG
F550 E974FC                          JMP     VIDEO_RETURN    ; EVERYTHING DONE

F553                         R16:                            ; BLANK_FIELD_DOWN
F553 8ADE                            MOV     BL,DH           ; SET BLANK COUNT TO EVERYTHING IN FIELD
F555 EBEB                            JMP     R14             ; CLEAR THE FIELD
                             GRAPHICS_DOWN   ENDP

                             ;------ ROUTINE TO MOVE ONE ROW OF INFORMATION

F557                         R17     PROC    NEAR
F557 8ACA                            MOV     CL,DL           ; NUMBER OF BYTES IN THE ROW
F559 56                              PUSH    SI
F55A 57                              PUSH    DI              ; SAVE POINTERS
F55B F3                              REP     MOVSB           ; MOVE THE EVEN FIELD
F55C A4
F55D 5F                              POP     DI
F55E 5E                              POP     SI
F55F 81C60020                        ADD     SI,2000H
F563 81C70020                        ADD     DI,2000H                ; POINT TO THE ODD FIELD
F567 56                              PUSH    SI
F568 57                              PUSH    DI              ; SAVE THE POINTERS
F569 8ACA                            MOV     CL,DL           ; COUNT BACK
```

```
LOC OBJ          LINE    SOURCE

F56B F3                          REP    MOVSB           ; MOVE THE ODD FIELD
F56C A4
F56D 5F                          POP    DI
F56E 5E                          POP    SI              ; POINTERS BACK
F56F C3                          RET                    ; RETURN TO CALLER
                         R17     ENDP

                         ;------ CLEAR A SINGLE ROW

F570                     R18     PROC   NEAR
F570 8ACA                        MOV    CL,DL           ; NUMBER OF BYTES IN FIELD
F572 57                          PUSH   DI              ; SAVE POINTER
F573 F3                          REP    STOSB           ; STORE THE NEW VALUE
F574 AA
F575 5F                          POP    DI              ; POINTER BACK
F576 81C70020                    ADD    DI,2000H                ; POINT TO ODD FIELD
F57A 57                          PUSH   DI
F57B 8ACA                        MOV    CL,DL
F57D F3                          REP    STOSB           ; FILL THE ODD FILELD
F57E AA
F57F 5F                          POP    DI
F580 C3                          RET                    ; RETURN TO CALLER
                         R18     ENDP
                         ;--------------------------------------------
                         ; GRAPHICS WRITE
                         ;  THIS ROUTINE WRITES THE ASCII CHARACTER TO THE CURRENT
                         ;  POSITION ON THE SCREEN.
                         ; ENTRY --
                         ;  AL = CHARACTER TO WRITE
                         ;  BL = COLOR ATTRIBUTE TO BE USED FOR FOREGROUND COLOR
                         ;       IF BIT 7 IS SET, THE CHAR IS XOR'D INTO THE REGEN BUFFER
                         ;       (0 IS USED FOR THE BACKGROUND COLOR)
                         ;  CX = NUMBER OF CHARS TO WRITE
                         ;  DS = DATA SEGMENT
                         ;  ES = REGEN SEGMENT
                         ; EXIT --
                         ;  NOTHING IS RETURNED
                         ;
                         ; GRAPHICS READ
                         ;   THIS ROUTINE READS THE ASCII CHARACTER AT THE CURRENT CURSOR
                         ;   POSITION ON THE SCREEN BY MATCHING THE DOTS ON THE SCREEN TO THE
                         ;   CHARACTER GENERATOR CODE POINTS
                         ; ENTRY --
                         ;  NONE  ( 0 IS ASSUMED AS THE BACKGROUND COLOR
                         ; EXIT --
                         ;  AL = CHARACTER READ AT THAT POSITION (0 RETURNED IF NONE FOUND)
                         ;
                         ; FOR BOTH ROUTINES, THE IMAGES USED TO FORM CHARS ARE CONTAINED IN ROM
                         ;  FOR THE 1ST 128 CHARS.  TO ACCESS CHARS IN THE SECOND HALF, THE USER
                         ;  MUST INITIALIZE THE VECTOR AT INTERRUPT 1FH (LOCATION 0007CH) TO
                         ;  POINT TO THE USER SUPPLIED TABLE OF GRAPHIC IMAGES (8X8 BOXES).
                         ;  FAILURE TO DO SO WILL CAUSE IN STRANGE RESULTS
                         ;--------------------------------------------
                                 ASSUME CS:CODE,DS:DATA,ES:DATA
F581                     GRAPHICS_WRITE  PROC    NEAR
F581 B400                        MOV    AH,0            ; ZERO TO HIGH OF CODE POINT
F583 50                          PUSH   AX              ; SAVE CODE POINT VALUE

                         ;------ DETERMINE POSITION IN REGEN BUFFER TO PUT CODE POINTS

F584 E88501                      CALL   S26             ; FIND LOCATION IN REGEN BUFFER
F587 8BF8                        MOV    DI,AX           ; REGEN POINTER IN DI

                         ;------ DETERMINE REGION TO GET CODE POINTS FROM

F589 58                          POP    AX              ; RECOVER CODE POINT
F58A 3C80                        CMP    AL,80H          ; IS IT IN SECOND HALF
F58C 7306                        JAE    S1              ; YES

                         ;------ IMAGE IS IN FIRST HALF, CONTAINED IN ROM

F58E BE6EFA                      MOV    SI,0FA6EH       ; OFFSET CRT_CHAR_GEN-OFFSET OF IMAGES
F591 0E                          PUSH   CS                      ; SAVE SEGMENT ON STACK
F592 EB0F                        JMP    SHORT S2                ; DETERMINE_MODE

```

```
LOC OBJ                 LINE    SOURCE

                                ;------ IMAGE IS IN SECOND HALF, IN USER RAM

F594                            S1:                                     ; EXTEND_CHAR
F594 2C80                               SUB     AL,80H                  ; ZERO ORIGIN FOR SECOND HALF
F596 1E                                 PUSH    DS                      ; SAVE DATA POINTER
F597 2BF6                               SUB     SI,SI
F599 8EDE                               MOV     DS,SI                   ; ESTABLISH VECTOR ADDRESSING
                                        ASSUME  DS:ABS0
F59B C5367C00                           LDS     SI,EXT_PTR              ; GET THE OFFSET OF THE TABLE
F59F 8CDA                               MOV     DX,DS                   ; GET THE SEGMENT OF THE TABLE
                                        ASSUME  DS:DATA
F5A1 1F                                 POP     DS                      ; RECOVER DATA SEGMENT
F5A2 52                                 PUSH    DX                      ; SAVE TABLE SEGMENT ON STACK

                                ;------ DETERMINE GRAPHICS MODE IN OPERATION

F5A3                            S2:                                     ; DETERMINE_MODE
F5A3 D1E0                               SAL     AX,1                    ; MULTIPLY CODE POINT
F5A5 D1E0                               SAL     AX,1                    ;  VALUE BY 8
F5A7 D1E0                               SAL     AX,1
F5A9 03F0                               ADD     SI,AX                   ; SI HAS OFFSET OF DESIRED CODES
F5AB 803E490006      R                  CMP     CRT_MODE,6
F5B0 1F                                 POP     DS              ; RECOVER TABLE POINTER SEGMENT
F5B1 722C                               JC      S7              ; TEST FOR MEDIUM RESOLUTION MODE

                                ;------ HIGH RESOLUTION MODE
F5B3                            S3:                             ; HIGH_CHAR
F5B3 57                                 PUSH    DI              ; SAVE REGEN POINTER
F5B4 56                                 PUSH    SI              ; SAVE CODE POINTER
F5B5 B604                               MOV     DH,4            ; NUMBER OF TIMES THROUGH LOOP
F5B7                            S4:
F5B7 AC                                 LODSB                   ; GET BYTE FROM CODE POINTS
F5B8 F6C380                             TEST    BL,80H          ; SHOULD WE USE THE FUNCTION
F5BB 7516                               JNZ     S6              ;  TO PUT CHAR IN
F5BD AA                                 STOSB                   ; STORE IN REGEN BUFFER
F5BE AC                                 LODSB
F5BF                            S5:                             ;
F5BF 268885FF1F                         MOV     ES:[DI+2000H-1],AL      ; STORE IN SECOND HALF
F5C4 83C74F                             ADD     DI,79           ; MOVE TO NEXT ROW IN REGEN
F5C7 FECE                               DEC     DH              ; DONE WITH LOOP
F5C9 75EC                               JNZ     S4
F5CB 5E                                 POP     SI
F5CC 5F                                 POP     DI              ; RECOVER REGEN POINTER
F5CD 47                                 INC     DI              ; POINT TO NEXT CHAR POSITION
F5CE E2E3                               LOOP    S3              ; MORE CHARS TO WRITE
F5D0 E9F4FB                             JMP     VIDEO_RETURN

F5D3                            S6:
F5D3 263205                             XOR     AL,ES:[DI]      ; EXCLUSIVE OR WITH CURRENT
F5D6 AA                                 STOSB                   ; STORE THE CODE POINT
F5D7 AC                                 LODSB                   ; AGAIN FOR ODD FIELD
F5D8 263285FF1F                         XOR     AL,ES:[DI+2000H-1]      ;
F5DD EBE0                               JMP     S5              ; BACK TO MAINSTREAM

                                ;------ MEDIUM RESOLUTION WRITE
F5DF                            S7:                             ; MED_RES_WRITE
F5DF 8AD3                               MOV     DL,BL           ; SAVE HIGH COLOR BIT
F5E1 D1E7                               SAL     DI,1            ; OFFSET*2 SINCE 2 BYTES/CHAR
F5E3 E8D100                             CALL    S19             ; EXPAND BL TO FULL WORD OF COLOR
F5E6                            S8:                             ; MED_CHAR
F5E6 57                                 PUSH    DI              ; SAVE REGEN POINTER
F5E7 56                                 PUSH    SI                      ; SAVE THE CODE POINTER
F5E8 B604                               MOV     DH,4            ; NUMBER OF LOOPS
F5EA                            S9:
F5EA AC                                 LODSB                   ; GET CODE POINT
F5EB E8DE00                             CALL    S21             ; DOUBLE UP ALL THE BITS
F5EE 23C3                               AND     AX,BX           ; CONVERT THEM TO FOREGROUND COLOR ( 0 BACK )
F5F0 F6C280                             TEST    DL,80H          ; IS THIS XOR FUNCTION
F5F3 7407                               JZ      S10             ; NO, STORE IT IN AS IT IS
F5F5 263225                             XOR     AH,ES:[DI]      ; DO FUNCTION WITH HALF
F5F8 26324501                           XOR     AL,ES:[DI+1]    ;  AND WITH OTHER HALF
F5FC                            S10:                            ;
F5FC 268825                             MOV     ES:[DI],AH      ; STORE FIRST BYTE
F5FF 26884501                           MOV     ES:[DI+1],AL    ; STORE SECOND BYTE
F603 AC                                 LODSB                   ; GET CODE POINT
F604 E8C500                             CALL    S21
```

```
LOC OBJ                 SOURCE

F607 23C3                       AND     AX,BX           ; CONVERT TO COLOR
F609 F6C280                     TEST    DL,80H          ; AGAIN, IS THIS XOR FUNCTION
F60C 740A                       JZ      S11             ; NO, JUST STORE THE VALUES
F60E 2632A50020                 XOR     AH,ES:[DI+2000H]        ; FUNCTION WITH FIRST HALF
F613 2632850120                 XOR     AL,ES:[DI+2001H]        ; AND WITH SECOND HALF
F618                    S11:                            ;
F618 2688A50020                 MOV     ES:[DI+2000H],AH
F61D 2688850120                 MOV     ES:[DI+2000H+1],AL      ; STORE IN SECOND PORTION OF BUFFER
F622 83C750                     ADD     DI,80           ; POINT TO NEXT LOCATION
F625 FECE                       DEC     DH
F627 75C1                       JNZ     S9              ; KEEP GOING
F629 5E                         POP     SI              ; RECOVER CODE PONTER
F62A 5F                         POP     DI              ; RECOVER REGEN POINTER
F62B 83C702                     ADD     DI,2            ; POINT TO NEXT CHAR POSITION
F62E E2B6                       LOOP    S8              ; MORE TO WRITE
F630 E994FB                     JMP     VIDEO_RETURN
                        GRAPHICS_WRITE  ENDP
                        ;---------------------------------------
                        ; GRAPHICS READ
                        ;---------------------------------------
F633                    GRAPHICS_READ   PROC    NEAR
F633 E8D600                     CALL    S26             ; CONVERTED TO OFFSET IN REGEN
F636 8BF0                       MOV     SI,AX           ; SAVE IN SI
F638 83EC08                     SUB     SP,8            ; ALLOCATE SPACE TO SAVE THE READ CODE POINT
F63B 8BEC                       MOV     BP,SP           ; POINTER TO SAVE AREA

                        ;------ DETERMINE GRAPHICS MODES

F63D 803E490006      R          CMP     CRT_MODE,6
F642 06                         PUSH    ES
F643 1F                         POP     DS              ; POINT TO REGEN SEGMENT
F644 721A                       JC      S13             ; MEDIUM RESOLUTION

                        ;------ HIGH RESOLUTION READ

                        ;------ GET VALUES FROM REGEN BUFFER AND CONVERT TO CODE POINT
F646 B604                       MOV     DH,4            ; NUMBER OF PASSES
F648                    S12:
F648 8A04                       MOV     AL,[SI]         ; GET FIRST BYTE
F64A 884600                     MOV     [BP],AL         ; SAVE IN STORAGE AREA
F64D 45                         INC     BP              ; NEXT LOCATION
F64E 8A840020                   MOV     AL,[SI+2000H]   ; GET LOWER REGION BYTE
F652 884600                     MOV     [BP],AL         ; ADJUST AND STORE
F655 45                         INC     BP
F656 83C650                     ADD     SI,80           ; POINTER INTO REGEN
F659 FECE                       DEC     DH              ; LOOP CONTROL
F65B 75EB                       JNZ     S12             ; DO IT SOME MORE
F65D EB1790                     JMP     S15             ; GO MATCH THE SAVED CODE POINTS

                        ;------ MEDIUM RESOLUTION READ
F660                    S13:                            ; MED_RES_READ
F660 D1E6                       SAL     SI,1            ; OFFSET*2 SINCE 2 BYTES/CHAR
F662 B604                       MOV     DH,4            ; NUMBER OF PASSES
F664                    S14:
F664 E88800                     CALL    S23             ; GET PAIR BYTES FROM REGEN INTO SINGLE SAVE
F667 81C60020                   ADD     SI,2000H        ; GO TO LOWER REGION
F66B E88100                     CALL    S23             ; GET THIS PAIR INTO SAVE
F66E 81EEB01F                   SUB     SI,2000H-80     ; ADJUST POINTER BACK INTO UPPER
F672 FECE                       DEC     DH
F674 75EE                       JNZ     S14             ; KEEP GOING UNTIL ALL 8 DONE

                        ;-------- SAVE AREA HAS CHARACTER IN IT, MATCH IT
F676                    S15:                            ; FIND_CHAR
F676 BF6EFA                     MOV     DI,0FA6EH       ; OFFSET CRT_CHAR_GEN-ESTABLISH ADDRESSING
F679 0E                         PUSH    CS
F67A 07                         POP     ES                      ; CODE POINTS IN CS
F67B 83ED08                     SUB     BP,8            ; ADJUST POINTER TO BEGINNING OF SAVE AREA
F67E 8BF5                       MOV     SI,BP
F680 FC                         CLD                     ; ENSURE DIRECTION
F681 B000                       MOV     AL,0            ; CURRENT CODE POINT BEING MATCHED
F683                    S16:
F683 16                         PUSH    SS              ; ESTABLISH ADDRESSING TO STACK
F684 1F                         POP     DS              ; FOR THE STRING COMPARE
F685 BA8000                     MOV     DX,128          ; NUMBER TO TEST AGAINST
F688                    S17:
F688 56                         PUSH    SI              ; SAVE SAVE AREA POINTER
```

```
LOC OBJ             LINE     SOURCE

F689 57             4511              PUSH    DI              ; SAVE CODE POINTER
F68A B90800         4512              MOV     CX,8            ; NUMBER OF BYTES TO MATCH
F68D F3             4513              REPE    CMPSB           ; COMPARE THE 8 BYTES
F68E A6
F68F 5F             4514              POP     DI              ; RECOVER THE POINTERS
F690 5E             4515              POP     SI
F691 741E           4516              JZ      S18             ; IF ZERO FLAG SET, THEN MATCH OCCURRED
F693 FEC0           4517              INC     AL              ; NO MATCH, MOVE ON TO NEXT
F695 83C708         4518              ADD     DI,8            ; NEXT CODE POINT
F698 4A             4519              DEC     DX              ; LOOP CONTROL
F699 75ED           4520              JNZ     S17             ; DO ALL OF THEM
                    4521
                    4522     ;------ CHAR NOT MATCHED, MIGHT BE IN USER SUPPLIED SECOND HALF
                    4523
F69B 3C00           4524              CMP     AL,0            ; AL<> 0 IF ONLY 1ST HALF SCANNED
F69D 7412           4525              JE      S18             ; IF = 0, THEN ALL HAS BEEN SCANNED
F69F 2BC0           4526              SUB     AX,AX
F6A1 8ED8           4527              MOV     DS,AX           ; ESTABLISH ADDRESSING TO VECTOR
                    4528              ASSUME  DS:ABS0
F6A3 C43E7C00       4529              LES     DI,EXT_PTR      ; GET POINTER
F6A7 8CC0           4530              MOV     AX,ES           ; SEE IF THE POINTER REALLY EXISTS
F6A9 0BC7           4531              OR      AX,DI           ; IF ALL 0, THEN DOESN'T EXIST
F6AB 7404           4532              JZ      S18             ; NO SENSE LOOKING
F6AD B080           4533              MOV     AL,128         ; ORIGIN FOR SECOND HALF
F6AF EBD2           4534              JMP     S16             ; GO BACK AND TRY FOR IT
                    4535              ASSUME  DS:DATA
                    4536
                    4537     ;------ CHARACTER IS FOUND ( AL=0 IF NOT FOUND )
F6B1                4538     S18:
F6B1 83C408         4539              ADD     SP,8            ; READJUST THE STACK, THROW AWAY SAVE
F6B4 E910FB         4540              JMP     VIDEO_RETURN    ; ALL DONE
                    4541     GRAPHICS_READ    ENDP
                    4542     ;-------------------------------------------
                    4543     ; EXPAND_MED_COLOR
                    4544     ;  THIS ROUTINE EXPANDS THE LOW 2 BITS IN BL TO
                    4545     ;  FILL THE ENTIRE BX REGISTER
                    4546     ; ENTRY --
                    4547     ;  BL = COLOR TO BE USED ( LOW 2 BITS )
                    4548     ; EXIT --
                    4549     ;  BX = COLOR TO BE USED ( 8 REPLICATIONS OF THE 2 COLOR BITS )
                    4550     ;-------------------------------------------
F6B7                4551     S19      PROC    NEAR
F6B7 80E303         4552              AND     BL,3            ; ISOLATE THE COLOR BITS
F6BA 8AC3           4553              MOV     AL,BL           ; COPY TO AL
F6BC 51             4554              PUSH    CX              ; SAVE REGISTER
F6BD B90300         4555              MOV     CX,3            ; NUMBER OF TIMES TO DO THIS
F6C0                4556     S20:
F6C0 D0E0           4557              SAL     AL,1
F6C2 D0E0           4558              SAL     AL,1            ; LEFT SHIFT BY 2
F6C4 0AD8           4559              OR      BL,AL           ; ANOTHER COLOR VERSION INTO BL
F6C6 E2F8           4560              LOOP    S20             ; FILL ALL OF BL
F6C8 8AFB           4561              MOV     BH,BL           ; FILL UPPER PORTION
F6CA 59             4562              POP     CX              ; REGISTER BACK
F6CB C3             4563              RET                     ; ALL DONE
                    4564     S19      ENDP
                    4565     ;-------------------------------------------
                    4566     ; EXPAND_BYTE
                    4567     ;  THIS ROUTINE TAKES THE BYTE IN AL AND DOUBLES ALL
                    4568     ;  OF THE BITS, TURNING THE 8 BITS INTO 16 BITS.
                    4569     ;  THE RESULT IS LEFT IN AX
                    4570     ;-------------------------------------------
F6CC                4571     S21      PROC    NEAR
F6CC 52             4572              PUSH    DX              ; SAVE REGISTERS
F6CD 51             4573              PUSH    CX
F6CE 53             4574              PUSH    BX
F6CF BA0000         4575              MOV     DX,0            ; RESULT REGISTER
F6D2 B90100         4576              MOV     CX,1            ; MASK REGISTER
F6D5                4577     S22:
F6D5 8BD8           4578              MOV     BX,AX           ; BASE INTO TEMP
F6D7 23D9           4579              AND     BX,CX           ; USE MASK TO EXTRACT A BIT
F6D9 0BD3           4580              OR      DX,BX           ; PUT INTO RESULT REGISTER
F6DB D1E0           4581              SHL     AX,1
F6DD D1E1           4582              SHL     CX,1            ; SHIFT BASE AND MASK BY 1
F6DF 8BD8           4583              MOV     BX,AX           ; BASE TO TEMP
F6E1 23D9           4584              AND     BX,CX           ; EXTRACT THE SAME BIT
F6E3 0BD3           4585              OR      DX,BX           ; PUT INTO RESULT
```

```
LOC OBJ                SOURCE

F6E5 D1E1                      SHL     CX,1            ; SHIFT ONLY MASK NOW, MOVING TO NEXT BASE
F6E7 73EC                      JNC     S22             ; USE MASK BIT COMING OUT TO TERMINATE
F6E9 8BC2                      MOV     AX,DX           ; RESULT TO PARM REGISTER
F6EB 5B                        POP     BX
F6EC 59                        POP     CX              ; RECOVER REGISTERS
F6ED 5A                        POP     DX
F6EE C3                        RET                     ; ALL DONE
                       S21     ENDP
                       ;-------------------------------------------
                       ; MED_READ_BYTE
                       ; THIS ROUTINE WILL TAKE 2 BYTES FROM THE REGEN BUFFER,
                       ;  COMPARE AGAINST THE CURRENT FOREGROUND COLOR, AND PLACE
                       ;  THE CORRESPONDING ON/OFF BIT PATTERN INTO THE CURRENT
                       ;  POSITION IN THE SAVE AREA
                       ; ENTRY --
                       ;  SI,DS = POINTER TO REGEN AREA OF INTEREST
                       ;  BX = EXPANDED FOREGROUND COLOR
                       ;  BP = POINTER TO SAVE AREA
                       ; EXIT --
                       ;  BP IS INCREMENT AFTER SAVE
                       ;-------------------------------------------
F6EF                   S23     PROC    NEAR
F6EF 8A24                      MOV     AH,[SI]         ; GET FIRST BYTE
F6F1 8A4401                    MOV     AL,[SI+1]       ; GET SECOND BYTE
F6F4 B900C0                    MOV     CX,0C000H       ; 2 BIT MASK TO TEST THE ENTRIES
F6F7 B200                      MOV     DL,0            ; RESULT REGISTER
F6F9                   S24:
F6F9 85C1                      TEST    AX,CX           ; IS THIS SECTION BACKGROUND?
F6FB F8                        CLC                     ; CLEAR CARRY IN HOPES THAT IT IS
F6FC 7401                      JZ      S25             ; IF ZERO, IT IS BACKGROUND
F6FE F9                        STC                     ; WASN'T, SO SET CARRY
F6FF D0D2              S25:    RCL     DL,1            ; MOVE THAT BIT INTO THE RESULT
F701 D1E9                      SHR     CX,1
F703 D1E9                      SHR     CX,1            ; MOVE THE MASK TO THE RIGHT BY 2 BITS
F705 73F2                      JNC     S24             ; DO IT AGAIN IF MASK DIDN'T FALL OUT
F707 885600                    MOV     [BP],DL         ; STORE RESULT IN SAVE AREA
F70A 45                        INC     BP              ; ADJUST POINTER
F70B C3                        RET                     ; ALL DONE
                       S23     ENDP
                       ;----------------------------------------
                       ; V4_POSITION
                       ;  THIS ROUTINE TAKES THE CURSOR POSITION CONTAINED IN
                       ;  THE MEMORY LOCATION, AND CONVERTS IT INTO AN OFFSET
                       ;  INTO THE REGEN BUFFER, ASSUMING ONE BYTE/CHAR.
                       ;  FOR MEDIUM RESOLUTION GRAPHICS, THE NUMBER MUST
                       ;  BE DOUBLED.
                       ; ENTRY -- NO REGISTERS,MEMORY LOCATION CURSOR_POSN IS USED
                       ; EXIT--
                       ;  AX CONTAINS OFFSET INTO REGEN BUFFER
                       ;----------------------------------------
F70C                   S26     PROC    NEAR
F70C A15000         R          MOV     AX,CURSOR_POSN          ; GET CURRENT CURSOR
F70F                   GRAPH_POSN      LABEL   NEAR
F70F 53                        PUSH    BX                      ; SAVE REGISTER
F710 8BD8                      MOV     BX,AX                   ; SAVE A COPY OF CURRENT CURSOR
F712 8AC4                      MOV     AL,AH                   ; GET ROWS TO AL
F714 F6264A00       R          MUL     BYTE PTR CRT_COLS       ; MULTIPLY BY BYTES/COLUMN
F718 D1E0                      SHL     AX,1                    ; MULTIPLY * 4 SINCE 4 ROWS/BYTE
F71A D1E0                      SHL     AX,1
F71C 2AFF                      SUB     BH,BH                   ; ISOLATE COLUMN VALUE
F71E 03C3                      ADD     AX,BX                   ; DETERMINE OFFSET
F720 5B                        POP     BX                      ; RECOVER POINTER
F721 C3                        RET                             ; ALL DONE
                       S26     ENDP
                       ;-------------------------------------------
                       ; WRITE_TTY
                       ; THIS INTERFACE PROVIDES A TELETYPE LIKE INTERFACE TO THE
                       ;   VIDEO CARD.  THE INPUT CHARACTER IS WRITTEN TO THE CURRENT
                       ;   CURSOR POSITION, AND THE CURSOR IS MOVED TO THE NEXT POSITION.
                       ;   IF THE CURSOR LEAVES THE LAST COLUMN OF THE FIELD, THE COLUMN
                       ;   IS SET TO ZERO, AND THE ROW VALUE IS INCREMENTED.  IF THE ROW
                       ;   ROW VALUE LEAVES THE FIELD, THE CURSOR IS PLACED ON THE LAST ROW,
                       ;   FIRST COLUMN, AND THE ENTIRE SCREEN IS SCROLLED UP ONE LINE.
                       ;   WHEN THE SCREEN IS SCROLLED UP, THE ATTRIBUTE FOR FILLING THE
                       ;   NEWLY BLANKED LINE IS READ FROM THE CURSOR POSITION ON THE PREVIOUS
                       ;   LINE BEFORE THE SCROLL, IN CHARACTER MODE.  IN GRAPHICS MODE,
```

LOC OBJ LINE SOURCE

```
                ;    THE 0 COLOR IS USED.
                ; ENTRY --
                ;   (AH) = CURRENT CRT MODE
                ;   (AL) = CHARACTER TO BE WRITTEN
                ;          NOTE THAT BACK SPACE, CAR RET, BELL AND LINE FEED ARE HANDLED
                ;          AS COMMANDS RATHER THAN AS DISPLAYABLE GRAPHICS
                ;   (BL) = FOREGROUND COLOR FOR CHAR WRITE IF CURRENTLY IN A GRAPHICS MODE
                ; EXIT --
                ;    ALL REGISTERS SAVED
                ;------------------------------------------
                        ASSUME  CS:CODE,DS:DATA
F722            WRITE_TTY       PROC    NEAR
F722 50                 PUSH    AX              ; SAVE REGISTERS
F723 50                 PUSH    AX              ; SAVE CHAR TO WRITE
F724 B403               MOV     AH,3
F726 CD10               INT     10H             ; READ THE CURRENT CURSOR POSITION
F728 58                 POP     AX              ; RECOVER CHAR
                ;------ DX NOW HAS THE CURRENT CURSOR POSITION
F729 3C08               CMP     AL,8            ; IS IT A BACKSPACE
F72B 7459               JE      U8              ; BACK_SPACE
F72D 3C0D               CMP     AL,0DH          ; IS IT CARRIAGE RETURN
F72F 745E               JE      U9              ; CAR_RET
F731 3C0A               CMP     AL,0AH          ; IS IT A LINE FEED
F733 745E               JE      U10             ; LINE_FEED
F735 3C07               CMP     AL,07H          ; IS IT A BELL
F737 7461               JE      U11             ; BELL
                ;------ WRITE THE CHAR TO THE SCREEN
F739 8A3E6200     R     MOV     BH,ACTIVE_PAGE  ; GET THE CURRENT ACTIVE PAGE
F73D B40A               MOV     AH,10           ; WRITE CHAR ONLY
F73F B90100             MOV     CX,1            ; ONLY ONE CHAR
F742 CD10               INT     10H             ; WRITE THE CHAR
                ;------ POSITION THE CURSOR FOR NEXT CHAR
F744 FEC2               INC     DL
F746 3A164A00     R     CMP     DL,BYTE PTR CRT_COLS    ; TEST FOR COLUMN OVERFLOW
F74A 7536               JNZ     U7              ; SET_CURSOR
F74C B200               MOV     DL,0            ; COLUMN FOR CURSOR
F74E 80FE18             CMP     DH,24
F751 752D               JNZ     U6              ; SET_CURSOR_INC
                ;------ SCROLL REQUIRED
F753            U1:
F753 B402               MOV     AH,2
F755 B700               MOV     BH,0
F757 CD10               INT     10H             ; SET THE CURSOR
                ;------ DETERMINE VALUE TO FILL WITH DURING SCROLL
F759 A04900       R     MOV     AL,CRT_MODE     ; GET THE CURRENT MODE
F75C 3C04               CMP     AL,4
F75E 7206               JC      U2              ; READ-CURSOR
F760 3C07               CMP     AL,7
F762 B700               MOV     BH,0            ; FILL WITH BACKGROUND
F764 7506               JNE     U3              ; SCROLL-UP
F766            U2:                             ; READ-CURSOR
F766 B408               MOV     AH,8
F768 CD10               INT     10H             ; READ CHAR/ATTR AT CURRENT CURSOR
F76A 8AFC               MOV     BH,AH           ; STORE IN BH
F76C            U3:                             ; SCROLL-UP
F76C B80106             MOV     AX,601H         ; SCROLL ONE LINE
F76F B90000             MOV     CX,0            ; UPPER LEFT CORNER
F772 B618               MOV     DH,24           ; LOWER RIGHT ROW
F774 8A164A00     R     MOV     DL,BYTE PTR CRT_COLS    ; LOWER RIGHT COLUMN
F778 FECA               DEC     DL
F77A            U4:                             ; VIDEO-CALL-RETURN
F77A CD10               INT     10H             ; SCROLL UP THE SCREEN
F77C            U5:                             ; TTY-RETURN
F77C 58                 POP     AX              ; RESTORE THE CHARACTER
```

```
LOC OBJ                  SOURCE

F77D E947FA                     JMP     VIDEO_RETURN    ; RETURN TO CALLER

F780                     U6:                            ; SET-CURSOR-INC
F780 FEC6                       INC     DH              ; NEXT ROW
F782                     U7:                            ; SET-CURSOR
F782 B402                       MOV     AH,2
F784 EBF4                       JMP     U4              ; ESTABLISH THE NEW CURSOR

                         ;------ BACK SPACE FOUND

F786                     U8:
F786 80FA00                     CMP     DL,0            ; ALREADY AT END OF LINE
F789 74F7                       JE      U7              ; SET_CURSOR
F78B FECA                       DEC     DL              ; NO -- JUST MOVE IT BACK
F78D EBF3                       JMP     U7              ; SET_CURSOR

                         ;------ CARRIAGE RETURN FOUND

F78F                     U9:
F78F B200                       MOV     DL,0            ; MOVE TO FIRST COLUMN
F791 EBEF                       JMP     U7              ; SET_CURSOR

                         ;------ LINE FEED FOUND

F793                     U10:
F793 80FE18                     CMP     DH,24           ; BOTTOM OF SCREEN
F796 75E8                       JNE     U6              ; YES, SCROLL THE SCREEN
F798 EBB9                       JMP     U1              ; NO, JUST SET THE CURSOR

                         ;------ BELL FOUND

F79A                     U11:
F79A B302                       MOV     BL,2            ; SET UP COUNT FOR BEEP
F79C E8C7EE                     CALL    BEEP            ; SOUND THE POD BELL
F79F EBDB                       JMP     U5              ; TTY_RETURN
                         WRITE_TTY      ENDP
                         ;------------------------------------------
                         ; LIGHT PEN
                         ;       THIS ROUTINE TESTS THE LIGHT PEN SWITCH AND THE LIGHT
                         ;         PEN TRIGGER.  IF BOTH ARE SET, THE LOCATION OF THE LIGHT
                         ;         PEN IS DETERMINED.  OTHERWISE, A RETURN WITH NO INFORMATION
                         ;         IS MADE.
                         ;       ON EXIT:
                         ;       (AH) = 0 IF NO LIGHT PEN INFORMATION IS AVAILABLE
                         ;               BX,CX,DX ARE DESTROYED
                         ;       (AH) = 1 IF LIGHT PEN IS AVAILABLE
                         ;               (DH,DL) = ROW,COLUMN OF CURRENT LIGHT PEN POSITION
                         ;               (CH) = RASTER POSITION
                         ;               (BX) = BEST GUESS AT PIXEL HORIZONTAL POSITION
                         ;------------------------------------------
                                ASSUME  CS:CODE,DS:DATA
                         ;------ SUBTRACT_TABLE
F7A1                     V1     LABEL   BYTE
F7A1 0303050503030304           DB      3,3,5,5,3,3,3,4 ;
F7A9                     READ_LPEN      PROC    NEAR

                         ;------ WAIT FOR LIGHT PEN TO BE DEPRESSED

F7A9 B400                       MOV     AH,0            ; SET NO LIGHT PEN RETURN CODE
F7AB 8B166300        R          MOV     DX,ADDR_6845            ; GET BASE ADDRESS OF 6845
F7AF 83C206                     ADD     DX,6                    ; POINT TO STATUS REGISTER
F7B2 EC                         IN      AL,DX                   ; GET STATUS REGISTER
F7B3 A804                       TEST    AL,4                    ; TEST LIGHT PEN SWITCH
F7B5 7578                       JNZ     V6                      ; NOT SET, RETURN

                         ;------ NOW TEST FOR LIGHT PEN TRIGGER

F7B7 A802                       TEST    AL,2                    ; TEST LIGHT PEN TRIGGER
F7B9 747E                       JZ      V7                      ; RETURN WITHOUT RESETTING TRIGGER

                         ;------ TRIGGER HAS BEEN SET, READ THE VALUE IN

F7BB B410                       MOV     AH,16           ; LIGHT PEN REGISTERS ON 6845

                         ;------ INPUT REGS POINTED TO BY AH, AND CONVERT TO ROW COLUMN IN DX

F7BD 8B166300        R          MOV     DX,ADDR_6845    ; ADDRESS REGISTER FOR 6845
```

```
LOC OBJ                 LINE    SOURCE

F7C1 8AC4                               MOV     AL,AH           ; REGISTER TO READ
F7C3 EE                                 OUT     DX,AL           ; SET IT UP
F7C4 42                                 INC     DX              ; DATA REGISTER
F7C5 EC                                 IN      AL,DX           ; GET THE VALUE
F7C6 8AE8                               MOV     CH,AL           ; SAVE IN CX
F7C8 4A                                 DEC     DX              ; ADDRESS REGISTER
F7C9 FEC4                               INC     AH
F7CB 8AC4                               MOV     AL,AH           ; SECOND DATA REGISTER
F7CD EE                                 OUT     DX,AL
F7CE 42                                 INC     DX              ; POINT TO DATA REGISTER
F7CF EC                                 IN      AL,DX           ; GET SECOND DATA VALUE
F7D0 8AE5                               MOV     AH,CH           ; AX HAS INPUT VALUE

                                ;------ AX HAS THE VALUE READ IN FROM THE 6845

F7D2 8A1E4900        R                  MOV     BL,CRT_MODE
F7D6 2AFF                               SUB     BH,BH           ; MODE VALUE TO BX
F7D8 2E8A9FA1F7      R                  MOV     BL,CS:V1[BX]    ; DETERMINE AMOUNT TO SUBTRACT
F7DD 2BC3                               SUB     AX,BX           ; TAKE IT AWAY
F7DF 2B064E00        R                  SUB     AX,CRT_START    ; CONVERT TO CORRECT PAGE ORIGIN
F7E3 7903                               JNS     V2              ; IF POSITIVE, DETERMINE MODE
F7E5 B80000                             MOV     AX,0            ; <0 PLAYS AS 0

                                ;------ DETERMINE MODE OF OPERATION

F7E8                            V2:                             ; DETERMINE_MODE
F7E8 B103                               MOV     CL,3            ; SET *8 SHIFT COUNT
F7EA 803E490004      R                  CMP     CRT_MODE,4      ; DETERMINE IF GRAPHICS OR ALPHA
F7EF 722A                               JB      V4              ; ALPHA_PEN
F7F1 803E490007      R                  CMP     CRT_MODE,7
F7F6 7423                               JE      V4              ; ALPHA_PEN

                                ;------ GRAPHICS MODE

F7F8 B228                               MOV     DL,40           ; DIVISOR FOR GRAPHICS
F7FA F6F2                               DIV     DL              ; DETERMINE ROW(AL) AND COLUMN(AH)
                                                                ;  AL RANGE 0-99, AH RANGE 0-39
                                ;------ DETERMINE GRAPHIC ROW POSITION

F7FC 8AE8                               MOV     CH,AL           ; SAVE ROW VALUE IN CH
F7FE 02ED                               ADD     CH,CH           ; *2 FOR EVEN/ODD FIELD
F800 8ADC                               MOV     BL,AH           ; COLUMN VALUE TO BX
F802 2AFF                               SUB     BH,BH           ; MULTIPLY BY 8 FOR MEDIUM RES
F804 803E490006      R                  CMP     CRT_MODE,6      ; DETERMINE MEDIUM OR HIGH RES
F809 7504                               JNE     V3              ; NOT_HIGH_RES
F80B B104                               MOV     CL,4            ; SHIFT VALUE FOR HIGH RES
F80D D0E4                               SAL     AH,1            ; COLUMN VALUE TIMES 2 FOR HIGH RES
F80F                            V3:                             ; NOT_HIGH_RES
F80F D3E3                               SHL     BX,CL           ; MULTIPLY *16 FOR HIGH RES

                                ;------ DETERMINE ALPHA CHAR POSITION

F811 8AD4                               MOV     DL,AH           ; COLUMN VALUE FOR RETURN
F813 8AF0                               MOV     DH,AL           ; ROW VALUE
F815 D0EE                               SHR     DH,1            ; DIVIDE BY 4
F817 D0EE                               SHR     DH,1            ;  FOR VALUE IN 0-24 RANGE
F819 EB12                               JMP     SHORT V5        ; LIGHT_PEN_RETURN_SET

                                ;------ ALPHA MODE ON LIGHT PEN

F81B                            V4:                             ; ALPHA_PEN
F81B F6364A00        R                  DIV     BYTE PTR CRT_COLS       ; DETERMINE ROW,COLUMN VALUE
F81F 8AF0                               MOV     DH,AL           ; ROWS TO DH
F821 8AD4                               MOV     DL,AH           ; COLS TO DL
F823 D2E0                               SAL     AL,CL           ; MULTIPLY ROWS * 8
F825 8AE8                               MOV     CH,AL           ; GET RASTER VALUE TO RETURN REG
F827 8ADC                               MOV     BL,AH           ; COLUMN VALUE
F829 32FF                               XOR     BH,BH           ;  TO BX
F82B D3E3                               SAL     BX,CL
F82D                            V5:                             ; LIGHT_PEN_RETURN_SET
F82D B401                               MOV     AH,1            ; INDICATE EVERTHING SET
F82F                            V6:                             ; LIGHT_PEN_RETURN
F82F 52                                 PUSH    DX              ; SAVE RETURN VALUE (IN CASE)
F830 8B166300        R                  MOV     DX,ADDR_6845    ; GET BASE ADDRESS
F834 83C207                             ADD     DX,7            ; POINT TO RESET PARM
F837 EE                                 OUT     DX,AL           ; ADDRESS, NOT DATA, IS IMPORTANT
```

```
LOC OBJ                 LINE    SOURCE

F838 5A                                 POP     DX              ; RECOVER VALUE
F839                            V7:                             ; RETURN_NO_RESET
F839 5F                                 POP     DI
F83A 5E                                 POP     SI
F83B 1F                                 POP     DS                      ; DISCARD SAVED BX,CX,DX
F83C 1F                                 POP     DS
F83D 1F                                 POP     DS
F83E 1F                                 POP     DS
F83F 07                                 POP     ES
F840 CF                                 IRET
                                READ_LPEN       ENDP
                                ;--- 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
                                ;-------------------------------------------
                                        ASSUME  CS:CODE,DS:DATA
F841                            MEMORY_SIZE_DETERMINE   PROC    FAR
F841 FB                                 STI                     ; INTERRUPTS BACK ON
F842 1E                                 PUSH    DS              ; SAVE SEGMENT
F843 B84000             R               MOV     AX,DATA         ; ESTABLISH ADDRESSING
F846 8ED8                               MOV     DS,AX
F848 A11300             R               MOV     AX,MEMORY_SIZE  ; GET VALUE
F84B 1F                                 POP     DS              ; RECOVER SEGMENT
F84C 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 EQUPMENT
                                ;       PORT 3FA = INTERRUPT ID REGISTER OF 8250
                                ;               BITS 7-3 ARE ALWAYS 0
                                ;       PORT 378 = OUTPUT PORT OF PRINTER -- 8255 PORT THAT
                                ;               CAN BE READ AS WELL AS WRITTEN
                                ; OUTPUT
                                ;       (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 3,2 = PLANAR RAM SIZE (00=16K,01=32K,10=48K,11=64K)
                                ;       BIT 1 NOT USED
                                ;       BIT 0 = IPL FROM DISKETTE -- THIS BIT INDICATES THAT THERE ARE DISKETTE
                                ;               DRIVES ON THE SYSTEM
                                ;
                                ;       NO OTHER REGISTERS AFFECTED
                                ;-------------------------------------------
                                        ASSUME  CS:CODE,DS:DATA
```

```
LOC OBJ                   SOURCE

F84D                      EQUIPMENT       PROC    FAR
F84D FB                           STI                     ; INTERRUPTS BACK ON
F84E 1E                           PUSH    DS              ; SAVE SEGMENT REGISTER
F84F B84000          R            MOV     AX,DATA         ; ESTABLISH ADDRESSING
F852 8ED8                         MOV     DS,AX
F854 A11000          R            MOV     AX,EQUIP_FLAG   ; GET THE CURRENT SETTINGS
F857 1F                           POP     DS              ; RECOVER SEGMENT
F858 CF                           IRET                    ; RETURN TO CALLER
                          EQUIPMENT       ENDP
                          ;--- INT 15 --------------------------------
                          ; CASSETTE I/O
                          ;       (AH) = 0  TURN CASSETTE MOTOR ON
                          ;       (AH) = 1  TURN CASSETTE MOTOR OFF
                          ;       (AH) = 2  READ 1 OR MORE 256 BYTE BLOCKS FROM CASSETTE
                          ;               (ES,BX) = POINTER TO DATA BUFFER
                          ;               (CX) = COUNT OF BYTES TO READ
                          ;               ON EXIT:
                          ;               (ES,BX) = POINTER TO LAST BYTE READ + 1
                          ;               (DX) = COUNT OF BYTES ACTUALLY READ
                          ;               (CY) = 0 IF NO ERROR OCCURRED
                          ;                    = 1 IF ERROR OCCURRED
                          ;               (AH) = ERROR RETURN IF (CY)= 1
                          ;                       = 01 IF CRC ERROR WAS DETECTED
                          ;                       = 02 IF DATA TRANSITIONS ARE LOST
                          ;                       = 04 IF NO DATA WAS FOUND
                          ;       (AH) = 3 WRITE 1 OR MORE 256 BYTE BLOCKS TO CASSETTE
                          ;               (ES,BX) = POINTER TO DATA BUFFER
                          ;               (CX) = COUNT OF BYTES TO WRITE
                          ;               ON EXIT:
                          ;               (EX,BX) = POINTER TO LAST BYTE WRITTEN + 1
                          ;               (CX) = 0
                          ;       (AH) = ANY OTHER THAN ABOVE VALUES CAUSES (CY)= 1
                          ;               AND (AH)= 80 TO BE RETURNED (INVALID COMMAND).
                          ;------------------------------------------
                                  ASSUME  DS:DATA, ES:NOTHING,SS:NOTHING,CS:CODE
F859                      CASSETTE_IO     PROC    FAR
F859 FB                           STI                             ; INTERRUPTS BACK ON
F85A 1E                           PUSH    DS                      ; ESTABLISH ADDRESSING TO DATA
F85B 50                           PUSH    AX
F85C B84000          R            MOV     AX, DATA
F85F 8ED8                         MOV     DS, AX
F861 802671007F      R            AND     BIOS_BREAK, 7FH         ; MAKE SURE BREAK FLAG IS OFF
F866 58                           POP     AX
F867 E80400                       CALL    W1                      ; CASSETTE_IO_CONT
F86A 1F                           POP     DS
F86B CA0200                       RET     2               ; INTERRUPT RETURN
                          CASSETTE_IO     ENDP
F86E                      W1      PROC    NEAR
                          ;------------------------------------------
                          ; PURPOSE:
                          ;  TO CALL APPROPRIATE ROUTINE DEPENDING ON REG AH
                          ;
                          ;  AH           ROUTINE
                          ;______________________________________
                          ;  0            MOTOR ON
                          ;  1            MOTOR OFF
                          ;  2            READ CASSETTE BLOCK
                          ;  3            WRITE CASSETTE BLOCK
                          ;______________________________________

F86E 0AE4                         OR      AH,AH           ;TURN ON MOTOR?
F870 7413                         JZ      MOTOR_ON        ;YES, DO IT
F872 FECC                         DEC     AH              ;TURN OFF MOTOR?
F874 7418                         JZ      MOTOR_OFF       ;YES, DO IT
F876 FECC                         DEC     AH              ;READ CASSETTE BLOCK?
F878 741A                         JZ      READ_BLOCK      ;YES, DO IT
F87A FECC                         DEC     AH              ;WRITE CASSETTE BLOCK?
F87C 7503                         JNZ     W2              ; NOT_DEFINED
F87E E92701                       JMP     WRITE_BLOCK     ;YES, DO IT

F881                      W2:                             ;COMMAND NOT DEFINED
F881 B480                         MOV     AH,080H         ;ERROR, UNDEFINED OPERATION
F883 F9                           STC                     ;ERROR FLAG
F884 C3                           RET
                          W1      ENDP

```

```
LOC  OBJ                SOURCE

F885                    MOTOR_ON        PROC    NEAR
                        ;---------------------------------
                        ; PURPOSE:
                        ;   TO TURN ON CASSETTE MOTOR
                        ;----------------------------------
F885 E461                       IN      AL,PORT_B               ;READ CASSETTE OUTPUT
F887 24F7                       AND     AL,NOT 08H              ; CLEAR BIT TO TURN ON MOTOR
F889 E661               W3:     OUT     PORT_B,AL               ;WRITE IT OUT
F88B 2AE4                       SUB     AH,AH                   ;CLEAR AH
F88D C3                         RET
                        MOTOR_ON        ENDP

F88E                    MOTOR_OFF       PROC    NEAR
                        ;-----------------------------------
                        ; PURPOSE:
                        ;   TO TURN CASSETTE MOTOR OFF
                        ;-------------------------------------
F88E E461                       IN      AL,PORT_B               ;READ CASSETTE OUTPUT
F890 0C08                       OR      AL,08H                  ; SET BIT TO TURN OFF
F892 EBF5                       JMP     W3                      ;WRITE IT, CLEAR ERROR, RETURN
                        MOTOR_OFF       ENDP
F894                    READ_BLOCK      PROC    NEAR
                        ;--------------------------------------------
                        ; PURPOSE:
                        ;   TO 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 MEM
                        ;   CX CONTAINS DECREMENTED BYTE COUNT
                        ;   DX CONTAINS NUMBER OF BYTES ACTUALLY READ
                        ;
                        ;   CARRY FLAG IS CLEAR IF NO ERROR DETECTED
                        ;   CARRY FLAG IS SET IF CRC ERROR DETECTED
                        ;--------------------------------------------
F894 53                         PUSH    BX                      ;SAVE BX
F895 51                         PUSH    CX                      ;SAVE CX
F896 56                         PUSH    SI                      ; SAVE SI
F897 BE0700                     MOV     SI, 7                   ; SET UP RETRY COUNT FOR LEADER
F89A E8C201                     CALL    BEGIN_OP                ; BEGIN BY STARTING MOTOR
F89D                    W4:                                     ; SEARCH FOR LEADER
F89D E462                       IN      AL,PORT_C               ;GET INTIAL VALUE
F89F 2410                       AND     AL,010H                 ;MASK OFF EXTRANEOUS BITS
F8A1 A26B00         R           MOV     LAST_VAL,AL             ;SAVE IN LOC LAST_VAL
F8A4 BA7A3F                     MOV     DX,16250                ; # OF TRANSITIONS TO LOOK FOR

F8A7                    W5:                                     ; WAIT_FOR_EDGE
F8A7 F606710080     R           TEST    BIOS_BREAK, 80H         ; CHECK FOR BREAK KEY
F8AC 7403                       JZ      W6                      ; JUMP IF NO BREAK KEY
F8AE E98A00                     JMP     W17                     ; JUMP IF BREAK KEY HIT

F8B1 4A                 W6:     DEC     DX
F8B2 7503                       JNZ     W7                      ; JUMP IF BEGINNING OF LEADER
F8B4 E98400                     JMP     W17                     ; JUMP IF NO LEADER FOUND

F8B7 E8C600             W7:     CALL    READ_HALF_BIT           ;IGNORE FIRST EDGE
F8BA E3EB                       JCXZ    W5                      ; JUMP IF NO EDGE DETECTED
F8BC BA7803                     MOV     DX,0378H                ; CHECK FOR HALF BITS
F8BF B90002                     MOV     CX,200H                 ;MUST HAVE AT LEAST THIS MANY ONE SIZE
                                                                ;PULSES BEFORE CHECKNG FOR SYNC BIT (0)
F8C2 E421                       IN      AL, 021H                ; INTERRUPT MASK REGISTER
F8C4 0C01                       OR      AL,1                    ; DISABLE TIMER INTERRUPTS
F8C6 E621                       OUT     021H, AL
F8C8                    W8:                                     ; SEARCH-LDR
F8C8 F606710080     R           TEST    BIOS_BREAK, 80H         ; CHECK FOR BREAK KEY
F8CD 756C                       JNZ     W17                     ; JUMP IF BREAK KEY HIT
F8CF 51                         PUSH    CX                      ;SAVE REG CX
F8D0 E8AD00                     CALL    READ_HALF_BIT           ;GET PULSE WIDTH
F8D3 0BC9                       OR      CX, CX                  ; CHECK FOR TRANSITION
F8D5 59                         POP     CX                      ;RESTORE ONE BIT COUNTER
F8D6 74C5                       JZ      W4                      ; JUMP IF NO TRANSITION
F8D8 3BD3                       CMP     DX,BX                   ;CHECK PULSE WIDTH
```

```
LOC OBJ                 LINE    SOURCE

F8DA E304                               JCXZ    W9                      ;IF CX=0 THEN WE CAN LOOK
                                                                        ;FOR SYNC BIT (0)
F8DC 73BF                               JNC     W4                      ; JUMP IF ZERO BIT (NOT GOOD LEADER)
F8DE E2E8                               LOOP    W8                      ;DEC CX AND READ ANOTHER HALF ONE BIT
F8E0                            W9:                                     ; FIND-SYNC
F8E0 72E6                               JC      W8                      ; JUMP IF ONE BIT (STILL LEADER)

                                ; A SYNCH BIT HAS BEEN FOUND.  READ SYN CHARACTER:

F8E2 E89B00                             CALL    READ_HALF_BIT           ;SKIP OTHER HALF OF SYNC BIT (0)
F8E5 E86A00                             CALL    READ_BYTE               ; READ SYN BYTE
F8E8 3C16                               CMP     AL, 16H                 ; SYNCHRONIZATION CHARACTER
F8EA 7549                               JNE     W16                     ; JUMP IF BAD LEADER FOUND.

                                ;------ GOOD CRC SO READ DATA BLOCK(S)
F8EC 5E                                 POP     SI                      ; RESTORE REGS
F8ED 59                                 POP     CX
F8EE 5B                                 POP     BX
                                ;------------------------------------------
                                ;  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 MEM
                                ;  CX CONTAINS DECREMENTED BYTE COUNT
                                ;  DX CONTAINS NUMBER OF BYTES ACTUALLY READ
                                ;------------------------------------------
F8EF 51                                 PUSH    CX                      ;SAVE BYTE COUNT
F8F0                            W10:                                    ;COME HERE BEFORE EACH
                                                                        ;256 BYTE BLOCK IS READ
F8F0 C7066900FFFF    R                  MOV     CRC_REG,0FFFFH          ;INIT CRC REG
F8F6 BA0001                             MOV     DX,256                  ;SET DX TO DATA BLOCK SIZE
F8F9                            W11:                                    ; RD_BLK
F8F9 F606710080      R                  TEST    BIOS_BREAK, 80H         ; CHECK FOR BREAK KEY
F8FE 7523                               JNZ     W13                     ; JUMP IF BREAK KEY HIT
F900 E84F00                             CALL    READ_BYTE               ;READ BYTE FROM CASSETTE
F903 721E                               JC      W13                     ;CY SET INDICATES NO DATA TRANSITIONS
F905 E305                               JCXZ    W12                     ;IF WE'VE ALREADY REACHED
                                                                        ;END OF MEMORY BUFFER
                                                                        ;SKIP REST OF BLOCK
F907 268807                             MOV     ES:[BX],AL              ;STORE DATA BYTE AT BYTE PTR
F90A 43                                 INC     BX                      ;INC BUFFER PTR
F90B 49                                 DEC     CX                      ;DEC BYTE COUNTER
F90C                            W12:                    ; LOOP UNTIL DATA BLOCK HAS BEEN READ FROM CASSETTE.
F90C 4A                                 DEC     DX                      ;DEC BLOCK CNT
F90D 7FEA                               JG      W11                     ; RD_BLK
F90F E84000                             CALL    READ_BYTE               ;NOW READ TWO CRC BYTES
F912 E83D00                             CALL    READ_BYTE
F915 2AE4                               SUB     AH,AH                   ;CLEAR AH
F917 813E69000F1D    R                  CMP     CRC_REG,1D0FH           ;IS THE CRC CORRECT
F91D 7506                               JNE     W14                     ;IF NOT EQUAL CRC IS BAD
F91F E306                               JCXZ    W15                     ;IF BYTE COUNT IS ZERO
                                                                        ;THEN WE HAVE READ ENOUGH
                                                                        ;SO WE WILL EXIT
F921 EBCD                               JMP     W10                     ;STILL MORE, SO READ ANOTHER BLOCK
F923                            W13:                                    ;MISSING-DATA
                                                                        ;NO DATA TRANSITIONS SO
F923 B401                               MOV     AH,01H                  ;SET AH=02 TO INDICATE
                                                                        ;DATA TIMEOUT
F925                            W14:                                    ; BAD-CRC
F925 FEC4                               INC     AH                      ;EXIT EARLY ON ERROR
                                                                        ;SET AH=01 TO INDICATE CRC ERROR
F927                            W15:                                    ; RD-BLK-EX
F927 5A                                 POP     DX                      ;CALCULATE COUNT OF
F928 2BD1                               SUB     DX,CX                   ;DATA BYTES ACTUALLY READ
                                                                        ;RETURN COUNT IN REG DX
F92A 50                                 PUSH    AX                      ;SAVE AX (RET CODE)
F92B F6C403                             TEST    AH, 03H                 ; CHECK FOR ERRORS
F92E 7513                               JNZ     W18                     ; JUMP IF ERROR DETECTED
F930 E81F00                             CALL    READ_BYTE               ;READ TRAILER
F933 EB0E                               JMP     SHORT W18               ;SKIP TO TURN OFF MOTOR
F935                            W16:                                    ; BAD-LEADER
F935 4E                                 DEC     SI                      ; CHECK RETRIES
```

```
LOC OBJ          LINE    SOURCE

F936 7403                        JZ      W17                 ; JUMP IF TOO MANY RETRIES
F938 E962FF                      JMP     W4                  ; JUMP IF NOT TOO MANY RETRIES
F93B                     W17:                                ; NO VALID DATA FOUND
                         ;------ NO DATA FROM CASSETTE ERROR, I.E. TIMEOUT

F93B 5E                          POP     SI                  ; RESTORE REGS
F93C 59                          POP     CX                  ;RESTORE REGS
F93D 5B                          POP     BX
F93E 2BD2                        SUB     DX,DX               ;ZERO NUMBER OF BYTES READ
F940 B404                        MOV     AH,04H              ;TIME OUT ERROR (NO LEADER)
F942 50                          PUSH    AX
F943                     W18:                                ; MOT-OFF
F943 E421                        IN      AL, 021H            ; RE_ENABLE INTERRUPTS
F945 24FE                        AND     AL, 0FFH- 1
F947 E621                        OUT     021H, AL
F949 E842FF                      CALL    MOTOR_OFF           ;TURN OFF MOTOR
F94C 58                          POP     AX                  ;RESTORE RETURN CODE
F94D 80FC01                      CMP     AH,01H              ;SET CARRY IF ERROR (AH>0)
F950 F5                          CMC
F951 C3                          RET                         ;FINISHED
                         READ_BLOCK      ENDP
                         ;------------------------------------------
F952                     READ_BYTE       PROC    NEAR
                         ; PURPOSE:
                         ;  TO READ A BYTE FROM CASSETTE
                         ;
                         ; ON EXIT REG AL CONTAINS READ DATA BYTE
                         ;--------------------------------------------
F952 53                          PUSH    BX                  ;SAVE REGS BX,CX
F953 51                          PUSH    CX
F954 B108                        MOV     CL,8H               ; SET BIT COUNTER FOR 8 BITS
F956                     W19:                                ; BYTE-ASM
F956 51                          PUSH    CX                  ;SAVE CX
                         ;------------------------------------------
                         ;  READ DATA BIT FROM CASSETTE
                         ;------------------------------------------
F957 E82600                      CALL    READ_HALF_BIT       ;READ ONE PULSE
F95A E320                        JCXZ    W21                 ;IF CX=0 THEN TIMEOUT
                                                             ;BECAUSE OF NO DATA TRANSITIONS
F95C 53                          PUSH    BX                  ;SAVE 1ST HALF BIT'S
                                                             ;PULSE WIDTH (IN BX)
F95D E82000                      CALL    READ_HALF_BIT       ;READ COMPLEMENTARY PULSE
F960 58                          POP     AX                  ;COMPUTE DATA BIT
F961 E319                        JCXZ    W21                 ;IF CX=0 THEN TIMEOUT DUE TO
                                                             ;NO DATA TRANSITIONS
F963 03D8                        ADD     BX,AX               ;PERIOD
F965 81FBF006                    CMP     BX, 06F0H           ; CHECK FOR ZERO BIT
F969 F5                          CMC                         ; CARRY IS SET IF ONE BIT
F96A 9F                          LAHF                        ;SAVE CARRY IN AH
F96B 59                          POP     CX                  ;RESTORE CX
                                                             ;NOTE:
                                                             ; MS BIT OF BYTE IS READ FIRST.
                                                             ; REG CH IS SHIFTED LEFT WITH
                                                             ; CARRY BEING INSERTED INTO LS
                                                             ;   BIT OF CH.
                                                             ; AFTER ALL 8 BITS HAVE BEEN
                                                             ;   READ, THE MS BIT OF THE DATA BYTE
                                                             ;   WILL BE IN THE MS BIT OF REG CH
F96C D0D5                        RCL     CH,1                ;ROTATE REG CH LEFT WITH CARRY TO
                                                             ;   LS BIT OF REG CH
F96E 9E                          SAHF                        ;RESTORE CARRY FOR CRC ROUTINE
F96F E8D900                      CALL    CRC_GEN             ;GENERATE CRC FOR BIT
F972 FEC9                        DEC     CL                  ;LOOP TILL ALL 8 BITS OF DATA
                                                             ;ASSEMBLED IN REG CH
F974 75E0                        JNZ     W19                 ; BYTE_ASM
F976 8AC5                        MOV     AL,CH               ;RETURN DATA BYTE IN REG AL
F978 F8                          CLC
F979                     W20:                                ; RD-BYT-EX
F979 59                          POP     CX                  ;RESTORE REGS CX,BX
F97A 5B                          POP     BX
F97B C3                          RET                         ;FINISHED
F97C                     W21:                                ; NO-DATA
F97C 59                          POP     CX                  ;RESTORE CX
F97D F9                          STC                         ;INDICATE ERROR
```

```
LOC OBJ                 LINE    SOURCE

F97E EBF9                              JMP     W20                         ; RD_BYT_EX
                                READ_BYTE       ENDP
                                ;------------------------------------------
F980                            READ_HALF_BIT   PROC    NEAR
                                ; PURPOSE:
                                ;  TO COMPUTE TIME TILL NEXT DATA
                                ;  TRANSITION (EDGE)
                                ;
                                ; ON ENTRY:
                                ;  EDGE_CNT CONTAINS LAST EDGE COUNT
                                ;
                                ; ON EXIT:
                                ;  AX CONTAINS OLD LAST EDGE COUNT
                                ;  BX CONTAINS PULSE WIDTH (HALF BIT)
                                ;---------------------------------------
F980 B96400                            MOV     CX, 100                     ; SET TIME TO WAIT FOR BIT
F983 8A266B00          R               MOV     AH,LAST_VAL                 ;GET PRESENT INPUT VALUE
F987                            W22:                                        ; RD-H-BIT
F987 E462                              IN      AL,PORT_C                   ;INPUT DATA BIT
F989 2410                              AND     AL,010H                     ;MASK OFF EXTRANEOUS BITS
F98B 3AC4                              CMP     AL,AH                       ;SAME AS BEFORE?
F98D E1F8                              LOOPE   W22                         ;LOOP TILL IT CHANGES
F98F A26B00            R               MOV     LAST_VAL,AL                 ;UPDATE LAST_VAL WITH NEW VALUE
F992 B000                              MOV     AL,0                        ;READ TIMER'S COUNTER COMMAND
F994 E643                              OUT     TIM_CTL,AL                  ;LATCH COUNTER
F996 E440                              IN      AL,TIMER0                   ;GET LS BYTE
F998 8AE0                              MOV     AH,AL                       ;SAVE IN AH
F99A E440                              IN      AL,TIMER0                   ;GET MS BYTE
F99C 86C4                              XCHG    AL,AH                       ;XCHG AL,AH
F99E 8B1E6700          R               MOV     BX,EDGE_CNT                 ;BX GETS LAST EDGE COUNT
F9A2 2BD8                              SUB     BX,AX                       ;SET BX EQUAL TO HALF BIT PERIOD
F9A4 A36700            R               MOV     EDGE_CNT,AX                 ;UPDATE EDGE COUNT;
F9A7 C3                                RET
                                READ_HALF_BIT   ENDP
                                ;----------------------------------------
F9A8                            WRITE_BLOCK     PROC    NEAR
                                ;
                                ; WRITE 1 OR MORE 256 BYTE BLOCKS TO CASSETTE.
                                ;       THE DATA IS PADDED TO FILL OUT THE LAST 256 BYTE BLOCK.
                                ;
                                ;  ON ENTRY:
                                ;   BX POINTS TO MEMORY BUFFER ADDRESS
                                ;   CX CONTAINS NUMBER OF BYTES TO WRITE
                                ;
                                ;  ON EXIT:
                                ;   BX POINTS 1 BYTE PAST LAST BYTE WRITTEN TO CASSETTE
                                ;   CX IS ZERO
                                ;------------------------------------------
F9A8 53                                PUSH    BX
F9A9 51                                PUSH    CX
F9AA E461                              IN      AL,PORT_B                   ;DISABLE SPEAKER
F9AC 24FD                              AND     AL,NOT 02H
F9AE 0C01                              OR      AL, 01H                     ; ENABLE TIMER
F9B0 E661                              OUT     PORT_B,AL
F9B2 B0B6                              MOV     AL,0B6H                     ; SET UP TIMER -- MODE 3 SQUARE WAVE
F9B4 E643                              OUT     TIM_CTL,AL
F9B6 E8A600                            CALL    BEGIN_OP                    ; START MOTOR AND DELAY
F9B9 B8A004                            MOV     AX,1184                     ; SET NORMAL BIT SIZE
F9BC E88500                            CALL    W31                         ; SET_TIMER
F9BF B90008                            MOV     CX,0800H                    ;SET CX FOR LEADER BYTE COUNT
F9C2                            W23:                                        ; WRITE LEADER
F9C2 F9                                STC                                 ; WRITE ONE BITS
F9C3 E86800                            CALL    WRITE_BIT                   ;
F9C6 E2FA                              LOOP    W23                         ; LOOP 'TIL LEADER IS WRITTEN
F9C8 F8                                CLC                                 ;WRITE SYNC BIT (0)
F9C9 E86200                            CALL    WRITE_BIT
F9CC 59                                POP     CX                          ;RESTORE REGS CX,BX
F9CD 5B                                POP     BX
F9CE B016                              MOV     AL, 16H                     ; WRITE SYN CHARACTER
F9D0 E84400                            CALL    WRITE_BYTE                  ;
```

```
LOC OBJ                       SOURCE

                              ;--------------------------------------------
                              ;  WRITE 1 OR MORE 256 BYTE BLOCKS TO CASSETTE
                              ;
                              ;  ON ENTRY:
                              ;    BX POINTS TO MEMORY BUFFER ADDRESS
                              ;    CX CONTAINS NUMBER OF BYTES TO WRITE
                              ;
                              ;  ON EXIT:
                              ;    BX POINTS 1 BYTE PAST LAST BYTE WRITTEN TO CASSETTE
                              ;    CX IS ZERO
                              ;--------------------------------------------
F9D3                          WR_BLOCK:
F9D3 C7066900FFFF    R                MOV     CRC_REG,0FFFFH          ;INIT CRC
F9D9 BA0001                           MOV     DX,256                  ;FOR 256 BYTES
F9DC                          W24:                                    ; WR-BLK
F9DC 268A07                           MOV     AL,ES:[BX]              ;READ BYTE FROM MEM
F9DF E83500                           CALL    WRITE_BYTE              ;WRITE IT TO CASSETTE
F9E2 E302                             JCXZ    W25                     ;UNLESS CX=0, ADVANCE PTRS & DEC COUNT
F9E4 43                               INC     BX                      ;INC BUFFER POINTER
F9E5 49                               DEC     CX                      ;DEC BYTE COUNTER
F9E6                          W25:                                    ; SKIP-ADV
F9E6 4A                               DEC     DX                      ;DEC BLOCK CNT
F9E7 7FF3                             JG      W24                     ;LOOP TILL 256 BYTE BLOCK
                                                                      ;  IS WRITTEN TO TAPE
                              ;------------------- WRITE CRC --------------
                              ;  WRITE 1'S COMPLEMENT OF CRC REG TO CASSETTE
                              ;  WHICH IS CHECKED FOR CORRECTNESS WHEN THE BLOCK IS READ
                              ;
                              ; REG AX IS MODIFIED
                              ;-------------------------------------------
F9E9 A16900          R                MOV     AX,CRC_REG              ;WRITE THE ONE'S COMPLEMENT OF THE
                                                                      ;  TWO BYTE CRC TO TAPE
F9EC F7D0                             NOT     AX                      ;FOR 1'S COMPLEMENT
F9EE 50                               PUSH    AX                      ;SAVE IT
F9EF 86E0                             XCHG    AH,AL                   ;WRITE MS BYTE FIRST
F9F1 E82300                           CALL    WRITE_BYTE              ;WRITE IT
F9F4 58                               POP     AX                      ;GET IT BACK
F9F5 E81F00                           CALL    WRITE_BYTE              ;NOW WRITE LS BYTE
F9F8 0BC9                             OR      CX,CX                   ;IS BYTE COUNT EXHAUSTED?
F9FA 75D7                             JNZ     WR_BLOCK                ;JUMP IF NOT DONE YET
F9FC 51                               PUSH    CX                      ;SAVE REG CX
F9FD B92000                           MOV     CX, 32                  ;WRITE OUT TRAILER BITS
FA00                          W26:                                    ; TRAIL-LOOP
FA00 F9                               STC
FA01 E82A00                           CALL    WRITE_BIT
FA04 E2FA                             LOOP    W26                     ; WRITE UNTIL TRAILER WRITTEN
FA06 59                               POP     CX                      ;RESTORE REG CX
FA07 B0B0                             MOV     AL, 0B0H                ; TURN TIMER2 OFF
FA09 E643                             OUT     TIM_CTL, AL
FA0B B80100                           MOV     AX, 1
FA0E E83300                           CALL    W31                     ; SET_TIMER
FA11 E87AFE                           CALL    MOTOR_OFF               ;TURN MOTOR OFF
FA14 2BC0                             SUB     AX,AX                   ;NO ERRORS REPORTED ON WRITE OP
FA16 C3                               RET                             ;FINISHED
                              WRITE_BLOCK     ENDP
                              ;-------------------------------------------
FA17                          WRITE_BYTE      PROC    NEAR
                              ;  WRITE A BYTE TO CASSETTE.
                              ;  BYTE TO WRITE IS IN REG AL.
                              ;--------------------------------------------
FA17 51                               PUSH    CX                      ;SAVE REGS CX,AX
FA18 50                               PUSH    AX
FA19 8AE8                             MOV     CH,AL                   ;AL=BYTE TO WRITE.
                                                                      ;   (MS BIT WRITTEN FIRST)
FA1B B108                             MOV     CL,8                    ;FOR 8 DATA BITS IN BYTE.
                                                                      ;   NOTE: TWO EDGES PER BIT
FA1D                          W27:                                    ; DISASSEMBLE THE DATA BIT
FA1D D0D5                             RCL     CH,1                    ;ROTATE MS BIT INTO CARRY
FA1F 9C                               PUSHF                           ;SAVE FLAGS.
                                                                      ;   NOTE: DATA BIT IS IN CARRY
FA20 E80B00                           CALL    WRITE_BIT               ;WRITE DATA BIT
FA23 9D                               POPF                            ;RESTORE CARRY FOR CRC CALC
FA24 E82400                           CALL    CRC_GEN                 ;COMPUTE CRC ON DATA BIT
FA27 FEC9                             DEC     CL                      ;LOOP TILL ALL 8 BITS DONE
FA29 75F2                             JNZ     W27                     ; JUMP IF NOT DONE YET
FA2B 58                               POP     AX                      ;RESTORE REGS AX,CX
FA2C 59                               POP     CX
```

```
LOC OBJ                 SOURCE

FA2D C3                         RET                             ;WE ARE FINISHED
                        WRITE_BYTE      ENDP
                        ;---------------------------------------
FA2E                    WRITE_BIT       PROC    NEAR
                        ; PURPOSE:
                        ;
                        ;  TO WRITE A DATA BIT TO CASSETTE
                        ;  CARRY FLAG CONTAINS DATA BIT
                        ;  I.E. IF SET   DATA BIT IS A ONE
                        ;       IF CLEAR DATA BIT IS A ZERO
                        ;
                        ;  NOTE: TWO EDGES ARE WRITTEN PER BIT
                        ;        ONE BIT  HAS 500 USEC BETWEEN EDGES
                        ;            FOR A 1000 USEC PERIOD (1 MILLISEC)
                        ;
                        ;        ZERO BIT HAS 250 USEC BETWEEN EDGES
                        ;            FOR A  500 USEC PERIOD (.5 MILLISEC)
                        ; CARRY FLAG IS DATA BIT
                        ;------------------------------------------
                                                                ;ASSUME IT'S A '1'
FA2E B8A004                     MOV     AX,1184                 ; SET AX TO NOMINAL ONE SIZE
FA31 7203                       JC      W28                     ; JUMP IF ONE BIT
FA33 B85002                     MOV     AX,592                  ; NO, SET TO NOMINAL ZERO SIZE
FA36                    W28:                                    ; WRITE-BIT-AX
FA36 50                         PUSH    AX                      ;WRITE BIT WITH PERIOD EQ TO VALUE AX
FA37                    W29:
FA37 E462                       IN      AL,PORT_C               ;INPUT TIMER_0 OUTPUT
FA39 2420                       AND     AL,020H
FA3B 74FA                       JZ      W29                     ;LOOP TILL HIGH
FA3D                    W30:
FA3D E462                       IN      AL,PORT_C               ;NOW WAIT TILL TIMER'S OUTPUT IS LOW
FA3F 2420                       AND     AL,020H
FA41 75FA                       JNZ     W30
                                                                ;RELOAD TIMER WITH PERIOD
                                                                ;FOR NEXT DATA BIT
FA43 58                         POP     AX                      ;RESTORE PERIOD COUNT
FA44                    W31:                                    ; SET TIMER
FA44 E642                       OUT     042H, AL                ; SET LOW BYTE OF TIMER 2
FA46 8AC4                       MOV     AL, AH
FA48 E642                       OUT     042H, AL                ; SET HIGH BYTE OF TIMER 2
FA4A C3                         RET
                        WRITE_BIT       ENDP
                        ;---------------------------------------
FA4B                    CRC_GEN         PROC    NEAR
                        ;  UPDATE CRC REGISTER WITH NEXT DATA BIT
                        ;
                        ;  CRC IS USED TO DETECT READ ERRORS
                        ;
                        ;  ASSUMES DATA BIT IS IN CARRY
                        ;
                        ;  REG AX IS MODIFIED
                        ;  FLAGS ARE MODIFIED
                        ;---------------------------------------
FA4B A16900        R            MOV     AX,CRC_REG
                                                                ;THE FOLLOWING INSTUCTIONS
                                                                ;WILL SET THE OVERFLOW FLAG
                                                                ;IF CARRY AND MS BIT OF CRC
                                                                ;ARE UNEQUAL
FA4E D1D8                       RCR     AX,1
FA50 D1D0                       RCL     AX,1
FA52 F8                         CLC                             ;CLEAR CARRY
FA53 7104                       JNO     W32                     ;SKIP IF NO OVERFLOW
                                                                ;IF DATA BIT XORED WITH
                                                                ; CRC REG BIT 15 IS ONE
FA55 351008                     XOR     AX,0810H                ;THEN XOR CRC REG WITH
                                                                ; 0810H
FA58 F9                         STC                             ;SET CARRY
FA59                    W32:
FA59 D1D0                       RCL     AX,1                    ;ROTATE CARRY (DATA BIT)
                                                                ;INTO CRC REG
FA5B A36900        R            MOV     CRC_REG,AX              ;UPDATE CRC_REG
FA5E C3                         RET                             ;FINISHED
                        CRC_GEN         ENDP
                        ;------------------------------------------
FA5F                    BEGIN_OP        PROC    NEAR            ; START TAPE AND DELAY
                        ;
                        ;------------------------------------------
```

```
LOC OBJ                  SOURCE

FA5F E823FE                      CALL    MOTOR_ON                ;TURN ON MOTOR
FA62 B342                        MOV     BL,42H                  ;DELAY FOR TAPE DRIVE
                                                                 ;TO GET UP TO SPEED  (1/2 SEC)
FA64                     W33:
FA64 B90007                      MOV     CX,700H                 ;INNER LOOP= APPROX. 10 MILLISEC
FA67 E2FE                W34:    LOOP    W34
FA69 FECB                        DEC     BL
FA6B 75F7                        JNZ     W33
FA6D C3                          RET
                         BEGIN_OP        ENDP
                         ;----------------------------------------
                         ; CHARACTER GENERATOR GRAPHICS FOR 320X200 AND 640X200 GRAPHICS
                         ;----------------------------------------
FA6E                     CRT_CHAR_GEN    LABEL   BYTE
FA6E 0000000000000000            DB      000H,000H,000H,000H,000H,000H,000H,000H ; D_00
FA76 7E81A581BD99817E            DB      07EH,081H,0A5H,081H,0BDH,099H,081H,07EH ; D_01
FA7E 7EFFDBFFC3E7FF7E            DB      07EH,0FFH,0DBH,0FFH,0C3H,0E7H,0FFH,07EH ; D_02
FA86 6CFEFEFE7C381000            DB      06CH,0FEH,0FEH,0FEH,07CH,038H,010H,000H ; D_03
FA8E 10387CFE7C381008            DB      010H,038H,07CH,0FEH,07CH,038H,010H,008H ; D_04
FA96 387C38FEFE7C387C            DB      038H,07CH,038H,0FEH,0FEH,07CH,038H,07CH ; D_05
FA9E 1010387CFE7C387C            DB      010H,010H,038H,07CH,0FEH,07CH,038H,07CH ; D_06
FAA6 0000183C3C180000            DB      000H,000H,018H,03CH,03CH,018H,000H,000H ; D_07
FAAE FFFFE7C3C3E7FFFF            DB      0FFH,0FFH,0E7H,0C3H,0C3H,0E7H,0FFH,0FFH ; D_08
FAB6 003C664242663C00            DB      000H,03CH,066H,042H,042H,066H,03CH,000H ; D_09
FABE FFC399BDBD99C3FF            DB      0FFH,0C3H,099H,0BDH,0BDH,099H,0C3H,0FFH ; D_0A
FAC6 0F070F7DCCCCCC78            DB      00FH,007H,00FH,07DH,0CCH,0CCH,0CCH,078H ; D_0B
FACE 3C6666663C187E18            DB      03CH,066H,066H,066H,03CH,018H,07EH,018H ; D_0C
FAD6 3F333F303070F0E0            DB      03FH,033H,03FH,030H,030H,070H,0F0H,0E0H ; D_0D
FADE 7F637F636367E6C0            DB      07FH,063H,07FH,063H,063H,067H,0E6H,0C0H ; D_0E
FAE6 995A3CE7E73C5A99            DB      099H,05AH,03CH,0E7H,0E7H,03CH,05AH,099H ; D_0F

FAEE 80E0F8FEF8E08000            DB      080H,0E0H,0F8H,0FEH,0F8H,0E0H,080H,000H ; D_10
FAF6 020E3EFE3E0E0200            DB      002H,00EH,03EH,0FEH,03EH,00EH,002H,000H ; D_11
FAFE 183C7E18187E3C18            DB      018H,03CH,07EH,018H,018H,07EH,03CH,018H ; D_12
FB06 6666666666006600            DB      066H,066H,066H,066H,066H,000H,066H,000H ; D_13
FB0E 7FDBDB7B1B1B1B00            DB      07FH,0DBH,0DBH,07BH,01BH,01BH,01BH,000H ; D_14
FB16 3E63386C6C38CC78            DB      03EH,063H,038H,06CH,06CH,038H,0CCH,078H ; D_15
FB1E 000000007E7E7E00            DB      000H,000H,000H,000H,07EH,07EH,07EH,000H ; D_16
FB26 183C7E187E3C18FF            DB      018H,03CH,07EH,018H,07EH,03CH,018H,0FFH ; D_17
FB2E 183C7E1818181800            DB      018H,03CH,07EH,018H,018H,018H,018H,000H ; D_18
FB36 181818187E3C1800            DB      018H,018H,018H,018H,07EH,03CH,018H,000H ; D_19
FB3E 00180CFE0C180000            DB      000H,018H,00CH,0FEH,00CH,018H,000H,000H ; D_1A
FB46 003060FE60300000            DB      000H,030H,060H,0FEH,060H,030H,000H,000H ; D_1B
FB4E 0000C0C0C0FE0000            DB      000H,000H,0C0H,0C0H,0C0H,0FEH,000H,000H ; D_1C
FB56 002466FF66240000            DB      000H,024H,066H,0FFH,066H,024H,000H,000H ; D_1D
FB5E 00183C7EFFFF0000            DB      000H,018H,03CH,07EH,0FFH,0FFH,000H,000H ; D_1E
FB66 00FFFF7E3C180000            DB      000H,0FFH,0FFH,07EH,03CH,018H,000H,000H ; D_1F

FB6E 0000000000000000            DB      000H,000H,000H,000H,000H,000H,000H,000H ; SP D_20
FB76 3078783030003000            DB      030H,078H,078H,030H,030H,000H,030H,000H ; ! D_21
FB7E 6C6C6C0000000000            DB      06CH,06CH,06CH,000H,000H,000H,000H,000H ; " D_22
FB86 6C6CFE6CFE6C6C00            DB      06CH,06CH,0FEH,06CH,0FEH,06CH,06CH,000H ; # D_23
FB8E 307CC0780CF83000            DB      030H,07CH,0C0H,078H,00CH,0F8H,030H,000H ; $ D_24
FB96 00C6CC183066C600            DB      000H,0C6H,0CCH,018H,030H,066H,0C6H,000H ; PER CENT D_25
FB9E 386C3876DCCC7600            DB      038H,06CH,038H,076H,0DCH,0CCH,076H,000H ; & D_26
FBA6 6060C00000000000            DB      060H,060H,0C0H,000H,000H,000H,000H,000H ; ' D_27
FBAE 1830606060301800            DB      018H,030H,060H,060H,060H,030H,018H,000H ; ( D_28
FBB6 6030181818306000            DB      060H,030H,018H,018H,018H,030H,060H,000H ; ) D_29
FBBE 00663CFF3C660000            DB      000H,066H,03CH,0FFH,03CH,066H,000H,000H ; * D_2A
FBC6 003030FC30300000            DB      000H,030H,030H,0FCH,030H,030H,000H,000H ; + D_2B
FBCE 0000000000303060            DB      000H,000H,000H,000H,000H,030H,030H,060H ; , D_2C
FBD6 000000FC00000000            DB      000H,000H,000H,0FCH,000H,000H,000H,000H ; - D_2D
FBDE 0000000000303000            DB      000H,000H,000H,000H,000H,030H,030H,000H ; . D_2E
FBE6 060C183060C08000            DB      006H,00CH,018H,030H,060H,0C0H,080H,000H ; / D_2F

FBEE 7CC6CEDEF6E67C00            DB      07CH,0C6H,0CEH,0DEH,0F6H,0E6H,07CH,000H ; 0 D_30
FBF6 307030303030FC00            DB      030H,070H,030H,030H,030H,030H,0FCH,000H ; 1 D_31
FBFE 78CC0C3860CCFC00            DB      078H,0CCH,00CH,038H,060H,0CCH,0FCH,000H ; 2 D_32
FC06 78CC0C380CCC7800            DB      078H,0CCH,00CH,038H,00CH,0CCH,078H,000H ; 3 D_33
FC0E 1C3C6CCCFE0C1E00            DB      01CH,03CH,06CH,0CCH,0FEH,00CH,01EH,000H ; 4 D_34
FC16 FCC0F80C0CCC7800            DB      0FCH,0C0H,0F8H,00CH,00CH,0CCH,078H,000H ; 5 D_35
FC1E 3860C0F8CCCC7800            DB      038H,060H,0C0H,0F8H,0CCH,0CCH,078H,000H ; 6 D_36
FC26 FCCC0C1830303000            DB      0FCH,0CCH,00CH,018H,030H,030H,030H,000H ; 7 D_37
FC2E 78CCCC78CCCC7800            DB      078H,0CCH,0CCH,078H,0CCH,0CCH,078H,000H ; 8 D_38
FC36 78CCCC7C0C187000            DB      078H,0CCH,0CCH,07CH,00CH,018H,070H,000H ; 9 D_39
FC3E 0030300000303000            DB      000H,030H,030H,000H,000H,030H,030H,000H ; : D_3A
```

```
LOC OBJ                  LINE    SOURCE

FC46 0030300000303060                     DB      000H,030H,030H,000H,000H,030H,030H,060H ; ; D_3B
FC4E 183060C060301800                     DB      018H,030H,060H,0C0H,060H,030H,018H,000H ; < D_3C
FC56 0000FC0000FC0000                     DB      000H,000H,0FCH,000H,000H,0FCH,000H,000H ; = D_3D
FC5E 6030180C18306000                     DB      060H,030H,018H,00CH,018H,030H,060H,000H ; > D_3E
FC66 78CC0C1830003000                     DB      078H,0CCH,00CH,018H,030H,000H,030H,000H ; ? D_3F

FC6E 7CC6DEDEDEC07800                     DB      07CH,0C6H,0DEH,0DEH,0DEH,0C0H,078H,000H ; @ D_40
FC76 3078CCCCFCCCCC00                     DB      030H,078H,0CCH,0CCH,0FCH,0CCH,0CCH,000H ; A D_41
FC7E FC66667C6666FC00                     DB      0FCH,066H,066H,07CH,066H,066H,0FCH,000H ; B D_42
FC86 3C66C0C0C0663C00                     DB      03CH,066H,0C0H,0C0H,0C0H,066H,03CH,000H ; C D_43
FC8E F86C6666666CF800                     DB      0F8H,06CH,066H,066H,066H,06CH,0F8H,000H ; D D_44
FC96 FE6268786862FE00                     DB      0FEH,062H,068H,078H,068H,062H,0FEH,000H ; E D_45
FC9E FE6268786860F000                     DB      0FEH,062H,068H,078H,068H,060H,0F0H,000H ; F D_46
FCA6 3C66C0C0CE663E00                     DB      03CH,066H,0C0H,0C0H,0CEH,066H,03EH,000H ; G D_47
FCAE CCCCCCFCCCCCCC00                     DB      0CCH,0CCH,0CCH,0FCH,0CCH,0CCH,0CCH,000H ; H D_48
FCB6 7830303030307800                     DB      078H,030H,030H,030H,030H,030H,078H,000H ; I D_49
FCBE 1E0C0C0CCCCC7800                     DB      01EH,00CH,00CH,00CH,0CCH,0CCH,078H,000H ; J D_4A
FCC6 E6666C786C66E600                     DB      0E6H,066H,06CH,078H,06CH,066H,0E6H,000H ; K D_4B
FCCE F06060606266FE00                     DB      0F0H,060H,060H,060H,062H,066H,0FEH,000H ; L D_4C
FCD6 C6EEFEFED6C6C600                     DB      0C6H,0EEH,0FEH,0FEH,0D6H,0C6H,0C6H,000H ; M D_4D
FCDE C6E6F6DECEC6C600                     DB      0C6H,0E6H,0F6H,0DEH,0CEH,0C6H,0C6H,000H ; N D_4E
FCE6 386CC6C6C66C3800                     DB      038H,06CH,0C6H,0C6H,0C6H,06CH,038H,000H ; O D_4F

FCEE FC66667C6060F000                     DB      0FCH,066H,066H,07CH,060H,060H,0F0H,000H ; P D_50
FCF6 78CCCCCCDC781C00                     DB      078H,0CCH,0CCH,0CCH,0DCH,078H,01CH,000H ; Q D_51
FCFE FC66667C6C66E600                     DB      0FCH,066H,066H,07CH,06CH,066H,0E6H,000H ; R D_52
FD06 78CCE0701CCC7800                     DB      078H,0CCH,0E0H,070H,01CH,0CCH,078H,000H ; S D_53
FD0E FCB4303030307800                     DB      0FCH,0B4H,030H,030H,030H,030H,078H,000H ; T D_54
FD16 CCCCCCCCCCCCFC00                     DB      0CCH,0CCH,0CCH,0CCH,0CCH,0CCH,0FCH,000H ; U D_55
FD1E CCCCCCCCCC783000                     DB      0CCH,0CCH,0CCH,0CCH,0CCH,078H,030H,000H ; V D_56
FD26 C6C6C6D6FEEEC600                     DB      0C6H,0C6H,0C6H,0D6H,0FEH,0EEH,0C6H,000H ; W D_57
FD2E C6C66C38386CC600                     DB      0C6H,0C6H,06CH,038H,038H,06CH,0C6H,000H ; X D_58
FD36 CCCCCC7830307800                     DB      0CCH,0CCH,0CCH,078H,030H,030H,078H,000H ; Y D_59
FD3E FEC68C183266FE00                     DB      0FEH,0C6H,08CH,018H,032H,066H,0FEH,000H ; Z D_5A
FD46 7860606060607800                     DB      078H,060H,060H,060H,060H,060H,078H,000H ; [ D_5B
FD4E C06030180C060200                     DB      0C0H,060H,030H,018H,00CH,006H,002H,000H ; BACKSLASH D_5C
FD56 7818181818187800                     DB      078H,018H,018H,018H,018H,018H,078H,000H ; ] D_5D
FD5E 10386CC600000000                     DB      010H,038H,06CH,0C6H,000H,000H,000H,000H ; CIRCUMFLEX D_5E
FD66 00000000000000FF                     DB      000H,000H,000H,000H,000H,000H,000H,0FFH ; _ D_5F

FD6E 3030180000000000                     DB      030H,030H,018H,000H,000H,000H,000H,000H ;   D_60
FD76 0000780C7CCC7600                     DB      000H,000H,078H,00CH,07CH,0CCH,076H,000H ; LOWER CASE A D_61
FD7E E060607C6666DC00                     DB      0E0H,060H,060H,07CH,066H,066H,0DCH,000H ; L.C. B D_62
FD86 000078CCC0CC7800                     DB      000H,000H,078H,0CCH,0C0H,0CCH,078H,000H ; L.C. C D_63
FD8E 1C0C0C7CCCCC7600                     DB      01CH,00CH,00CH,07CH,0CCH,0CCH,076H,000H ; L.C. D D_64
FD96 000078CCFCC07800                     DB      000H,000H,078H,0CCH,0FCH,0C0H,078H,000H ; L.C. E D_65
FD9E 386C60F06060F000                     DB      038H,06CH,060H,0F0H,060H,060H,0F0H,000H ; L.C. F D_66
FDA6 000076CCCC7C0CF8                     DB      000H,000H,076H,0CCH,0CCH,07CH,00CH,0F8H ; L.C. G D_67
FDAE E0606C766666E600                     DB      0E0H,060H,06CH,076H,066H,066H,0E6H,000H ; L.C. H D_68
FDB6 3000703030307800                     DB      030H,000H,070H,030H,030H,030H,078H,000H ; L.C. I D_69
FDBE 0C000C0C0CCCCC78                     DB      00CH,000H,00CH,00CH,00CH,0CCH,0CCH,078H ; L.C. J D_6A
FDC6 E060666C786CE600                     DB      0E0H,060H,066H,06CH,078H,06CH,0E6H,000H ; L.C. K D_6B
FDCE 7030303030307800                     DB      070H,030H,030H,030H,030H,030H,078H,000H ; L.C. L D_6C
FDD6 0000CCFEFED6C600                     DB      000H,000H,0CCH,0FEH,0FEH,0D6H,0C6H,000H ; L.C. M D_6D
FDDE 0000F8CCCCCCCC00                     DB      000H,000H,0F8H,0CCH,0CCH,0CCH,0CCH,000H ; L.C. N D_6E
FDE6 000078CCCCCC7800                     DB      000H,000H,078H,0CCH,0CCH,0CCH,078H,000H ; L.C. O D_6F

FDEE 0000DC66667C60F0                     DB      000H,000H,0DCH,066H,066H,07CH,060H,0F0H ; L.C. P D_70
FDF6 000076CCCC7C0C1E                     DB      000H,000H,076H,0CCH,0CCH,07CH,00CH,01EH ; L.C. Q D_71
FDFE 0000DC766660F000                     DB      000H,000H,0DCH,076H,066H,060H,0F0H,000H ; L.C. R D_72
FE06 00007CC0780CF800                     DB      000H,000H,07CH,0C0H,078H,00CH,0F8H,000H ; L.C. S D_73
FE0E 10307C3030341800                     DB      010H,030H,07CH,030H,030H,034H,018H,000H ; L.C. T D_74
FE16 0000CCCCCCCC7600                     DB      000H,000H,0CCH,0CCH,0CCH,0CCH,076H,000H ; L.C. U D_75
FE1E 0000CCCCCC783000                     DB      000H,000H,0CCH,0CCH,0CCH,078H,030H,000H ; L.C. V D_76
FE26 0000C6D6FEFE6C00                     DB      000H,000H,0C6H,0D6H,0FEH,0FEH,06CH,000H ; L.C. W D_77
FE2E 0000C66C386CC600                     DB      000H,000H,0C6H,06CH,038H,06CH,0C6H,000H ; L.C. X D_78
FE36 0000CCCCCC7C0CF8                     DB      000H,000H,0CCH,0CCH,0CCH,07CH,00CH,0F8H ; L.C. Y D_79
FE3E 0000FC983064FC00                     DB      000H,000H,0FCH,098H,030H,064H,0FCH,000H ; L.C. Z D_7A
FE46 1C3030E030301C00                     DB      01CH,030H,030H,0E0H,030H,030H,01CH,000H ;   D_7B
FE4E 1818180018181800                     DB      018H,018H,018H,000H,018H,018H,018H,000H ;   D_7C
FE56 E030301C3030E000                     DB      0E0H,030H,030H,01CH,030H,030H,0E0H,000H ;   D_7D
FE5E 76DC000000000000                     DB      076H,0DCH,000H,000H,000H,000H,000H,000H ;   D_7E
FE66 0010386CC6C6FE00                     DB      000H,010H,038H,06CH,0C6H,0C6H,0FEH,000H ; DELTA D_7F
```

```
LOC OBJ                    SOURCE

                           ;--- INT 1A -------------------------------
                           ; TIME_OF_DAY
                           ;  THIS ROUTINE ALLOWS THE CLOCK TO BE SET/READ
                           ;
                           ; INPUT
                           ;   (AH) = 0    READ THE CURRENT CLOCK SETTING
                           ;               RETURNS CX = HIGH PORTION OF COUNT
                           ;                       DX = LOW PORTION OF COUNT
                           ;                       AL = 0 IF TIMER HAS NOT PASSED 24 HOURS SINCE LAST READ
                           ;                          <>0 IF ON ANOTHER DAY
                           ;   (AH) = 1    SET THE CURRENT CLOCK
                           ;       CX = HIGH PORTION OF COUNT
                           ;       DX = LOW PORTION OF COUNT
                           ; NOTE: COUNTS OCCUR AT THE RATE OF 1193180/65536 COUNTS/SEC
                           ;       (OR ABOUT 18.2 PER SECOND -- SEE EQUATES BELOW)
                           ;--------------------------------------------
                                   ASSUME  CS:CODE,DS:DATA
FE6E                       TIME_OF_DAY     PROC    FAR
FE6E FB                            STI                     ; INTERRUPTS BACK ON
FE6F 1E                            PUSH    DS              ; SAVE SEGMENT
FE70 50                            PUSH    AX              ; SAVE PARM
FE71 B84000          R             MOV     AX,DATA
FE74 8ED8                          MOV     DS,AX           ; ESTABLISH ADDRESSING TO VALUES
FE76 58                            POP     AX              ; GET BACK INPUT PARM
FE77 0AE4                          OR      AH,AH           ; AH=0
FE79 7407                          JZ      T2              ; READ_TIME
FE7B FECC                          DEC     AH              ; AH=1
FE7D 7416                          JZ      T3              ; SET_TIME
FE7F                       T1:                             ; TOD_RETURN
FE7F FB                            STI                     ; INTERRUPTS BACK ON
FE80 1F                            POP     DS              ; RECOVER SEGMENT
FE81 CF                            IRET                    ; RETURN TO CALLER

FE82                       T2:                             ; READ_TIME
FE82 FA                            CLI                     ; NO TIMER INTERRUPTS WHILE READING
FE83 A07000          R             MOV     AL,TIMER_OFL
FE86 C606700000      R             MOV     TIMER_OFL,0     ; GET OVERFLOW, AND RESET THE FLAG
FE8B 8B0E6E00        R             MOV     CX,TIMER_HIGH
FE8F 8B166C00        R             MOV     DX,TIMER_LOW
FE93 EBEA                          JMP     T1                      ; TOD_RETURN

FE95                       T3:                             ; SET_TIME
FE95 FA                            CLI                     ; NO INTERRUPTS WHILE WRITING
FE96 89166C00        R             MOV     TIMER_LOW,DX
FE9A 890E6E00        R             MOV     TIMER_HIGH,CX   ; SET THE TIME
FE9E C606700000      R             MOV     TIMER_OFL,0     ; RESET OVERFLOW
FEA3 EBDA                          JMP     T1              ; TOD_RETURN
                           TIME_OF_DAY     ENDP
                           ;--------------------------------------------
                           ; THIS ROUTINE HANDLES THE TIMER INTERRUPT FROM
                           ; CHANNEL 0 OF THE 8253 TIMER.  INPUT FREQUENCY IS 1.19318 MHZ
                           ; AND THE DIVISOR IS 65536, RESULTING IN APPROX. 18.2 INTERRUPTS
                           ; EVERY SECOND.
                           ;
                           ; THE INTERRUPT HANDLER MAINTAINS A COUNT OF INTERRUPTS SINCE POWER
                           ;  ON TIME, WHICH MAY BE USED TO ESTABLISH TIME OF DAY.
                           ; THE INTERRUPT HANDLER ALSO DECREMENTS THE MOTOR CONTROL COUNT
                           ;  OF THE DISKETTE, AND WHEN IT EXPIRES, WILL TURN OFF THE DISKETTE
                           ;  MOTOR, AND RESET THE MOTOR RUNNING FLAGS
                           ; THE INTERRUPT HANDLER WILL ALSO INVOKE A USER ROUTINE THROUGH INTERRUPT
                           ;  1CH AT EVERY TIME TICK.  THE USER MUST CODE A ROUTINE AND PLACE THE
                           ;  CORRECT ADDRESS IN THE VECTOR TABLE.
                           ;--------------------------------------------
FEA5                       TIMER_INT       PROC    FAR
FEA5 FB                            STI                     ; INTERRUPTS BACK ON
FEA6 1E                            PUSH    DS
FEA7 50                            PUSH    AX
FEA8 52                            PUSH    DX              ; SAVE MACHINE STATE
FEA9 B84000          R             MOV     AX,DATA
FEAC 8ED8                          MOV     DS,AX           ; ESTABLISH ADDRESSABILITY
FEAE FF066C00        R             INC     TIMER_LOW       ; INCREMENT TIME
FEB2 7504                          JNZ     T4              ; TEST_DAY
FEB4 FF066E00        R             INC     TIMER_HIGH      ; INCREMENT HIGH WORD OF TIME
FEB8                       T4:                             ; TEST_DAY
```

```
LOC OBJ                   SOURCE

FEB8 833E6E0018      R            CMP     TIMER_HIGH,018H ; TEST FOR COUNT EQUALLING 24 HOURS
FEBD 7519                         JNZ     T5              ; DISKETTE_CTL
FEBF 813E6C00B000    R            CMP     TIMER_LOW,0B0H
FEC5 7511                         JNZ     T5              ; DISKETTE_CTL
                          ;------ TIMER HAS GONE 24 HOURS
FEC7 C7066E000000    R            MOV     TIMER_HIGH,0
FECD C7066C000000    R            MOV     TIMER_LOW,0
FED3 C606700001      R            MOV     TIMER_OFL,1
                          ;------ TEST FOR DISKETTE TIME OUT
FED8                      T5:                             ; DISKETTE_CTL
FED8 FE0E4000        R            DEC     MOTOR_COUNT
FEDC 750B                         JNZ     T6              ; RETURN IF COUNT NOT OUT
FEDE 80263F00F0      R            AND     MOTOR_STATUS,0F0H       ; TURN OFF MOTOR RUNNING BITS
FEE3 B00C                         MOV     AL,0CH
FEE5 BAF203                       MOV     DX,03F2H                ; FDC CTL PORT
FEE8 EE                           OUT     DX,AL           ; TURN OFF THE MOTOR
FEE9                      T6:                             ; TIMER_RET:
FEE9 CD1C                         INT     1CH             ; TRANSFER CONTROL TO A USER ROUTINE
FEEB B020                         MOV     AL,EOI
FEED E620                         OUT     020H,AL         ; END OF INTERRUPT TO 8259
FEEF 5A                           POP     DX
FEF0 58                           POP     AX
FEF1 1F                           POP     DS              ; RESET MACHINE STATE
FEF2 CF                           IRET                    ; RETURN FROM INTERRUPT
                          TIMER_INT       ENDP
                          ;--------------------------------------------
                          ; THESE ARE THE VECTORS WHICH ARE MOVED INTO
                          ;  THE 8086 INTERRUPT AREA DURING POWER ON
                          ;--------------------------------------------
FEF3                      VECTOR_TABLE    LABEL   WORD            ; VECTOR TABLE FOR MOVE TO INTERRUPTS
FEF3 A5FE            R            DW      OFFSET TIMER_INT        ; INTERRUPT 8
FEF5 00F0            R            DW      CODE
FEF7 87E9            R            DW      OFFSET  KB_INT          ; INTERRUPT 9
FEF9 00F0            R            DW      CODE
FEFB 00000000                     DD      0                       ; INTERRUPT A
FEFF 00000000                     DD      0                       ; INTERRUPT B
FF03 00000000                     DD      0                       ; INTERRUPT C
FF07 00000000                     DD      0                       ; INTERRUPT D
FF0B 57EF            R            DW      OFFSET DISK_INT         ; INTERRUPT E
FF0D 00F0            R            DW      CODE
FF0F 00000000                     DD      0                       ; INTERRUPT F
FF13 65F0            R            DW      OFFSET VIDEO_IO         ; INTERRUPT 10H
FF15 00F0            R            DW      CODE
FF17 4DF8            R            DW      OFFSET EQUIPMENT        ; INTERRUPT 11H
FF19 00F0            R            DW      CODE
FF1B 41F8            R            DW      OFFSET MEMORY_SIZE_DETERMINE    ; INT 12H
FF1D 00F0            R            DW      CODE
FF1F 59EC            R            DW      OFFSET DISKETTE_IO      ; INTERRUPT 13H
FF21 00F0            R            DW      CODE
FF23 39E7            R            DW      OFFSET RS232_IO         ; INTERRUPT 14H
FF25 00F0            R            DW      CODE
FF27 59F8            R            DW      OFFSET CASSETTE_IO      ; INTERRUPT 15H
FF29 00F0            R            DW      CODE
FF2B 2EE8            R            DW      OFFSET KEYBOARD_IO      ; INTERRUPT 16H
FF2D 00F0            R            DW      CODE
FF2F D2EF            R            DW      OFFSET PRINTER_IO       ; INTERRUPT 17H
FF31 00F0            R            DW      CODE
```

```
LOC OBJ                  LINE    SOURCE

FF33 0000                              DW      00000H                  ; INTERRUPT 18H
FF35 00F6                              DW      0F600H                  ;  ROM BASIC ENTRY POINT

FF37 F2E6           R                  DW      OFFSET BOOT_STRAP       ; INTERRUPT 19H
FF39 00F0           R                  DW      CODE

FF3B 6EFE           R                  DW      TIME_OF_DAY             ; INTERRUPT 1AH -- TIME OF DAY
FF3D 00F0           R                  DW      CODE

FF3F 53FF           R                  DW      DUMMY_RETURN            ; INTERRUPT 1BH -- KEYBOARD BREAK ADDR
FF41 00F0           R                  DW      CODE

FF43 53FF           R                  DW      DUMMY_RETURN            ; INTERRUPT 1C -- TIMER BREAK ADDR
FF45 00F0           R                  DW      CODE

FF47 A4F0           R                  DW      VIDEO_PARMS             ; INTERRUPT 1D -- VIDEO PARAMETERS
FF49 00F0           R                  DW      CODE

FF4B C7EF           R                  DW      OFFSET DISK_BASE        ; INTERRUPT 1E -- DISK PARMS
FF4D 00F0           R                  DW      CODE

FF4F 00000000                          DD      0                       ; INTERRUPT 1F -- POINTER TO VIDEO EXT

FF53                           DUMMY_RETURN:
FF53 CF                                IRET                            ; DUMMY RETURN FOR BREAK FROM KEYBOARD
                               ;-- INT 5 ----------------------------------
                               ;       THIS LOGIC WILL BE INVOKED BY INTERRUPT 05H TO PRINT
                               ;       THE SCREEN. THE CURSOR POSITION AT THE TIME THIS ROUTINE
                               ;       IS INVOKED WILL BE SAVED AND RESTORED UPON COMPLETION. THE
                               ;       ROUTINE IS INTENDED TO RUN WITH INTERRUPTS ENABLED.
                               ;       IF A SUBSEQUENT 'PRINT SCREEN KEY IS DEPRESSED DURING THE
                               ;       TIME THIS ROUTINE IS PRINTING IT WILL BE IGNORED.
                               ;       ADDRESS 50:0 CONTAINS THE STATUS OF THE PRINT SCREEN:
                               ;
                               ;       50:0    =0      EITHER PRINT SCREEN HAS NOT BEEN CALLED
                               ;                       OR UPON RETURN FROM A CALL THIS INDICATES
                               ;                       A SUCCESSFUL OPERATION.
                               ;
                               ;               =1      PRINT SCREEN IS IN PROGRESS
                               ;
                               ;               =377    ERROR ENCOUNTERED DURING PRINTING
                               ;------------------------------------------
                                       ASSUME  CS:CODE,DS:XXDATA

FF54                           PRINT_SCREEN    PROC    FAR
FF54 FB                                STI                     ;MUST RUN WITH INTERRUPTS ENABLED
FF55 1E                                PUSH    DS              ;MUST USE 50:0 FOR DATA AREA STORAGE
FF56 50                                PUSH    AX
FF57 53                                PUSH    BX
FF58 51                                PUSH    CX              ;WILL USE THIS LATER FOR CURSOR LIMITS
FF59 52                                PUSH    DX              ;WILL HOLD CURRENT CURSOR POSITION
FF5A B85000                            MOV     AX,XXDATA       ;HEX 50
FF5D 8ED8                              MOV     DS,AX
FF5F 803E000001                        CMP     STATUS_BYTE,1   ;SEE IF PRINT ALREADY IN PROGRESS
FF64 745F                              JZ      EXIT            ; JUMP IF PRINT ALREADY IN PROGRESS
FF66 C606000001                        MOV     STATUS_BYTE,1   ;INDICATE PRINT NOW IN PROGRESS
FF6B B40F                              MOV     AH,15           ;WILL REQUEST THE CURRENT SCREEN MODE
FF6D CD10                              INT     10H             ;       [AL]=MODE
                                                               ;       [AH]=NUMBER COLUMNS/LINE
                                                               ;       [BH]=VISUAL PAGE
                                       ;***********************************
                                       ;       AT THIS POINT WE KNOW THE COLUMNS/LINE ARE IN
                                       ;       [AX] AND THE PAGE IF APPLICABLE IS IN [BH]. THE STACK
                                       ;       HAS DS,AX,BX,CX,DX PUSHED. [AL] HAS VIDEO MODE
                                       ;
                                       ;***********************************
FF6F 8ACC                              MOV     CL,AH           ;WILL MAKE USE OF [CX] REGISTER TO
FF71 B519                              MOV     CH,25           ;CONTROL ROW & COLUMNS
FF73 E85500                            CALL    CRLF            ;CARRIAGE RETURN LINE FEED ROUTINE
FF76 51                                PUSH    CX              ;SAVE SCREEN BOUNDS
FF77 B403                              MOV     AH,3            ;WILL NOW READ THE CURSOR.
FF79 CD10                              INT     10H             ;AND PRESERVE THE POSITION
FF7B 59                                POP     CX              ;RECALL SCREEN BOUNDS
FF7C 52                                PUSH    DX              ;RECALL [BH]=VISUAL PAGE
FF7D 33D2                              XOR     DX,DX           ;WILL SET CURSOR POSITION TO [0,0]
```

```
LOC OBJ                     SOURCE

                            ;*******************************************
                            ;       THE LOOP FROM PRI10 TO THE INSTRUCTION PRIOR TO PRI20
                            ;       IS THE LOOP TO READ EACH CURSOR POSITION FROM THE SCREEN
                            ;       AND PRINT.
                            ;*******************************************
FF7F B402                   PRI10:  MOV     AH,2            ;TO INDICATE CURSOR SET REQUEST
FF81 CD10                           INT     10H             ;NEW CURSOR POSITION ESTABLISHED
FF83 B408                           MOV     AH,8            ;TO INDICATE READ CHARACTER
FF85 CD10                           INT     10H             ;CHARACTER NOW IN [AL]
FF87 0AC0                           OR      AL,AL           ;SEE IF VALID CHAR
FF89 7502                           JNZ     PRI15           ;JUMP IF VALID CHAR
FF8B B020                           MOV     AL,' '          ;MAKE A BLANK
FF8D                        PRI15:
FF8D 52                             PUSH    DX              ;SAVE CURSOR POSITION
FF8E 33D2                           XOR     DX,DX           ;INDICATE PRINTER 1
FF90 32E4                           XOR     AH,AH           ;TO INDICATE PRINT CHAR IN [AL]
FF92 CD17                           INT     17H             ;PRINT THE CHARACTER
FF94 5A                             POP     DX              ;RECALL CURSOR POSITION
FF95 F6C425                         TEST    AH, 25H         ; TEST FOR PRINTER ERROR
FF98 7521                           JNZ     ERR10           ; JUMP IF ERROR DETECTED
FF9A FEC2                           INC     DL              ;ADVANCE TO NEXT COLUMN
FF9C 3ACA                           CMP     CL,DL           ;SEE IF AT END OF LINE
FF9E 75DF                           JNZ     PRI10           ;IF NOT PROCEED
FFA0 32D2                           XOR     DL,DL           ;BACK TO COLUMN 0
FFA2 8AE2                           MOV     AH,DL           ;[AH]=0
FFA4 52                             PUSH    DX              ;SAVE NEW CURSOR POSITION
FFA5 E82300                         CALL    CRLF            ; LINE FEED CARRIAGE RETURN
FFA8 5A                             POP     DX              ;RECALL CURSOR POSITION
FFA9 FEC6                           INC     DH              ;ADVANCE TO NEXT LINE
FFAB 3AEE                           CMP     CH,DH           ;FINISHED?
FFAD 75D0                           JNZ     PRI10           ;IF NOT CONTINUE
FFAF 5A                     PRI20:  POP     DX              ;RECALL CURSOR POSITION
FFB0 B402                           MOV     AH,2            ;TO INDICATE CURSOR SET REQUEST
FFB2 CD10                           INT     10H             ;CURSOR POSITION RESTORED
FFB4 C606000000                     MOV     STATUS_BYTE,0   ;INDICATE FINISHED
FFB9 EB0A                           JMP     SHORT EXIT      ;EXIT THE ROUTINE
FFBB 5A                     ERR10:  POP     DX              ;GET CURSOR POSITION
FFBC B402                           MOV     AH,2            ;TO REQUEST CURSOR SET
FFBE CD10                           INT     10H             ;CURSOR POSITION RESTORED
FFC0 C6060000FF             ERR20:  MOV     STATUS_BYTE,0FFH        ;INDICATE ERROR

FFC5 5A                     EXIT:   POP     DX              ;RESTORE ALL THE REGISTERS USED
FFC6 59                             POP     CX
FFC7 5B                             POP     BX
FFC8 58                             POP     AX
FFC9 1F                             POP     DS
FFCA CF                             IRET
                            PRINT_SCREEN    ENDP

                            ;------ CARRIAGE RETURN, LINE FEED SUBROUTINE

FFCB                        CRLF    PROC    NEAR
FFCB 33D2                           XOR     DX,DX           ;PRINTER 0
FFCD 32E4                           XOR     AH,AH           ;WILL NOW SEND INITIAL LF,CR TO PRINTER
FFCF B00A                           MOV     AL,12Q          ;LF
FFD1 CD17                           INT     17H             ;SEND THE LINE FEED
FFD3 32E4                           XOR     AH,AH           ;NOW FOR THE CR
FFD5 B00D                           MOV     AL,15Q          ;CR
FFD7 CD17                           INT     17H             ;SEND THE CARRIAGE RETURN
FFD9 C3                             RET
                            CRLF    ENDP
                            CODE    ENDS

                            ;---------------------------------
                            ; POWER ON RESET VECTOR
                            ;---------------------------------
FFFF                        VECTOR  SEGMENT AT 0FFFFH

                            ;------ POWER ON RESET

0000 EA5B0000F0       R             JMP     RESET

0005 30342F32342F38                 DB      '04/24/81'              ; RELEASE MARKER
     31
                            VECTOR  ENDS
                                    END
```

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.

NOTES

Appendix B. Assembly Instruction Set Reference

8088
REGISTER MODEL

AX:	AH	AL	ACCUMULATOR	GENERAL REGISTER FILE
BX:	BH	BL	BASE	
CX:	CH	CL	COUNT	
DX:	DH	DL	DATA	
	SP		STACK POINTER	
	BP		BASE POINTER	
	SI		SOURCE INDEX	
	DI		DESTINATION INDEX	
	IP		INSTRUCTION POINTER	
	FLAGSH	FLAGSL	STATUS FLAGS	
	CS		CODE SEGMENT	SEGMENT REGISTER FILE
	DS		DATA SEGMENT	
	SS		STACK SEGMENT	
	ES		EXTRA SEGMENT	

Instructions which reference the flag register file as a 16-bit object use the symbol FLAGS to represent the file:

15								7							0
X	X	X	X	OF	DF	IF	TF	SF	ZF	X	AF	X	PF	X	CF

X = Don't Care

AF:	AUXILIARY CARRY – BCD	8080 FLAGS
CF:	CARRY FLAG	
PF:	PARITY FLAG	
SF:	SIGN FLAG	
ZF:	ZERO FLAG	
DF:	DIRECTION FLAG (STRINGS)	8088 FLAGS
IF:	INTERRUPT ENABLE FLAG	
OF:	OVERFLOW FLAG (CF ⊕ SF)	
TF:	TRAP – SINGLE STEP FLAG	

OPERAND SUMMARY

"reg" field Bit Assignments:

16-Bit (w=1)	8-Bit (w=0)	Segment
000 AX	000 AL	00 ES
001 CX	001 CL	01 CS
010 DX	010 DL	10 SS
011 BX	011 BL	11 DS
100 SP	100 AH	
101 BP	101 CH	
110 SI	110 DH	
111 DI	111 BH	

APPENDIX B

SECOND INSTRUCTION BYTE SUMMARY

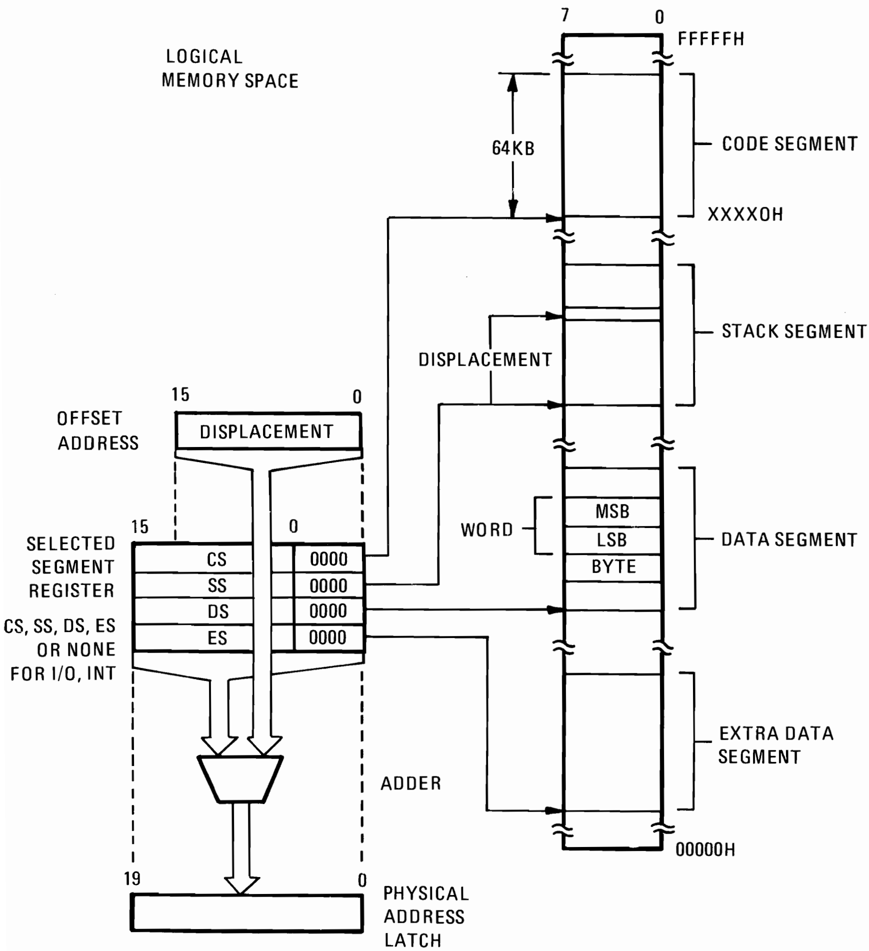
mod	xxx	r/m

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

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

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

MEMORY SEGMENTATION MODEL



SEGMENT OVERRIDE PREFIX

0	0	1	reg	1	1	0

USE OF SEGMENT OVERRIDE

OPERAND REGISTER	DEFAULT	WITH OVERRIDE PREFIX
IP (code address)	CS	Never
SP (stack address)	SS	Never
BP (stack address or stack marker)	SS	BP + DS or ES, or CS
SI or DI (not incl. strings)	DS	ES, SS, or CS
SI (implicit source addr for strings)	DS	ES, SS, or CS
DI (implicit dest addr for strings)	ES	Never

DATA TRANSFER

MOV = Move

Register/memory to/ from register

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

Immediate to register/memory

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

Immediate to register

1 0 1 1 w reg	data	data if w=1

Memory to accumulator

1 0 1 0 0 0 0 w	addr-low	addr-high

Accumulator to memory

1 0 1 0 0 0 1 w	addr-low	addr-high

Register/memory to segment register

1 0 0 0 1 1 1 0	mod 0 reg r/m

Segment register to register/memory

1 0 0 0 1 1 0 0	mod 0 reg r/m

PUSH = Push

Register/memory

1 1 1 1 1 1 1 1	mod 1 1 0 r/m

Register

0 1 0 1 0 reg

Segment register

0 0 0 reg 1 1 0

POP = Pop

Register/memory

1 0 0 0 1 1 1 1	mod 0 0 0 r/m

Register

0 1 0 1 1 reg

Segment register

0 0 0 reg 1 1 1

XCHG = Exchange
Register/memory with register

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

Register with accumulator

1	0	0	1	0	reg

IN = Input to AL/AX from
Fixed port

1	1	1	0	0	1	0	w	port

Variable port (DX)

1	1	1	0	1	1	0	w

OUT = Output from AL/AX to
Fixed port

1	1	1	0	0	1	1	w	port

Variable port (DX)

1	1	1	0	1	1	1	w

XLAT = Translate byte to AL

1	1	0	1	0	1	1	1

LEA = Load EA to register

1	0	0	0	1	1	0	1	mod reg r/m

LDS = Load pointer to DS

1	1	0	0	0	1	0	1	mod reg r/m

LES = Load pointer to ES

1	1	0	0	0	1	0	0	mod reg r/m

LAHF = Load AH with flags

1	0	0	1	1	1	1	1

SAHF = Store AH into flags

1	0	0	1	1	1	1	0

PUSHF = Push flags

1	0	0	1	1	1	0	0

POPF = Pop flags

1	0	0	1	1	1	0	1

ARITHMETIC

ADD = Add

Reg./memory with register to either

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

Immediate to register/memory

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

Immediate to accumulator

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

ADC = Add with carry

Reg./memory with register to either

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

Immediate to register/memory

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

Immediate to accumulator

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

INC = Increment

Register/memory

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

Register

0 1 0 0 0 reg

AAA = ASCII adjust for add

0 0 1 1 0 1 1 1

DAA = Decimal adjust for add

0 0 1 0 0 1 1 1

SUB = Subtract

Reg./memory and register to either

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

Immediate from register/memory

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

Immediate from accumulator

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

SBB = Subtract with borrow

Reg./memory and register to either

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

Immediate from register/memory

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

Immediate from accumulator

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

DEC = Decrement

Register/memory

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

Register

0 1 0 0 1 reg

NEG = Change sign

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

CMP = Compare

Register/memory and register

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

Immediate with register/memory

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

Immediate with accumulator

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

AAS = ASCII adjust for subtract

0 0 1 1 1 1 1 1

DAS = Decimal adjust for subtract

0 0 1 0 1 1 1 1

MUL = Multiply (unsigned)

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

IMUL = Integer multiply (signed)

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

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 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

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

1 1 1 1 1 1 1 1	mod 1 0 0 r/m

Direct intersegment

1 1 1 0 1 0 1 0	offset-low	offset-high
	seg-low	seg-high

Indirect intersegment

1 1 1 1 1 1 1 1	mod 1 0 1 r/m

RET = Return from CALL

Within segment

1 1 0 0 0 0 1 1

Within seg. adding immed to SP

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

Intersegment

1 1 0 0 1 0 1 1

Intersegment, adding immediate to SP

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

JE/JZ = Jump on equal/zero

0 1 1 1 0 1 0 0	disp

JL/JNGE = Jump on less/not greater or equal

0 1 1 1 1 1 0 0	disp

JLE/JNG = Jump on less or equal/not greater

0 1 1 1 1 1 1 0	disp

JB/JNAE = Jump on below/not above or equal

0 1 1 1 0 0 1 0	disp

JBE/JNA = Jump on below or equal/not above

0 1 1 1 0 1 1 0	disp

JP/JPE = Jump on parity/parity even

0 1 1 1 1 0 1 0	disp

JO = Jump on overflow

0 1 1 1 0 0 0 0	disp

JS = Jump on sign

0 1 1 1 1 0 0 0	disp

JNE/JNZ = Jump on not equal/not zero

0 1 1 1 0 1 0 1	disp

JNL/JGE = Jump on not less/greater or equal

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

Instruction	Condition	Interpretation
JE or JZ	ZF = 1	"equal" or "zero"
JL or JNGE	(SF xor OF) = 1	"less" or "not greater or equal"
JLE or JNG	((SP xor OF) or ZF) = 1	"less or equal" or "not greater"
JB or JNAE	CF = 1	"below" or "not above or equal"
JBE or JNA	(CF or ZF) = 1	"below or equal" or "not above"
JP or JPE	PF = 1	"parity" or "parity even"
JO	OF = 1	"overflow"
JS	SF = 1	"sign"
JNE or JNZ	ZF = 0	"not equal" or "not zero"
JNL or JGE	(SF xor OF) = 0	"not less" or "greater or equal"
JNLE or JG	((SF xor OF) or ZF) = 0	"not less or equal" or "greater"
JNB or JAE	CF = 0	"not below" or "above or equal"
JNBE or JA	(CF or ZF) = 0	"not below or equal" or "above"
JNP or JPO	PF = 0	"not parity" or "parity odd"
JNO	OF = 0	"not overflow"
JNS	SF = 0	"not sign"

*"Above" and "below" refer to the relation between two unsigned values, while "greater" and "less" refer to the relation between two signed values.

INT = **Interrupt**

Type specified

1	1	0	0	1	1	0	1	type

Type 3

1	1	0	0	1	1	0	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

CLC = Clear carry

1	1	1	1	1	1	0	0	0

STC = Set carry

1	1	1	1	1	0	0	1

CMC = Complement carry

1	1	1	1	0	1	0	1

NOP = No operation

1	0	0	1	0	0	0	0

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	0	1	0

STI = Set interrupt

1	1	1	1	1	0	1	1

HLT = Halt

1	1	1	1	0	1	0	0

WAIT = Wait

1	0	0	1	1	0	1	1

LOCK = Bus lock prefix

1	1	1	1	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

8088 INSTRUCTION SET MATRIX

HI \ LO	0	1	2	3	4	5	6	7
0	ADD b.f.r/m	ADD w.f.r/m	ADD b,t,r/m	ADD w,t,r/m	ADD b.ia	Add w.ia	PUSH ES	POP ES
1	ADC b.f.r/m	ADC w.f.r/m	ADC b,t,r/m	ADC w,t,r/m	ADC b.i	ADC w.i	PUSH SS	POP SS
2	AND b.f.r/m	AND w.f.r/m	AND b,t,r/m	AND w,t,r/m	AND b.i	AND w.i	SEG =ES	DAA
3	XOR b.f.r/m	XOR w,f,r/m	XOR b,t,r/m	XOR w,t,r/m	XOR b.i	XOR w.i	SEG =SS	AAA
4	INC AX	INC CX	INC DX	INC BX	INC SP	INC BP	INC SI	INC DI
5	PUSH AX	PUSH CX	PUSH DX	PUSH BX	PUSH SP	PUSH BP	PUSH SI	PUSH DI
6								
7	JO	JNO	JB/ JNAE	JNB/ JAE	JE/ JZ	JNE/ JNZ	JBE/ JNA	JNBE/ JA
8	Immed b.r/m	Immed w.r/m	Immed b.r/m	Immed is.r/m	TEST b.r/m	TEST w.r/m	XCHG b.r/m	XCHG w.r/m
9	NOP	XCHG CX	XCHG DX	XCHG BX	XCHG SP	XCHG BP	XCHG SI	XCHG DI
A	MOV m→AL	MOV m→AX	MOV AL→m	MOV AX→m	MOVS b	MOVS w	CMPS b	CMPS w
B	MOV i→AL	MOV i→CL	MOV i→DL	MOV i→BL	MOV i→AH	MOV i→CH	MOV i→DH	MOV i→BH
C			RET. (i+SP)	RET	LES	LDS	MOV b.i.r/m	MOV w.i.r/m
D	Shift b	Shift w	Shift b.v	Shift w.v	AAM	AAD		XLAT
E	LOOPNZ/ LOOPNE	LOOPZ/ LOOPE	LOOP	JCXZ	IN b	IN w	OUT b	OUT w
F	LOCK		REP	REP Z	HLT	CMC	Grp 1 b.r/m	Grp 1 w.r/m

b = byte operation
d = direct
f = from CPU reg
i = immediate
ia = immed. to accum.
id = indirect
is = immed. byte, sign ext.
l = long ie. intersegment

m = memory
r/m = EA is second byte
si = short intrasegment
sr = segment register
t = to CPU reg
v = variable
w = word operation
z = zero

8088 INSTRUCTION SET MATRIX

HI \ LO	8	9	A	B	C	D	E	F
0	OR b.f.r/m	OR w.f.r/m	OR b,t,r/m	OR w,t,r/m	OR b.i	OR w.i	PUSH CS	
1	SBB b.f.r/m	SBB w.f.r/m	SBB b,t,r/m	SBB w,t,r/m	SBB b.i	SBB w.i	PUSH DS	POP DS
2	SUB b.f.r/m	SUB w.f.r/m	SUB b,t,r/m	SUB w,t,r/m	SUB b.i	SUB w,i	SEG CS	DAS
3	CMP b.f.r/m	CMP w.f.r/m	CMP b,t,r/m	CMP w,t,r/m	CMP b.i	CMP w.i	SEG DS	AAS
4	DEC AX	DEC CX	DEC DX	DEC BX	DEC SP	DEC BP	DEC SI	DEC DI
5	POP AX	POP CX	POP DX	POP BX	POP SP	POP BP	POP SI	POP DI
6								
7	JS	JNS	JP/ JPE	JNP/ JPO	JL/ JNGE	JNL/ JGE	JLE/ JNG	JNLE/ JG
8	MOV b.f.r/m	MOV w.f.r/m	MOV b,t,r/m	MOV w,t,r/m	MOV sr.t.r/m	LEA	MOV sr.f.r/m	POP r/m
9	CBW	CWD	CALL l.d	WAIT	PUSHF	POPF	SAHF	LAHF
A	TEST b.i	TEST w.i	STOS b	STOS w	LODS b	LODS w	SCAS b	SCAS w
B	MOV i→AX	MOV i→CX	MOV i→DX	MOV i→BX	MOV i→SP	MOV i→BP	MOV i→SI	MOV i→DI
C			RET l,(i+SP)	RET l	INT Type 3	INT (Any)	INTO	IRET
D	ESC 0	ESC 1	ESC 2	ESC 3	ESC 4	ESC 5	ESC 6	ESC 7
E	CALL d	JMP d	JMP l.d	JMP si.d	IN v.b	IN v.w	OUT v.b	OUT v.w
F	CLC	STC	CLI	STI	CLD	STD	Grp 2 b.r/m	Grp 2 w.r/m

where

mod ☐ r/m	000	001	010	011	100	101	100	111
Immed	ADD	OR	ADC	SBB	AND	SUB	XOR	CMP
Shift	ROL	ROR	RCL	RCR	SHL/SAL	SHR	-	SAR
Grp 1	TEST	-	NOT	NEG	MUL	IMUL	DIV	IDIV
Grp 2	INC	DEC	CALL id	CALL l,id	JMP id	JMP l,id	PUSH	-

APPENDIX B

INSTRUCTION SET INDEX

Appendix C. Of Characters Keystrokes and Color

VALUE		AS CHARACTERS			AS TEXT ATTRIBUTES		
					COLOR/GRAPHICS MONITOR ADAPTER		IBM MONOCHROME DISPLAY ADAPTER
HEX	DEC	SYMBOL	KEYSTROKES	MODES	BACKGROUND	FOREGROUND	
00	0	BLANK (NULL)	CTRL 2		BLACK	BLACK	NON-DISPLAY
01	1	☺	CTRL A		BLACK	BLUE	UNDERLINE
02	2	☻	CTRL B		BLACK	GREEN	NORMAL
03	3	♥	CTRL C		BLACK	CYAN	NORMAL
04	4	♦	CTRL D		BLACK	RED	NORMAL
05	5	♣	CTRL E		BLACK	MAGENTA	NORMAL
06	6	♠	CTRL F		BLACK	BROWN	NORMAL
07	7	•	CTRL G		BLACK	LIGHT GREY	NORMAL
08	8	◘	CTRL H, BACKSPACE, SHIFT BACKSPACE		BLACK	DARK GREY	NON-DISPLAY
09	9	○	CTRL I		BLACK	LIGHT BLUE	HIGH INTENSITY UNDERLINE
0A	10	◙	CTRL J, CTRL ↵		BLACK	LIGHT GREEN	HIGH INTENSITY
0B	11	♂	CTRL K		BLACK	LIGHT GREEN	HIGH INTENSITY
0C	12	♀	CTRL L,		BLACK	LIGHT RED	HIGH INTENSITY
0D	13	♪	CTRL M, ↵ SHIFT ↵		BLACK	LIGHT MAGENTA	HIGH INTENSITY
0E	14	♫	CTRL N		BLACK	YELLOW	HIGH INTENSITY
0F	15	☼	CTRL O		BLACK	WHITE	HIGH INTENSITY
10	16	►	CTRL P		BLUE	BLACK	NORMAL
11	17	◄	CTRL Q		BLUE	BLUE	UNDERLINE
12	18	↕	CTRL R		BLUE	GREEN	NORMAL
13	19	‼	CTRL S		BLUE	CYAN	NORMAL
14	20	¶	CTRL T		BLUE	RED	NORMAL
15	21	§	CTRL U			MAGENTA	NORMAL
16	22	▬	CTRL V		BLUE	BROWN	NORMAL
17	23	↨	CTRL W		BLUE	LIGHT GREY	NORMAL

VALUE		AS CHARACTERS			AS TEXT ATTRIBUTES		
					COLOR/GRAPHICS MONITOR ADAPTER		IBM MONOCHROME DISPLAY ADAPTER
HEX	DEC	SYMBOL	KEYSTROKES	MODES	BACKGROUND	FOREGROUND	
18	24	↑	CTRL X		BLUE	DARK GREY	HIGH INTENSITY
19	25	↓	CTRL Y		BLUE	LIGHT BLUE	HIGH INTENSITY UNDERLINE
1A	26	→	CTRL Z		BLUE	LIGHT GREEN	HIGH INTENSITY
1B	27	←	CTRL [, ESC, SHIFT ESC, CTRL ESC		BLUE	LIGHT CYAN	HIGH INTENSITY
1C	28	∟	CTRL \		BLUE	LIGHT RED	HIGH INTENSITY
1D	29	↔	CTRL]		BLUE	LIGHT MAGENTA	HIGH INTENSITY
1E	30	▲	CTRL 6		BLUE	YELLOW	HIGH INTENSITY
1F	31	▼	CTRL –		BLUE	WHITE	HIGH INTENSITY
20	32	BLANK (SPACE	SPACE BAR, SHIFT SPACE, CTRL SPACE, ALT SPACE		GREEN	BLACK	NORMAL
21	33	!	!	SHIFT	GREEN	BLUE	UNDERLINE
22	34	"	"	SHIFT	GREEN	GREEN	NORMAL
23	35	#	#	SHIFT	GREEN	CYAN	NORMAL
24	36	$	$	SHIFT	GREEN	RED	NORMAL
25	37	%	%	SHIFT	GREEN	MAGENTA	NORMAL
26	38	&	&	SHIFT	GREEN	BROWN	NORMAL
27	39	'	'		GREEN	LIGHT GREY	NORMAL
28	40	(	(	SHIFT	GREEN	DARK GREY	HIGH INTENSITY
29	41	)	)	SHIFT	GREEN	LIGHT BLUE	HIGH INTENSITY UNDERLINE
2A	42	*	*	NOTE 1	GREEN	LIGHT GREEN	HIGH INTENSITY
2B	43	+	+	SHIFT	GREEN	LIGHT CYAN	HIGH INTENSITY
2C	44	,	,		GREEN	LIGHT RED	HIGH INTENSITY
2D	45	–	–		GREEN	LIGHT MAGENTA	HIGH INTENSITY
2E	46	.	.	NOTE 2	GREEN	YELLOW	HIGH INTENSITY

VALUE		AS CHARACTERS			AS TEXT ATTRIBUTES		
					COLOR/GRAPHICS MONITOR ADAPTER		IBM MONOCHROME DISPLAY ADAPTER
HEX	DEC	SYMBOL	KEYSTROKES	MODES	BACKGROUND	FOREGROUND	
2F	47	/	/		GREEN	WHITE	HIGH INTENSITY
30	48	0	0	NOTE 3	CYAN	BLACK	NORMAL
31	49	1	1	NOTE 3	CYAN	BLUE	UNDERLINE
32	50	2	2	NOTE 3	CYAN	GREEN	NORMAL
33	51	3	3	NOTE 3	CYAN	CYAN	NORMAL
34	52	4	4	NOTE 3	CYAN	RED	NORMAL
35	53	5	5	NOTE 3	CYAN	MAGENTA	NORMAL
36	54	6	6	NOTE 3	CYAN	BROWN	NORMAL
37	55	7	7	NOTE 3	CYAN	LIGHT GREY	NORMAL
38	56	8	8	NOTE 3	CYAN	DARK GREY	HIGH INTENSITY
39	57	9	9	NOTE 3	CYAN	LIGHT BLUE	HIGH INTENSITY UNDERLINE
3A	58	:	:	SHIFT	CYAN	LIGHT GREEN	HIGH INTENSITY
3B	59	;	;		CYAN	LIGHT CYAN	HIGH INTENSITY
3C	60	<	<	SHIFT	CYAN	LIGHT RED	HIGH INTENSITY
3D	61	=	=		CYAN	LIGHT MAGENTA	HIGH INTENSITY
3E	62	>	>	SHIFT	CYAN	YELLOW	HIGH INTENSITY
3F	63	?	?	SHIFT	CYAN	WHITE	HIGH INTENSITY
40	64	@	@	SHIFT	RED	BLACK	NORMAL
41	65	A	A	NOTE 4	RED	BLUE	UNDERLINE
42	66	B	B	NOTE 4	RED	GREEN	NORMAL
43	67	C	C	NOTE 4	RED	CYAN	NORMAL
44	68	D	D	NOTE 4	RED	RED	NORMAL
45	69	E	E	NOTE 4	RED	MAGENTA	NORMAL
46	70	F	F	NOTE 4	RED	BROWN	NORMAL
47	71	G	G	NOTE 4	RED	LIGHT GREY	NORMAL
48	72	H	H	NOTE 4	RED	DARK GREY	HIGH INTENSITY
49	73	I	I	NOTE 4	RED	LIGHT BLUE	HIGH INTENSITY UNDERLINE
4A	74	J	J	NOTE 4	RED	LIGHT GREEN	HIGH INTENSITY

VALUE		AS CHARACTERS			AS TEXT ATTRIBUTES		
					COLOR/GRAPHICS MONITOR ADAPTER		IBM MONOCHROME DISPLAY ADAPTER
HEX	DEC	SYMBOL	KEYSTROKES	MODES	BACKGROUND	FOREGROUND	
4B	75	K	K	NOTE 4	RED	LIGHT CYAN	HIGH INTENSITY
4C	76	L	L	NOTE 4	RED	LIGHT RED	HIGH INTENSITY
4D	77	M	M	NOTE 4	RED	LIGHT MAGENTA	HIGH INTENSITY
4E	78	N	N	NOTE 4	RED	YELLOW	HIGH INTENSITY
4F	79	O	O	NOTE 4	RED	WHITE	HIGH INTENSITY
50	80	P	P	NOTE 4	MAGENTA	BLACK	NORMAL
51	81	Q	Q	NOTE 4	MAGENTA	BLUE	UNDERLINE
52	82	R	R	NOTE 4	MAGENTA	GREEN	NORMAL
53	83	S	S	NOTE 4	MAGENTA	CYAN	NORMAL
54	84	T	T	NOTE 4	MAGENTA	RED	NORMAL
55	85	U	U	NOTE 4	MAGENTA	MAGENTA	NORMAL
56	86	V	V	NOTE 4	MAGENTA	BROWN	NORMAL
57	87	W	W	NOTE 4	MAGENTA	LIGHT GREY	NORMAL
58	88	X	X	NOTE 4	MAGENTA	DARK GREY	HIGH INTENSITY
59	89	Y	Y	NOTE 4	MAGENTA	LIGHT BLUE	HIGH INTENSITY UNDERLINE
5A	90	Z	Z	NOTE 4	MAGENTA	LIGHT GREEN	HIGH INTENSITY
5B	91	[	[		MAGENTA	LIGHT CYAN	HIGH INTENSITY
5C	92	\	\		MAGENTA	LIGHT RED	HIGH INTENSITY
5D	93	]	]		MAGENTA	LIGHT MAGENTA	HIGH INTENSITY
5E	94	∧	∧	SHIFT	MAGENTA	YELLOW	HIGH INTENSITY
5F	95	_	_	SHIFT	MAGENTA	WHITE	HIGH INTENSITY
60	96	'	'		YELLOW	BLACK	NORMAL
61	97	a	a	NOTE 5	YELLOW	BLUE	UNDERLINE
62	98	b	b	NOTE 5	YELLOW	GREEN	NORMAL
63	99	c	c	NOTE 5	YELLOW	CYAN	NORMAL
64	100	d	d	NOTE 5	YELLOW	RED	NORMAL
65	101	e	e	NOTE 5	YELLOW	MAGENTA	NORMAL
66	102	f	f	NOTE 5	YELLOW	BROWN	NORMAL

VALUE		AS CHARACTERS			AS TEXT ATTRIBUTES		
					COLOR/GRAPHICS MONITOR ADAPTER		IBM MONOCHROME DISPLAY ADAPTER
HEX	DEC	SYMBOL	KEYSTROKES	MODES	BACKGROUND	FOREGROUND	
67	103	g	g	NOTE 5	YELLOW	LIGHT GREY	NORMAL
68	104	h	h	NOTE 5	YELLOW	DARK GREY	HIGH INTENSITY
69	105	i	i	NOTE 5	YELLOW	LIGHT BLUE	HIGH INTENSITY UNDERLINE
6A	106	j	j	NOTE 5	YELLOW	LIGHT GREEN	HIGH INTENSITY
6B	107	k	k	NOTE 5	YELLOW	LIGHT CYAN	HIGH INTENSITY
6C	108	l	l	NOTE 5	YELLOW	LIGHT RED	HIGH INTENSITY
6D	109	m	m	NOTE 5	YELLOW	LIGHT MAGENTA	HIGH INTENSITY
6E	110	n	n	NOTE 5	YELLOW	YELLOW	HIGH INTENSITY
6F	111	o	o	NOTE 5	YELLOW	WHITE	HIGH INTENSITY
70	112	p	p	NOTE 5	WHITE	BLACK	REVERSE VIDEO
71	113	q	q	NOTE 5	WHITE	BLUE	UNDERLINE
72	114	r	r	NOTE 5	WHITE	GREEN	NORMAL
73	115	s	s	NOTE 5	WHITE	CYAN	NORMAL
74	116	t	t	NOTE 5	WHITE	RED	NORMAL
75	117	u	u	NOTE 5	WHITE	MAGENTA	NORMAL
76	118	v	v	NOTE 5	WHITE	BROWN	NORMAL
77	119	w	w	NOTE 5	WHITE	LIGHT GREY	NORMAL
78	120	x	x	NOTE 5	WHITE	DARK GREY	REVERSE VIDEO
79	121	y	y	NOTE 5	WHITE	LIGHT BLUE	HIGH INTENSITY UNDERLINE
7A	122	z	z	NOTE 5	WHITE	LIGHT GREEN	HIGH INTENSITY
7B	123	{	{	SHIFT	WHITE	LIGHT CYAN	HIGH INTENSITY
7C	124	¦	¦	SHIFT	WHITE	LIGHT RED	HIGH INTENSITY
7D	125	}	}	SHIFT	WHITE	LIGHT MAGENTA	HIGH INTENSITY
7E	126	~	~	SHIFT	WHITE	YELLOW	HIGH INTENSITY
7F	127	△	CTRL ←		WHITE	WHITE	HIGH INTENSITY

APPENDIX C

VALUE		AS CHARACTERS			AS TEXT ATTRIBUTES		
					COLOR/GRAPHICS MONITOR ADAPTER		IBM MONOCHROME DISPLAY ADAPTER
HEX	DEC	SYMBOL	KEYSTROKES	MODES	BACKGROUND	FOREGROUND	
* * * 80 HEX – FF HEX ARE FLASHING IN BOTH COLOR & IBM MONOCHROME * * * *							
80	128	Ç	ALT 128	NOTE 6	BLACK	BLACK	NON-DISPLAY
81	129	ü	ALT 129	NOTE 6	BLACK	BLUE	UNDERLINE
82	130	é	ALT 130	NOTE 6	BLACK	GREEN	NORMAL
83	131	â	ALT 131	NOTE 6	BLACK	CYAN	NORMAL
84	132	ä	ALT 132	NOTE 6	BLACK	RED	NORMAL
85	133	à	ALT 133	NOTE 6	BLACK	MAGENTA	NORMAL
86	134	å	ALT 134	NOTE 6	BLACK	BROWN	NORMAL
87	135	ç	ALT 135	NOTE 6	BLACK	LIGHT GREY	NORMAL
88	136	ê	ALT 136	NOTE 6	BLACK	DARK GREY	NON-DISPLAY
89	137	ë	ALT 137	NOTE 6	BLACK	LIGHT BLUE	HIGH INTENSITY UNDERLINED
8A	138	è	ALT 138	NOTE 6	BLACK	LIGHT GREEN	HIGH INTENSITY
8B	139	ï	ALT 139	NOTE 6	BLACK	LIGHT CYAN	HIGH INTENSITY
8C	140	î	ALT 140	NOTE 6	BLACK	LIGHT RED	HIGH INTENSITY
8D	141	ì	ALT 141	NOTE 6	BLACK	LIGHT MAGENTA	HIGH INTENSITY
8E	142	Ä	ALT 142	NOTE 6	BLACK	YELLOW	HIGH INTENSITY
8F	143	Å	ALT 143	NOTE 6	BLACK	WHITE	HIGH INTENSITY
90	144	É	ALT 144	NOTE 6	BLUE	BLACK	NORMAL
91	145	æ	ALT 145	NOTE 6	BLUE	BLUE	UNDERLINE
92	146	Æ	ALT 146	NOTE 6	BLUE	GREEN	NORMAL
93	147	ô	ALT 147	NOTE 6	BLUE	CYAN	NORMAL
94	148	ö	ALT 148	NOTE 6	BLUE	RED	NORMAL
95	149	ò	ALT 149	NOTE 6	BLUE	MAGENTA	NORMAL
96	150	û	ALT 150	NOTE 6	BLUE	BROWN	NORMAL
97	151	ù	ALT 151	NOTE 6	BLUE	LIGHT GREY	NORMAL
98	152	ÿ	ALT 152	NOTE 6	BLUE	DARK GREY	HIGH INTENSITY
99	153	Ö	ALT 153	NOTE 6	BLUE	LIGHT BLUE	HIGH INTENSITY UNDERLINE
9A	154	Ü	ALT 154	NOTE 6	BLUE	LIGHT GREEN	HIGH INTENSITY

VALUE		AS CHARACTERS			AS TEXT ATTRIBUTES		
					COLOR/GRAPHICS MONITOR ADAPTER		IBM MONOCHROME DISPLAY ADAPTER
HEX	DEC	SYMBOL	KEYSTROKES	MODES	BACKGROUND	FOREGROUND	
9B	155	¢	ALT 155	NOTE 6	BLUE	LIGHT CYAN	HIGH INTENSITY
9C	156	£	ALT 156	NOTE 6	BLUE	LIGHT RED	HIGH INTENSITY
9D	157	¥	ALT 157	NOTE 6	BLUE	LIGHT MAGENTA	HIGH INTENSITY
9E	158	Pts	ALT 158	NOTE 6	BLUE	YELLOW	HIGH INTENSITY
9F	159	ƒ	ALT 159	NOTE 6	BLUE	WHITE	HIGH INTENSITY
A0	160	á	ALT 160	NOTE 6	GREEN	BLACK	NORMAL
A1	161	í	ALT 161	NOTE 6	GREEN	BLUE	UNDERLINE
A2	162	ó	ALT 162	NOTE 6	GREEN	GREEN	NORMAL
A3	163	ú	ALT 163	NOTE 6	GREEN	CYAN	NORMAL
A4	164	ñ	ALT 164	NOTE 6	GREEN	RED	NORMAL
A5	165	Ñ	ALT 165	NOTE 6	GREEN	MAGENTA	NORMAL
A6	166	ª	ALT 166	NOTE 6	GREEN	BROWN	NORMAL
A7	167	º	ALT 167	NOTE 6	GREEN	LIGHT GREY	NORMAL
A8	168	¿	ALT 168	NOTE 6	GREEN	DARK GREY	HIGH INTENSITY
A9	169	⌐	ALT 169	NOTE 6	GREEN	LIGHT BLUE	HIGH INTENSITY UNDERLINE
AA	170	¬	ALT 170	NOTE 6	GREEN	LIGHT GREEN	HIGH INTENSITY
AB	171	½	ALT 171	NOTE 6	GREEN	LIGHT CYAN	HIGH INTENSITY
AC	172	¼	ALT 172	NOTE 6	GREEN	LIGHT RED	HIGH INTENSITY
AD	173	¡	ALT 173	NOTE 6	GREEN	LIGHT MAGENTA	HIGH INTENSITY
AE	174	«	ALT 174	NOTE 6	GREEN	YELLOW	HIGH INTENSITY
AF	175	»	ALT 175	NOTE 6	GREEN	WHITE	HIGH INTENSITY
B0	176	¼ DOTS ON	ALT 176	NOTE 6	CYAN	BLACK	NORMAL
B1	177	½ DOTS ON	ALT 177	NOTE 6	CYAN	BLUE	UNDERLINE
B2	178	¾ DOTS ON	ALT 178	NOTE 6	CYAN	GREEN	NORMAL
B3	179	│	ALT 179	NOTE 6	CYAN	CYAN	NORMAL
B4	180	┤	ALT 180	NOTE 6	CYAN	RED	NORMAL
B5	181	╡	ALT 181	NOTE 6	CYAN	MAGENTA	NORMAL
B6	182	╢	ALT 182	NOTE 6	CYAN	BROWN	NORMAL

VALUE		AS CHARACTERS			AS TEXT ATTRIBUTES		
					COLOR/GRAPHICS MONITOR ADAPTER		IBM MONOCHROME DISPLAY ADAPTER
HEX	DEC	SYMBOL	KEYSTROKES	MODES	BACKGROUND	FOREGROUND	
B7	183	╖	ALT 183	NOTE 6	CYAN	LIGHT GREY	NORMAL
B8	184	╕	ALT 184	NOTE 6	CYAN	DARK GREY	HIGH INTENSITY
B9	185	╣	ALT 185	NOTE 6	CYAN	LIGHT BLUE	HIGH INTENSITY UNDERLINE
BA	186	║	ALT 186	NOTE 6	CYAN	LIGHT GREEN	HIGH INTENSITY
BB	187	╗	ALT 187	NOTE 6	CYAN	LIGHT CYAN	HIGH INTENSITY
BC	188	╝	ALT 188	NOTE 6	CYAN	LIGHT RED	HIGH INTENSITY
BD	189	╜	ALT 189	NOTE 6	CYAN	LIGHT MAGENTA	HIGH INTENSITY
BE	190	╛	ALT 190	NOTE 6	CYAN	YELLOW	HIGH INTENSITY
BF	191	┐	ALT 191	NOTE 6	CYAN	WHITE	HIGH INTENSITY
C0	192	└	ALT 192	NOTE 6	RED	BLACK	NORMAL
C1	193	┴	ALT 193	NOTE 6	RED	BLUE	UNDERLINE
C2	194	┬	ALT 194	NOTE 6	RED	GREEN	NORMAL
C3	195	├	ALT 195	NOTE 6	RED	CYAN	NORMAL
C4	196	─	ALT 196	NOTE 6	RED	RED	NORMAL
C5	197	┼	ALT 197	NOTE 6	RED	MAGENTA	NORMAL
C6	198	╞	ALT 198	NOTE 6	RED	BROWN	NORMAL
C7	199	╟	ALT 199	NOTE 6	RED	LIGHT GREY	NORMAL
C8	200	╚	ALT 200	NOTE 6	RED	DARK GREY	HIGH INTENSITY
C9	201	╔	ALT 201	NOTE 6	RED	LIGHT BLUE	HIGH INTENSITY UNDERLINE
CA	202	╩	ALT 202	NOTE 6	RED	LIGHT GREEN	HIGH INTENSITY
CB	203	╦	ALT 203	NOTE 6	RED	LIGHT CYAN	HIGH INTENSITY
CC	204	╠	ALT 204	NOTE 6	RED	LIGHT RED	HIGH INTENSITY
CD	205	═	ALT 205	NOTE 6	RED	LIGHT MAGENTA	HIGH INTENSITY
CE	206	╬	ALT 206	NOTE 6	RED	YELLOW	HIGH INTENSITY
CF	207	╧	ALT 207	NOTE 6	RED	WHITE	HIGH INTENSITY
D0	208	╨	ALT 208	NOTE 6	MAGENTA	BLACK	NORMAL

VALUE		AS CHARACTERS			AS TEXT ATTRIBUTES		
					COLOR/GRAPHICS MONITOR ADAPTER		IBM MONOCHROME DISPLAY ADAPTER
HEX	DEC	SYMBOL	KEYSTROKES	MODES	BACKGROUND	FOREGROUND	
D1	209	╤	ALT 209	NOTE 6	MAGENTA	BLUE	UNDERLINE
D2	210	╥	ALT 210	NOTE 6	MAGENTA	GREEN	NORMAL
D3	211	╙	ALT 211	NOTE 6	MAGENTA	CYAN	NORMAL
D4	212	╘	ALT 212	NOTE 6	MAGENTA	RED	NORMAL
D5	213	╒	ALT 213	NOTE 6	MAGENTA	MAGENTA	NORMAL
D6	214	╓	ALT 214	NOTE 6	MAGENTA	BROWN	NORMAL
D7	215	╫	ALT 215	NOTE 6	MAGENTA	LIGHT GREY	NORMAL
D8	216	╪	ALT 216	NOTE 6	MAGENTA	DARK GREY	HIGH INTENSITY
D9	217	┘	ALT 217	NOTE 6	MAGENTA	LIGHT BLUE	HIGH INTENSITY UNDERLINE
DA	218	┌	ALT 218	NOTE 6	MAGENTA	LIGHT GREEN	HIGH INTENSITY
DB	219	█	ALT 219	NOTE 6	MAGENTA	LIGHT CYAN	HIGH INTENSITY
DC	220	▄	ALT 220	NOTE 6	MAGENTA	LIGHT RED	HIGH INTENSITY
DD	221	▌	ALT 221	NOTE 6	MAGENTA	LIGHT MAGENTA	HIGH INTENSITY
DE	222	▐	ALT 222	NOTE 6	MAGENTA	YELLOW	HIGH INTENSITY
DF	223	▀	ALT 223	NOTE 6	MAGENTA	WHITE	HIGH INTENSITY
E0	224	α	ALT 224	NOTE 6	YELLOW	BLACK	NORMAL
E1	225	β	ALT 225	NOTE 6	YELLOW	BLUE	UNDERLINE
E2	226	Γ	ALT 226	NOTE 6	YELLOW	GREEN	NORMAL
E3	227	π	ALT 227	NOTE 6	YELLOW	CYAN	NORMAL
E4	228	Σ	ALT 228	NOTE 6	YELLOW	RED	NORMAL
E5	229	σ	ALT 229	NOTE 6	YELLOW	MAGENTA	NORMAL
E6	230	μ	ALT 230	NOTE 6	YELLOW	BROWN	NORMAL
E7	231	τ	ALT 231	NOTE 6	YELLOW	LIGHT GREY	NORMAL
E8	232	Φ	ALT 232	NOTE 6	YELLOW	DARK GREY	HIGH INTENSITY
E9	233	θ	ALT 233	NOTE 6	YELLOW	LIGHT BLUE	HIGH INTENSITY UNDERLINE
EA	234	Ω	ALT 234	NOTE 6	YELLOW	LIGHT GREEN	HIGH INTENSITY
EB	235	δ	ALT 235	NOTE 6	YELLOW	LIGHT CYAN	HIGH INTENSITY

VALUE		AS CHARACTERS			AS TEXT ATTRIBUTES		
					COLOR/GRAPHICS MONITOR ADAPTER		IBM MONOCHROME DISPLAY ADAPTER
HEX	DEC	SYMBOL	KEYSTROKES	MODES	BACKGROUND	FOREGROUND	
EC	236	∞	ALT 236	NOTE 6	YELLOW	LIGHT RED	HIGH INTENSITY
ED	237	φ	ALT 237	NOTE 6	YELLOW	LIGHT MAGENTA	HIGH INTENSITY
EE	238	∈	ALT 238	NOTE 6	YELLOW	YELLOW	HIGH INTENSITY
EF	239	∩	ALT 239	NOTE 6	YELLOW	WHITE	HIGH INTENSITY
F0	240	≡	ALT 240	NOTE 6	WHITE	BLACK	REVERSE VIDEO
F1	241	±	ALT 241	NOTE 6	WHITE	BLUE	UNDERLINE
F2	242	≥	ALT 242	NOTE 6	WHITE	GREEN	NORMAL
F3	243	≤	ALT 243	NOTE 6	WHITE	CYAN	NORMAL
F4	244	⌠	ALT 244	NOTE 6	WHITE	RED	NORMAL
F5	245	⌡	ALT 245	NOTE 6	WHITE	MAGENTA	NORMAL
F6	246	÷	ALT 246	NOTE 6	WHITE	BROWN	NORMAL
F7	247	≈	ALT 247	NOTE 6	WHITE	LIGHT GREY	NORMAL
F8	248	°	ALT 248	NOTE 6	WHITE	DARK GREY	REVERSE VIDEO
F9	249	●	ALT 249	NOTE 6	WHITE	LIGHT BLUE	HIGH INTENSITY UNDERLINE
FA	250	•	ALT 250	NOTE 6	WHITE	LIGHT GREEN	HIGH INTENSITY
FB	251	√	ALT 251	NOTE 6	WHITE	LIGHT CYAN	HIGH INTENSITY
FC	252	^{n}	ALT 252	NOTE 6	WHITE	LIGHT RED	HIGH INTENSITY
FD	253	^{2}	ALT 253	NOTE 6	WHITE	LIGHT MAGENTA	HIGH INTENSITY
FE	254	■	ALT 254	NOTE 6	WHITE	YELLOW	HIGH INTENSITY
FF	255	BLANK	ALT 255	NOTE 6	WHITE	WHITE	HIGH INTENSITY

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

DECIMAL VALUE	➡	0	16	32	48	64	80	96	112
⬇	HEXA-DECIMAL VALUE	0	1	2	3	4	5	6	7
0	0	BLANK (NULL)	►	BLANK (SPACE)	0	@	P	‘	p
1	1	☺	◄	!	1	A	Q	a	q
2	2	☻	↕	"	2	B	R	b	r
3	3	♥	‼	#	3	C	S	c	s
4	4	♦	¶	$	4	D	T	d	t
5	5	♣	§	%	5	E	U	e	u
6	6	♠	▬	&	6	F	V	f	v
7	7	•	↨	'	7	G	W	g	w
8	8	◘	↑	(	8	H	X	h	x
9	9	○	↓	)	9	I	Y	i	y
10	A	◙	→	*	:	J	Z	j	x
11	B	♂	←	+	;	K	[	k	{
12	C	♀	∟	,	<	L	\	l	¦
13	D	♪	↔	—	=	M	]	m	}
14	E	♫	▲	.	>	N	^	n	~
15	F	☼	▼	/	?	O	_	o	△

Character Set (80-FF) Quick Reference

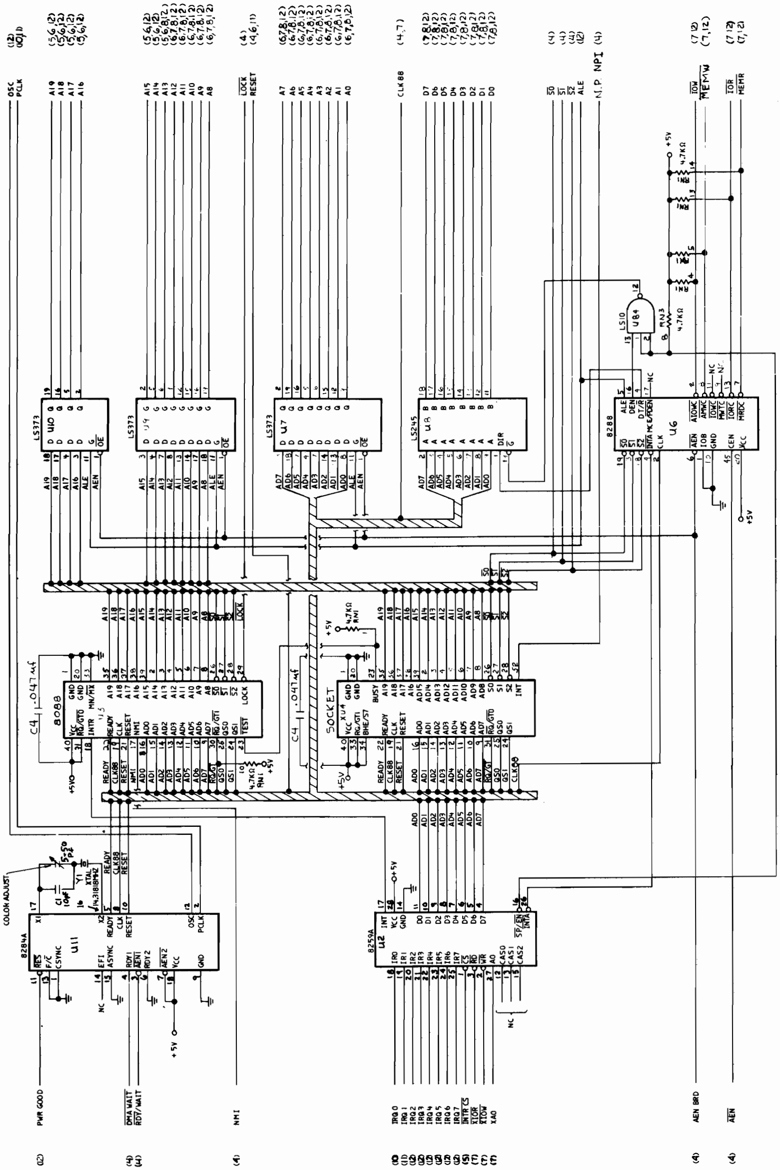
DECIMAL VALUE	➡	128	144	160	176	192	208	224	240
⬇	HEXA-DECIMAL VALUE	8	9	A	B	D	C	E	F
0	0	Ç	É	á	1/4 Dots On	└	╨	∝	≡
1	1	ü	Æ	í	1/2 Dots On	┴	╤	β	±
2	2	é	FE	ó	3/4 Dots On	┬	╥	γ	≥
3	3	â	ô	ú	│	├	╙	π	≤
4	4	ä	ö	ñ	┤	─	╘	Σ	⌠
5	5	à	ò	Ñ	╡	┼	╒	σ	⌡
6	6	å	û	ª	╢	╞	╓	μ	÷
7	7	ç	ù	º	╖	╟	╫	τ	≈
8	8	ê	ÿ	¿	╕	╚	╪	Φ	°
9	9	ë	Ö	⌐	╣	╔	┘	θ	•
10	A	è	Ü	¬	║	╩	┌	Ω	·
11	B	ï	¢	½	╗	╦	█	δ	√
12	C	î	£	¼	╝	╠	▄	∞	η
13	D	ì	¥	¡	╜	═	▌	∅	²
14	E	Ä	Pts	«	╛	╬	▐	∈	■
15	F	Å	ƒ	»	┐	╧	▀	∩	BLANK 'FF'

APPENDIX C

NOTES

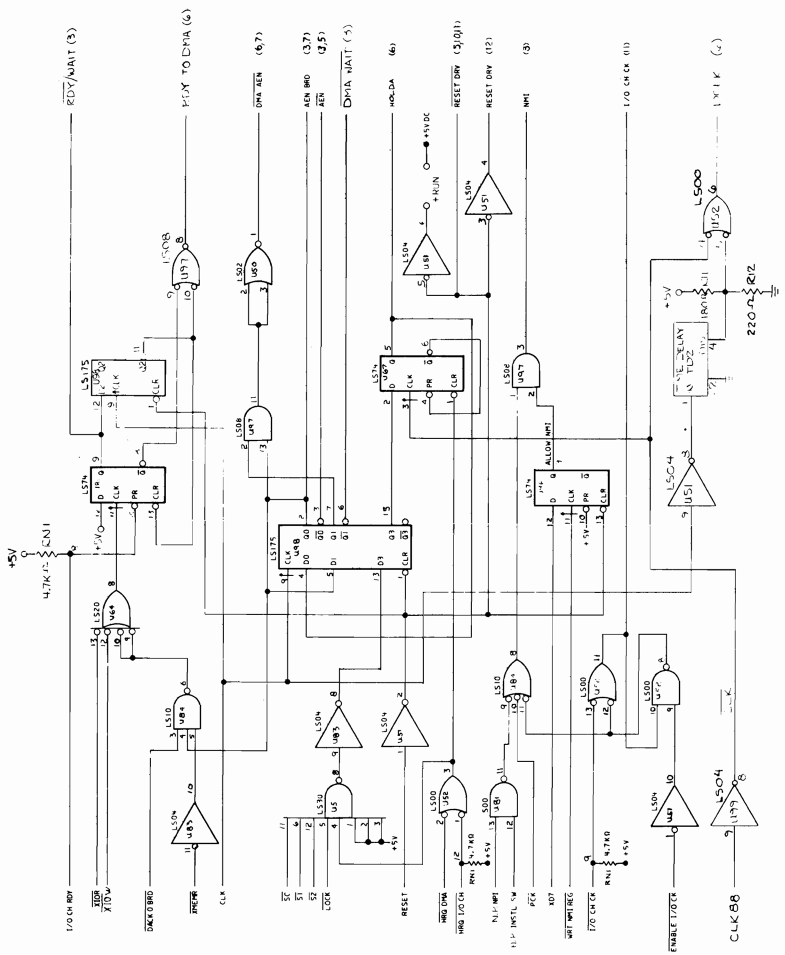
APPENDIX D LOGIC DIAGRAMS

Contents:



System Board (Processor And Support) Logic 3 of 12

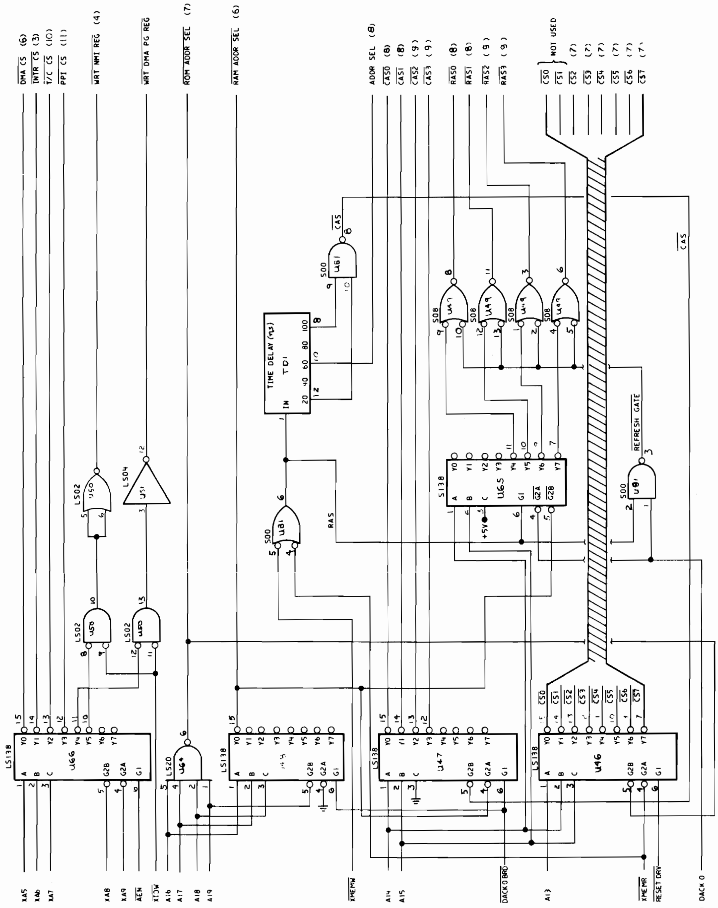
Note: Logics one and two of twelve are not applicable



System Board (Wait State Generator) Logic 4 of 12

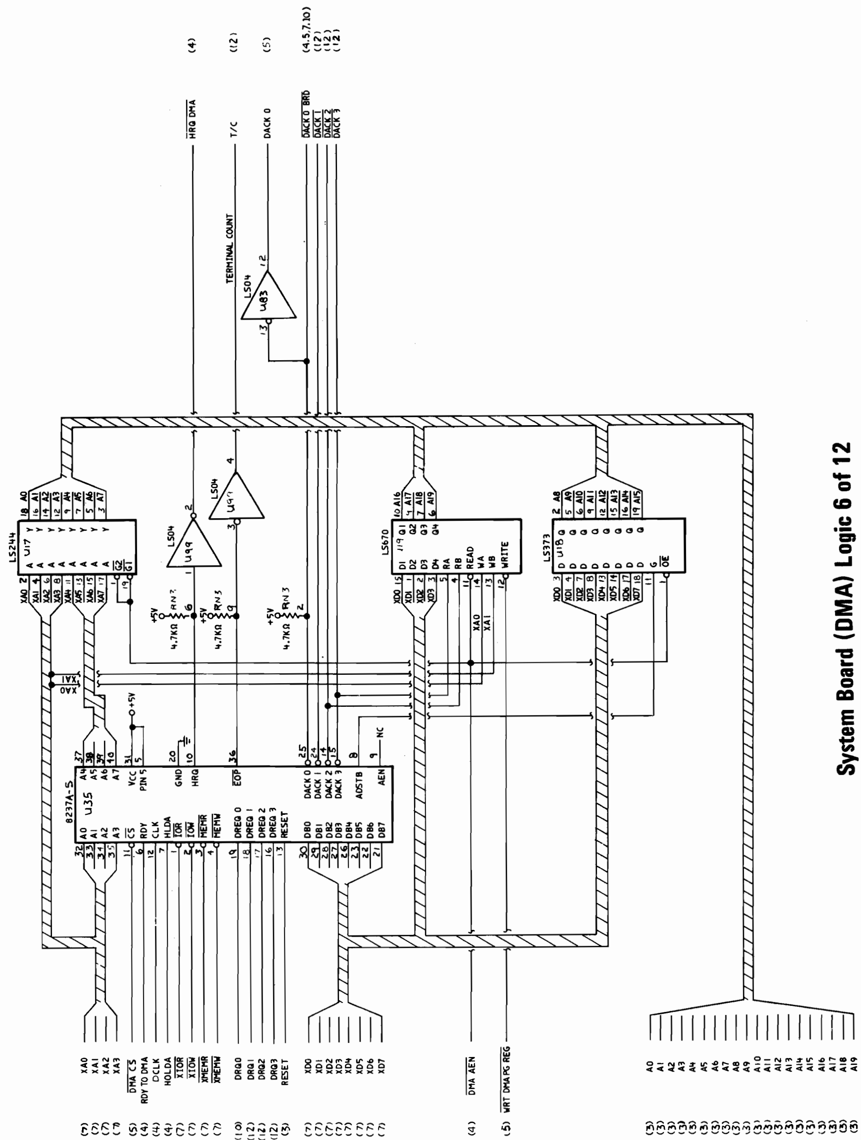
APPENDIX D

SYSTEM BOARD (DEVICE DECODES)

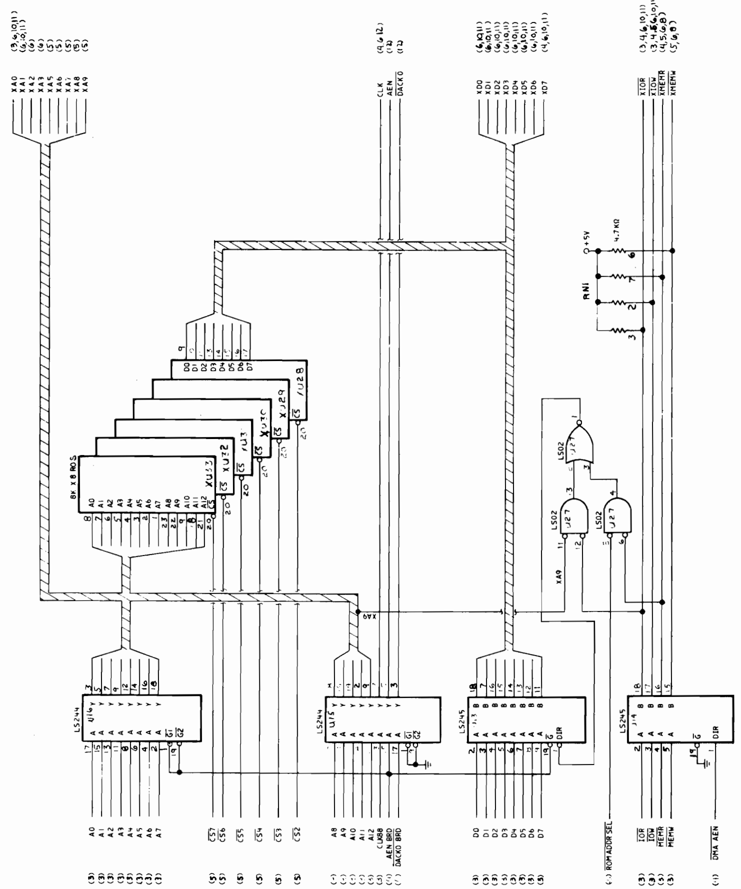


System Board (Device Decodes) Logic 5 of 12

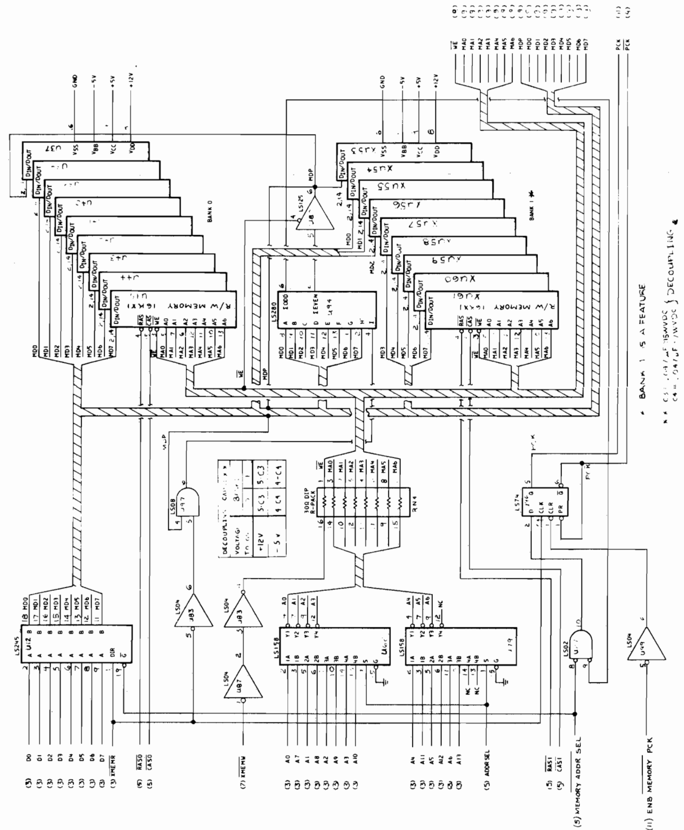
SYSTEM BOARD (DMA)



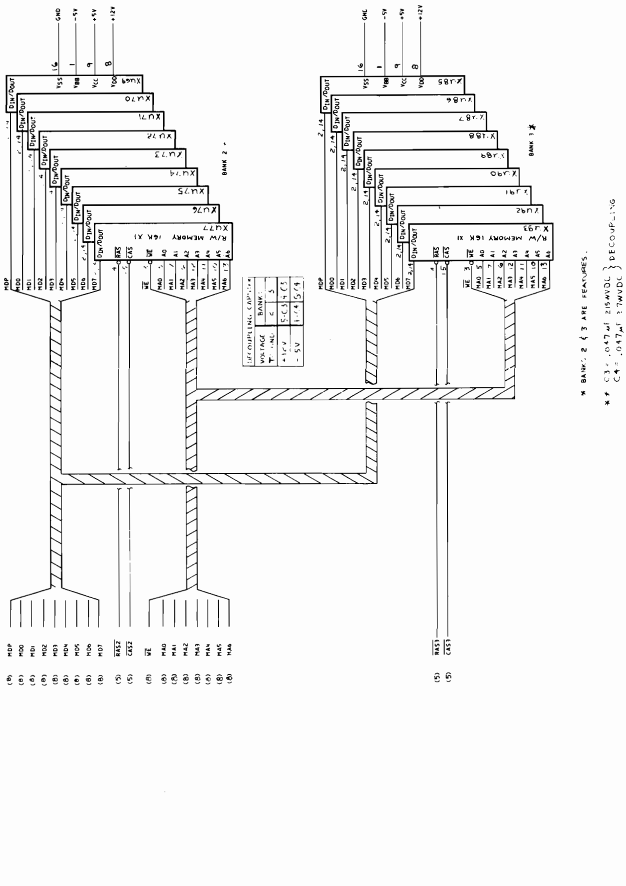
System Board (DMA) Logic 6 of 12



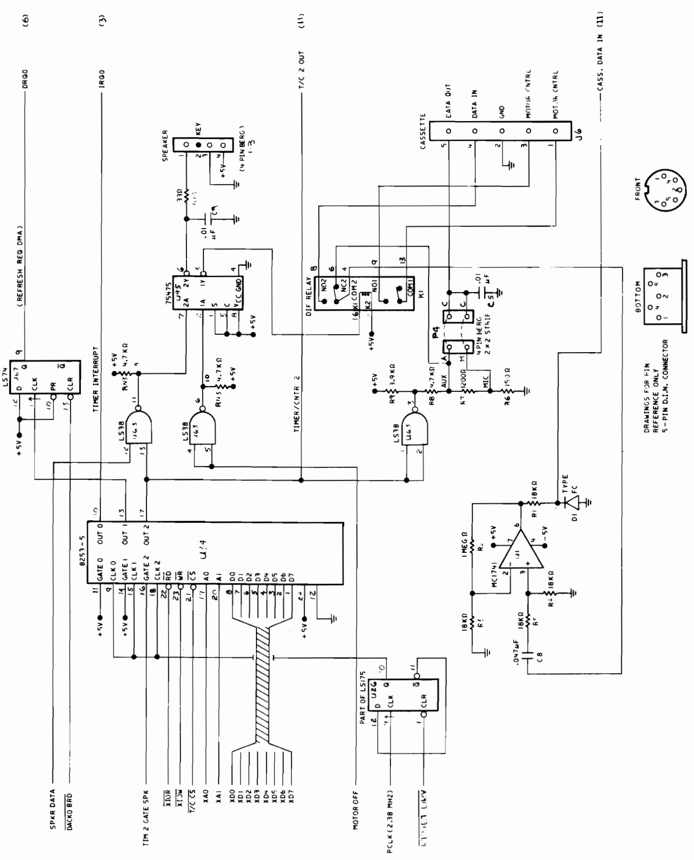
System Board (ROS And Bus Driver) Logic 7 of 12



System Board (Dynamic Memory) Logic 8 of 12

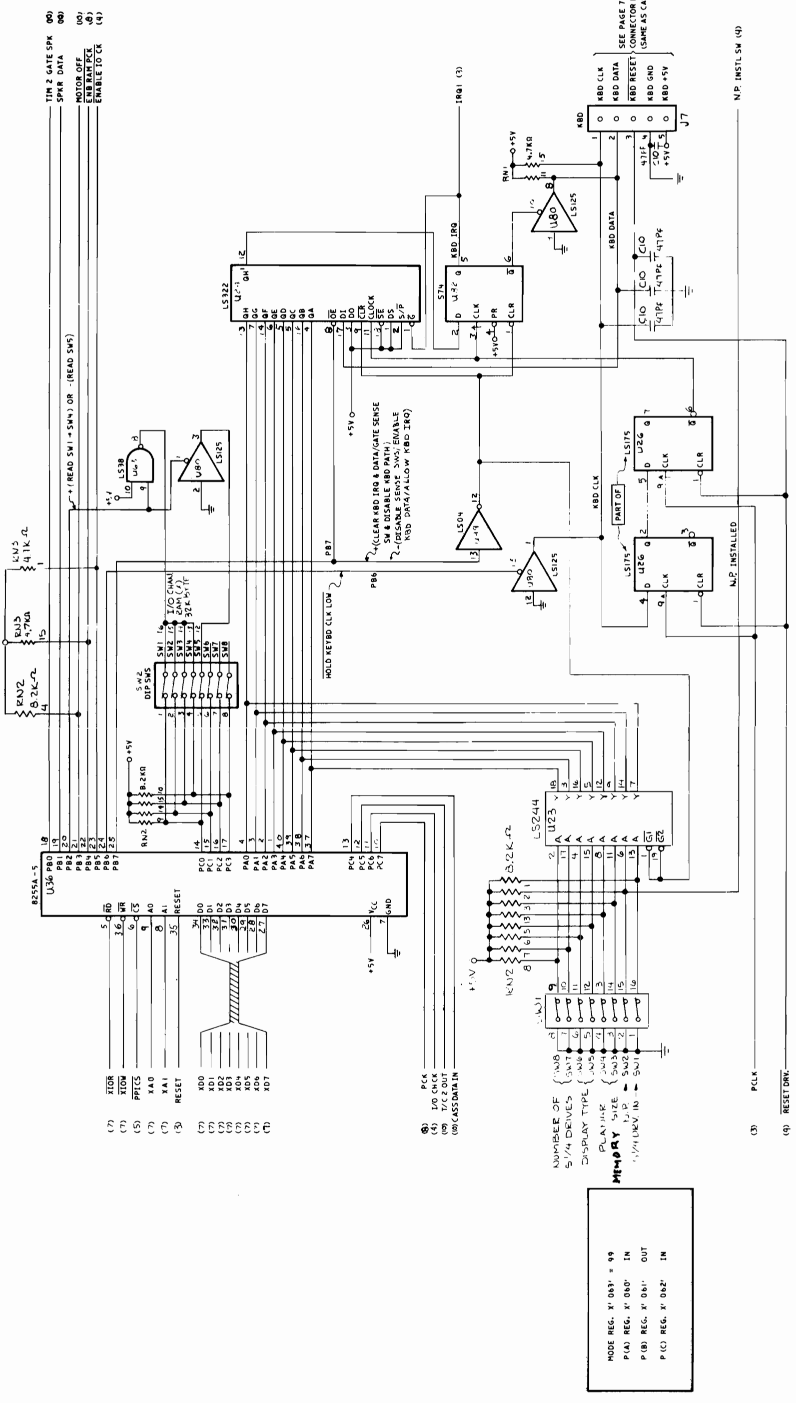


System Board (Dynamic Memory Extended) Logic 9 of 12



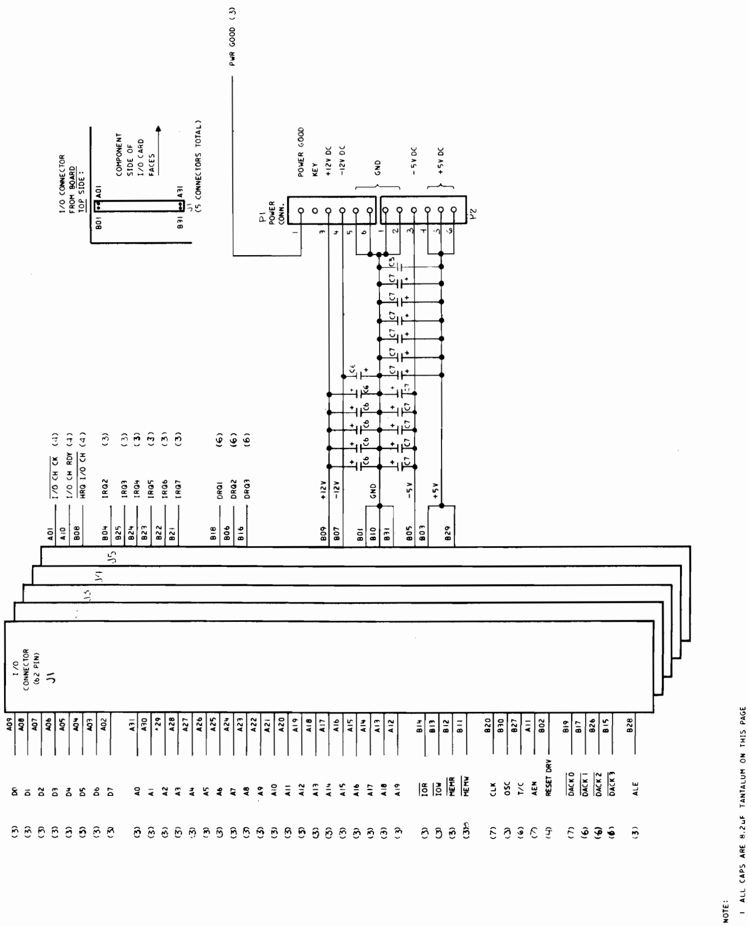
System Board (Speaker/Cassette/Timer/Counter) Logic 10 of 12

APPENDIX D

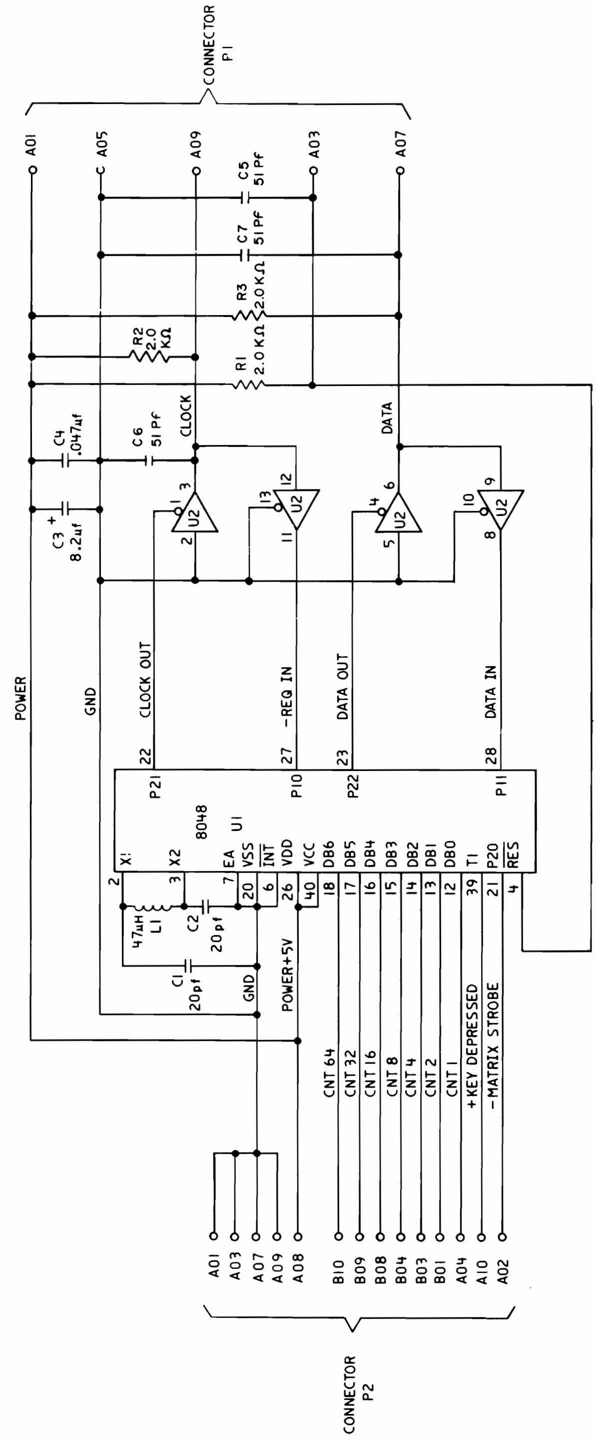


System Board (Keyboard/Sense/Control) Logic 11 of 12

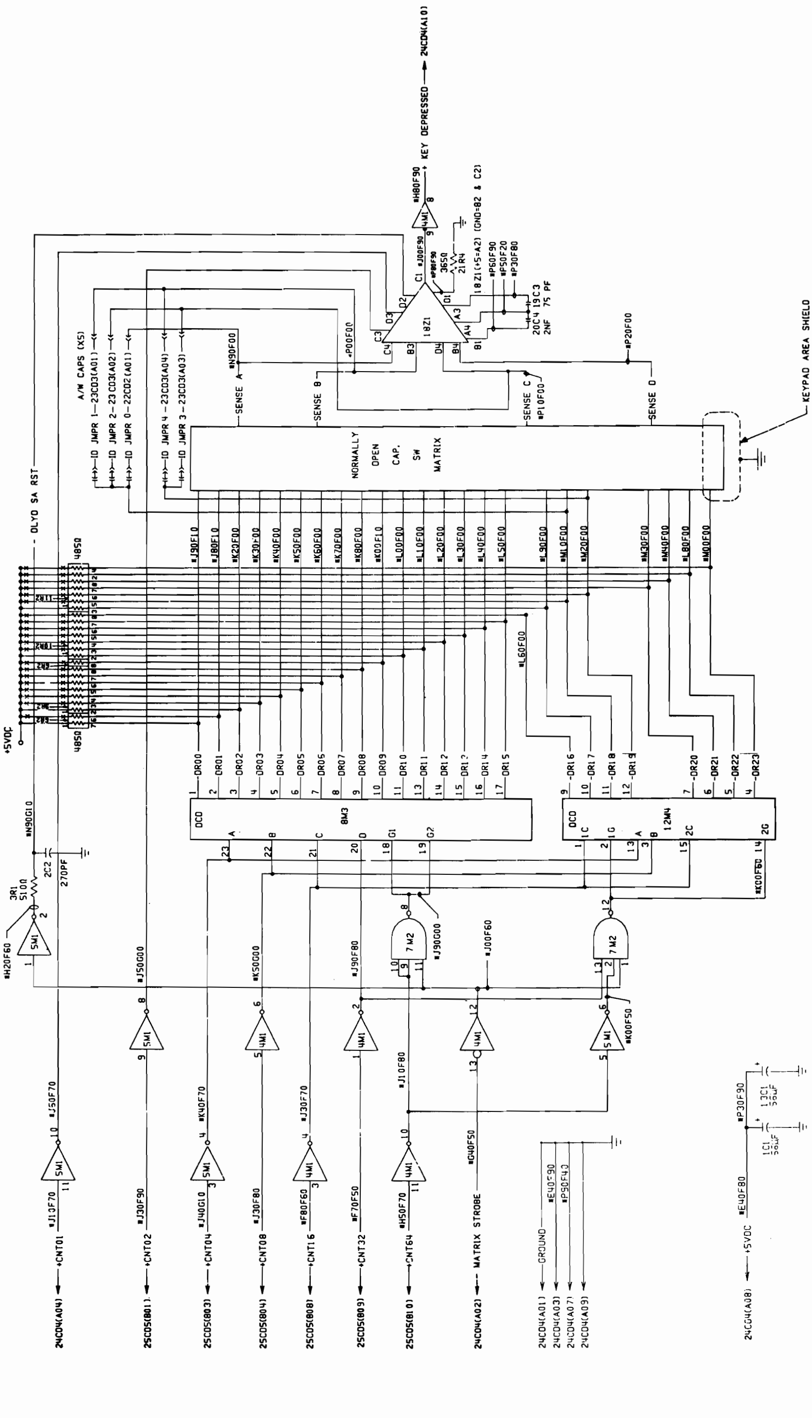
SYSTEM BOARD (I/O CHANNEL)



System Board (I/O Channel) Logic 12 of 12

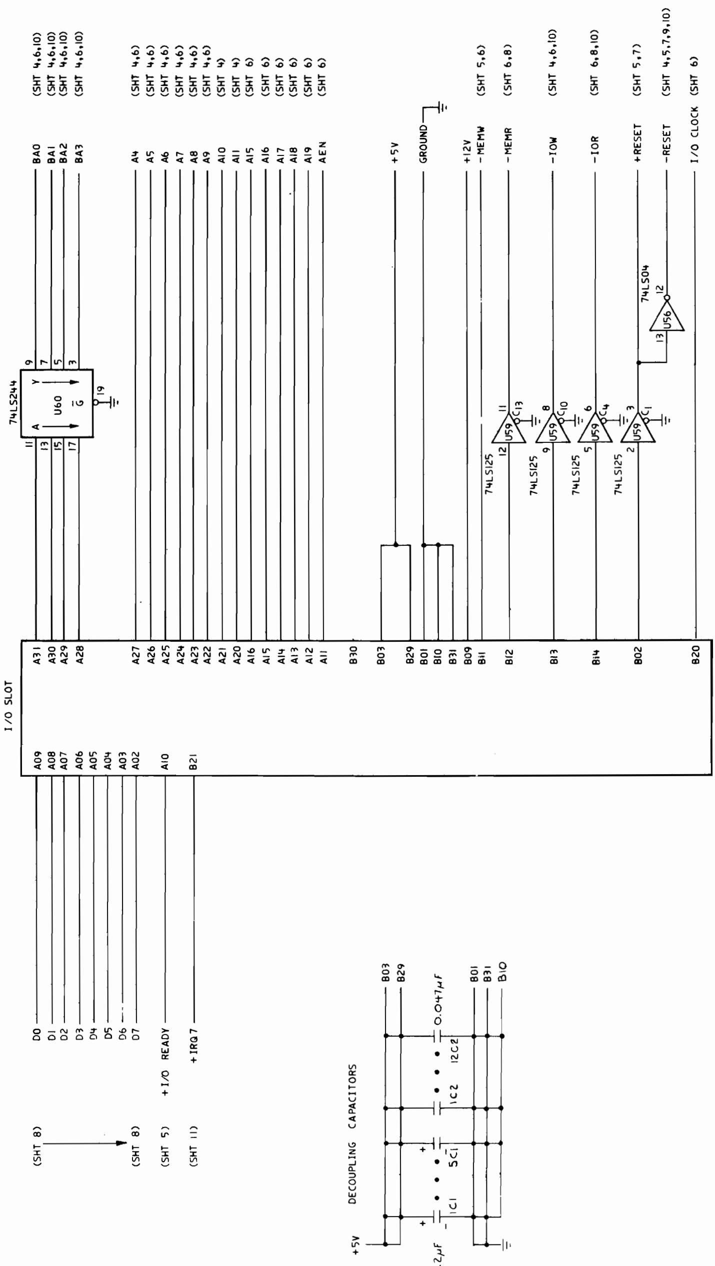


Keyboard Logic 1 of 2



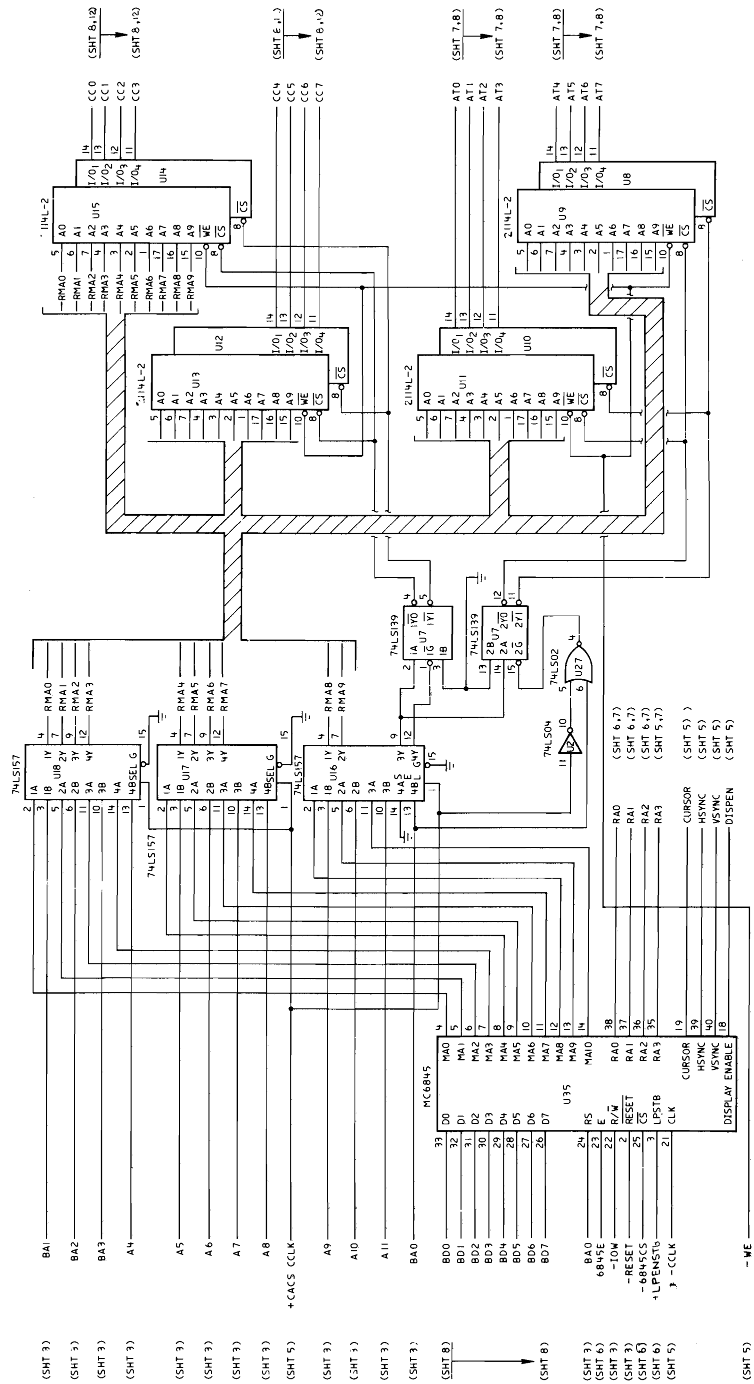
Keyboard Logic 2 of 2

APPENDIX D



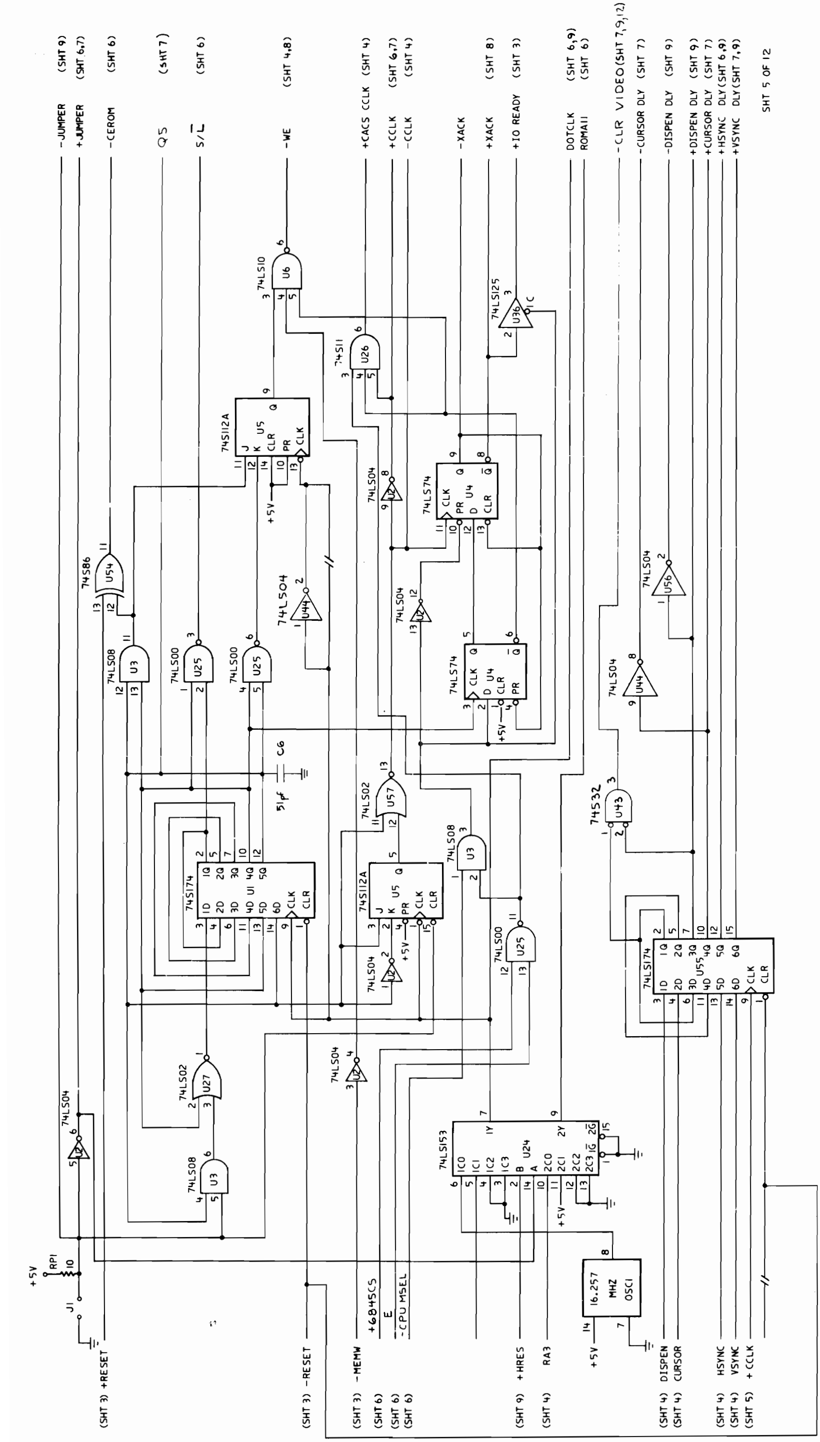
IBM Monochrome Display And Parallel Printer Adapter Logic 3 of 12

Note: Logics one and two of twelve are not applicable.

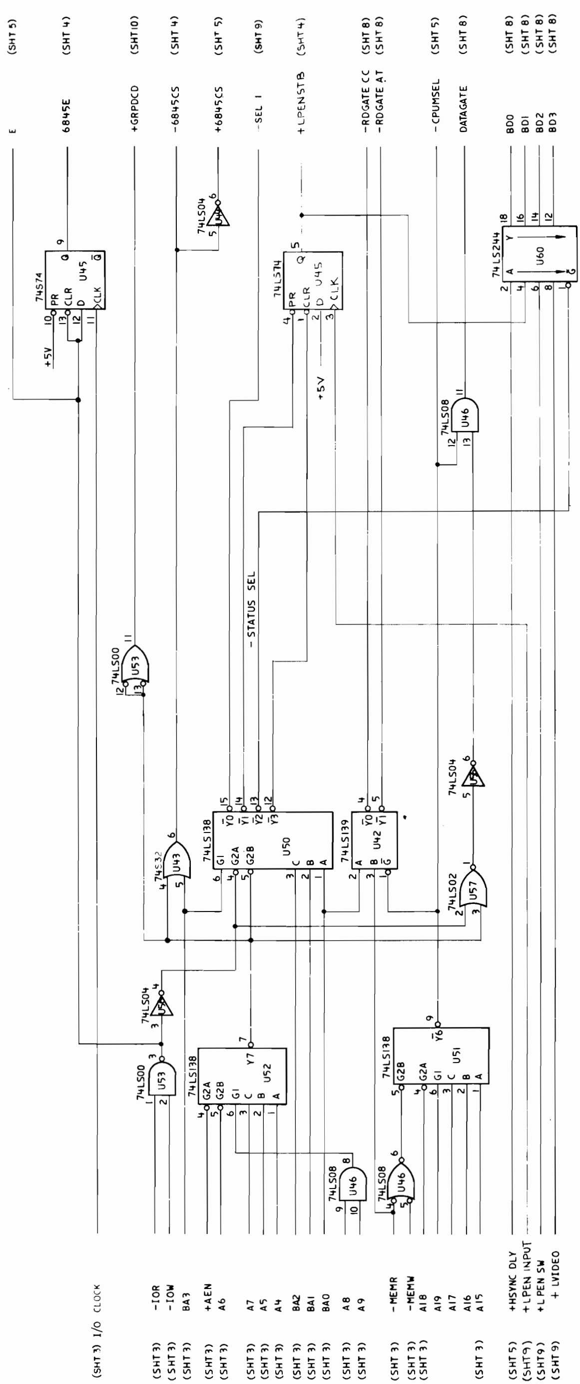


IBM Monochrome Display And Parallel Printer Adapter Logic 4 of 12

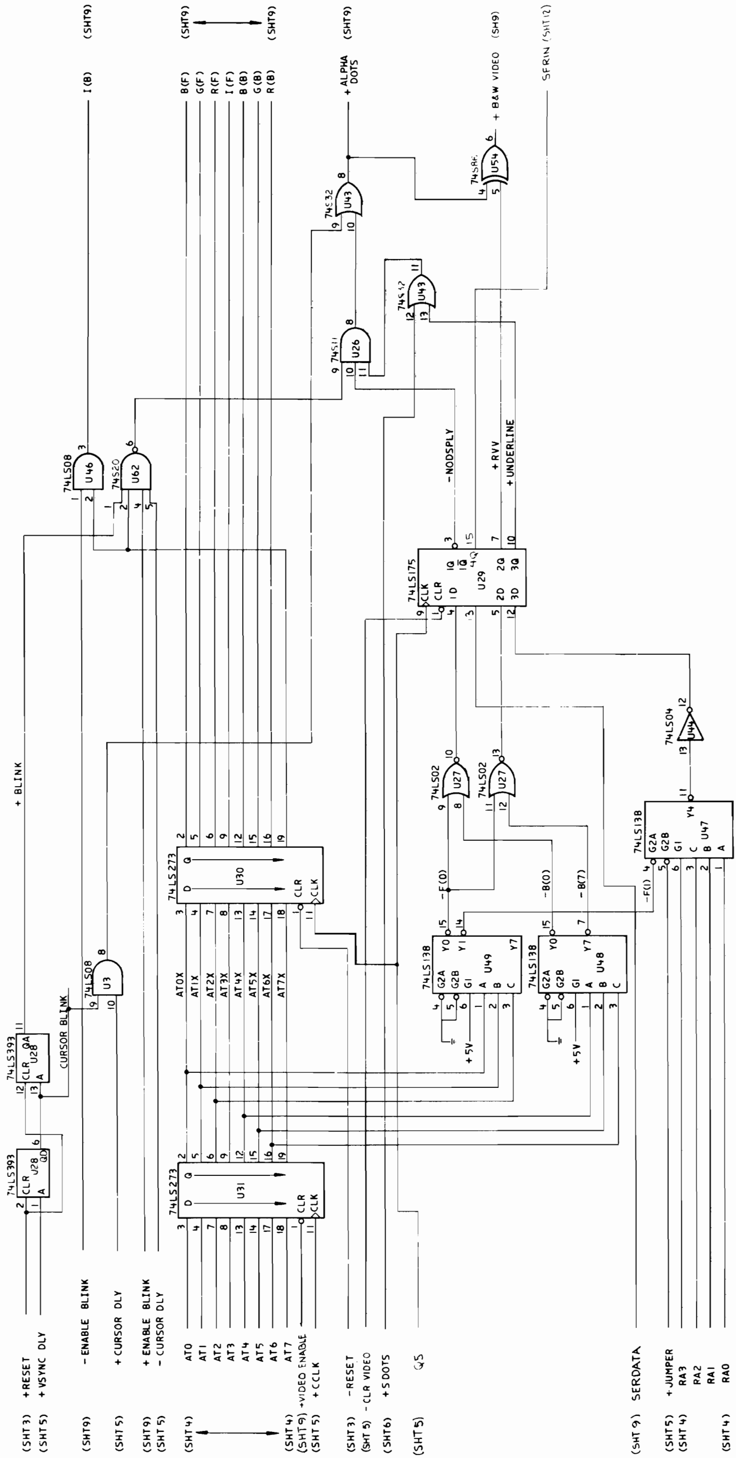
APPENDIX D



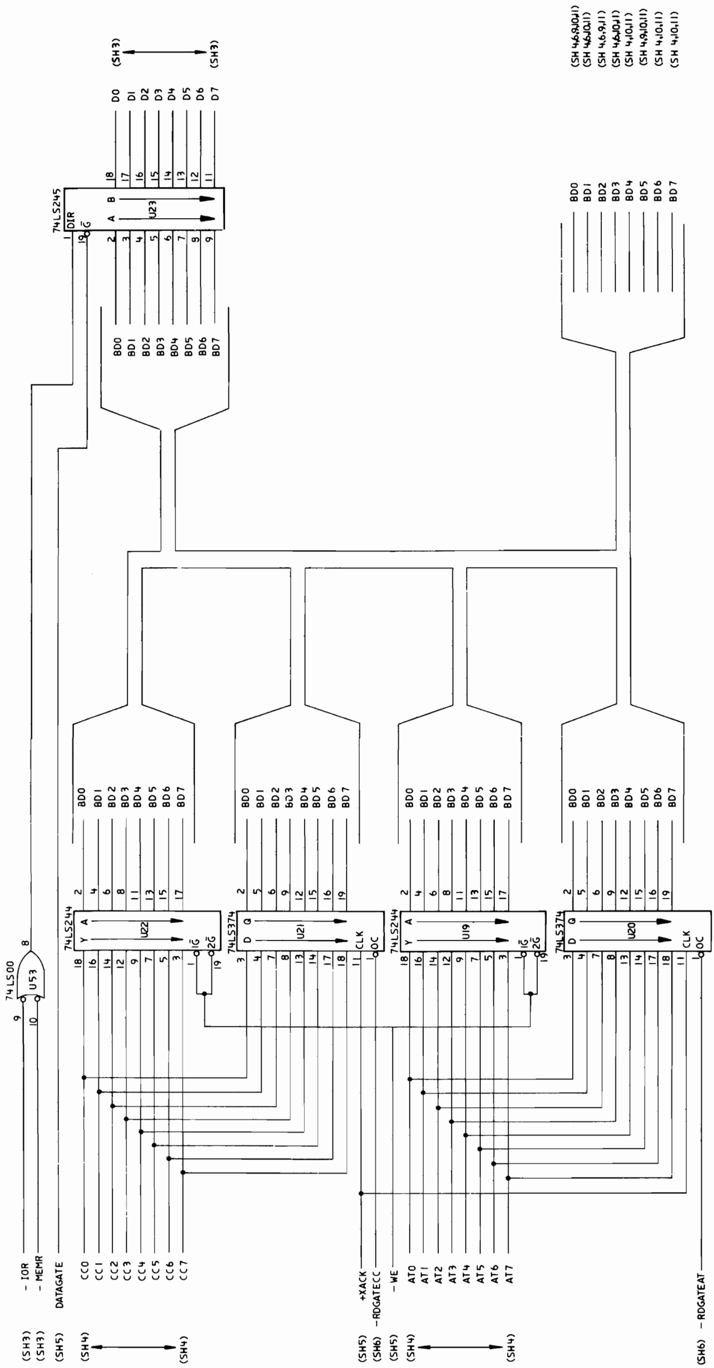
IBM Monochrome Display And Parallel Printer Adapter Logic 5 of 12



IBM Monochrome Display And Parallel Printer Adapter Logic 6 of 12

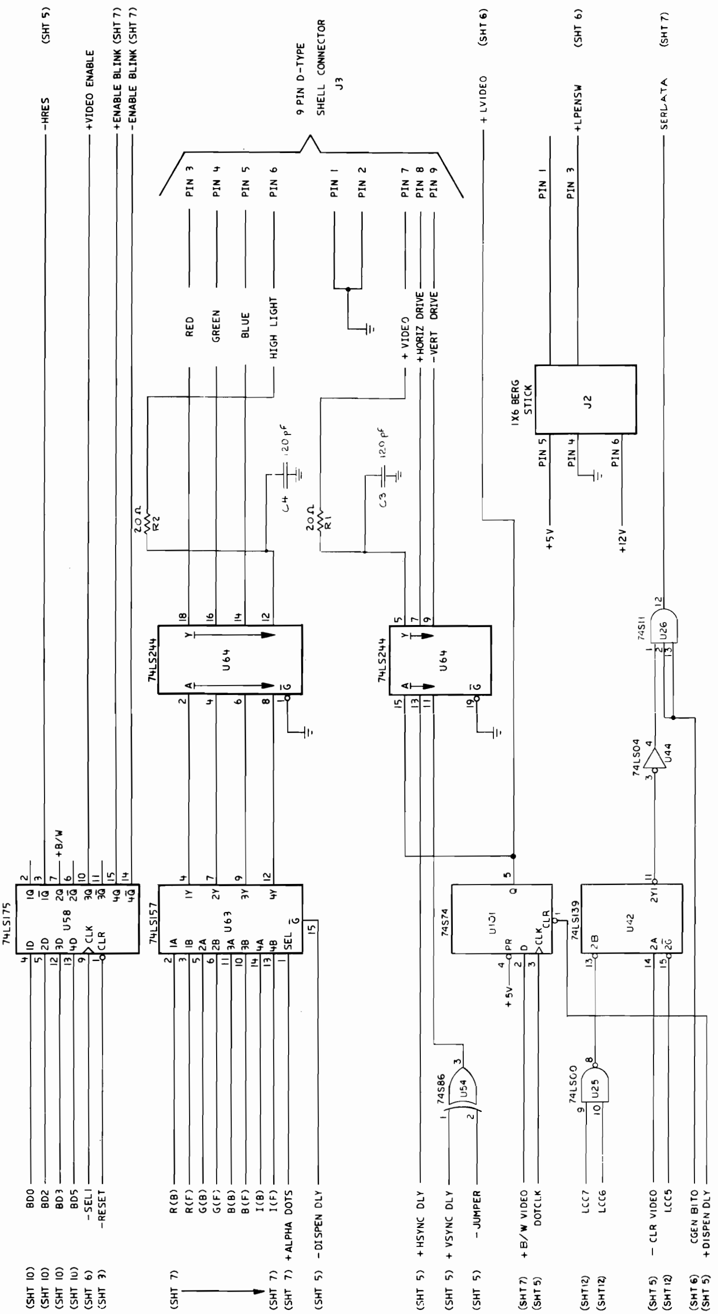


IBM Monochrome Display And Parallel Printer Adapter Logic 7 of 12

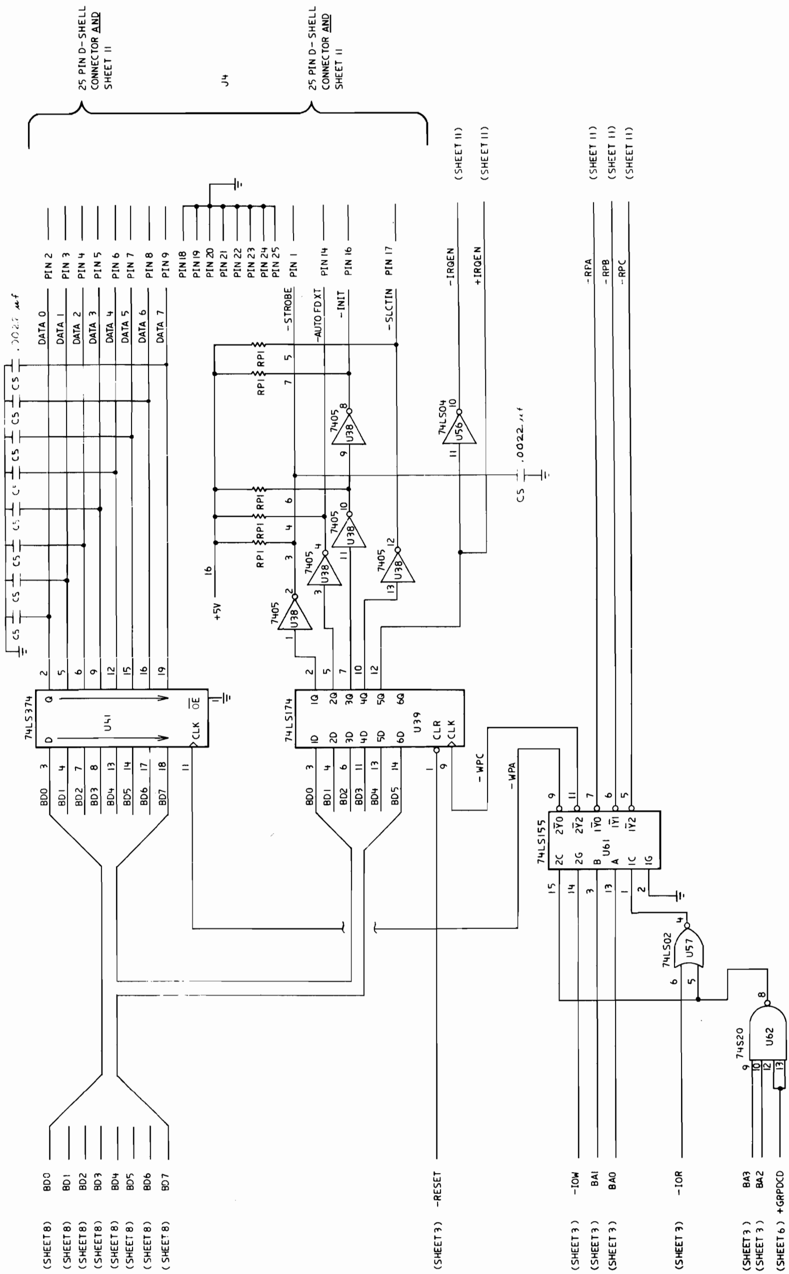


IBM Monochrome Display And Parallel Printer Adapter Logic 8 of 12

APPENDIX D

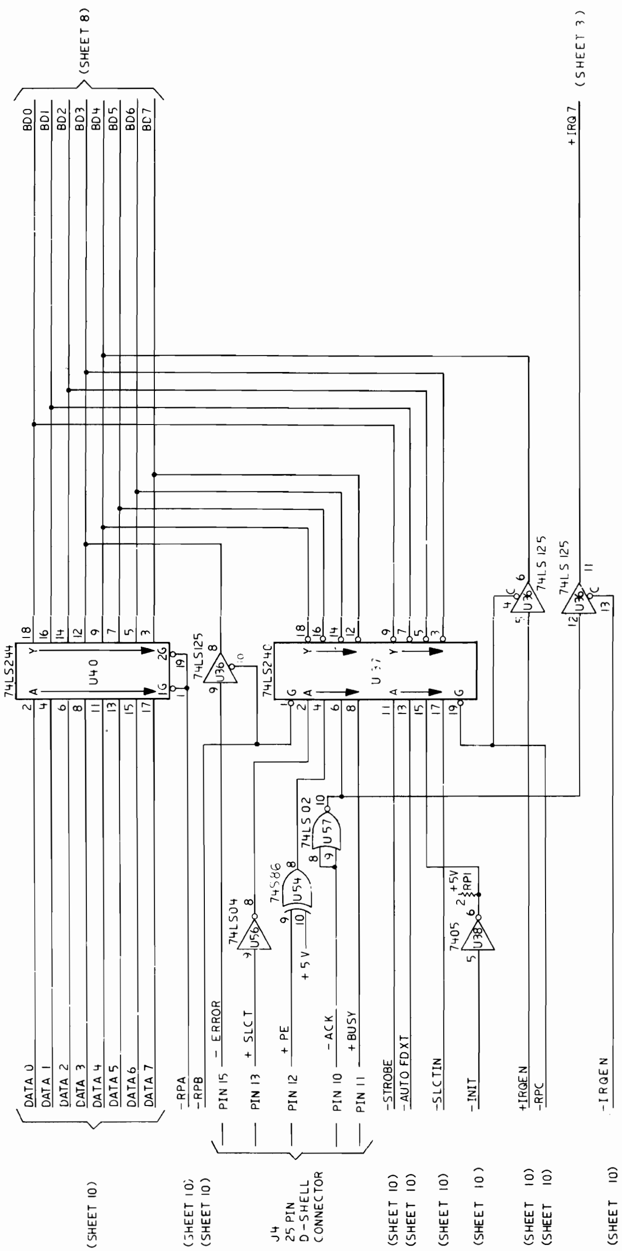


IBM Monochrome Display And Parallel Printer Adapter Logic 9 of 12



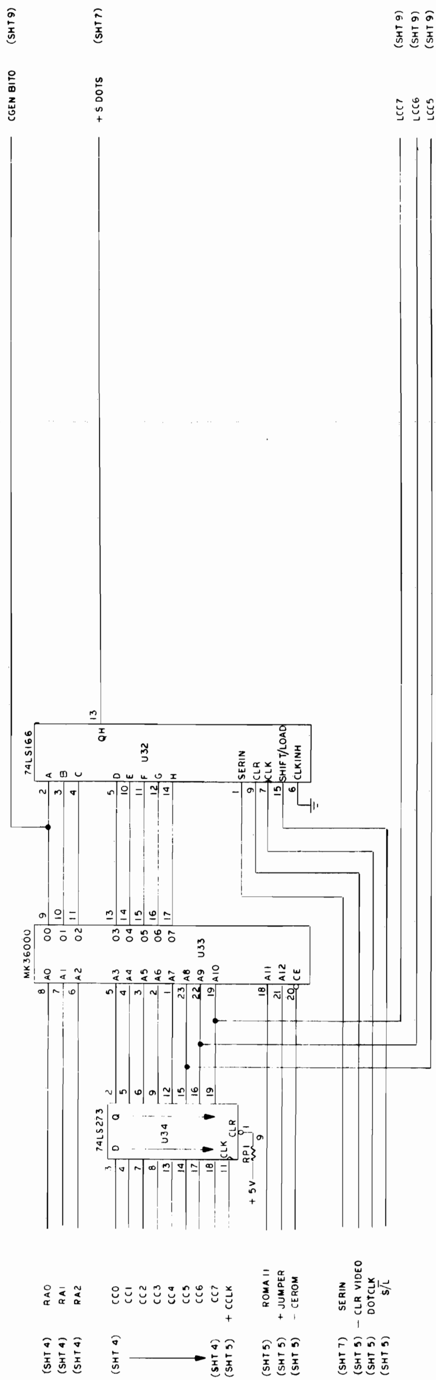
IBM Monochrome Display And Parallel Printer Adapter Logic 10 of 12

APPENDIX D



IBM Monochrome Display And Parallel Printer Adapter Logic 11 of 12

IBM MONOCHROME DISPLAY AND PARALLEL PRINTER ADAPTER



IBM Monochrome Display And Parallel Printer Adapter Logic 12 of 12

APPENDIX D

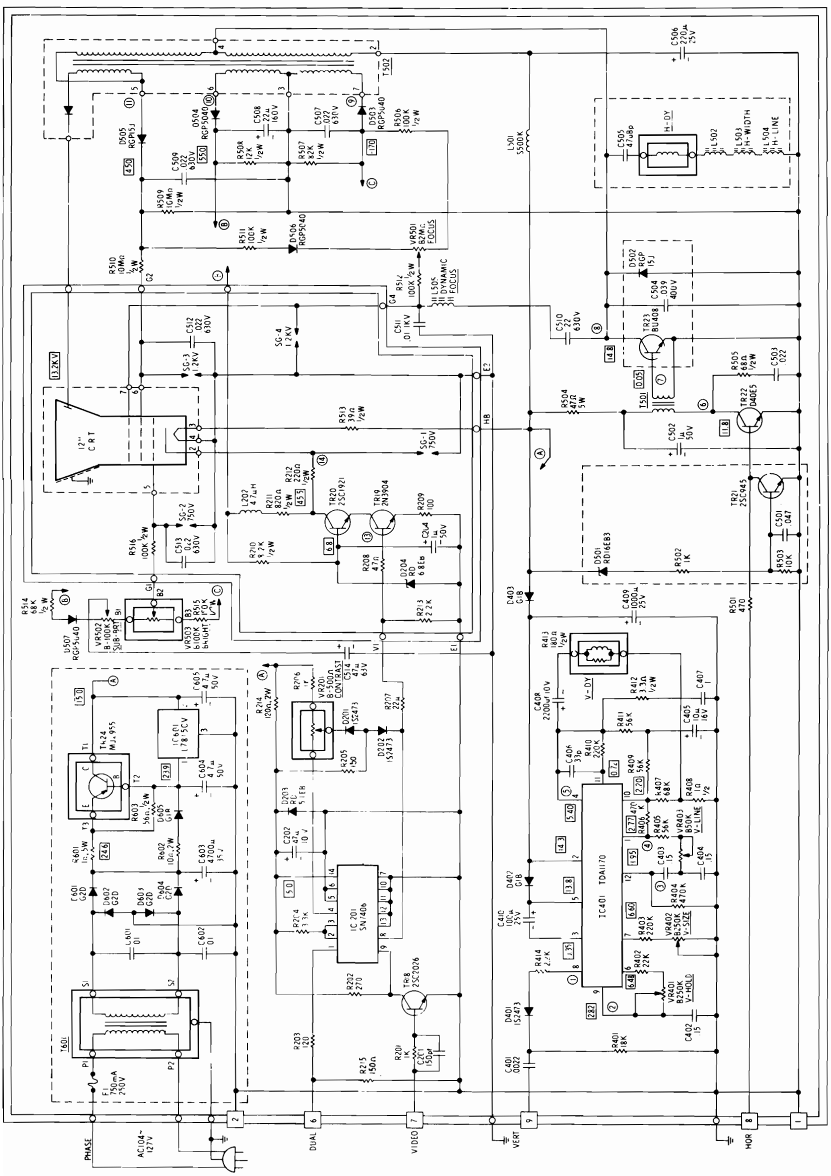
DANGER

HAZARDOUS VOLTAGES UP TO 450 VOLTS EXIST ON THE PRINTED CIRCUIT BOARDS

NOTES

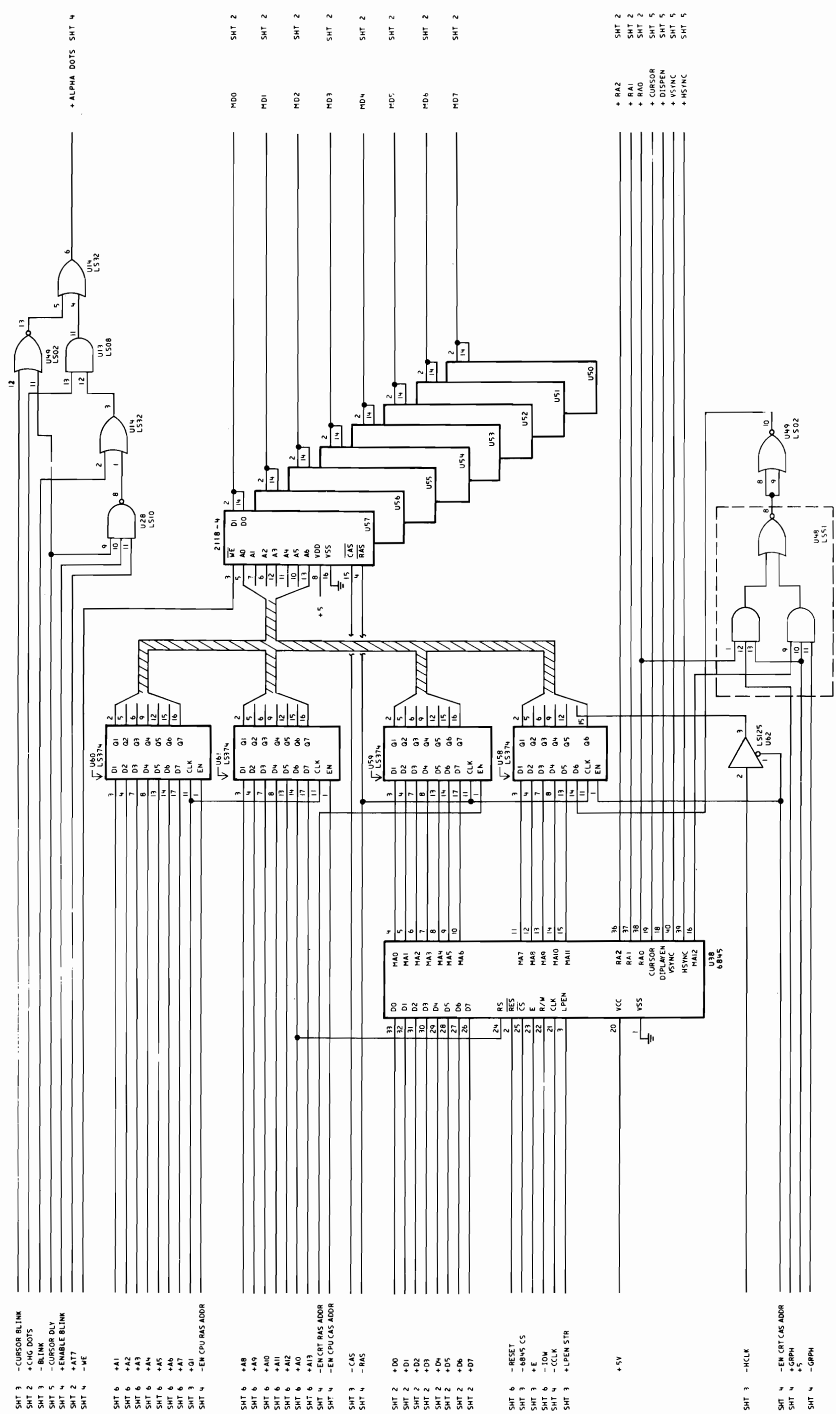
1. RESISTOR VALUES ARE IN (OHM) Ω K=1000Ω M=1,000,000Ω
2. ALL RESISTOR ARE 1/4W EXCEPT WHERE OTHERWISE INDICATED.
3. ALL CAPACITORS ARE 50V EXCEPT WHERE OTHERWISE INDICATED.
4. CAPACITORS VALUES ARE μF UNLESS OTHERWISE INDICATED $\mu = \mu F = 10^{-6}F$.
5. AC WIRING INFORMATION:
   PHASE = BLACK/BROWN WIRE
   NEUTRAL = WHITE/BLUE WIRE
   GROUND = GREEN and YELLOW WIRE

   IMPORTANT: THE PHASE WIRE MUST GO TO THE FUSED SIDE OF TRANSFORMER.

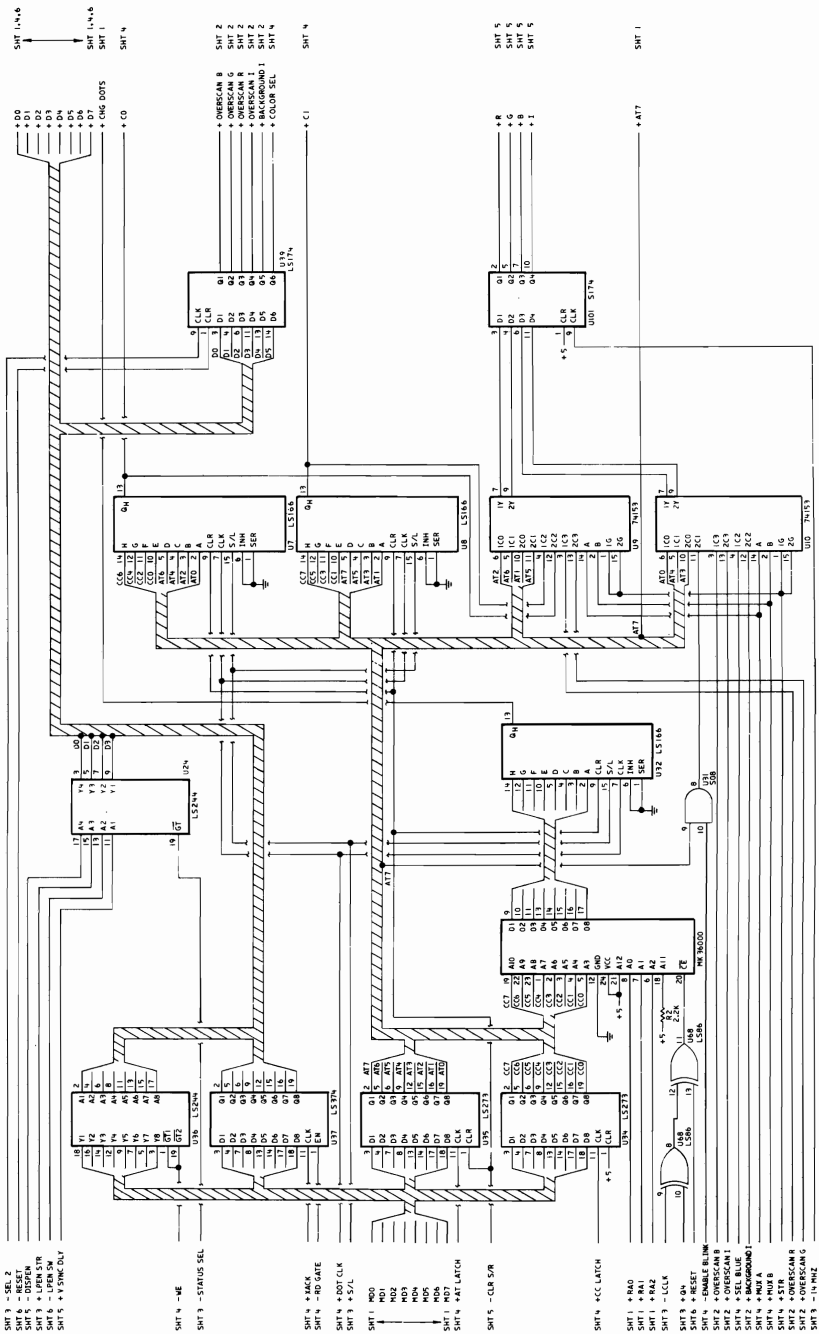


IBM Monochrome Display

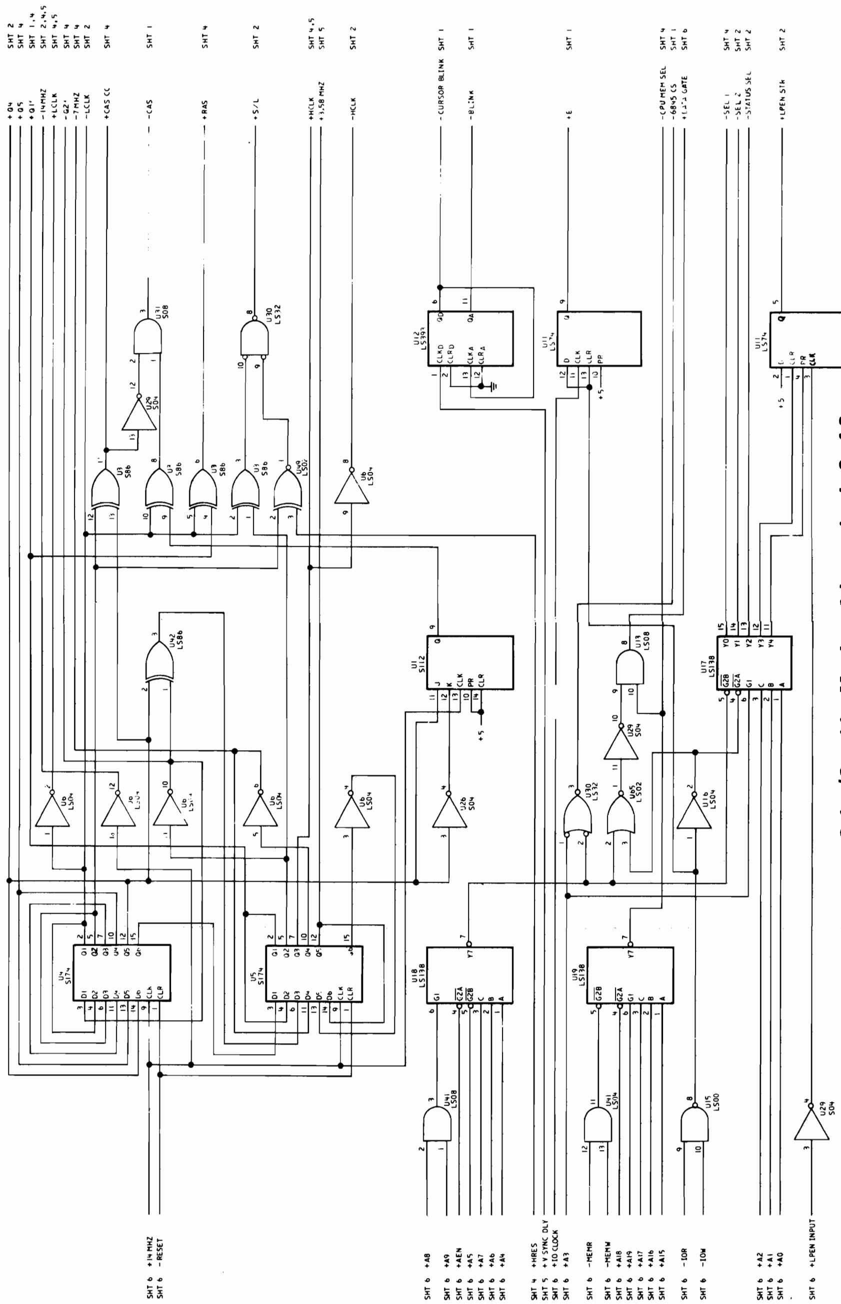
COLOR/GRAPHICS MONITOR ADAPTER



Color/Graphics Monitor Adapter Logic 1 of 6

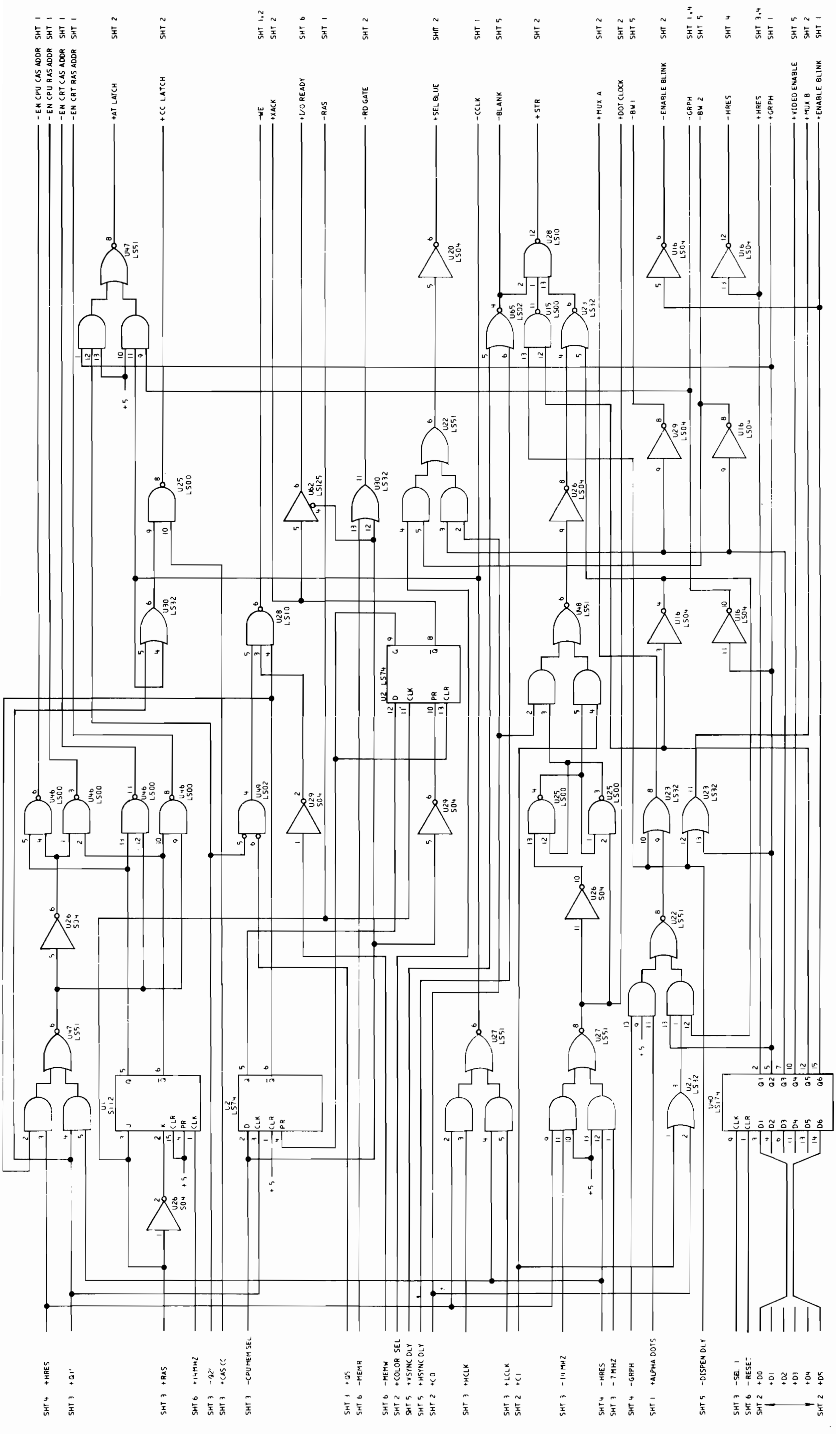


Color/Graphics Monitor Adapter Logic 2 of 6



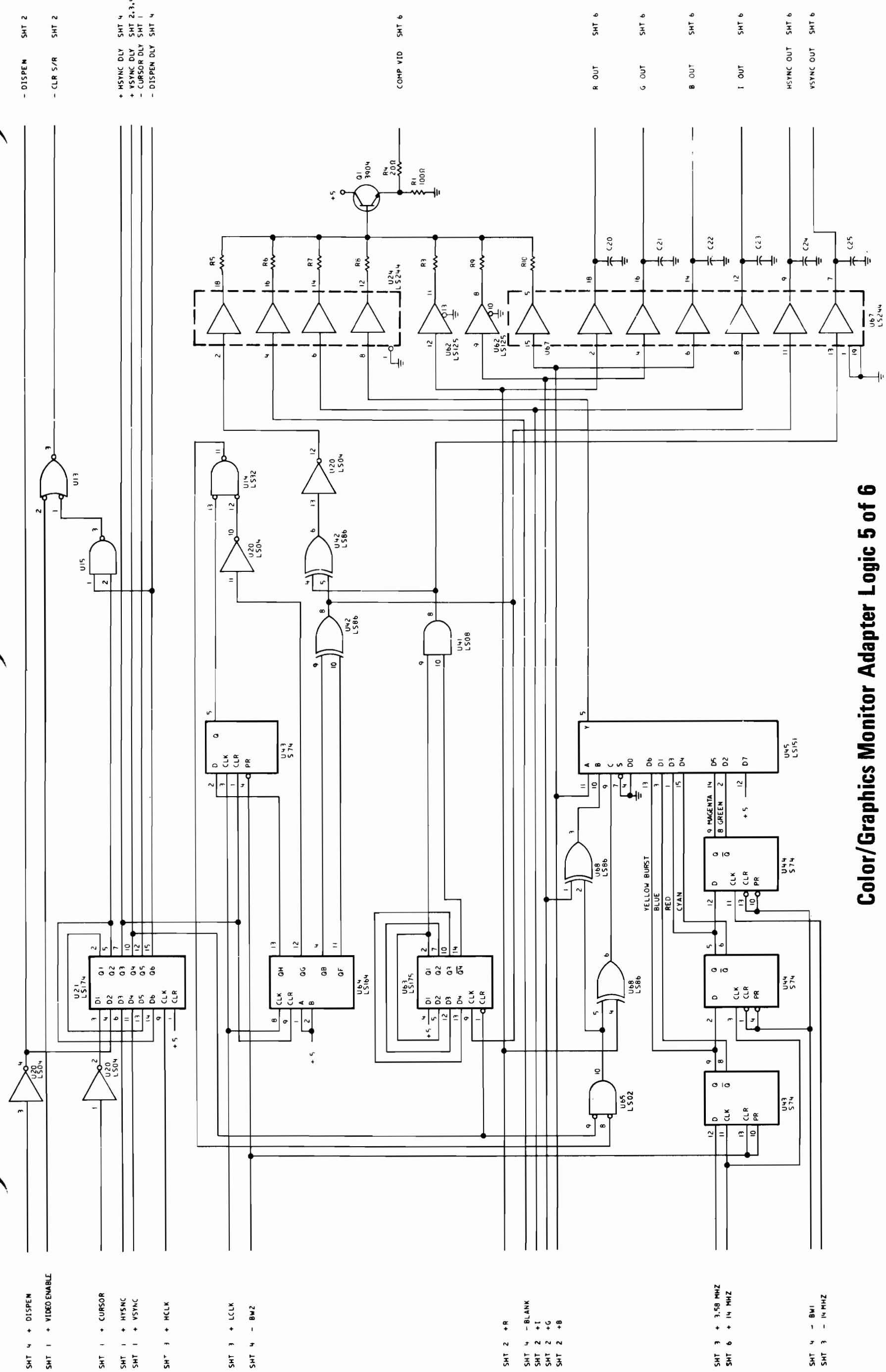
Color/Graphics Monitor Adapter Logic 3 of 6

COLOR/GRAPHICS MONITOR ADAPTER

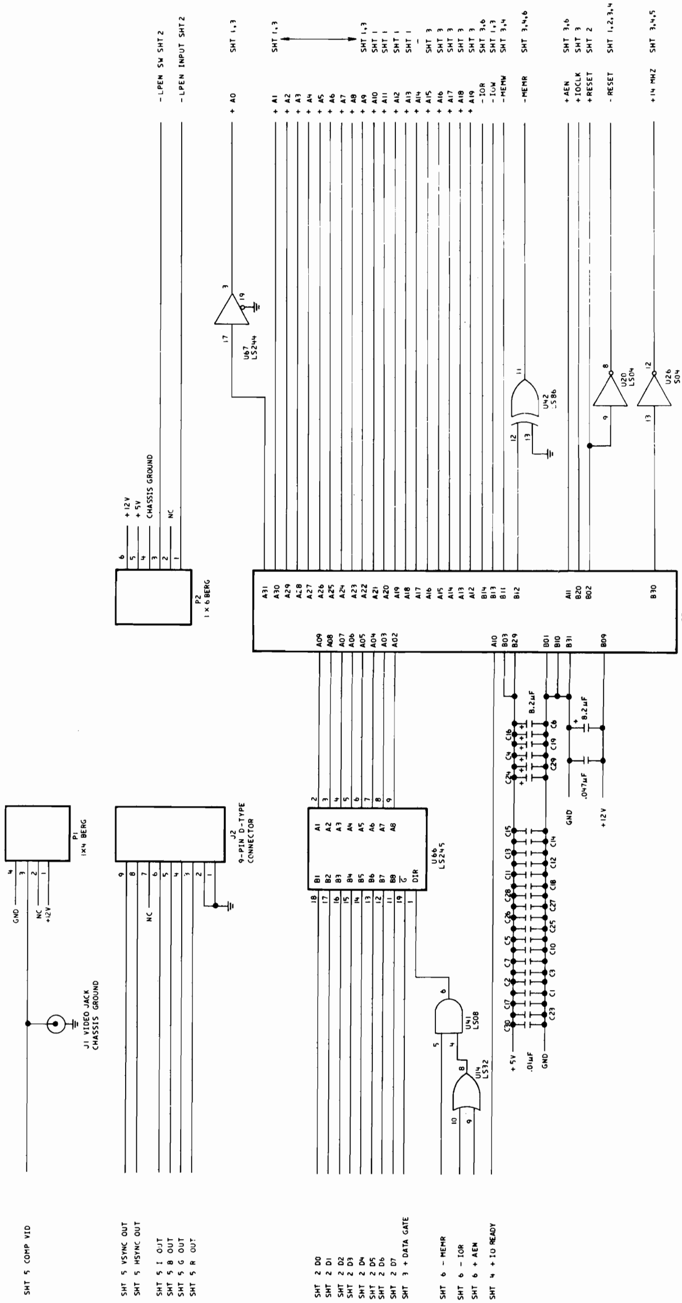


Color/Graphics Monitor Adapter Logic 4 of 6

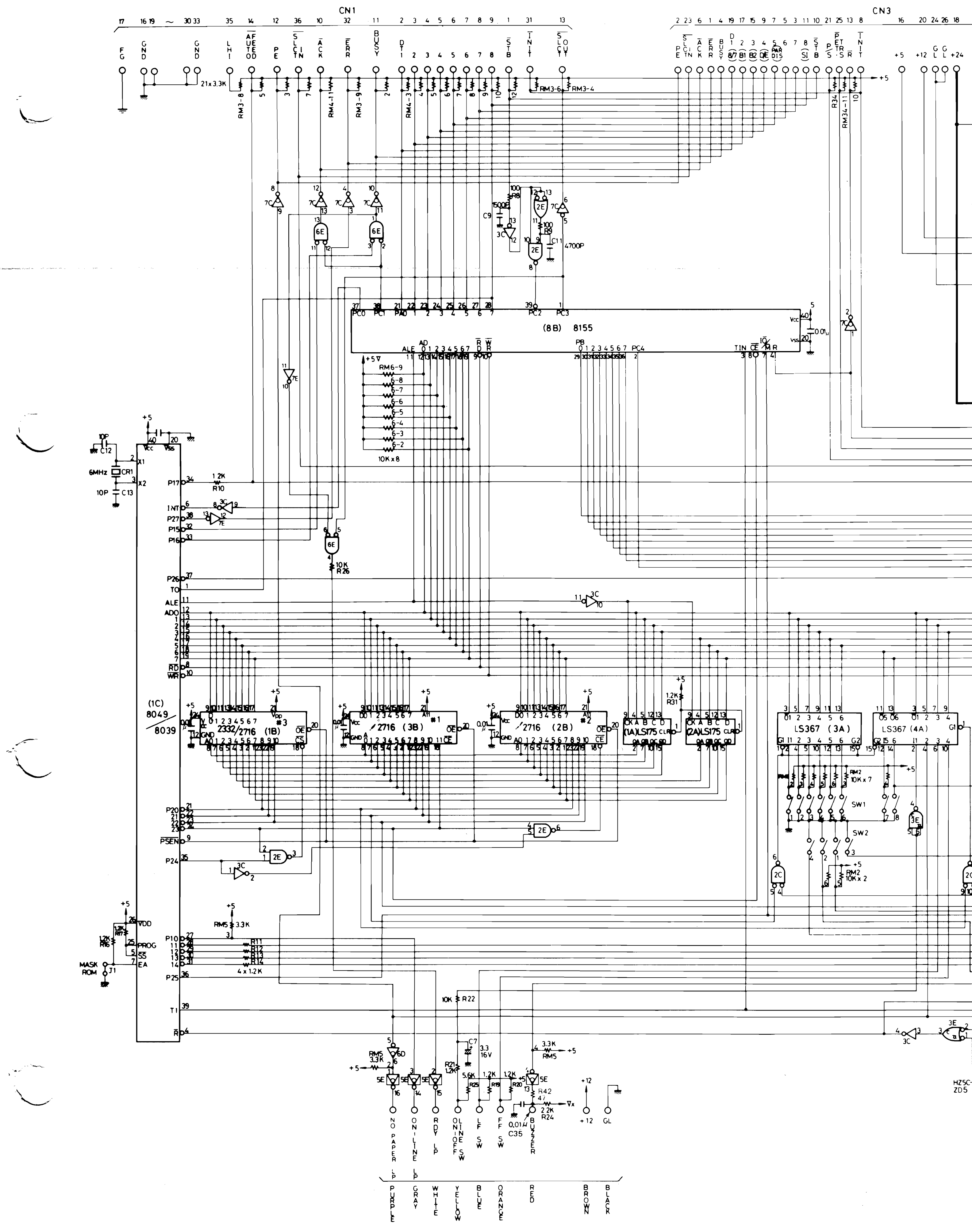
COLOR/GRAPHICS MONITOR ADAPTER



Color/Graphics Monitor Adapter Logic 5 of 6

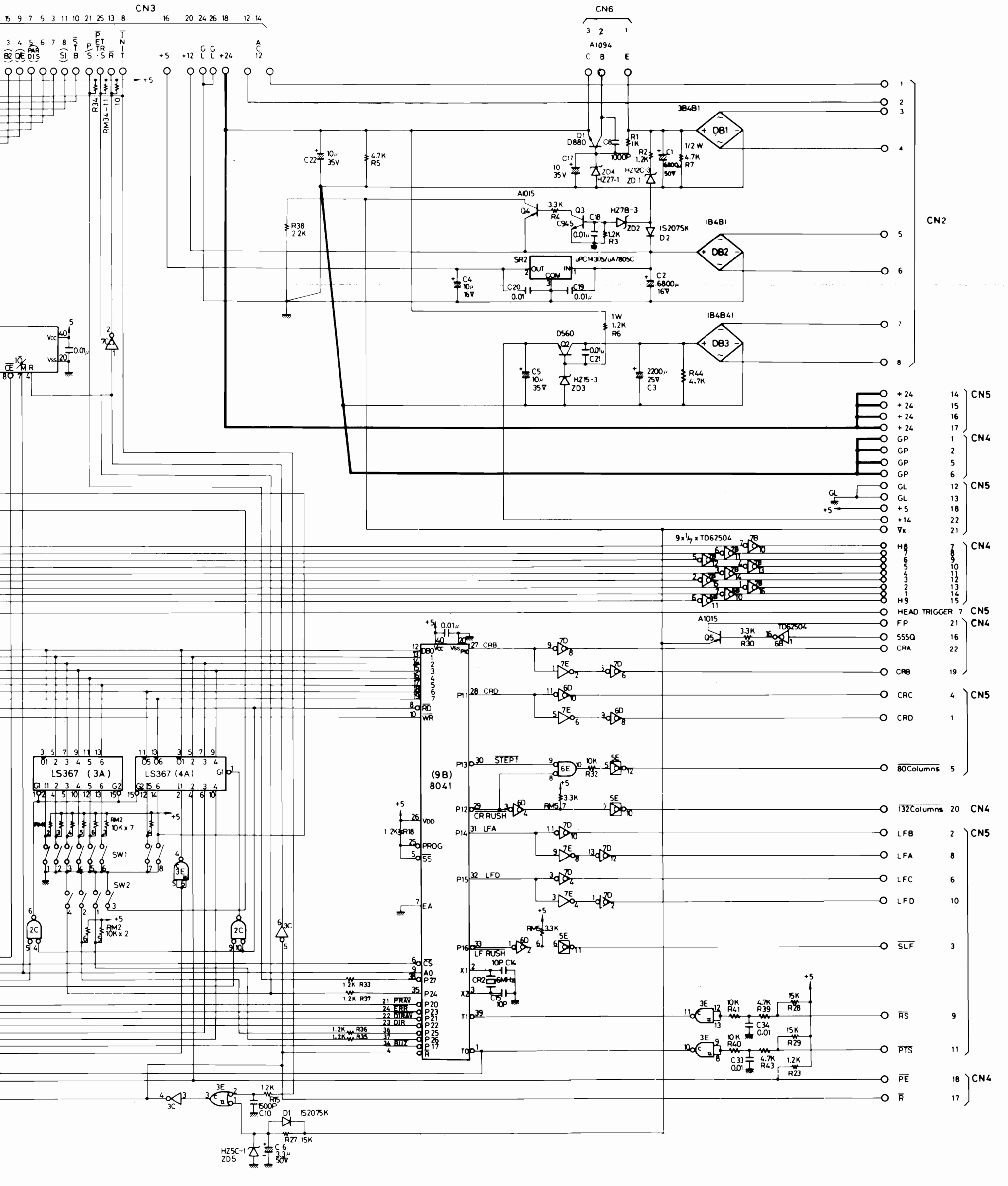


Color/Graphics Monitor Adapter Logic 6 of 6



CN1
CN3
FG
GND
LHI
AFUEOD
PE
SLCTN
ACK
ERR
BUSY
DT1 2 3 4 5 6 7 8
STB
INIT
SLCTY
21x3.3K
RM3-8
RM4-11
RM3-9
RM4-3
RM3-6
RM3-4
RM34-11
(8B) 8155
PC0 PC1 PA0
PC2 PC3
PB
PC4
ALE
AD
RD
WR
TIN
CE
M R
Vcc
Vss
0.01μ
RM6-9
10K x 8
4700P
C9
C11
1500P
R8
R9
7C
6E
2E
3C
7E
(1C)
8049/8039
6MHz
CR1
10P
C12
C13
X1
X2
P17
R10
1.2K
INT
P27
P15
P16
P26
T0
ALE
AD0
RD
WR
10K
R26
R31
2332/2716 (1B)
2716 (3B)
2716 (2B)
(1A)LS175
(2A)LS175
LS367 (3A)
LS367 (4A)
RM2 10K x 7
SW1
SW2
RM2 10K x 2
2C
3E
P20
PSEN
P24
VDD
PROG
SS
EA
MASK ROM
J1
R16
R17
P10
P25
T1
R
RM5 3.3K
R11
R12
R13
R14
4 x 1.2K
R22
5E
6D
C7
3.3 16V
R21
R25
R19
R20
5.6K
1.2K
R42
2.2K
R24
0.01μ
C35
+12
GL
NO PAPER LP
ON-LINE LP
RDY LP
ON LINE OFF SW
LF SW
FF SW
BUZZER
PURPLE
GRAY
WHITE
YELLOW
BLUE
ORANGE
RED
BROWN
BLACK
HZ5C
ZD5

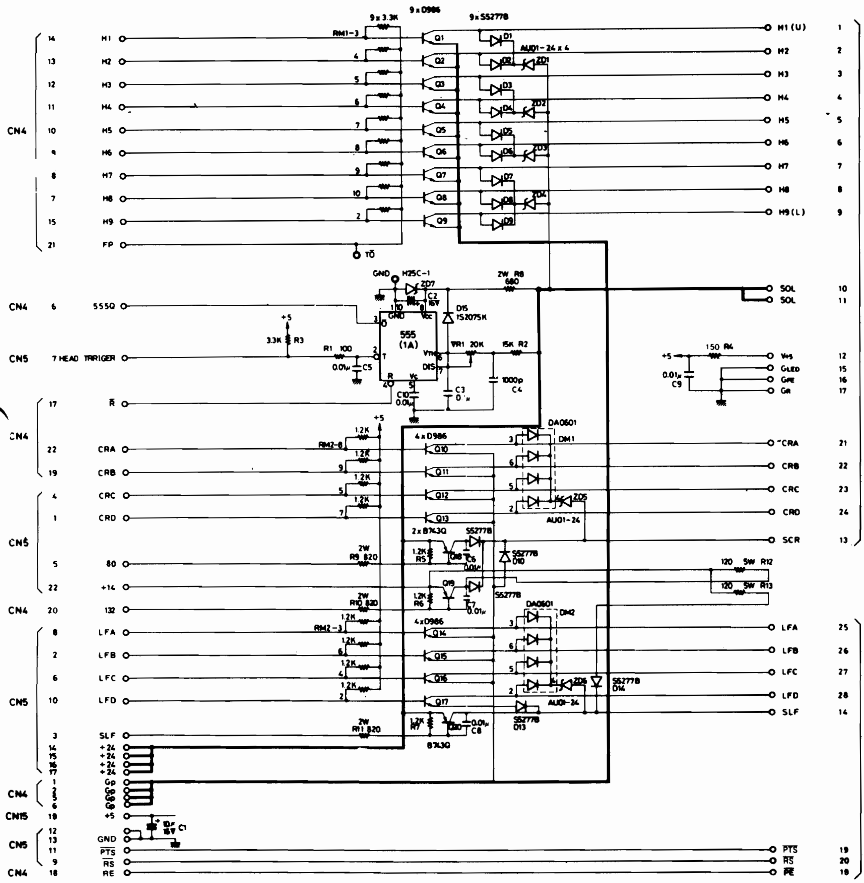
IBM 80 CPS MATRIX PRINTER



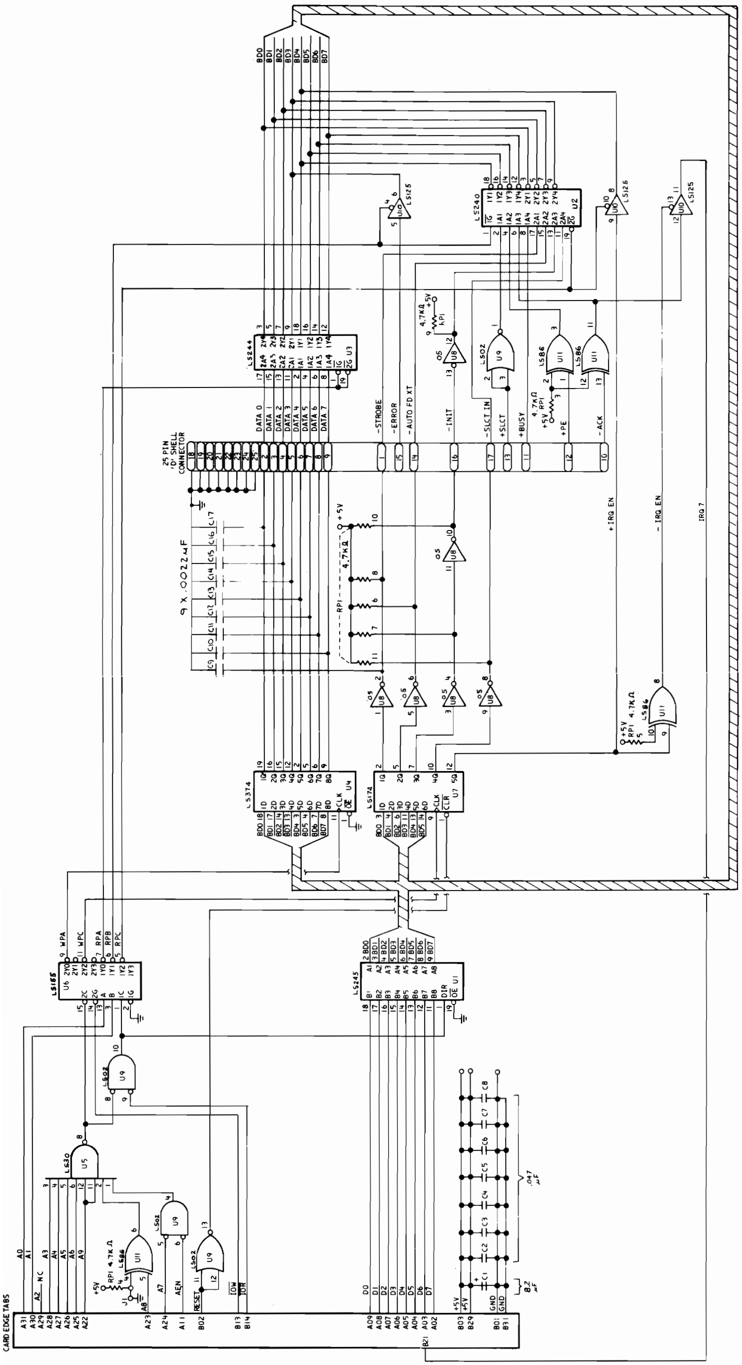
IBM 80 CPS Dot Matrix Printer Diagram of Control Circuit

APPENDIX D

IBM 80 CPS DIAGRAM OF DRIVER CIRCUIT MATRIX PRINTER

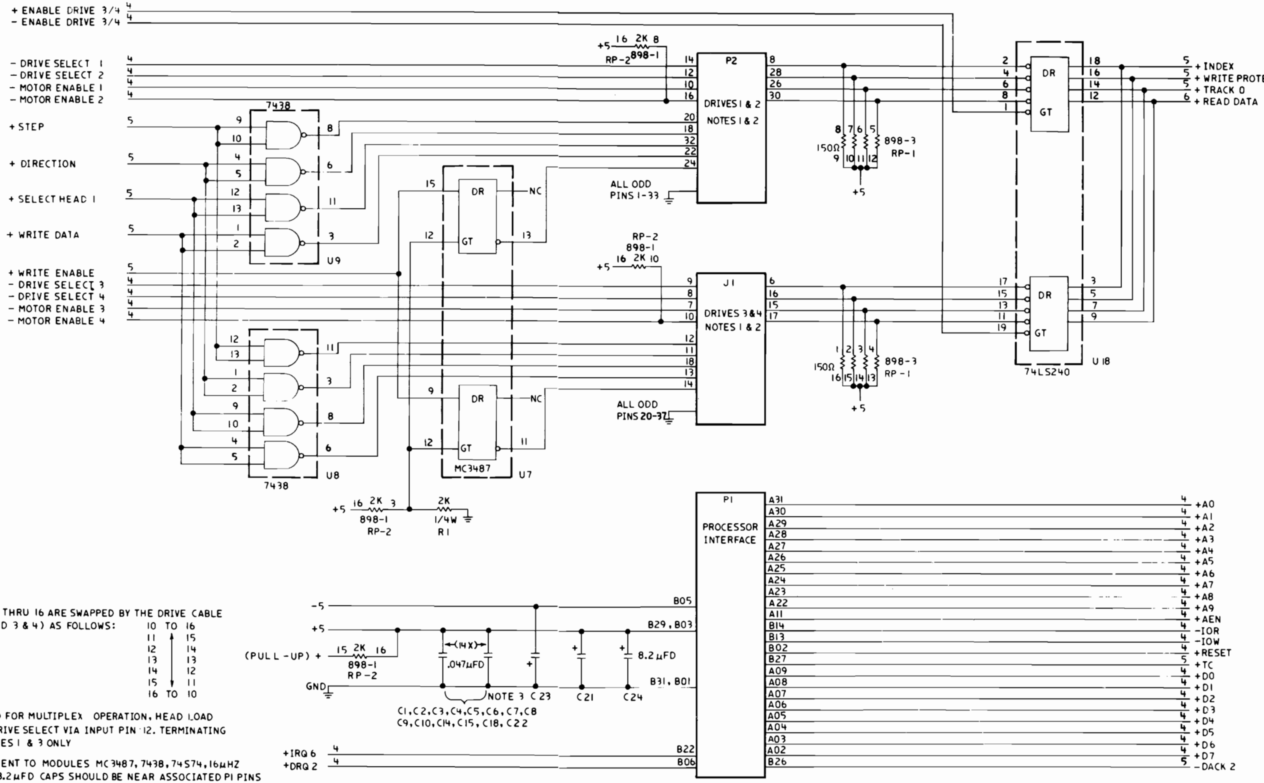


PARALLEL PRINTER ADAPTER



5¼" DISKETTE DRIVE ADAPTER

Note: Logics one and two of six are not applicable.



NOTES:

1. SIGNALS ON DRIVE PINS 10 THRU 16 ARE SWAPPED BY THE DRIVE CABLE BETWEEN DRIVES 1 & 2 (AND 3 & 4) AS FOLLOWS:

10	TO 16
11	15
12	14
13	13
14	12
15	11
16	TO 10

2. ALL DRIVES ARE JUMPERED FOR MULTIPLEX OPERATION, HEAD LOAD WITH DRIVE SELECT AND DRIVE SELECT VIA INPUT PIN 12. TERMINATING R-PACS ARE LEFT IN DRIVES 1 & 3 ONLY

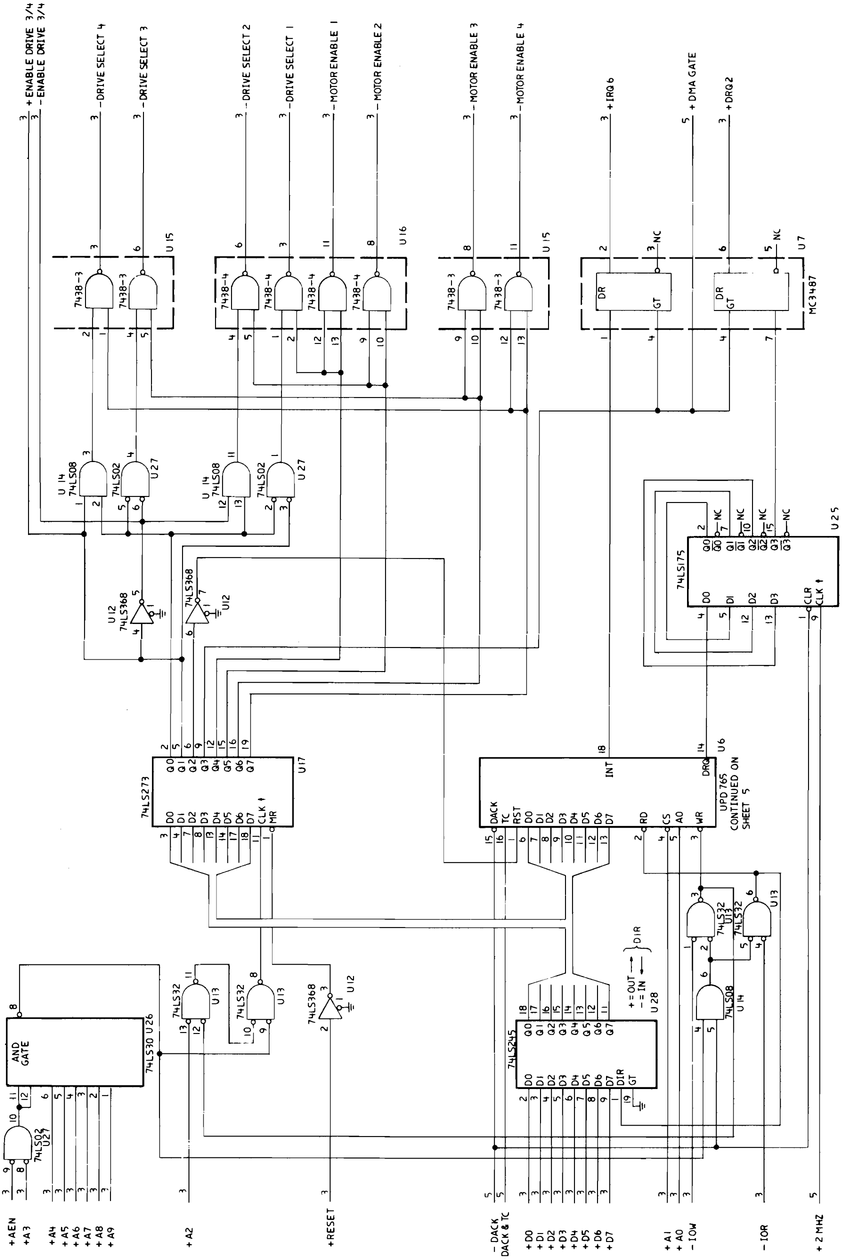
3. .047µFD SHOULD BE ADJACENT TO MODULES MC3487, 7438, 74S74, 16µHZ OSC, VCO AND CHG PUMP. 8.2µFD CAPS SHOULD BE NEAR ASSOCIATED P1 PINS

$\boxed{4}$ ALL SIGNAL LINES HIGHER THAN OR EQUAL TO 1MHZ SHOULD BE KEPT TO THE SHORTEST POSSIBLE LENGTH. THIS IS A PRIMARY DESIGN GOAL

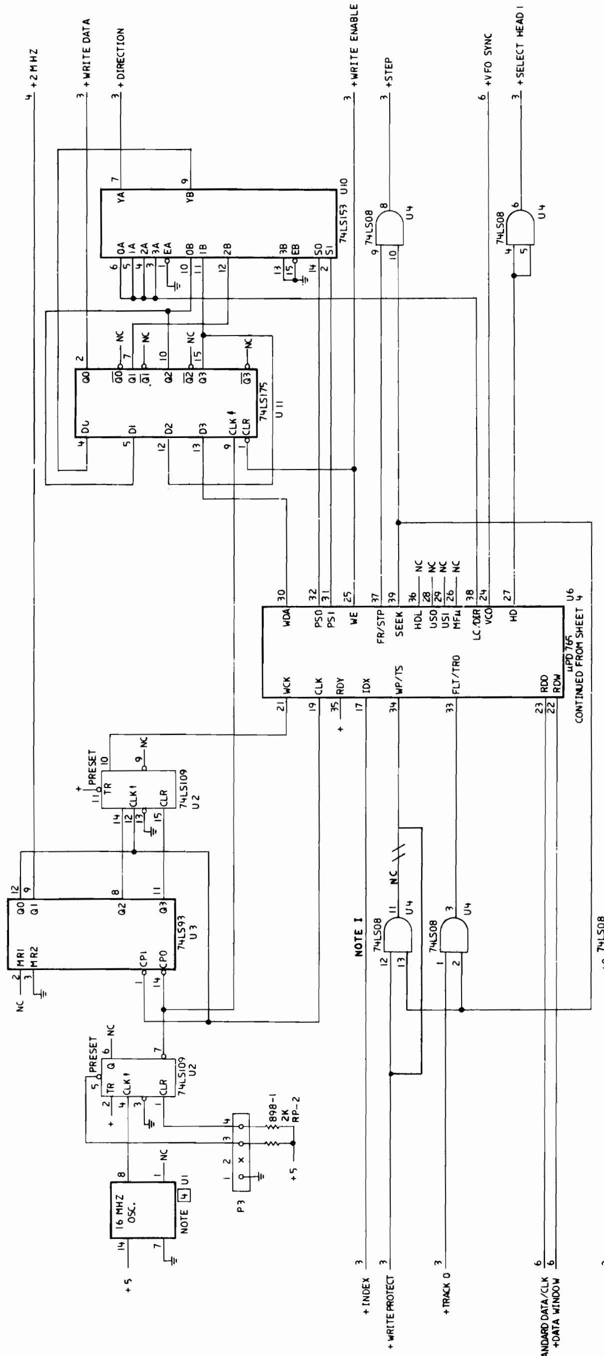
5. MAKE NO CONNECTION TO UNUSED PINS ON THE VCO, CHARGE PUMP & DATA SEPARATOR MODULES

6. ALL VOLTAGE AND GROUND CONNECTIONS TO THE VCO, CHARGE PUMP AND ASSOCIATED DISCRETE COMPONENTS SHOULD BE SEPARATE FROM OTHER CIRCUITS AND THEN JOINED TO THE OTHER CIRCUITS AT ONE POINT

5¼" Diskette Drive Adapter Logic 3 of 6

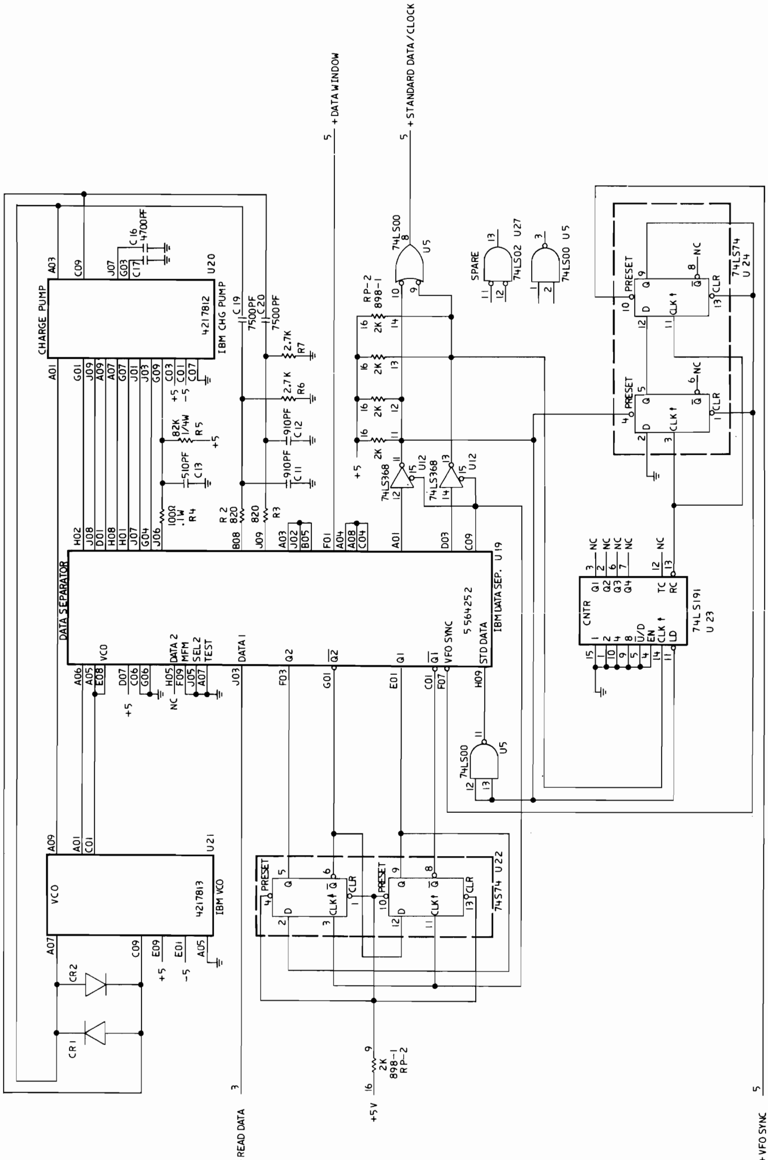


5¼" Diskette Drive Adapter Logic 4 of 6

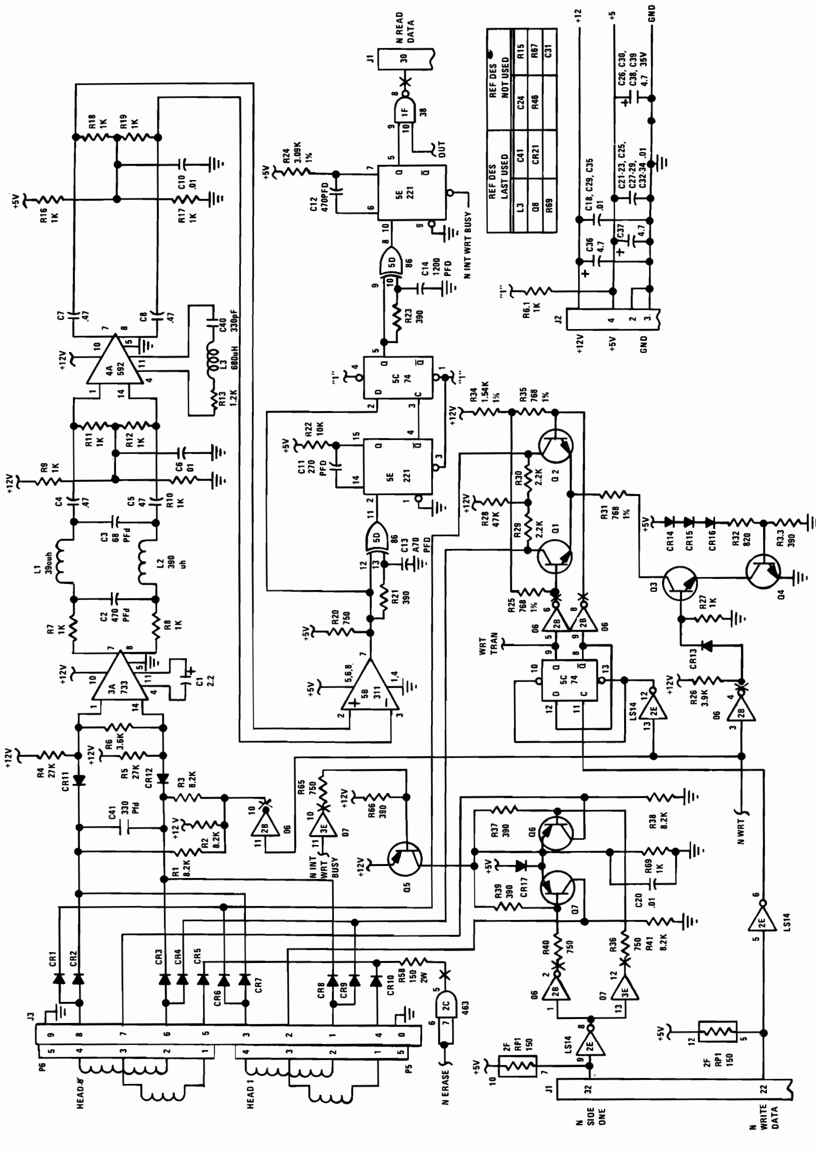


NOTE: U4 (74LS08) PINS 12 AND 13 ARE CONNECTED ONLY ON CARDS BUILT USING RAW CARD P/N 5001293

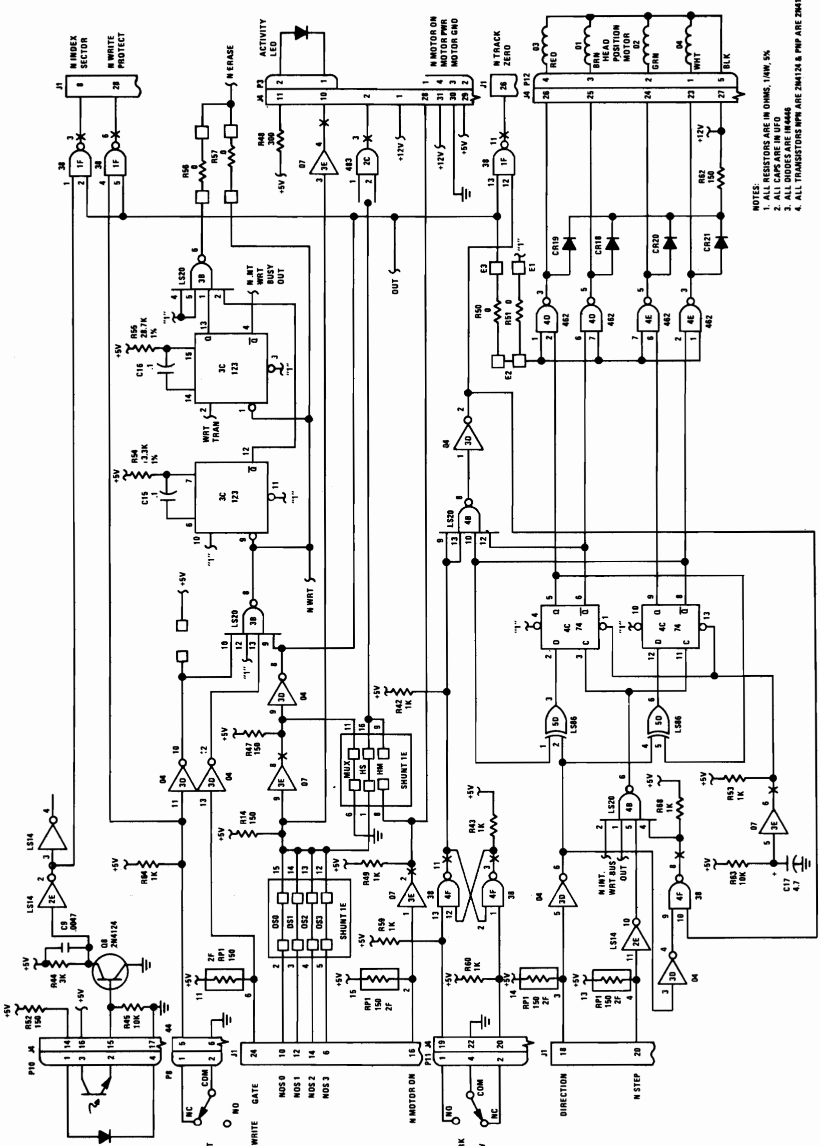
5¼" Diskette Drive Adapter Logic 5 of 6



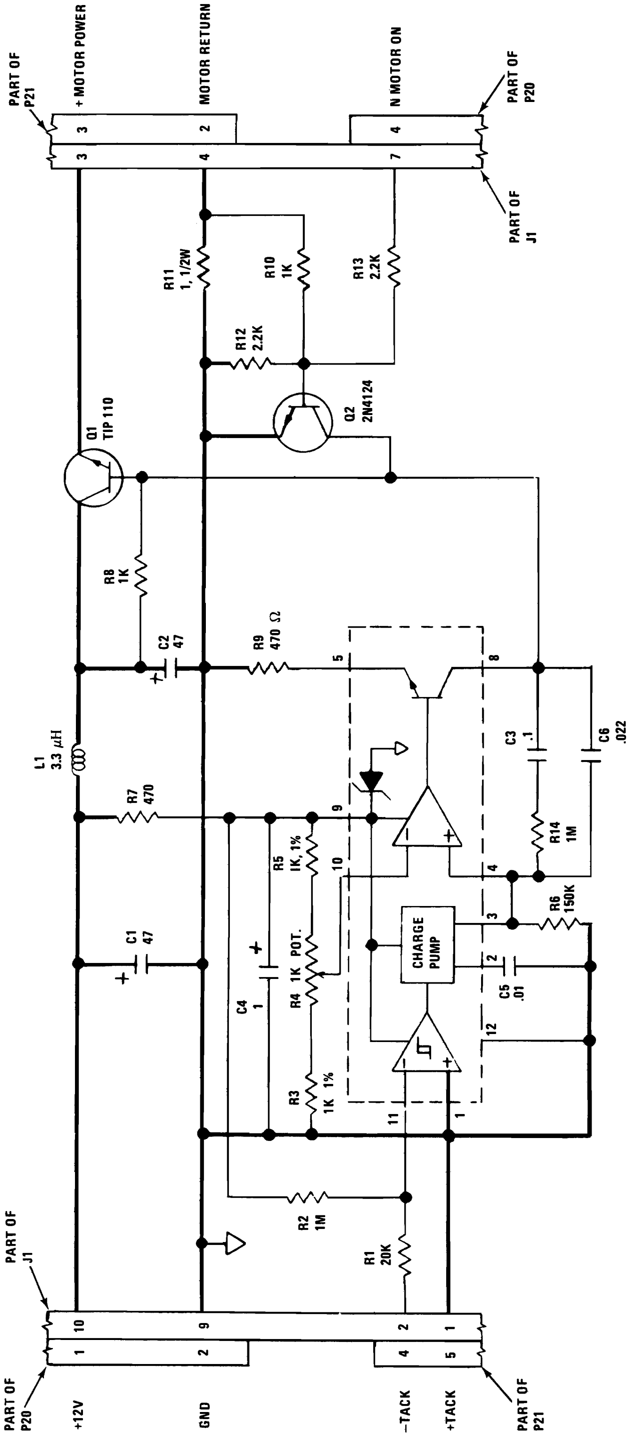
5¼" Diskette Drive Adapter Logic 6 of 6



5¼" Diskette Drive Logic 1 of 3

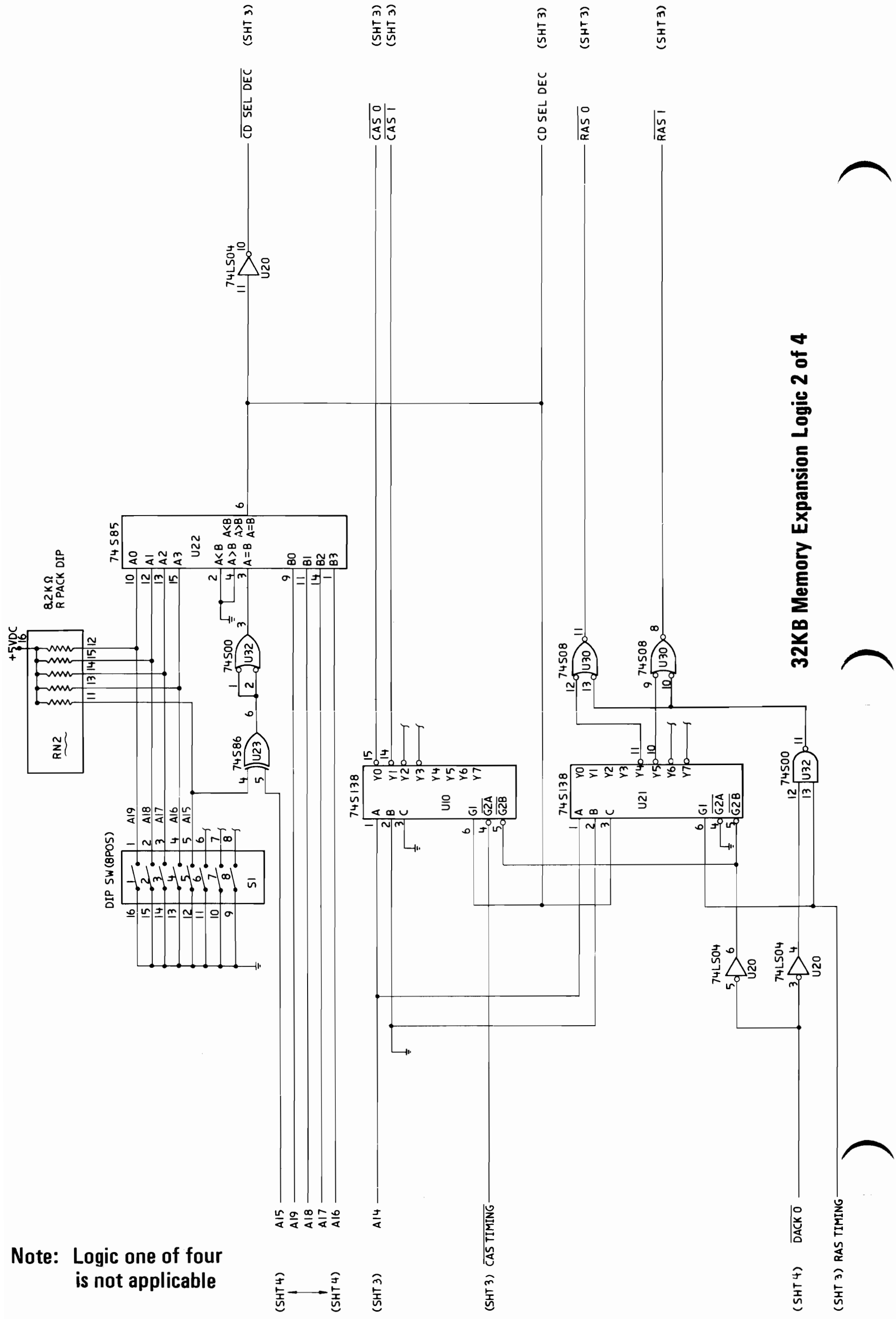


5¼" Diskette Drive Logic 2 of 3



NOTES:
1. RESISTORS ARE IN OHMS, ± 5%, 1/4 W.
2. 1% RESISTORS ARE 1/8 W.
3. CAPACITORS ARE IN μ F, ± 20%, 35 V.

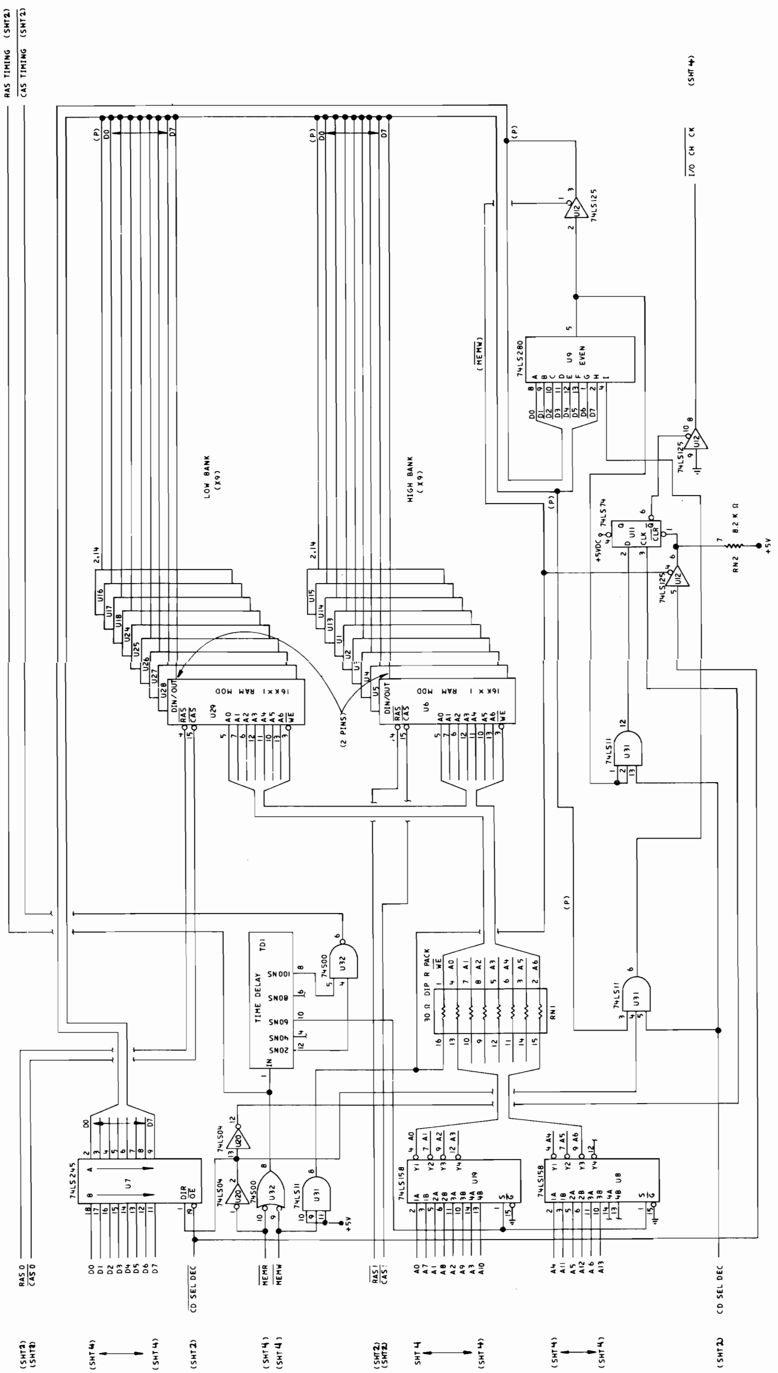
5¼" Diskette Drive Logic 3 of 3



32KB Memory Expansion Logic 2 of 4

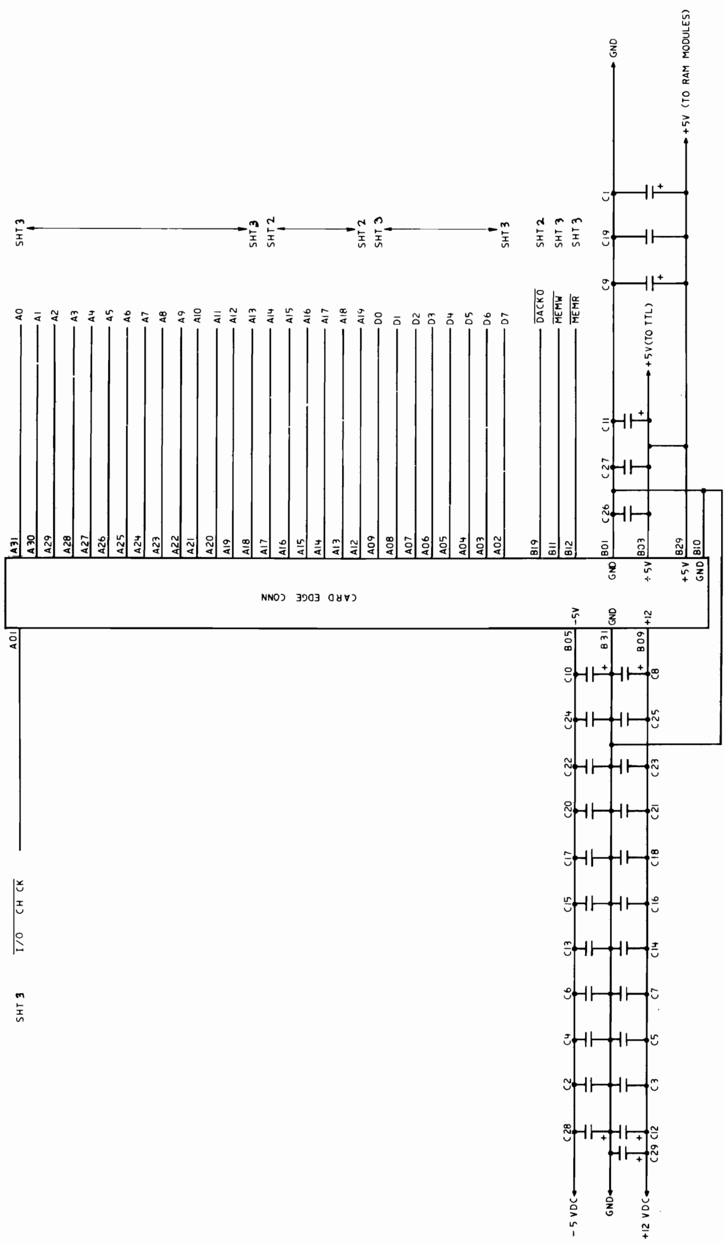
Note: Logic one of four is not applicable

32 KB MEMORY EXPANSION

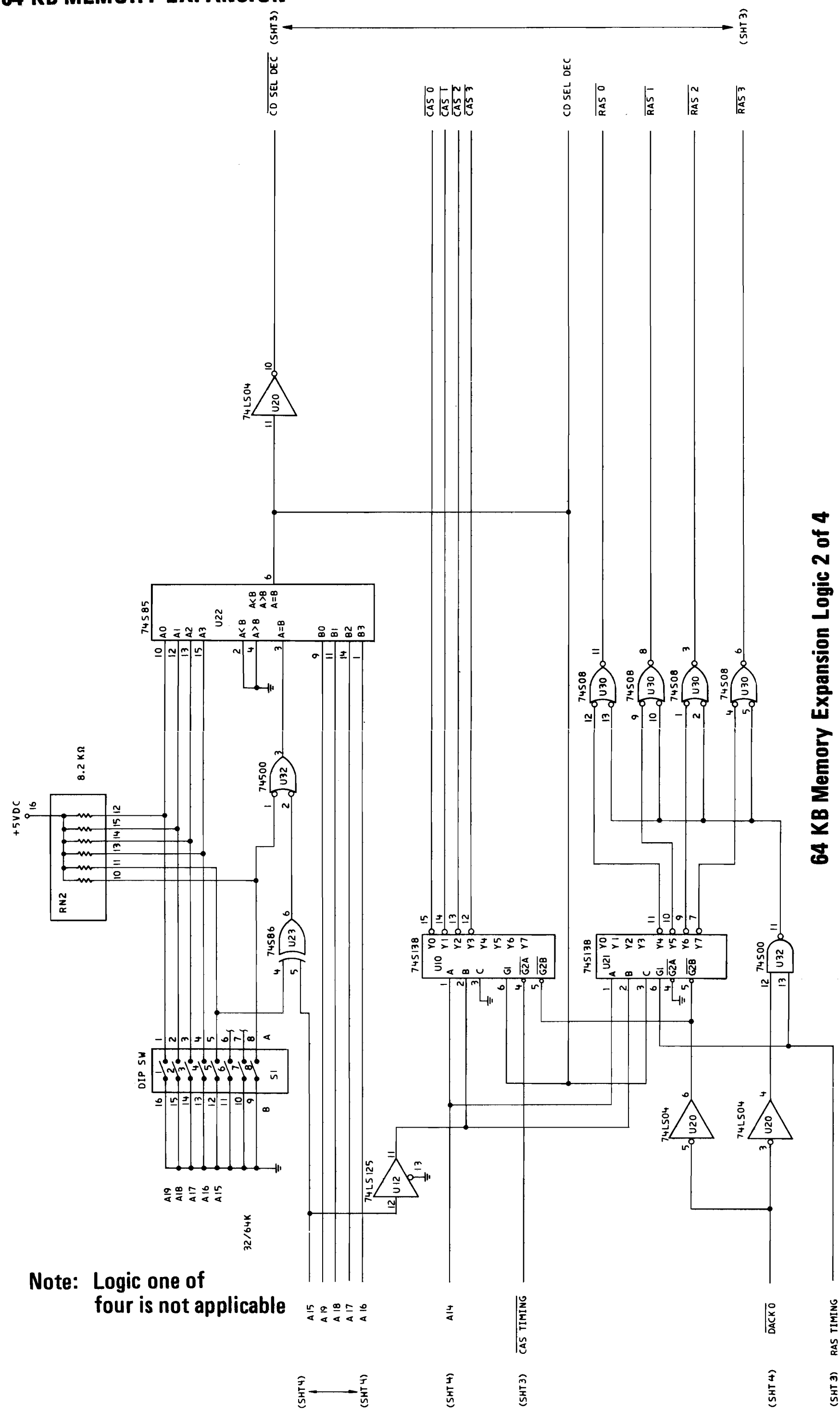


32KB Memory Expansion Logic 3 of 4

APPENDIX D



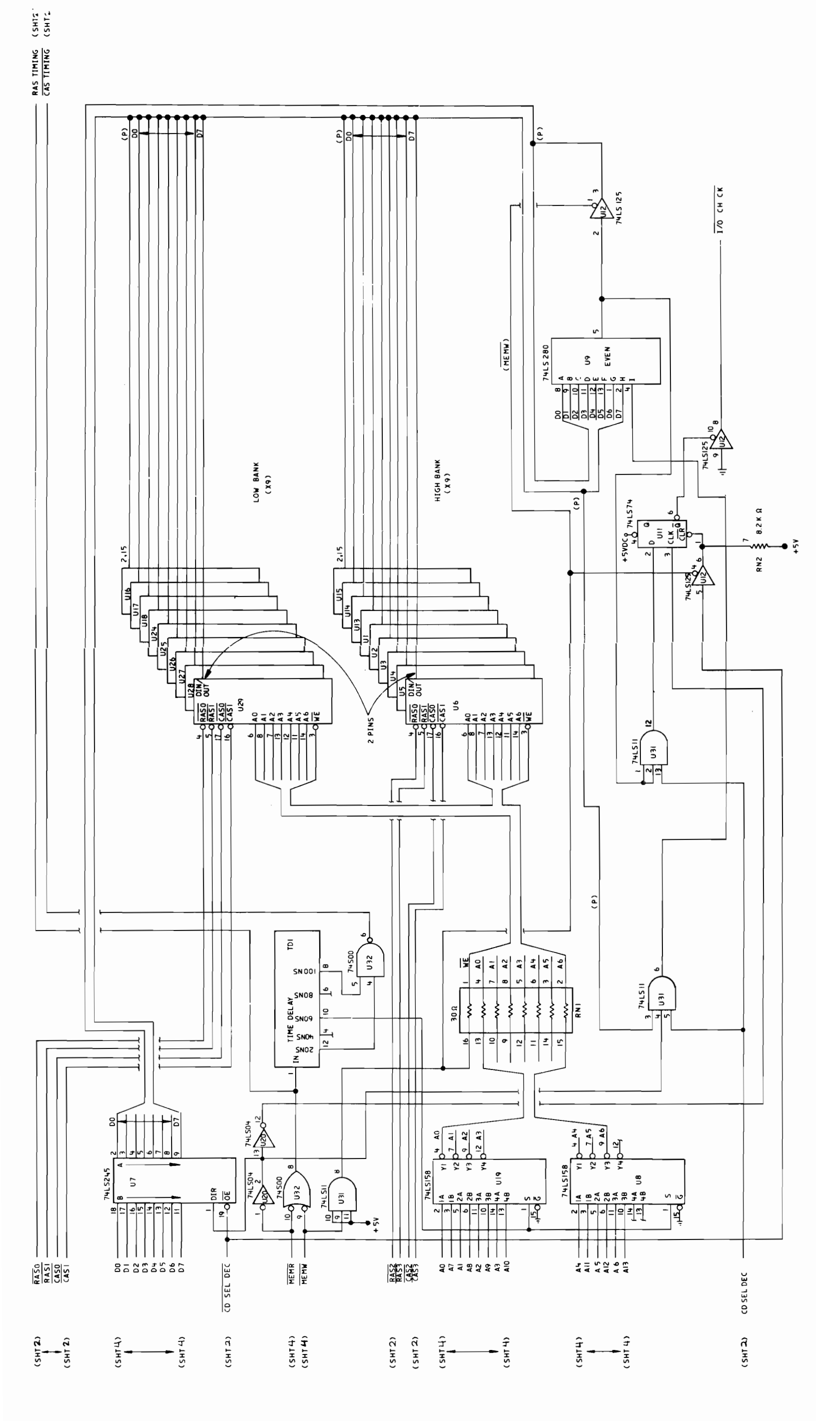
32KB Memory Expansion Logic 4 of 4



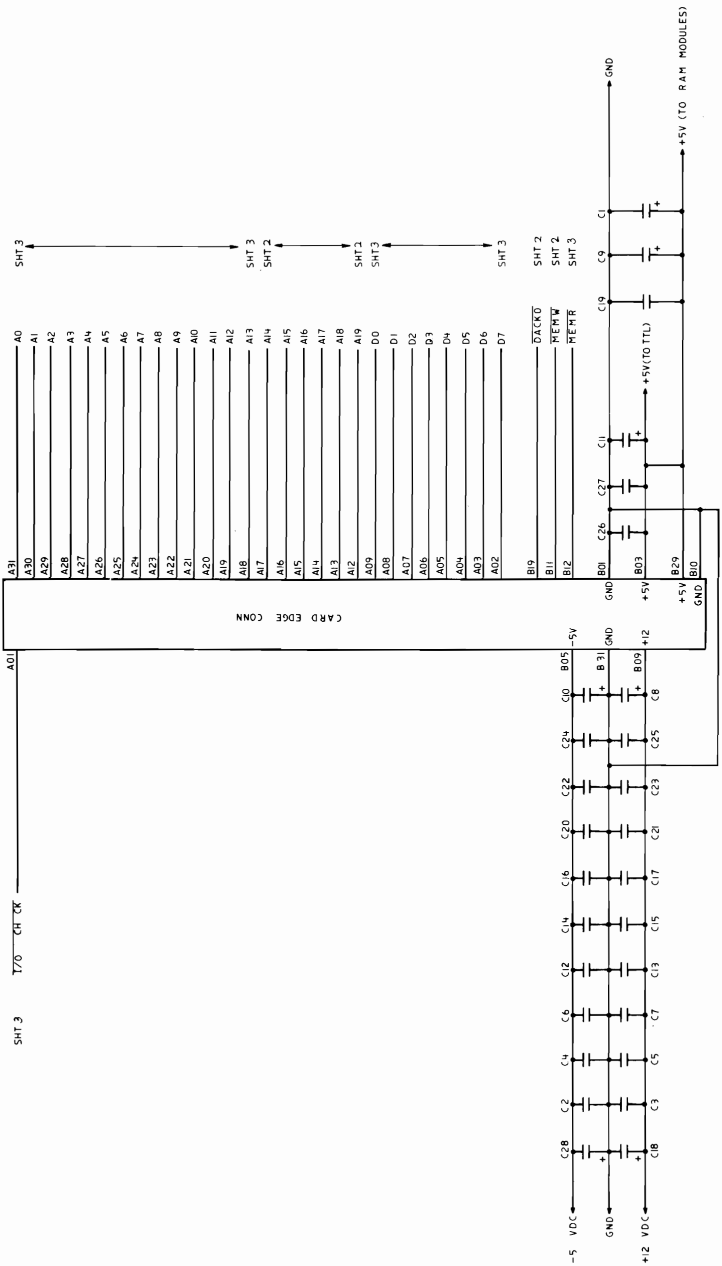
Note: Logic one of four is not applicable

64 KB Memory Expansion Logic 2 of 4

APPENDIX D



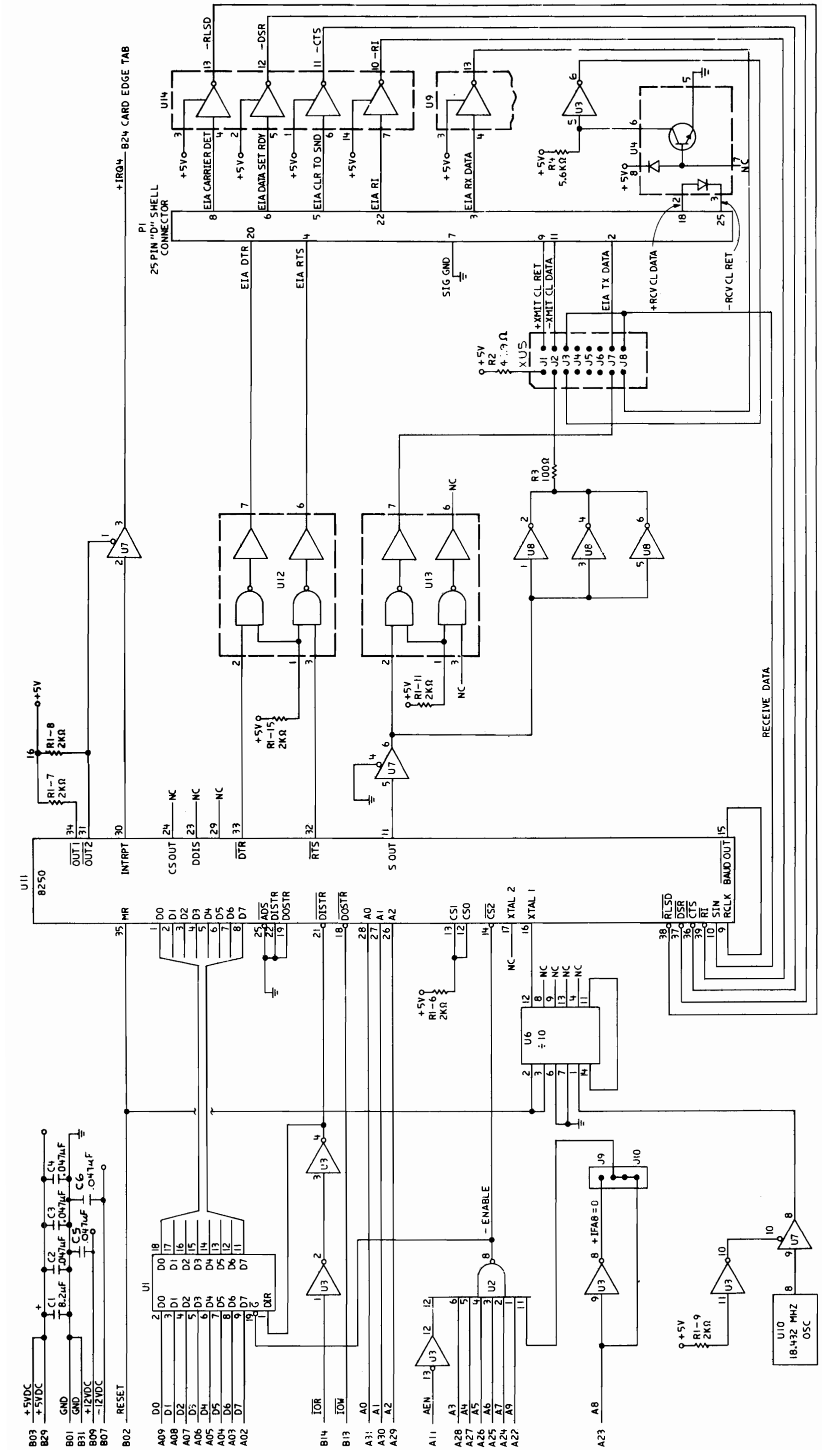
64KB Memory Expansion Logic 3 of 4



64 KB Memory Expansion Logic 4 of 4

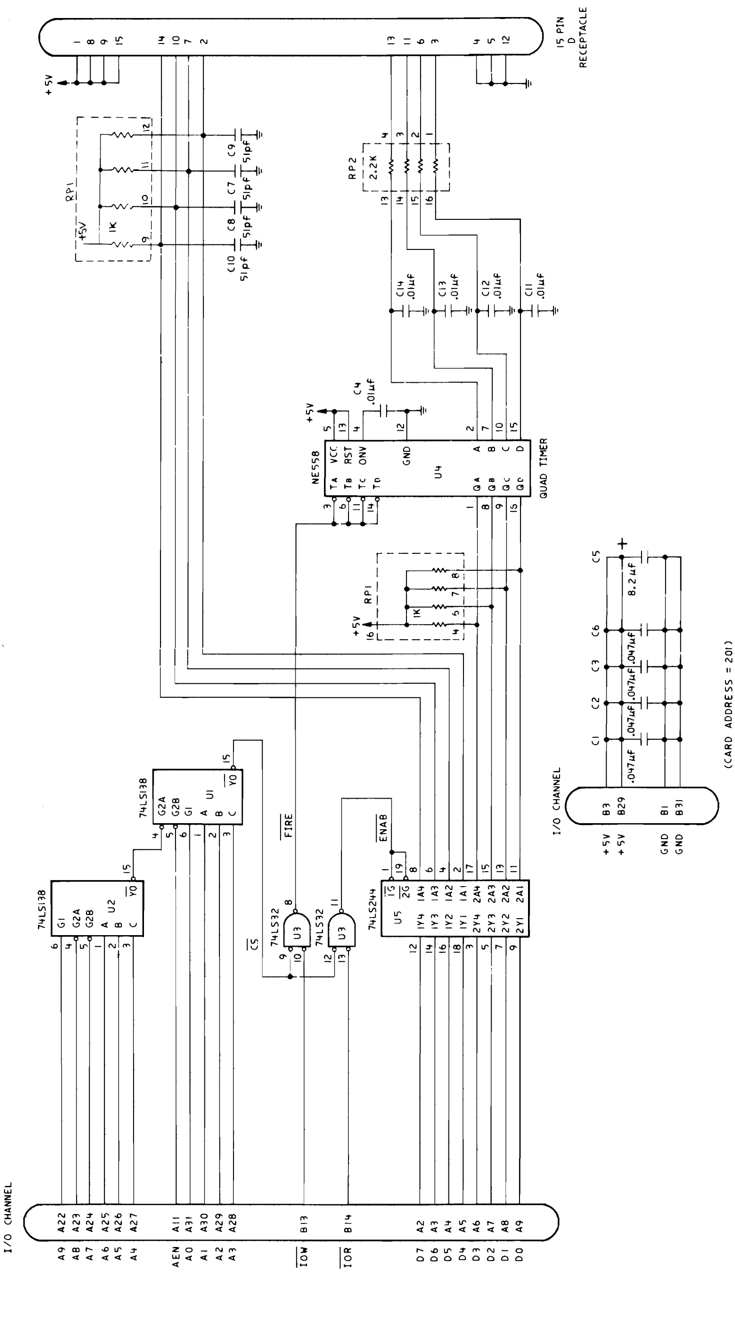
APPENDIX D

ASYNCHRONOUS COMMUNICATIONS ADAPTER



Asynchronous Communications Adapter

GAME CONTROL ADAPTER



NOTES

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

GLOSSARY

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. Syonymous with alphameric.
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 repetoire 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 it's 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 inseperably 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: Pertaininng 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.

NOTES

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

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

Numerics

NOTES

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 anyway 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 ______________

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 123 BOCA RATON, FLORIDA 33432

POSTAGE WILL BE PAID BY ADDRESSEE

IBM PERSONAL COMPUTER
SALES & SERVICE
P.O. BOX 1328-C
BOCA RATON, FLORIDA 33432

Fold here

Tape Please do not staple Tape