Bit 5 This bit is the command parameter buffer full (CPBF) bit and indicates that the parameter register contains a parameter. It is set when the processor deposits a parameter in the parameter register, and reset when the 8273 accepts the parameter.

Bit 6 This bit is the command buffer full (CBF) bit and, when set, it indicates that a byte is present in the command register. This bit is normally not used.

Bit 7 This bit is the command busy (CBSY) bit and indicates when the 8273 is in the command phase. It is set when the processor writes a command into the command register, starting the command phase. It is reset when the last parameter is deposited in the parameter register and accepted by the 8273, completing the command phase.

Initializing the Adapter (Typical Sequence)

Before initialization of the 8273 protocol controller, the support devices on the card must be initialized to the proper modes of operation.

Configuration of the 8255A-5 programmable peripheral interface is accomplished by selecting the mode-set address for the 8255 (see the "SDLC Communications Adapter Device Addresses" table later in this section) and writing the appropriate control word to the device (hex 98) to set ports A, B, and C to the modes described previously in this section.

Next, a bit pattern is output to port C which disallows interrupts, sets wrap mode on, and gates the external clock pins (address = hex 382, data = hex 0D). The adapter is now isolated from the communications interface.

Using bit 4 of port B, the 8273 reset line is brought high, held and then dropped. This resets the internal registers of the 8273.

The 8253-5's counter 1 and 2 terminal-count values are now set to values which will provide the desired time delay before a level 4 interrupt is generated. These interrupts may be used to indicate to the communication software that a pre-determined period of time has elapsed without a result interrupt (interrupt level 3). The terminal count-values for these counters are set for any time delay which the programmer requires. Counter 0 is also set at this time to mode 3 (generates square wave signal, used to drive counter 2 input).

To setup the counter modes, the address for the 8253 counter mode register is selected (see the "SDLC Communications Adapter Device Addresses" table, later in this section), and the control word for each individual counter is written to the device separately. The control-word format and bit definitions for the 8253 are shown below. Note that the two most-significant bits of the control word select each individual counter, and each counter mode is defined separately.

Once the support devices have been initialized to the proper modes and the 8273 has been reset, the 8273 protocol controller is ready to be configured for the operating mode that defines the communications environment in which it will be used.

Control Word Format

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SC1	SC0	RL1	RL0	M2	M1	M0	BCD

Definitions of Control

SC - Select Counter:

SC1	SC0	
0	0	Select Counter 0
0	1	Select Counter 1
1	0	Select Counter 2
1	1	Illegal

RL - Read/Load:

RL1	RL0	
0	0	Counter Latching operation
1	0	Read/Load most significant byte (MSB)
0	1	Read/Load least significant byte (LSB)
1	1	Read/Load least significant byte first, then most significant byte.

M - Mode:

M2	M1	M0	Mode
0	0	0	Mode 0
0	0	1	Mode 1
X	1	0	Mode 2
X	1	1	Mode 3
1	0	0	Mode 4
1	0	1	Mode 5

BCD:

0	Binary Counter 16-bits
1	Binary Coded Decimal (BCD) Counter (4 Decades)

8253-5 Programmable Interval Timer Control Word

Initialization/Configuration Commands

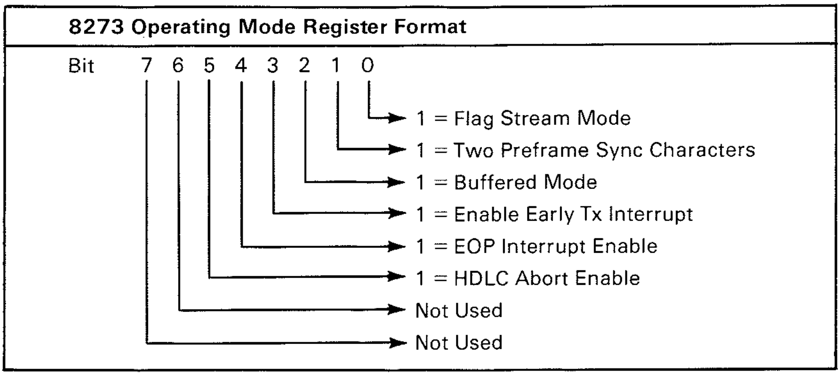
The initialization/configuration commands manipulate internal registers of the 8273, which define operating modes. After chip reset, the 8273 defaults to all 1's in the mode registers. The initialization/configuration commands either set or reset specified bits in the registers depending on the type of command. One parameter is required with the commands. The parameter is actually the bit pattern (mask) used by the set or reset command to manipulate the register bits.

Set commands perform a logical OR operation of the parameter (mask) of the internal register. This mask contains 1's where register bits are to be set. Zero (0's) in the mask cause no change to the corresponding register bit.

Reset commands perform a logical AND operation of the parameter (mask) and internal register. The mask 0 is reset to register bit, and 1 to cause no change.

The following are descriptions of each bit of the operating, serial I/O, one-bit delay, and data transfer mode registers.

Operating Mode Register


8273 Operating Mode Register Format

Bit 0

If bit 0 is set to a 1, flags are sent immediately if the transmitter was idle when the bit was set. If a transmit or transmit-transparent command was active, flags are sent immediately after transmit completion. This mode is ignored if loop transmit is active or the one-bit-delay mode register is set for one-bit delay. If bit 0 is reset (to 0), the transmitter sends idles on the next character boundary if idle or, after transmission is complete, if the transmitter was active at bit-0 reset time.

Bit 1

If bit 1 is set to a 1, the 8273 sends two characters before the first flag of a frame. These characters are hex 00 if NRZI is set or hex 55 if NRZI is not set. (See "Serial I/O Mode Register," for NRZI encoding mode format.)

Bit 2

If bit 2 is set to a 1, the 8273 buffers the first two bytes of a received frame (the bytes are not passed to memory). Resetting this bit (to 0) causes these bytes to be passed to and from memory.

Bit 3

This bit indicates to the 8273 when to generate an end-of-frame interrupt. If bit 3 is set, an early interrupt is generated when the last data character has been passed to the 8273. If the processor responds to the early interrupt with another transmit command before the final flag is sent, the final-flag interrupt will not be generated and a new frame will begin when the current frame is complete. Thus, frames may be sent separated by a single flag. A reset condition causes an interrupt to be generated only following a final flag.

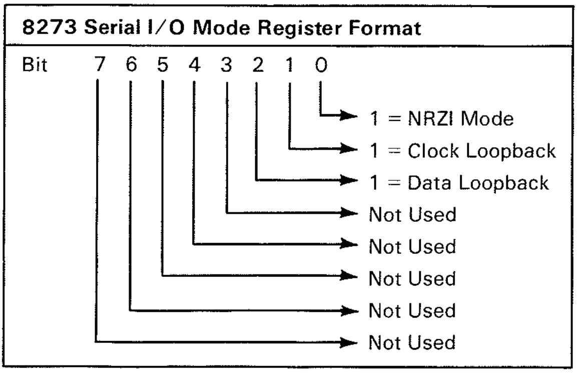
Bit 4

This is the EOP-interrupt-mode function and is not used on the SDLC communications adapter. This bit should always be in the reset condition.

Bit 5

This bit is always reset for SDLC operation, which causes the 8273 protocol controller to recognize eight ones (0 1 1 1 1 1 1 1) as an abort character.

Serial I/O Mode Register



8273 Serial I/O Mode Register Format

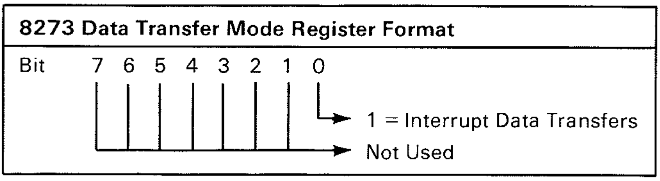
Bit 7 6 5 4 3 2 1 0

Bit 0 Set to 1, this bit specifies NRZI encoding and decoding. Resetting this bit specifies that transmit and receive data be treated as a normal positive-logic bit stream.

Bit 1 When bit 1 is set to 1, the transmit clock is internally routed to the receive-clock circuitry. It is normally used with the loopback bit (bit 2). The reset condition causes the transmit and receive clocks to be routed to their respective 8273 I/O pins.

Bit 2 When bit 2 is set, the transmitted data is internally routed to the received data circuitry. The reset condition causes the transmitted and received data to be routed to their respective 8273 I/O pins.

Data Transfer Mode Register

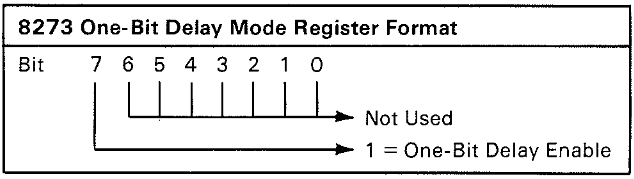


8273 Data Transfer Mode Register Format

Bit 7 6 5 4 3 2 1 0

When the data transfer mode register is set, the 8273 protocol controller will interrupt when data bytes are required for transmission, or are available from a reception. If a transmit or receive interrupt occurs and the status register indicates that there is no transmit or receive interrupt result, the interrupt is a transmit or receive data request, respectively. Reset of this register causes DMA requests to be performed with no interrupts to the processor.

One-Bit Delay Mode Register



8273 One-Bit Delay Mode Register Format

When one-bit delay is set, the 8273 retransmits the received data stream one-bit delayed. Reset of this bit stops the one-bit delay mode.

The table below is a summary of all set and reset commands associated with the 8273 mode registers. The set or reset mask used to define individual bits is treated as a single parameter. No result or interrupt is generated by the 8273 after execution of these commands.

Register	Command	Hex Code	Parameter
One-Bit Delay Mode	Set	A4	Set Mask
	Reset	64	Reset Mask
Data Transfer Mode	Set	97	Set Mask
	Reset	57	Reset Mask
Operating Mode	Set	91	Set Mask
	Reset	51	Reset Mask
Serial I/O Mode	Set	A0	Set Mask
	Reset	60	Reset Mask

8273 SDLC Protocol Controller Mode Register Commands

Command Phase

Although the 8273 is a full duplex device, there is only one command register. Thus, the command register must be used for only one command sequence at a time and the transmitter and receiver may never be simultaneously in a command phase.

The system software starts the command phase by selecting the 8273 command register address and writing a command byte into the register. The following table lists command and parameter information for the 8273 protocol controller. If further information is required by the 8273 prior to execution of the command, the system software must write this information into the parameter register.

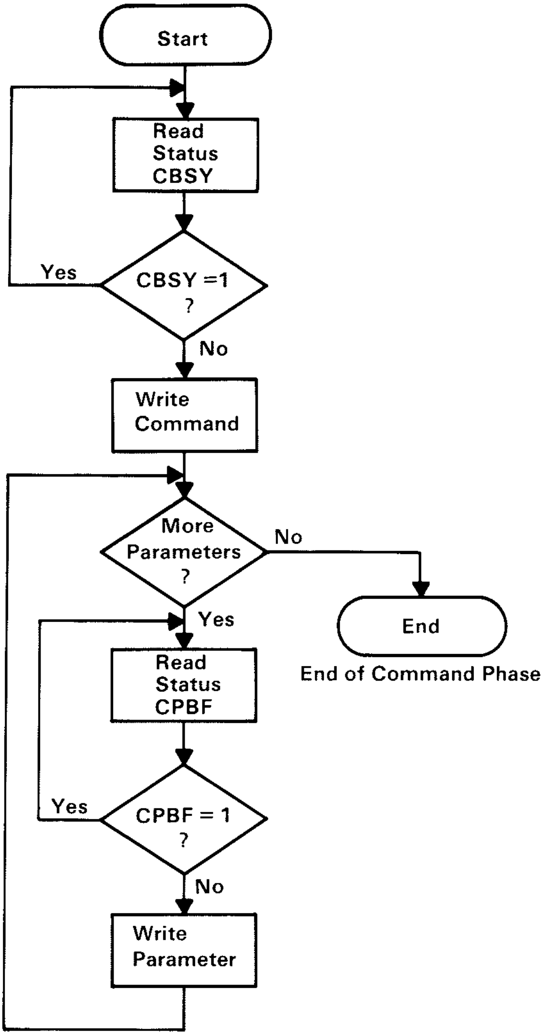
Command Description	Command (Hex)	Parameter	Results	Result Port	Completion Interrupt
Set One-Bit Delay	A4	Set Mask	None	—	No
Reset One-Bit Delay	64	Reset Mask	None	—	No
Set Data Transfer Mode	97	Set Mask	None	—	No
Reset Data Transfer Mode	57	Reset Mask	None	—	No
Set Operating Mode	91	Set Mask	None	—	No
Reset Operating Mode	51	Reset Mask	None	—	No
Set Serial I/O Mode	A0	Set Mask	None	—	No
Reset Serial I/O Mode	60	Reset Mask	None	—	No
General Receive	C0	80,81	RIC,R0,R1, A,C	RXI/R	Yes
Selective Receive	C1	80,81,A1, A2	RIC,R0,R1, A,C	RXI/R	Yes
Receive Disable	C5	None	None	—	No
Transmit Frame	C8	L0,L1,A,C	TIC	TXI/R	Yes
Transmit Transparent	C9	L0,L1	TIC	TXI/R	Yes
Abort Transmit Frame	CC	None	TIC	TXI/R	Yes
Abort Transmit Transparent	CD	None	TIC	TXI/R	Yes
Read Port A	22	None	Port Value	Result	No
Read Port B	23	None	Port Value	Result	No
Set Port B Bit	A3	Set Mask	None	—	No
Reset Port B Bit	63	Reset Mask	None	—	No

8273 Command Summary Key

B0 — Least significant byte of the receiver buffer length.
B1 — Most significant byte of the receiver buffer length.
L0 — Least significant byte of the Tx frame length.
L1 — Most significant byte of the Tx frame length.
A1 — Receive frame address match field one.
A2 — Receive frame address match field two.
A — Address field of received frame. If non-buffered mode is specified, this result is not provided.
C — Control field of received frame. If non-buffered mode is specified, this result is not provided.
RXI/R — Receive interrupt result register.
TXI/R — Transmit interrupt result register.
R0 — Least significant byte of the length of the frame received.
R1 — Most significant byte of the length of the frame received.
RIC — Receiver interrupt result code.
TIC — Transmitter interrupt result code.

8273 SDLC Protocol Controller Commands

A flowchart of the command phase is shown below. Handshaking of the command and parameter bytes is accomplished by the CBSY and CPBF bits of the status register. A command may not be written if the 8273 is busy (CBSY = 1). The original command will be overwritten if a second command is issued while CBSY = 1. The flowchart also indicates a parameter buffer full check. The processor must wait until CPBF = 0 before writing a parameter to the parameter register. Previous parameters are overwritten and lost if a parameter is written while CPBF = 1.



8273 SDLC Protocol Controller Command Phase Flowchart

Execution Phase

During the execution phase, the operation specified by the command phase is performed. If DMA is utilized for data transfers, no processor involvement is required.

For interrupt-driven transfers the 8273 raises the appropriate INT pin (TxINT or RxINT). When the processor responds to the interrupt, it must determine the cause by examining the status register and the associated IRA (interrupt result available) bit of the status register. If IRA = 0, the interrupt is a data transfer request. If IRA = 1, an operation is complete and the associated interrupt result register must be read to determine completion status.

Result Phase

During the result phase, the 8273 notifies the processor of the outcome of a command execution. This phase is initiated by either a successful completion or error detection during execution.

Some commands such as reading or writing the I/O ports provide immediate results. These results are made available to the processor in the 8273 result register. Presence of a valid immediate result is indicated by the CRBF (command result buffer full) bit of the status register.

Non-immediate results deal with the transmitter and receiver. These results are provided in the TxI/R (transmit interrupt result) or RxI/R (receiver interrupt result) registers, respectively. The 8273 notifies the processor that a result is available with the TxIRA and RxIRA bits of the status register. Results consist of one-byte result interrupt code indicating the condition for the interrupt and, if required, one or more bytes supplying additional information. The “Result Code Summary” table later in this section provides information on the format and decode of the transmitter and receiver results.

The following are typical frame transmit and receive sequences. These examples assume DMA is utilized for data transfer operations.

Transmit

Before a frame can be transmitted, the DMA controller is supplied, by the communication software, the starting address for the desired information field. The 8273 is then commanded to transmit a frame (by issuing a transmit frame command).

After a command, but before transmission begins, the 8273 needs some more information (parameters). Four parameters are required for the transmit frame command; the frame address field byte, the frame control field byte, and two bytes which are the least significant and most significant bytes of the information field byte length. Once all four parameters are loaded, the 8273 makes RTS (request to send) active and waits for CTS (clear to send) to go active from the modem interface. Once CTS is active, the 8273 starts the frame transmission. While the 8273 is transmitting the opening flag, address field, and control field, it starts making transmitter DMA requests. These requests continue at character (byte) boundaries until the pre-loaded number of bytes of information field have been transmitted. At this point, the requests stop, the FCS (frame check sequence) and closing flag are transmitted, and the TxINT line is raised, signaling the processor the frame transmission is complete and the result should be read. Note that after the initial command and parameter loading, no processor intervention was required (since DMA is used for data transfers) until the entire frame was transmitted.

General Receive

Receiver operation is very similar. Like the initial transmit sequence, the processor's DMA controller is loaded with a starting address for a receive data buffer and the 8273 is commanded to receive. Unlike the transmitter, there are two different receive commands; a general receive, where all received frames are transferred to memory, and selective receive, where only frames having an address field matching one of two preprogrammed 8273 address fields are transferred to memory.

(This example covers a general receive operation.) After the receive command, two parameters are required before the receiver becomes active; the least significant and most significant bytes of the receiver buffer length. Once these bytes are loaded, the receiver is active and the processor may return to other tasks. The next frame appearing at the receiver input is transferred to memory using receiver DMA requests. When the closing flag is received, the 8273 checks the FCS and raises its RxINT line. The processor can then read the results, which indicate if the frame was error-free or not. (If the received frame had been longer than the pre-loaded buffer length, the processor would have been notified of that occurrence earlier with a receiver error interrupt. Like the transmit example, after the initial command, the processor is free for other tasks until a frame is completely received.

Selective Receive

In selective receive, two parameters (A1 and A2) are required in addition to those for general receive. These parameters are two address match bytes. When commanded to selective receive, the 8273 passes to memory or the processor only those frames having an address field matching either A1 or A2. This command is usually used for secondary stations with A1 designating the secondary address and A2 being the "all parties" address. If only one match byte is needed, A1 and A2 should be equal. As in general receive, the 8273 counts the incoming data bytes and interrupts the processor if the received frame is larger than the preset receive buffer length.

Result Code Summary

	Hex Code	Result	Status After Interrupt
Transmit	0C	Early Transmit Interrupt	Transmitter Active
	0D	Frame Transmit Complete	Idle or Flags
	0E	DMA Underrun	Abort
	0F	Clear to Send Error	Abort
	10	Abort Complete	Idle or Flags
Receive	X0	A1 Match or General Receive	Active
	X1	A2 Match	Active
	03	CRC Error	Active
	04	Abort Detected	Active
	05	Idle Detected	Disabled
	06	EOP Detected	Disabled
	07	Frame Less Than 32 Bits	Active
	08	DMA Overrun	Disabled
	09	Memory Buffer Overflow	Disabled
	0A	Carrier Detect Failure	Disabled
	0B	Receiver Interrupt Overrun	Disabled

Note: X decodes to number of bits in partial byte received.

The first two codes in the receive result code table result from the error free reception of a frame. Since SDLC allows frames of arbitrary length (>32 bits), the high order bits of the receive result report the number of valid received bits in the last received information field byte. The chart below shows the decode of this receive result bit.

X	Bits Received in Last Byte
E	All Eight Bits of Last Byte
0	Bit0 Only
8	Bit1-Bit0
4	Bit2-Bit0
C	Bit3-Bit0
2	Bit4-Bit0
A	Bit5-Bit0
6	Bit6-Bit0

Address and Interrupt Information

The following tables provide address and interrupt information for the SDLC adapter:

Hex Code	Device	Register Name	Function
380	8255	Port A Data	Internal/External Sensing
381	8255	Port B Data	External Modem Interface
382	8255	Port C Data	Internal Control
383	8255	Mode Set	8255 Mode Initialization
384	8253	Counter 0 LSB	Square Wave Generator
384	8253	Counter 0 MSB	Square Wave Generator
385	8253	Counter 1 LSB	Inactivity Time-Outs
385	8253	Counter 1 MSB	Inactivity Time-Outs
386	8253	Counter 2 LSB	Inactivity Time-Outs
386	8253	Counter 2 MSB	Inactivity Time-Outs
387	8253	Mode Register	8253 Mode Set
388	8273	Command/Status	Out=Command In=Status
389	8273	Parameter/Result	Out=Parameter In=Status
38A	8273	Transmit INT Status	DMA/INT
38B	8273	Receive INT Status	DMA/INT
38C	8273	Data	DPC (Direct Program Control)

SDLC Communications Adapter Device Addresses

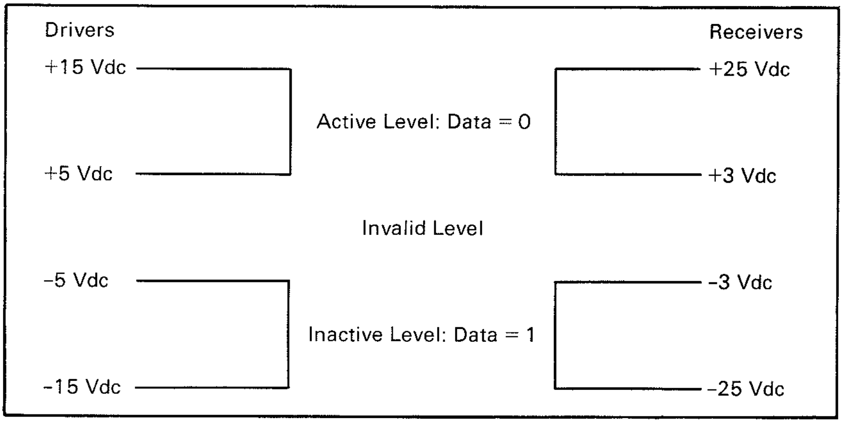
Interrupt Level 3	Transmit/Receive Interrupt
Interrupt Level 4	Timer 1 Interrupt Timer 2 Interrupt Clear to Send Changed Data Set Ready Changed
DMA Level One is used for Transmit and Receive	

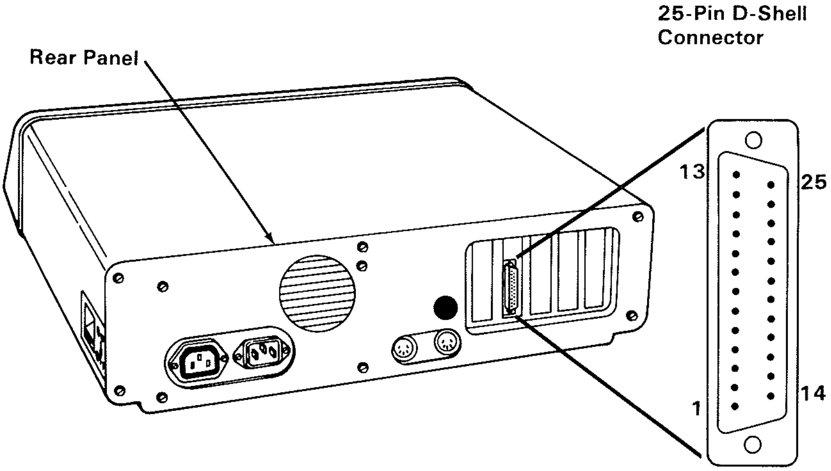
Interrupt Information

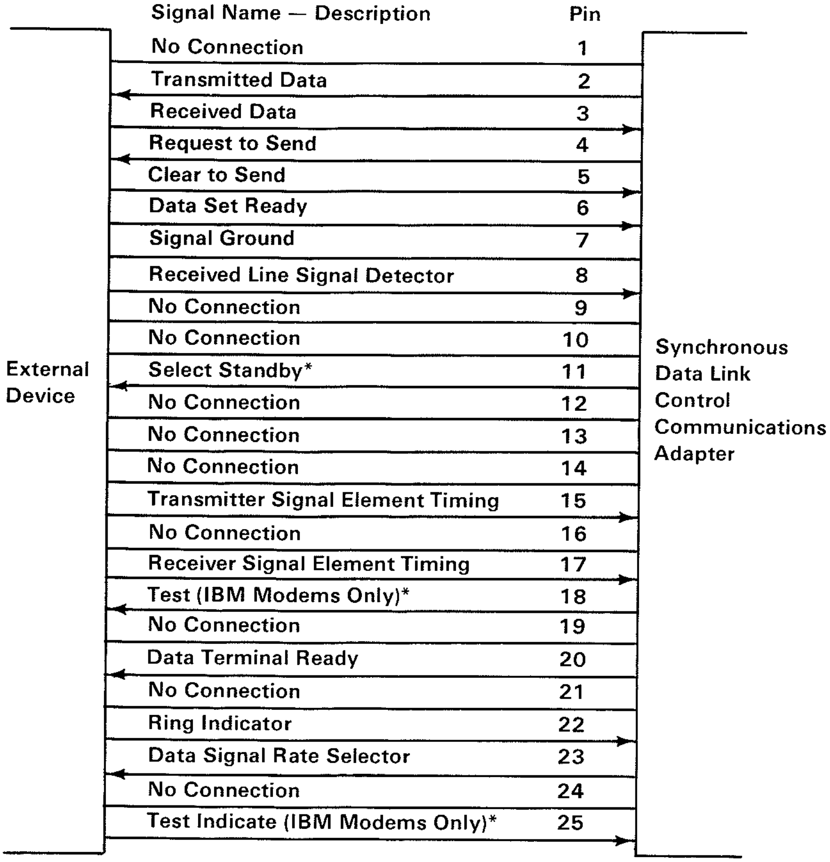
Interface Information

The SDLC communications adapter conforms to interface signal levels standardized by the Electronics Industries Association RS-232C Standard. These levels are shown in the figure below.

Additional lines used but not standardized by EIA are pins 11, 18, and 25. These lines are designated as select standby, test and test indicate, respectively. Select Standby is used to support the switched network backup facility of a modem providing this option. Test and test indicate support a modem wrap function on modems which are designed for business machine controlled modem wraps. Two jumpers on the adapter (P1 and P2) are used to connect test and test indicate to the interface, if required (see Appendix D for these jumpers).







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

*Not standardized by EIA (Electronics Industry Association).

Connector Specifications

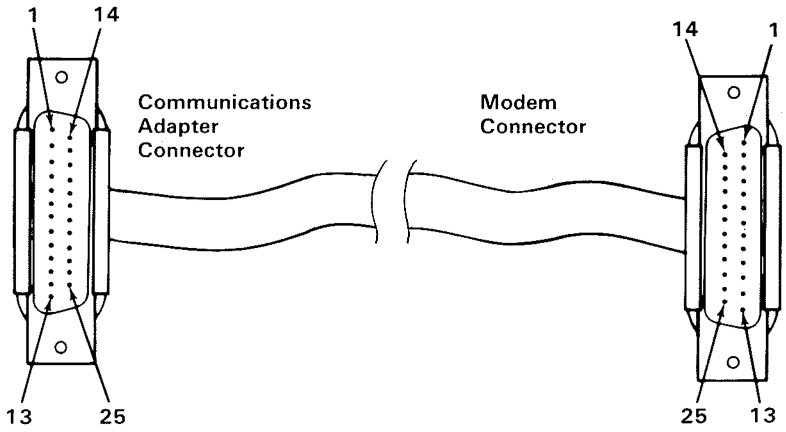
Notes:

IBM Communications Adapter Cable

The IBM Communications Adapter Cable is a ten foot cable for connection of an IBM communications adapter to a modem or other RS-232C DCE (data communications equipment). It is fully shielded and provides a high quality, low noise channel for interface between the communications adapter and DCE.

The connector ends are 25-pin D-shell connectors. All pin connections conform with the EIA RS-232C standard. In addition, connection is provided on pins 11, 18 and 25. These pins are designated as select standby, test and test indicate, respectively, on some modems. Select standby is used to support the switched network backup facility, if applicable. Test and test indicate support a modem wrap function on modems designed for business machine controlled modem wraps.

The IBM Communications Adapter Cable connects the following pins on the 25-pin D-shell connectors.



Communications Adapter Connector Pin #	Name	Modem Connector Pin #
NC	Outer Cable Shield	1
2	Transmitted Data	2
3	Received Data	3
4	Request to Send	4
5	Clear to Send	5
6	Data Set Ready	6
7	Signal Ground (Inner Lead Shields)	7
8	Received Line Signal Detector	8
NC		NC
NC		NC
11	Select Standby	11
NC		NC
NC		NC
NC		NC
15	Transmitter Signal Element Timing	15
NC		NC
17	Receiver Signal Element Timing	17
18	Test	18
NC		NC
20	Data Terminal Ready	20
NC		NC
22	Ring Indicator	22
23	Data Signal Rate Selector	23
NC		NC
25	Test Indicate	25

SECTION 2: ROM BIOS AND SYSTEM USAGE

Notes:

ROM BIOS

The basic input/output system (BIOS) resides in ROM on the system board and provides device level control for the major I/O devices in the system. Additional ROM modules may be located on option adapters to provide device level control for that option adapter. BIOS routines enable the assembly language programmer to perform block (disk and diskette) or character-level I/O operations without concern for device address and operating characteristics. System services, such as time-of-day and memory size determination, are provided by the BIOS.

The goal is to provide an operational interface to the system and relieve the programmer of the concern about the characteristics of hardware devices. The BIOS interface insulates the user from the hardware, thus allowing new devices to be added to the system, yet retaining the BIOS level interface to the device. In this manner, user programs become transparent to hardware modifications and enhancements.

The IBM Personal Computer MACRO Assembler manual and the IBM Personal Computer Disk Operating System (DOS) manual provide useful programming information related to this section. A complete listing of the BIOS is given in Appendix A.

Use of BIOS

Access to BIOS is through the 8088 software interrupts. Each BIOS entry point is available through its own interrupt, which can be found in the "8088 Software Interrupt Listing."

The software interrupts, hex 10 through hex 1A, each access a different BIOS routine. For example, to determine the amount of memory available in the system,

```
  INT   12H
```

will invoke the BIOS routine for determining memory size and will return the value to the caller.

Parameter Passing

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

If a BIOS function has several possible operations, the AH register is used at input to indicate the desired operation. For example, to set the time of day, the following code is required:

```
MOV  AH,1              ;function is to set time of day.
MOV  CX,HIGH_COUNT     ;establish the current time.
MOV  DX,LOW_COUNT
INT  1AH               ;set the time.
```

To read the time of day:

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

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

Address (Hex)	Interrupt Number	Name	BIOS Entry
0-3	0	Divide by Zero	D_EOI
4-7	1	Single Step	D_EOI
8-B	2	Nonmaskable	NMI_INT
C-F	3	Breakpoint	D_EOI
10-13	4	Overflow	D_EOI
14-17	5	Print Screen	PRINT_SCREEN
18-1B	6	Reserved	D_EOI
1D-1F	7	Reserved	D_EOI
20-23	8	Time of Day	TIMER_INT
24-27	9	Keyboard	KB_INT
28-2B	A	Reserved	D_EOI
2C-2F	B	Communications	D_EOI
30-33	C	Communications	D_EOI
34-37	D	Disk	D_EOI
38-3B	E	Diskette	DISK_INT
3C-3F	F	Printer	D_EOI
40-43	10	Video	VIDEO_IO
44-47	11	Equipment Check	EQUIPMENT
48-4B	12	Memory	MEMORY_SIZE_DETERMINE
4C-4F	13	Diskette/Disk	DISKETTE_IO
50-53	14	Communications	RS232_IO
54-57	15	Cassette	CASSETTE_IO
58-5B	16	Keyboard	KEYBOARD_IO
5C-5F	17	Printer	PRINTER_IO
60-63	18	Resident BASIC	F600:0000
64-67	19	Bootstrap	BOOT_STRAP
68-6B	1A	Time of Day	TIME_OF_DAY
6C-6F	1B	Keyboard Break	DUMMY_RETURN
70-73	1C	Timer Tick	DUMMY_RETURN
74-77	1D	Video Initialization	VIDEO_PARMS
78-7B	1E	Diskette Parameters	DISK_BASE
7C-7F	1F	Video Graphics Chars	0

8088 Software Interrupt Listing

Vectors with Special Meanings

Interrupt Hex 1B – Keyboard Break Address

This vector points to the code to be exercised when the Ctrl and Break keys are pressed on the keyboard. The vector is invoked while responding to the keyboard interrupt, and control should be returned through an IRET instruction. The power-on routines initialize this vector to point to an IRET instruction, so that nothing will occur when the Ctrl and Break keys are pressed unless the application program sets a different value.

Control may be retained by this routine, with the following problems. The Break may have occurred during interrupt processing, so that one or more End of Interrupt commands must be sent to the 8259 controller. Also, all I/O devices should be reset in case an operation was underway at that time.

Interrupt Hex 1C – Timer Tick

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

Interrupt Hex 1D – Video Parameters

This vector points to a data region containing the parameters required for the initialization of the 6845 on the video card. Note that there are four separate tables, and all four must be reproduced if all modes of operation are to be supported. The power-on routines initialize this vector to point to the parameters contained in the ROM video routines.

Interrupt Hex 1E – Diskette Parameters

This vector points to a data region containing the parameters required for the diskette drive. The power-on routines initialize the vector to point to the parameters contained in the ROM diskette routine. These default parameters represent the specified values for any IBM drives attached to the machine. Changing this parameter block may be necessary to reflect the specifications of the other drives attached.

Interrupt Hex 1F – Graphics Character Extensions

When operating in the graphics modes of the IBM Color/Graphics Monitor Adapter (320 by 200 or 640 by 200), the read/write character interface will form the character from the ASCII code point, using a set of dot patterns. The dot patterns for the first 128 code points are contained in ROM. To access the second 128 code points, this vector must be established to point at a table of up to 1K bytes, where each code point is represented by eight bytes of graphic information. At power-on, this vector is initialized to 000:0, and it is the responsibility of the user to change this vector if the additional code points are required.

Interrupt Hex 40 – Reserved

When an IBM Fixed Disk Drive Adapter is installed, the BIOS routines use interrupt hex 40 to revector the diskette pointer.

Interrupt Hex 41 – Fixed Disk Parameters

This vector points to a data region containing the parameters required for the fixed disk drive. The power-on routines initialize the vector to point to the parameters contained in the ROM disk routine. These default parameters represent the specified values for any IBM Fixed Disk Drives attached to the machine. Changing this parameter block may be necessary to reflect the specifications of the other fixed disk drives attached.

Other Read/Write Memory Usage

The IBM BIOS routines use 256 bytes of memory starting at absolute hex 400 to hex 4FF. Locations hex 400 to 407 contain the base addresses of any RS-232C cards attached to the system. Locations hex 408 to 40F contain the base addresses of the printer adapter.

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

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 Break 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
100-17F	40-5F	Reserved
180-19F	60-67	Reserved for User Software Interrupts
1A0-1FF	68-7F	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

BASIC and DOS Reserved Interrupts

Address (Hex)	Mode	Function
400-48F	ROM BIOS	See BIOS Listing
490-4EF		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

Reserved Memory Locations

If you do DEF SEG (Default workspace segment):

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

Example

```
100 Print  PEEK (&H2E) + 256*PEEK (&H2F)
                 L             H
 100       | Hex 64  |  Hex 00  |
```

BASIC Workspace Variables

Starting Address in Hex	
00000	BIOS Interrupt Vectors
00080	Available Interrupt Vectors
00400	BIOS Data Area
00500	User Read/Write Memory
C8000	Disk Adapter
F0000	Read Only Memory
FE000	Bios Program Area

BIOS Memory Map

BIOS Programming Hints

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

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

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

Adapter Cards with System-Accessible ROM Modules

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

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

Byte 0: Hex 55
Byte 1: Hex AA
Byte 2: A length indicator representing the number of 512 byte blocks in the ROM (length/512).
A checksum is also done to test the integrity of the ROM module. Each byte in the defined ROM is summed modulo hex 100. This sum must be 0 for the module to be deemed valid.

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

Notes:

Keyboard Encoding and Usage

Encoding

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

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

Character Codes

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

Key Number	Base Case	Upper Case	Ctrl	Alt
1	Esc	Esc	Esc	-1
2	1	!	-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

Character Codes (Part 1 of 3)

Key Number	Base Case	Upper Case	Ctrl	Alt
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
28	CR	CR	LF (010)	-1
29 Ctrl	-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
42 Shift	-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
54 Shift	-1	-1	-1	-1
55	*	(Note 2)	(Note 1)	-1
56 Alt	-1	-1	-1	-1
57	SP	SP	SP	SP
58 Caps 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)

Character Codes (Part 2 of 3)

Key Number	Base Case	Upper Case	Ctrl	Alt
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)
69 Num Lock	-1	-1	Pause (Note 2)	-1
70 Scroll Lock	-1	-1	Break (Note 2)	-1

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

Character Codes (Part 3 of 3)

Keys 71 to 83 have meaning only in base case, in Num Lock (or shifted) states, or in Ctrl state. It should be noted that the shift key temporarily reverses the current Num Lock state.

Key Number	Num Lock	Base Case	Alt	Ctrl
71	7	Home (Note 1)	-1	Clear Screen
72	8	↑ (Note 1)	-1	-1
73	9	Page Up (Note 1)	-1	Top of Text and Home
74	-	--------------------	-1	-1
75	4	← (Note 1)	-1	Reverse Word (Note 1)
76	5	-1	-1	-1
77	6	→ (Note 1)	-1	Advance Word (Note 1)
78	+	+	-1	-1
79	1	End (Note 1)	-1	Erase to EOL (Note 1)
80	2	↓ (Note 1)	-1	-1
81	3	Page Down (Note 1)	-1	Erase to EOS (Note 1)
82	0	Ins	-1	-1
83	.	Del (Notes 1,2)	Note 2	Note 2

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

Extended Codes

Extended Functions

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

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 to F10 Function Keys Base Case
71	Home
72	↑
73	Page Up and Home Cursor
75	←
77	→
79	End
80	↓
81	Page Down and Home Cursor
82	Ins (Insert)
83	Del (Delete)
84-93	F11 to F20 (Uppercase F1 to F10)
94-103	F21 to F30 (Ctrl F1 to F10)
104-113	F31 to F40 (Alt F1 to F10)
114	Ctrl PrtSc (Start/Stop Echo to Printer)
115	Ctrl← (Reverse Word)
116	Ctrl→ (Advance Word)
117	Ctrl End [Erase to End of Line (EOL)]
118	Ctrl PgDn [Erase to End of Screen (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 PgUp (Top 25 Lines of Text and Home Cursor)

Keyboard Extended Functions

Shift States

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

Shift

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

Ctrl

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

Alt

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

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

Caps Lock

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

Scroll Lock

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

Shift Key Priorities and Combinations

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

Special Handling

System Reset

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

Break

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

Pause

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

Print Screen

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

Other Characteristics

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

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

Keyboard Usage

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

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 backward 25 lines and home	PgUp	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 forward 25 lines and 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, and so on
Start/Stop Echo screen to printer	Ctrl Prt Sc (Key 55)	Any time
Delete from cursor to EOS	Ctrl PgDn	Text, command entry
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

Keyboard - Commonly Used Functions (Part 1 of 2)

Function	Key(s)	Comment
Exit insert mode	Ins	Line editor
Cancel current line	Esc	Command entry, text entry
Suspend system (pause)	Ctrl Num Lock	Stop list, stop program, and so on Resumes on any key
Break interrupt	Ctrl Break	Interrupt current process
System reset	Alt Ctrl Del	Reboot
Top of document and home cursor	Ctrl PgUp	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

Keyboard - Commonly Used Functions (Part 2 of 2)

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 Num Lock
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

BASIC Screen Editor Special Functions

Function	Key
Suspend	Ctrl Num Lock
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 until match	F2
Copy remaining	F3
Skip character	Del
Skip until match	F4
Enter skip mode	Ins
Exit insert mode	Ins
Make new line the template	F5
String separator in REPLACE	F6
End of file in keyboard input	F6

DOS Special Functions

BIOS Cassette Logic

Software Algorithms – Interrupt Hex 15

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

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

Cassette Status	Description
AH = 00	No Errors
AH = 01	Cyclic Redundancy Check (CRC) Error in 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.

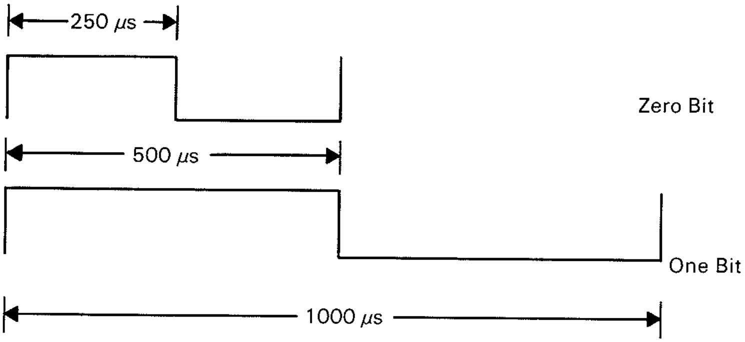
Cassette Write

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

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

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

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



Cassette Read

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

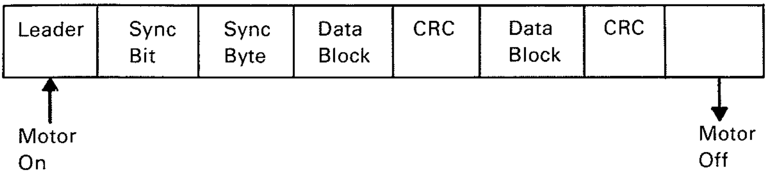
The read-block routine then searches for the leader and must detect all 1 bits for approximately 1/4 of the leader length before it can look for the sync (0) bit. After the sync bit is detected, the sync byte (ASCII character hex 16) is read. If the sync byte is read correctly, the data portion can be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

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

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

Data Record Architecture

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



Component	Description
Leader	256 Bytes (of All 1's)
Sync Bit	One 0 Bit
Sync Byte	ASCII Character Hex 16
Data Blocks	256 Bytes in Length
CRC	2 Bytes for each Data Block

Data Record Components

Error Recovery

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

APPENDIX A: ROM BIOS LISTINGS

System ROM BIOS

Fixed Disk ROM BIOS

```
LOC OBJ                 LINE    SOURCE

                        $TITLE(BIOS FOR IBM PERSONAL COMPUTER)

                        ;-----------------------------------------------------------
                        ;       THE  BIOS ROUTINES ARE MEANT TO BE ACCESSED THROUGH     :
                        ;       SOFTWARE INTERRUPTS ONLY.  ANY ADDRESSES PRESENT IN     :
                        ;       THE LISTINGS  ARE INCLUDED  ONLY FOR  COMPLETENESS,     :
                        ;       NOT FOR  REFERENCE.   APPLICATIONS WHICH  REFERENCE     :
                        ;       ABSOLUTE   ADDRESSES   WITHIN   THE  CODE   SEGMENT     :
                        ;       VIOLATE THE STRUCTURE AND DESIGN OF BIOS.               :
                        ;-----------------------------------------------------------

                        ;----------------------------------------
                        ;               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 ADDR 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 KB SENSE DATA
                        ;----------------------------------------
                        ;       8088 INTERRUPT LOCATIONS        :
                        ;----------------------------------------
----                    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
0060                                    ORG     18H*4
0060                    BASIC_PTR       LABEL   WORD            ; ENTRY POINT FOR CASSETTE BASIC
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
0100                                    ORG     040H*4          ; ROUTINE
0100 ????               IO_ROM_INIT     DW      ?               ;
0102 ????               IO_ROM_SEG      DW      ?               ; OPTIONAL ROM SEGMENT
0400                                    ORG     400H
0400                    DATA_AREA       LABEL   BYTE            ; ABSOLUTE LOCATION OF DATA SEGMENT
0400                    DATA_WORD       LABEL   WORD
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             :
                        ;----------------------------------------
----                    DATA            SEGMENT AT 40H
```

```
LOC OBJ            SOURCE

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
  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 OP IS A WRITE, REQUIRES DELAY
0040 ??            MOTOR_COUNT     DB      ?               ; TIME OUT COUNTER FOR DRIVE TURN OFF
  0025             MOTOR_WAIT      EQU     37              ; TWO SEC OF COUNT FOR MOTOR TURN OFF

0041 ??            DISKETTE_STATUS DB      ?               ; 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
```

```
LOC OBJ                 LINE    SOURCE

  0001                  149     BAD_CMD         EQU     01H             ; BAD COMMAND PASSED TO DISKETTE I/O
                        150
0042 (7                 151     NEC_STATUS      DB      7 DUP(?)        ; STATUS BYTES FROM NEC
     ??
     )
                        152
                        153     ;----------------------------------------
                        154     ;       VIDEO DISPLAY DATA AREA         :
                        155     ;----------------------------------------
0049 ??                 156     CRT_MODE        DB      ?               ; CURRENT CRT MODE
004A ????               157     CRT_COLS        DW      ?               ; NUMBER OF COLUMNS ON SCREEN
004C ????               158     CRT_LEN         DW      ?               ; LENGTH OF REGEN IN BYTES
004E ????               159     CRT_START       DW      ?               ; STARTING ADDRESS IN REGEN BUFFER
0050 (8                 160     CURSOR_POSN     DW      8 DUP(?)        ; CURSOR FOR EACH OF UP TO 8 PAGES
     ????
     )
0060 ????               161     CURSOR_MODE     DW      ?               ; CURRENT CURSOR MODE SETTING
0062 ??                 162     ACTIVE_PAGE     DB      ?               ; CURRENT PAGE BEING DISPLAYED
0063 ????               163     ADDR_6845       DW      ?               ; BASE ADDRESS FOR ACTIVE DISPLAY CARD
0065 ??                 164     CRT_MODE_SET    DB      ?               ; CURRENT SETTING OF THE 3X8 REGISTER
0066 ??                 165     CRT_PALETTE     DB      ?               ; CURRENT PALETTE SETTING COLOR CARD
                        166
                        167     ;----------------------------------------
                        168     ;          CASSETTE DATA AREA           :
                        169     ;----------------------------------------
0067 ????               170     EDGE_CNT        DW      ?               ; TIME COUNT AT DATA EDGE
0069 ????               171     CRC_REG         DW      ?               ; CRC REGISTER
006B ??                 172     LAST_VAL        DB      ?               ; LAST INPUT VALUE
                        173
                        174     ;----------------------------------------
                        175     ;            TIMER DATA AREA            :
                        176     ;----------------------------------------
006C ????               177     TIMER_LOW       DW      ?               ; LOW WORD OF TIMER COUNT
006E ????               178     TIMER_HIGH      DW      ?               ; HIGH WORD OF TIMER COUNT
0070 ??                 179     TIMER_OFL       DB      ?               ; TIMER HAS ROLLED OVER SINCE LAST READ
                        180     ;COUNTS_SEC     EQU     18
                        181     ;COUNTS_MIN     EQU     1092
                        182     ;COUNTS_HOUR    EQU     65543
                        183     ;COUNTS_DAY     EQU     1573040 = 1800B0H
                        184
                        185     ;----------------------------------------
                        186     ;          SYSTEM DATA AREA             :
                        187     ;----------------------------------------
0071 ??                 188     BIOS_BREAK      DB      ?               ; BIT 7 = 1 IF BREAK KEY HAS DEPRESSED
0072 ????               189     RESET_FLAG      DW      ?               ; WORD = 1234H IF KB RESET UNDERWAY
                        190     ;----------------------------------------
                        191     ;         FIXED DISK DATA AREA          :
                        192     ;----------------------------------------
0074 ????               193                     DW      ?               ;
0076 ????               194                     DW      ?               ;
                        195     ;----------------------------------------
                        196     ;     PRINTER AND RS232 TIMEOUT CTRS    :
                        197     ;----------------------------------------
0078 (4                 198     PRINT_TIM_OUT   DB      4 DUP(?)        ; PRINTER TIME OUT COUNTER
     ??
     )
007C (4                 199     RS232_TIM_OUT   DB      4 DUP(?)        ; RS232 TIME OUT COUNTER
     ??
     )
                        200     ;----------------------------------------
                        201     ;       EXTRA KEYBOARD DATA AREA        :
                        202     ;----------------------------------------
0080 ????               203     BUFFER_START    DW      ?
0082 ????               204     BUFFER_END      DW      ?
----                    205     DATA    ENDS
                        206     ;----------------------------------------
                        207     ;           EXTRA DATA AREA             :
                        208     ;----------------------------------------
----                    209     XXDATA          SEGMENT AT 50H
0000 ??                 210     STATUS_BYTE     DB      ?
----                    211     XXDATA          ENDS
                        212
                        213     ;----------------------------------------
                        214     ;       VIDEO DISPLAY BUFFER            :
                        215     ;----------------------------------------
----                    216     VIDEO_RAM       SEGMENT AT 0B800H
```

```
0000                REGEN           LABEL   BYTE
0000                REGENW          LABEL   WORD
0000 (16384                         DB      16384 DUP(?)
      ??
      )
----                VIDEO_RAM       ENDS
                    ;----------------------------------------
                    ;          ROM RESIDENT CODE              :
                    ;----------------------------------------
----                CODE            SEGMENT AT 0F000H
0000 (57344                         DB      57344 DUP(?)              ; FILL LOWEST 56K
      ??
      )

E000 31353031343736                 DB      '1501476 COPR. IBM 1981'  ; COPYRIGHT NOTICE
     20434F50522E20
     49424D20313938
     32

                    ;------------------------------------------------------------
                    ;       INITIAL RELIABILITY TESTS -- PHASE 1                :
                    ;------------------------------------------------------------
                                    ASSUME   CS:CODE,SS:CODE,ES:ABS0,DS:DATA
                    ;----------------------------------------
                    ;       DATA DEFINITIONS                 :
                    ;----------------------------------------
E016 D1E0           C1      DW      C11             ; RETURN ADDRESS

                    ;---------------------------------------------------------------------
                    ;       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.                    :
                    ;---------------------------------------------------------------------

E018                STGTST  PROC    NEAR
E018 B90040                 MOV     CX,4000H            ; SETUP CNT TO TEST A 16K BLK
E01B                STGTST_CNT:
E01B FC                     CLD                         ; SET DIR FLAG TO INCREMENT
E01C 8BD9                   MOV     BX,CX               ; SAVE BYTE CNT (4K FOR VIDEO OR 16K)
E01E B8AAAA                 MOV     AX,0AAAAH           ; GET DATA PATTERN TO WRITE
E021 BA55FF                 MOV     DX,0FF55H           ; SETUP OTHER DATA PATTERNS TO USE
E024 2BFF                   SUB     DI,DI               ; DI = OFFSET 0 RELATIVE TO ES REG
E026 F3                     REP     STOSB               ; WRITE STORAGE LOCATIONS
E027 AA
E028                C3:                                 ;  STG01
E028 4F                     DEC     DI                  ; POINT TO LAST BYTE JUST WRITTEN
E029 FD                     STD                         ; SET DIR FLAG TO GO BACKWARDS
E02A                C4:
E02A 8BF7                   MOV     SI,DI
E02C 8BCB                   MOV     CX,BX               ; SETUP BYTE CNT
E02E                C5:                                 ; INNER TEST LOOP
E02E AC                     LODSB                       ; READ OLD TST BYTE FROM STORAGE [SI]+
E02F 32C4                   XOR     AL,AH               ; DATA READ AS EXPECTED ?
E031 7525                   JNE     C7                  ;  NO - GO TO ERROR ROUTINE
E033 8AC2                   MOV     AL,DL               ; GET NEXT DATA PATTERN TO WRITE
E035 AA                     STOSB                       ; WRITE INTO LOCATION JUST READ [DI]+
E036 E2F6                   LOOP    C5                  ;  DECREMENT BYTE COUNT AND LOOP CX

E038 22E4                   AND     AH,AH               ; ENDING ZERO PATTERN WRITTEN TO STG ?
E03A 7416                   JZ      C6X                 ; YES - RETURN TO CALLER WITH AL=0
E03C 8AE0                   MOV     AH,AL               ; SETUP NEW VALUE FOR COMPARE
E03E 86F2                   XCHG    DH,DL               ; MOVE NEXT DATA PATTERN TO DL
E040 22E4                   AND     AH,AH               ; READING ZERO PATTERN THIS PASS ?
E042 7504                   JNZ     C6                  ; CONTINUE TEST SEQUENCE TILL ZERO DATA
E044 8AD4                   MOV     DL,AH               ;  ELSE SET ZERO FOR END READ PATTERN
E046 EBE0                   JMP     C3                  ;  AND MAKE FINAL BACKWARDS PASS
E048                C6:
```

```
E048 FC                  CLD                             ; SET DIR FLAG TO GO FORWARD
E049 47                  INC     DI                      ; SET POINTER TO BEG LOCATION
E04A 74DE                JZ      C4                      ; READ/WRITE FORWARD IN STG
E04C 4F                  DEC     DI                      ; ADJUST POINTER
E04D BA0100              MOV     DX,00001H               ; SETUP 01 FOR PARITY BIT
                                                         ;  AND 00 FOR END
E050 EBD6                JMP     C3                      ; READ/WRITE BACKWARD IN STG
E052            C6X:
E052 E462                IN      AL,PORT_C               ; DID A PARITY ERROR OCCUR ?
E054 24C0                AND     AL,0C0H                 ; ZERO FLAG WILL BE OFF PARITY ERROR
E056 B000                MOV     AL,000H                 ; AL=0 DATA COMPARE OK
E058            C7:
E058 FC                  CLD                             ; SET DEFAULT DIRCTN FLAG BACK TO INC
E059 C3                  RET
                STGTST  ENDP
                ;--------------------------------------------------------------
                ;        8088 PROCESSOR TEST                                  :
                ; DESCRIPTION                                                 :
                ;        VERIFY 8088 FLAGS, REGISTERS AND CONDITIONAL JUMPS   :
                ;--------------------------------------------------------------
                         ASSUME  CS:CODE,DS:NOTHING,ES:NOTHING,SS:NOTHING
E05B                     ORG     0E05BH
E05B            RESET    LABEL   FAR
E05B            START:
E05B FA                  CLI                             ; DISABLE INTERRUPTS
E05C B4D5                MOV     AH,0D5H                 ; SET SF, CF, ZF, AND AF FLAGS ON
E05E 9E                  SAHF
E05F 734C                JNC     ERR01                   ; GO TO ERR ROUTINE IF CF NOT SET
E061 754A                JNZ     ERR01                   ; GO TO ERR ROUTINE IF ZF NOT SET
E063 7B48                JNP     ERR01                   ; GO TO ERR ROUTINE IF PF NOT SET
E065 7946                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 733F                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 7139                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 7634                JBE     ERR01                   ; GO TO ERR ROUTINE IF CF ON
                                                         ; OR TO TO ERR ROUTINE IF ZF ON
E079 7832                JS      ERR01                   ; GO TO ERR ROUTINE IF SF ON
E07B 7A30                JP      ERR01                   ; GO TO ERR ROUTINE IF PF ON
E07D 9F                  LAHF                            ; LOAD FLAG IMAGE TO AH
E07E B105                MOV     CL,5                    ; LOAD CNT REG WITH SHIFT CNT
E080 D2EC                SHR     AH,CL                   ; SHIFT 'AF' INTO CARRY BIT POS
E082 7229                JC      ERR01                   ; GO TO ERR ROUTINE IF ON
E084 D0E4                SHL     AH,1                    ; CHECK THAT 'OF' IS CLEAR
E086 7025                JO      ERR01                   ; GO TO ERR ROUTINE IF ON

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

E088 B8FFFF              MOV     AX,0FFFFH               ; SETUP ONE'S PATTERN IN AX
E08B F9                  STC
E08C            C8:
E08C 8ED8                MOV     DS,AX                   ; WRITE PATTERN TO ALL REGS
E08E 8CDB                MOV     BX,DS
E090 8EC3                MOV     ES,BX
E092 8CC1                MOV     CX,ES
E094 8ED1                MOV     SS,CX
E096 8CD2                MOV     DX,SS
E098 8BE2                MOV     SP,DX
E09A 8BEC                MOV     BP,SP
E09C 8BF5                MOV     SI,BP
E09E 8BFE                MOV     DI,SI
E0A0 7307                JNC     C9                      ; TST1A
E0A2 33C7                XOR     AX,DI                   ; PATTERN MAKE IT THRU ALL REGS
E0A4 7507                JNZ     ERR01                   ; NO - GO TO ERR ROUTINE
E0A6 F8                  CLC
E0A7 EBE3                JMP     C8
E0A9            C9:                                      ; TST1A
E0A9 0BC7                OR      AX,DI                   ; ZERO PATTERN MAKE IT THRU?
E0AB 7401                JZ      C10                     ; YES - GO TO NEXT TEST
E0AD F4         ERR01:   HLT                             ; HALT SYSTEM
                ;-------------------------------------------------------
```

```
LOC OBJ                 SOURCE

                        ;       ROS CHECKSUM TEST I                           :
                        ; DESCRIPTION                                         :
                        ;       A CHECKSUM IS DONE FOR THE 8K ROS MODULE      :
                        ;       CONTAINING POD AND BIOS.                      :
                        ;------------------------------------------------------
E0AE                    C10:
                                                        ; ZERO IN AL ALREADY
E0AE E6A0                       OUT     0A0H,AL         ; DISABLE NMI INTERRUPTS
E0B0 E683                       OUT     83H,AL          ; INITIALZE DMA PAGE REG
E0B2 BAD803                     MOV     DX,3D8H
E0B5 EE                         OUT     DX,AL           ; DISABLE COLOR VIDEO
E0B6 FEC0                       INC     AL
E0B8 B2B8                       MOV     DL,0B8H
E0BA EE                         OUT     DX,AL           ; DISABLE B/W VIDEO,EN HIGH RES
E0BB B099                       MOV     AL,99H          ; SET 8255 A,C-INPUT,B-OUTPUT
E0BD E663                       OUT     CMD_PORT,AL     ; WRITE 8255 CMD/MODE REG
E0BF B0FC                       MOV     AL,0FCH         ; DISABLE PARITY CHECKERS AND
E0C1 E661                       OUT     PORT_B,AL       ;  GATE SNS SWS,CASS MOTOR OFF
E0C3 8CC8                       MOV     AX,CS           ; SETUP SS SEG REG
E0C5 8ED0                       MOV     SS,AX
E0C7 8ED8                       MOV     DS,AX           ; SET UP DATA SEG TO POINT TO
                                                        ;  ROM ADDRESS
                                ASSUME  SS:CODE
E0C9 B7E0                       MOV     BH,0E0H         ; SETUP STARTING ROS ADDR (E0000)
E0CB BC16E0                     MOV     SP,OFFSET C1    ; SETUP RETURN ADDRESS
E0CE E97B0B                     JMP     ROS_CHECKSUM
E0D1                    C11:
E0D1 750A                       JNE     ERR01           ; HALT SYSTEM IF ERROR
                        ;--------------------------------------------------------------
                        ;       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.                         :
                        ;--------------------------------------------------------------
E0D3 B004                       MOV     AL,04           ; DISABLE DMA CONTROLLER
E0D5 E608                       OUT     DMA08,AL

                        ;----- VERIFY THAT TIMER 1 FUNCTIONS OK

E0D7 B054                       MOV     AL,54H          ; SEL TIMER 1,LSB,MODE 2
E0D9 E643                       OUT     TIMER+3,AL
E0DB 8AC1                       MOV     AL,CL           ; SET INITIAL TIMER CNT TO 0
E0DD E641                       OUT     TIMER+1,AL
E0DF                    C12:                            ; TIMER1_BITS_ON
E0DF B040                       MOV     AL,40H          ; LATCH TIMER 1 COUNT
E0E1 E643                       OUT     TIMER+3,AL
E0E3 80FBFF                     CMP     BL,0FFH         ; YES - SEE IF ALL BITS GO OFF
E0E6 7407                       JE      C13             ; TIMER1_BITS_OFF
E0E8 E441                       IN      AL,TIMER+1      ; READ TIMER 1 COUNT
E0EA 0AD8                       OR      BL,AL           ; ALL BITS ON IN TIMER
E0EC E2F1                       LOOP    C12             ; TIMER1_BITS_ON
E0EE F4                         HLT                     ; TIMER 1 FAILURE, HALT SYS
E0EF                    C13:                            ; TIMER1_BITS_OFF
E0EF 8AC3                       MOV     AL,BL           ; SET TIMER 1 CNT
E0F1 2BC9                       SUB     CX,CX
E0F3 E641                       OUT     TIMER+1,AL
E0F5                    C14:                            ; TIMER_LOOP
E0F5 B040                       MOV     AL,40H          ; LATCH TIMER 1 COUNT
E0F7 E643                       OUT     TIMER+3,AL
E0F9 90                         NOP                     ; DELAY FOR TIMER
E0FA 90                         NOP
E0FB E441                       IN      AL,TIMER+1      ; READ TIMER 1 COUNT
E0FD 22D8                       AND     BL,AL
E0FF 7403                       JZ      C15             ; GO TO WRAP_DMA_REG
E101 E2F2                       LOOP    C14             ; TIMER_LOOP
E103 F4                         HLT                     ; TIMER ERROR - HALT SYSTEM

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

E104                    C15:                            ; WRAP_DMA_REG
E104 B012                       MOV     AL,18           ; SETUP DIVISOR FOR REFRESH
E106 E641                       OUT     TIMER+1,AL      ; WRITE TIMER 1 CNT REG
E108 E60D                       OUT     DMA+0DH,AL      ; SEND MASTER CLEAR TO DMA

```

```
LOC OBJ              SOURCE

                     ;----- WRAP DMA CHANNELS ADDRESS AND COUNT REGISTERS

E10A B0FF                    MOV    AL,0FFH              ; WRITE PATTERN FF TO ALL REGS
E10C                 C16:
E10C 8AD8                    MOV    BL,AL                ; SAVE PATTERN FOR COMPARE
E10E 8AF8                    MOV    BH,AL
E110 B90800                  MOV    CX,8                 ; SETUP LOOP CNT
E113 2BD2                    SUB    DX,DX                ; SETUP I/O PORT ADDR OF REG (0000)
E115                 C17:
E115 EE                      OUT    DX,AL                ; WRITE PATTERN TO REG, LSB
E116 50                      PUSH   AX
E117 EE                      OUT    DX,AL                ; MSB OF 16 BIT REG
E118 B80101                  MOV    AX,0101H             ; AX TO ANOTHER PAT BEFORE RD
E11B EC                      IN     AL,DX                ; READ 16-BIT DMA CH REG, LSB
E11C 8AE0                    MOV    AH,AL                ; SAVE LSB OF 16-BIT REG
E11E EC                      IN     AL,DX                ; READ MSB OF DMA CH REG
E11F 3BD8                    CMP    BX,AX                ; PATTERN READ AS WRITTEN?
E121 7401                    JE     C18                  ; YES - CHECK NEXT REG
E123 F4                      HLT                         ; NO - HALT THE SYSTEM
E124                 C18:                                ; NXT_DMA_CH
E124 42                      INC    DX                   ; SET I/O PORT TO NEXT CH REG
E125 E2EE                    LOOP   C17                  ; WRITE PATTERN TO NEXT REG
E127 FEC0                    INC    AL                   ; SET PATTERN TO 0
E129 74E1                    JZ     C16                  ; WRITE TO CHANNEL REGS

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

E12B 8EDB                    MOV    DS,BX                ; SET UP ABS0 INTO DS AND ES
E12D 8EC3                    MOV    ES,BX
                             ASSUME DS:ABS0,ES:ABS0

E12F B0FF                    MOV    AL,0FFH              ; SET CNT OF 64K FOR RAM REFRESH
E131 E601                    OUT    DMA+1,AL
E133 50                      PUSH   AX
E134 E601                    OUT    DMA+1,AL
E136 B20B                    MOV    DL,0BH               ; DX=000B
E138 B058                    MOV    AL,058H              ; SET DMA MODE,CH 0,READ,AUTOINT
E13A EE                      OUT    DX,AL                ; WRITE DMA MODE REG
E13B B000                    MOV    AL,0                 ; ENABLE DMA CONTROLLER
E13D E608                    OUT    DMA+8,AL             ; SETUP DMA COMMAND REG
E13F 50                      PUSH   AX
E140 E60A                    OUT    DMA+10,AL            ; ENABLE CHANNEL 0 FOR REFRESH
E142 B103                    MOV    CL,3
E144 B041                    MOV    AL,41H               ; SET MODE FOR CHANNEL 1
E146                 C18A:
E146 EE                      OUT    DX,AL
E147 FEC0                    INC    AL                   ; POINT TO NEXT CHANNEL
E149 E2FB                    LOOP   C18A
                     ;-------------------------------------------------------------
                     ;       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

E14B BA1302                  MOV    DX,0213H             ; ENABLE EXPANSION BOX
E14E B001                    MOV    AL,01H
E150 EE                      OUT    DX,AL
E151 8B2E7204                MOV    BP,DATA_WORD[OFFSET RESET_FLAG] ; SAVE 'RESET_FLAG' IN BP
E155 81FD3412                CMP    BP,1234H             ; WARM START?
E159 740A                    JE     C18B                 ; BYPASS STG TST.
E15B BC41F090                MOV    SP,OFFSET C2
E15F E9B6FE                  JMP    STGTST
E162                 C24:
E162 7401                    JE     C18B                 ; PROCEED IF STGTST OK
E164 F4                      HLT                         ; HALT IF NOT
E165                 C18B:
E165 2BFF                    SUB    DI,DI
E167 E460                    IN     AL,PORT_A            ; DETERMINE BASE RAM SIZE
E169 240C                    AND    AL,0CH               ; ISOLATE RAM SIZE SWS
E16B 0404                    ADD    AL, 4                ; CALCULATE MEMORY SIZE
E16D B10C                    MOV    CL, 12
```

```
LOC OBJ              SOURCE

E16F D3E0                    SHL     AX, CL
E171 8BC8                    MOV     CX, AX
E173 FC                      CLD                             ; SET DIR FLAG TO INCR
E174                 C19:
E174 AA                      STOSB                           ; FILL BASE RAM WITH DATA
E175 E2FD                    LOOP    C19                     ; LOOP TIL ALL ZERO
E177 892E7204                MOV     DATA_WORD[OFFSET RESET_FLAG],BP

                     ;----- DETERMINE IO CHANNEL RAM SIZE

E17B B0F8                    MOV     AL,0F8H                 ; ENABLE SWITCH 5
E17D E661                    OUT     PORT_B,AL
E17F E462                    IN      AL,PORT_C               ; READ SWITCHES
E181 2401                    AND     AL,00000001B            ; ISOLATE SWITCH 5
E183 B10C                    MOV     CL,12D
E185 D3C0                    ROL     AX,CL
E187 B0FC                    MOV     AL,0FCH                 ; DISABLE SW. 5
E189 E661                    OUT     PORT_B,AL
E18B E462                    IN      AL,PORT_C
E18D 240F                    AND     AL,0FH
E18F 0AC4                    OR      AL,AH                   ; COMBINE SWITCH VALUES
E191 8AD8                    MOV     BL,AL                   ; SAVE
E193 B420                    MOV     AH,32
E195 F6E4                    MUL     AH                      ; CALC. LENGTH
E197 A31504                  MOV     DATA_WORD[OFFSET IO_RAM_SIZE],AX        ;SAVE IT
E19A 7418                    JZ      C21
E19C BA0010                  MOV     DX,1000H                ; SEGMENT FOR I/O RAM
E19F 8AE0                    MOV     AH,AL
E1A1 B000                    MOV     AL,0
E1A3                 C20:                                    ; FILL_IO:
E1A3 8EC2                    MOV     ES,DX
E1A5 B90080                  MOV     CX,8000H                ; FILL 32K BYTES
E1A8 2BFF                    SUB     DI,DI
E1AA F3                      REP     STOSB
E1AB AA
E1AC 81C20008                ADD     DX,800H                 ; NEXT SEGMENT VALUE
E1B0 FECB                    DEC     BL
E1B2 75EF                    JNZ     C20                     ; FILL_IO
                     ;-------------------------------------------------------------
                     ;       INITIALIZE THE 8259 INTERRUPT CONTROLLER CHIP         :
                     ;-------------------------------------------------------------
E1B4                 C21:
E1B4 B013                    MOV     AL,13H                  ; ICW1 - EDGE, SNGL, ICW4
E1B6 E620                    OUT     INTA00,AL
E1B8 B008                    MOV     AL,8                    ; SETUP ICW2 - INT TYPE 8 (8-F)
E1BA E621                    OUT     INTA01,AL
E1BC B009                    MOV     AL,9                    ; SETUP ICW4 - BUFFRD,8086 MODE
E1BE E621                    OUT     INTA01,AL
E1C0 2BC0                    SUB     AX,AX                   ; POINT ES TO BEGIN
E1C2 8EC0                    MOV     ES,AX                   ;  OF R/W STORAGE
                     ;-------------------------------------------------------------------------
                     ;       CHECK FOR MANUFACTURING TEST 2 TO LOAD TEST PROGRAMS FROM KEYBOARD.:
                     ;-------------------------------------------------------------------------

                     ;----- SETUP STACK SEG AND SP

E1C4 B83000                  MOV     AX,STACK                ; GET STACK VALUE
E1C7 8ED0                    MOV     SS,AX                   ; SET THE STACK UP
E1C9 BC0001                  MOV     SP,OFFSET TOS           ; STACK IS READY TO GO
E1CC 81FD3412                CMP     BP,1234H                ; RESET_FLAG SET?
E1D0 7425                    JE      C25                     ; YES - SKIP MFG TEST
E1D2 2BFF                    SUB     DI,DI
E1D4 8EDF                    MOV     DS, DI
E1D6 BB2400                  MOV     BX, 24H
E1D9 C70747FF                MOV     WORD PTR [BX],OFFSET D11 ; SET UP KB INTERRUPT
E1DD 43                      INC     BX
E1DE 43                      INC     BX
E1DF 8C0F                    MOV     [BX],CS
E1E1 E85F04                  CALL    KBD_RESET               ; READ IN KB RESET CODE TO BL
E1E4 80FB65                  CMP     BL,065H                 ; IS THIS MANUFACTURING TEST 2?
E1E7 750E                    JNZ     C25                     ; JUMP IF NOT MAN. TEST
E1E9 B2FF                    MOV     DL,255                  ; READ IN TEST PROGRAM
E1EB                 C22:
E1EB E86204                  CALL    SP_TEST
E1EE 8AC3                    MOV     AL,BL
E1F0 AA                      STOSB
```

Appendix A

```
LOC OBJ                 SOURCE

E1F1 FECA                       DEC     DL
E1F3 75F6                       JNZ     C22                     ; JUMP IF NOT DONE YET
E1F5 CD3E                       INT     3EH                     ; SET INTERRUPT TYPE 62 ADDRESS F8H
E1F7                    C25:

                        ;----- SET UP THE BIOS INTERRUPT VECTORS TO TEMP INTERRUPT

E1F7 B92000                     MOV     CX,32                   ; FILL ALL 32 INTERRUPTS
E1FA 2BFF                       SUB     DI,DI                   ; FIRST INTERRUPT LOCATOIN
E1FC                    D3:
E1FC B847FF                     MOV     AX,OFFSET D11           ; MOVE ADDR OF INTR PROC TO TBL
E1FF AB                         STOSW
E200 8CC8                       MOV     AX,CS                   ; GET ADDR OF INTR PROC SEG
E202 AB                         STOSW
E203 E2F7                       LOOP    D3                      ; VECTBL0

                        ;----- SET UP OTHER INTERRUPTS AS NECESSARY

E205 C7060800C3E2               MOV     NMI_PTR,OFFSET NMI_INT  ; NMI INTERRUPT
E20B C706140054FF               MOV     INT5_PTR,OFFSET PRINT_SCREEN    ; PRINT SCREEN
E211 C706620000F6               MOV     BASIC_PTR+2,0F600H      ; SEGMENT FOR CASSETTE BASIC

                        ;---------------------------------------------------------------
                        ;       8259 INTERRUPT CONTROLLER TEST                          :
                        ; DESCRIPTION                                                   :
                        ;       READ/WRITE THE INTERRUPT MASK REGISTER (IMR) WITH ALL   :
                        ;       ONES AND ZEROES. ENABLE SYSTEM INTERRUPTS.  MASK DEVICE :
                        ;       INTERRUPTS OFF. CHECK FOR HOT INTERRUPTS (UNEXPECTED).  :
                        ;---------------------------------------------------------------

                        ;----- TEST THE IMR REGISTER

E217 BA2100                     MOV     DX,0021H                ; POINT INTR. CHIP ADDR 21
E21A B000                       MOV     AL,0                    ; SET IMR TO ZERO
E21C EE                         OUT     DX,AL
E21D EC                         IN      AL,DX                   ; READ IMR
E21E 0AC0                       OR      AL,AL                   ; IMR = 0?
E220 7515                       JNZ     D6                      ; GO TO ERR ROUTINE IF NOT 0
E222 B0FF                       MOV     AL,0FFH                 ; DISABLE DEVICE INTERRUPTS
E224 EE                         OUT     DX,AL                   ; WRITE TO IMR
E225 EC                         IN      AL,DX                   ; READ IMR
E226 0401                       ADD     AL,1                    ; ALL IMR BIT ON?
E228 750D                       JNZ     D6                      ; NO - GO TO ERR ROUTINE

                        ;----- CHECK FOR HOT INTERRUPTS

                        ;----- INTERRUPTS ARE MASKED OFF.  CHECK THAT NO INTERRUPTS OCCUR.

E22A 32E4                       XOR     AH,AH                   ; CLEAR AH REG
E22C FB                         STI                             ; ENABLE EXTERNAL INTERRUPTS
E22D 2BC9                       SUB     CX,CX                   ; WAIT 1 SEC FOR ANY INTRS THAT
E22F                    D4:
E22F E2FE                       LOOP    D4                      ;  MIGHT OCCUR
E231                    D5:
E231 E2FE                       LOOP    D5
E233 0AE4                       OR      AH,AH                   ; DID ANY INTERRUPTS OCCUR?
E235 7408                       JZ      D7                      ; NO - GO TO NEXT TEST
E237                    D6:
E237 BA0101                     MOV     DX,101H                 ; BEEP SPEAKER IF ERROR
E23A E89203                     CALL    ERR_BEEP                ; GO TO BEEP SUBROUTINE
E23D FA                         CLI
E23E F4                         HLT                             ; HALT THE SYSTEM
                        ;--------------------------------------------
                        ;       8253 TIMER CHECKOUT                 :
                        ; DESCRIPTION                               :
                        ;       VERIFY THAT THE SYSTEM TIMER (0)    :
                        ;       DOESN'T COUNT TOO FAST OR TOO SLOW. :
                        ;--------------------------------------------
E23F                    D7:
E23F B0FE                       MOV     AL,0FEH                 ; MASK ALL INTRS EXCEPT LVL 0
E241 EE                         OUT     DX,AL                   ; WRITE THE 8259 IMR
E242 B010                       MOV     AL,00010000B            ; SEL TIM 0, LSB, MODE 0, BINARY
E244 E643                       OUT     TIM_CTL,AL              ; WRITE TIMER CONTROL MODE REG
E246 B91600                     MOV     CX,16H                  ; SET PGM LOOP CNT
E249 8AC1                       MOV     AL,CL                   ; SET TIMER 0 CNT REG
E24B E640                       OUT     TIMER0,AL               ; WRITE TIMER 0 CNT REG
```

```
LOC OBJ               LINE   SOURCE

E24D                         D8:
E24D F6C4FF                         TEST   AH,0FFH                  ; DID TIMER 0 INTERRUPT OCCUR?
E250 7504                           JNZ    D9                       ; YES - CHECK TIMER OP FOR SLOW TIME
E252 E2F9                           LOOP   D8                       ; WAIT FOR INTR FOR SPECIFIED TIME
E254 EBE1                           JMP    D6                       ; TIMER 0 INTR DIDN'T OCCUR - ERR
E256                         D9:
E256 B112                           MOV    CL,18                    ; SET PGM LOOP CNT
E258 B0FF                           MOV    AL,0FFH                  ; WRITE TIMER 0 CNT REG
E25A E640                           OUT    TIMER0,AL
E25C B8FE00                         MOV    AX,0FEH
E25F EE                             OUT    DX,AL
E260                         D10:
E260 F6C4FF                         TEST   AH,0FFH                  ; DID TIMER 0 INTERRUPT OCCUR?
E263 75D2                           JNZ    D6                       ; YES - TIMER CNTING TOO FAST, ERR
E265 E2F9                           LOOP   D10                      ; WAIT FOR INTR FOR SPECIFIED TIME

                             ;----- ESTABLISH BIOS SUBROUTINE CALL INTERRUPT VECTORS

E267 1E                             PUSH   DS                       ; SAVE POINTER TO DATA AREA
E268 BF4000                         MOV    DI,OFFSET VIDEO_INT      ; SETUP ADDR TO INTR AREA
E26B 0E                             PUSH   CS
E26C 1F                             POP    DS                       ; SETUP ADDR OF VECTOR TABLE
E26D BE03FF90                       MOV    SI,OFFSET VECTOR_TABLE+16        ; START WITH VIDEO ENTRY
E271 B91000                         MOV    CX,16

                             ;----- SETUP TIMER 0 TO MODE 3

E274 B0FF                           MOV    AL,0FFH                  ; DISABLE ALL DEVICE INTERRUPTS
E276 EE                             OUT    DX,AL
E277 B036                           MOV    AL,36H                   ; SEL TIM 0,LSB,MSB,MODE 3
E279 E643                           OUT    TIMER+3,AL               ; WRITE TIMER MODE REG
E27B B000                           MOV    AL,0
E27D E640                           OUT    TIMER,AL                 ; WRITE LSB TO TIMER 0 REG
E27F                         E1A:
E27F A5                             MOVSW                           ; MOVE VECTOR TABLE TO RAM
E280 47                             INC    DI                       ; MOVE PAST SEGMENT POINTER
E281 47                             INC    DI
E282 E2FB                           LOOP   E1A
E284 E640                           OUT    TIMER,AL                 ; WRITE MSB TO TIMER 0 REG
E286 1F                             POP    DS                       ; RECOVER DATA SEG POINTER

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

E287 E8B903                         CALL   KBD_RESET                ; SEND SOFTWARE RESET TO KEYBRD
E28A 80FBAA                         CMP    BL,0AAH                  ; SCAN CODE 'AA' RETURNED?
E28D 741E                           JE     E6                       ; YES - CONTINUE (NON MFG MODE)
E28F B03C                           MOV    AL,3CH                   ; EN KBD, SET KBD CLK LINE LOW
E291 E661                           OUT    PORT_B,AL                ; WRITE 8255 PORT B
E293 90                             NOP
E294 90                             NOP
E295 E460                           IN     AL,PORT_A                ; WAS A BIT CLOCKED IN?
E297 24FF                           AND    AL,0FFH
E299 750E                           JNZ    E2                       ; YES - CONTINUE (NON MFG MODE)
E29B FE061204                       INC    DATA_AREA[OFFSET MFG_TST]        ; ELSE SET SW FOR MFG TEST MODE
E29F C70620006DE6                   MOV    INT_ADDR,OFFSET BLINK_INT        ; SETUP TIMER INTR TO BLINK LED
E2A5 B0FE                           MOV    AL,0FEH                  ; ENABLE TIMER INTERRUPT
E2A7 E621                           OUT    INTA01,AL
E2A9                         E2:                                    ; JUMPER_NOT_IN:
E2A9 B0CC                           MOV    AL,0CCH                  ; RESET THE KEYBOARD
E2AB E661                           OUT    PORT_B,AL

                             ;--------------------------------------------------------
                             ;       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.                                  :
                             ;--------------------------------------------------------
E2AD                         E6:
E2AD E460                           IN     AL,PORT_A                ; READ SENSE SWITCHES
E2AF B400                           MOV    AH,0
E2B1 A31004                         MOV    DATA_WORD[OFFSET EQUIP_FLAG],AX ; STORE SENSE SW INFO
E2B4                         E6A:
E2B4 2430                           AND    AL,30H                   ; ISOLATE VIDEO SWS
E2B6 7529                           JNZ    E7                       ; VIDEO SWS SET TO 0?
```

```
E2B8 C706400053FF              MOV     VIDEO_INT,OFFSET DUMMY_RETURN
E2BE E9A200                    JMP     E18_1                   ; SKIP VIDEO TESTS FOR BURN-IN

E2C3                           ORG     0E2C3H
E2C3                   NMI_INT PROC    NEAR
E2C3 50                        PUSH    AX                      ; SAVE ORIG CONTENTS OF AX
E2C4 E462                      IN      AL,PORT_C
E2C6 A8C0                      TEST    AL,0C0H                 ; PARITY CHECK?
E2C8 7415                      JZ      D14                     ; NO, EXIT FROM ROUTINE
E2CA BEDAFF90                  MOV     SI,OFFSET D1            ; ADDR OF ERROR MSG
E2CE A840                      TEST    AL,40H                  ; I/O PARITY CHECK
E2D0 7504                      JNZ     D13                     ; DISPLAY ERROR MSG
E2D2 BE23FF90                  MOV     SI,OFFSET D2            ; MUST BE PLANAR
E2D6                   D13:
E2D6 2BC0                      SUB     AX,AX                   ; INIT AND SET MODE FOR VIDEO
E2D8 CD10                      INT     10H                     ; CALL VIDEO_IO PROCEDURE
E2DA E8DD03                    CALL    P_MSG                   ; PRINT ERROR MSG
E2DD FA                        CLI
E2DE F4                        HLT                             ; HALT SYSTEM
E2DF                   D14:
E2DF 58                        POP     AX                      ; RESTORE ORIG CONTENTS OF AX
E2E0 CF                        IRET
                       NMI_INT ENDP
E2E1                   E7:                                     ; TEST_VIDEO:
E2E1 3C30                      CMP     AL,30H                  ; B/W CARD ATTACHED?
E2E3 7408                      JE      E8                      ; YES - SET MODE FOR B/W CARD
E2E5 FEC4                      INC     AH                      ; SET COLOR MODE FOR COLOR CD
E2E7 3C20                      CMP     AL,20H                  ; 80X25 MODE SELECTED?
E2E9 7502                      JNE     E8                      ; NO - SET MODE FOR 40X25
E2EB B403                      MOV     AH,3                    ; SET MODE FOR 80X25
E2ED                   E8:
E2ED 86E0                      XCHG    AH,AL                   ; SET_MODE
E2EF 50                        PUSH    AX                      ; SAVE VIDEO MODE ON STACK
E2F0 2AE4                      SUB     AH,AH                   ; INITIALIZE TO ALPHANUMERIC MD
E2F2 CD10                      INT     10H                     ; CALL VIDEO_IO
E2F4 58                        POP     AX                      ; RESTORE VIDEO SENSE SWS IN AH
E2F5 50                        PUSH    AX                      ; RESAVE VALUE
E2F6 BB00B0                    MOV     BX,0B000H               ; BEG VIDEO RAM ADDR B/W CD
E2F9 BAB803                    MOV     DX,3B8H                 ; MODE REG FOR B/W
E2FC B90010                    MOV     CX,4096                 ; RAM BYTE CNT FOR B/W CD
E2FF B001                      MOV     AL,1                    ; SET MODE FOR BW CARD
E301 80FC30                    CMP     AH,30H                  ; B/W VIDEO CARD ATTACHED?
E304 7408                      JE      E9                      ; YES - GO TEST VIDEO STG
E306 B7B8                      MOV     BH,0B8H                 ; BEG VIDEO RAM ADDR COLOR CD
E308 B2D8                      MOV     DL,0D8H                 ; MODE REG FOR COLOR CD
E30A B540                      MOV     CH,40H                  ; RAM BYTE CNT FOR COLOR CD
E30C FEC8                      DEC     AL                      ; SET MODE TO 0 FOR COLOR CD
E30E                   E9:                                     ; TEST_VIDEO_STG:
E30E EE                        OUT     DX,AL                   ; DISABLE VIDEO FOR COLOR CD
E30F 81FD3412                  CMP     BP,1234H                ; POD INITIATED BY KBD RESET?
E313 8EC3                      MOV     ES,BX                   ; POINT ES TO VIDEO RAM STG
E315 7407                      JE      E10                     ; YES - SKIP VIDEO RAM TEST
E317 8EDB                      MOV     DS,BX                   ; POINT DS TO VIDEO RAM STG
                               ASSUME  DS:NOTHING,ES:NOTHING
E319 E8FFFC                    CALL    STGTST_CNT              ; GO TEST VIDEO R/W STG
E31C 7532                      JNE     E17                     ; R/W STG FAILURE - BEEP SPK
                       ;-----------------------------------------------------------
                       ;       SETUP VIDEO DATA ON SCREEN FOR VIDEO LINE TEST.     :
                       ; DESCRIPTION                                               :
                       ;       ENABLE VIDEO SIGNAL AND SET MODE.                   :
                       ;       DISPLAY A HORIZONTAL BAR ON SCREEN.                 :
                       ;-----------------------------------------------------------
E31E                   E10:
E31E 58                        POP     AX                      ; GET VIDEO SENSE SWS (AH)
E31F 50                        PUSH    AX                      ; SAVE IT
E320 B400                      MOV     AH,0                    ; ENABLE VIDEO AND SET MODE
E322 CD10                      INT     10H                     ; VIDEO
E324 B82070                    MOV     AX,7020H                ; WRT BLANKS IN REVERSE VIDEO
E327 2BFF                      SUB     DI,DI                   ; SETUP STARTING LOC
E329 B92800                    MOV     CX,40                   ; NO. OF BLANKS TO DISPLAY
E32C F3                        REP     STOSW                   ; WRITE VIDEO STORAGE
E32D AB
                       ;---------------------------------------------------
                       ;       CRT INTERFACE LINES TEST                   :
                       ; DESCRIPTION                                      :
                       ;       SENSE ON/OFF TRANSITION OF THE VIDEO ENABLE :
```

```
                    ;       AND HORIZONTAL SYNC LINES.                       :
                    ;-------------------------------------------------------
E32E 58                     POP     AX                  ; GET VIDEO SENSE SW INFO
E32F 50                     PUSH    AX                  ; SAVE IT
E330 80FC30                 CMP     AH,30H              ; B/W CARD ATTACHED?
E333 BABA03                 MOV     DX,03BAH            ; SETUP ADDR OF BW STATUS PORT
E336 7402                   JE      E11                 ; YES - GO TEST LINES
E338 B2DA                   MOV     DL,0DAH             ; COLOR CARD IS ATTACHED
E33A                E11:                                ; LINE_TST:
E33A B408                   MOV     AH,8
E33C                E12:                                ; OFLOOP_CNT:
E33C 2BC9                   SUB     CX,CX
E33E                E13:
E33E EC                     IN      AL,DX               ; READ CRT STATUS PORT
E33F 22C4                   AND     AL,AH               ; CHECK VIDEO/HORZ LINE
E341 7504                   JNZ     E14                 ; ITS ON - CHECK IF IT GOES OFF
E343 E2F9                   LOOP    E13                 ; LOOP TILL ON OR TIMEOUT
E345 EB09                   JMP     SHORT E17           ; GO PRINT ERROR MSG
E347                E14:
E347 2BC9                   SUB     CX,CX
E349                E15:
E349 EC                     IN      AL,DX               ; READ CRT STATUS PORT
E34A 22C4                   AND     AL,AH               ; CHECK VIDEO/HORZ LINE
E34C 740A                   JZ      E16                 ; ITS ON - CHECK NEXT LINE
E34E E2F9                   LOOP    E15                 ; LOOP IF OFF TILL IT GOES ON
E350                E17:                                ;  CRT_ERR
E350 BA0201                 MOV     DX,102H
E353 E87902                 CALL    ERR_BEEP            ; GO BEEP SPEAKER
E356 EB06                   JMP     SHORT E18
E358                E16:                                ; NXT_LINE
E358 B103                   MOV     CL,3                ; GET NEXT BIT TO CHECK
E35A D2EC                   SHR     AH,CL
E35C 75DE                   JNZ     E12                 ; GO CHECK HORIZONTAL LINE
E35E                E18:                                ;  DISPLAY_CURSOR:
E35E 58                     POP     AX                  ; GET VIDEO SENSE SWS (AH)
E35F B400                   MOV     AH,0                ; SET MODE AND DISPLAY CURSOR
E361 CD10                   INT     10H                 ; CALL VIDEO I/O PROCEDURE

E363                E18_1:
E363 BA00C0                 MOV     DX,0C000H
E366                E18A:
E366 8EDA                   MOV     DS,DX
E368 2BDB                   SUB     BX,BX
E36A 8B07                   MOV     AX,[BX]             ; GET FIRST 2 LOCATIONS
E36C 53                     PUSH    BX
E36D 5B                     POP     BX                  ; LET BUS SETTLE
E36E 3D55AA                 CMP     AX,0AA55H           ; PRESENT?
E371 7505                   JNZ     E18B                ; NO? GO LOOK FOR OTHER MODULES
E373 E80E03                 CALL    ROM_CHECK           ; GO SCAN MODULE
E376 EB04                   JMP     SHORT E18C
E378                E18B:
E378 81C28000               ADD     DX,0080H            ; POINT TO NEXT 2K BLOCK
E37C                E18C:
E37C 81FA00C8               CMP     DX,0C800H           ; TOP OF VIDEO ROM AREA YET?
E380 7CE4                   JL      E18A                ; GO SCAN FOR ANOTHER MODULE
                    ;----------------------------------------------------------------
                    ; EXPANSION I/O BOX TEST                                         :
                    ;       CHECK TO SEE IF EXPANSION BOX PRESENT - IF INSTALLED,    :
                    ;       TEST DATA AND ADDRESS BUSES TO I/O BOX.                  :
                    ; ERROR='1801'                                                   :
                    ;----------------------------------------------------------------

                    ;----- DETERMINE IF BOX IS PRESENT

E382                EXP_IO:                             ; (CARD WAS ENABLED EARLIER)
E382 BA1002                 MOV     DX,0210H            ; CONTROL PORT ADDRESS
E385 B85555                 MOV     AX,5555H            ; SET DATA PATTERN
E388 EE                     OUT     DX,AL
E389 B001                   MOV     AL,01H
E38B EC                     IN      AL,DX               ; RECOVER DATA
E38C 3AC4                   CMP     AL,AH               ; REPLY?
E38E 7534                   JNE     E19                 ; NO RESPONSE, GO TO NEXT TEST
E390 F7D0                   NOT     AX                  ; MAKE DATA=AAAA
E392 EE                     OUT     DX,AL
E393 B001                   MOV     AL,01H
E395 EC                     IN      AL,DX               ; RECOVER DATA
E396 3AC4                   CMP     AL,AH
```

```
LOC OBJ                  LINE    SOURCE

E398 752A                                JNE     E19                     ; NO ANSWER=NEXT TEST

                                 ;----- CHECK ADDRESS AND DATA BUS

E39A                             EXP1:
E39A 8BD8                                MOV     BX,AX
E39C BA1402                              MOV     DX,0214H                ; LOAD DATA  REG ADDRESS
E39F 2E8807                              MOV     CS:[BX],AL              ; WRITE ADDRESS F0000+BX
E3A2 EE                                  OUT     DX,AL                   ; WRITE DATA
E3A3 90                                  NOP
E3A4 EC                                  IN      AL,DX                   ; READ DATA
E3A5 3AC7                                CMP     AL,BH
E3A7 7514                                JNE     EXP_ERR
E3A9 42                                  INC     DX                      ; DX=215H (ADDR. HI REG)
E3AA EC                                  IN      AL,DX
E3AB 3AC4                                CMP     AL,AH                   ; COMPARE TO HI ADDRESS
E3AD 750E                                JNE     EXP_ERR
E3AF 42                                  INC     DX                      ; DX-216H (ADDR. LOW REG)
E3B0 EC                                  IN      AL,DX
E3B1 3AC4                                CMP     AL,AH                   ; ADDR. LOW OK?
E3B3 7508                                JNE     EXP_ERR
E3B5 F7D0                                NOT     AX                      ; INVERT AX
E3B7 3CAA                                CMP     AL,0AAH                 ; BACK TO STARTING VALUE (AAAA) YET
E3B9 7409                                JE      E19                     ; GO ON TO NEXT TEST IF SO
E3BB EBDD                                JMP     EXP1                    ; LOOP BACK THROUGH WITH DATA OF 5555
E3BD                             EXP_ERR:
E3BD BEEDFE90                            MOV     SI,OFFSET F3B
E3C1 E8F602                              CALL    P_MSG
                                 ;-------------------------------------------------------------
                                 ;       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
E3C4                             E19:

                                 ;----- DETERMINE RAM SIZE ON PLANAR BOARD

E3C4 E8771B                              CALL    DDS
E3C7 A01000                              MOV     AL,BYTE PTR EQUIP_FLAG  ; GET SENSE SWS INFO
E3CA 240C                                AND     AL,0CH                  ; ISOLATE RAM SIZE SWS
E3CC B404                                MOV     AH,4
E3CE F6E4                                MUL     AH
E3D0 0410                                ADD     AL,16                   ; ADD BASIC 16K
E3D2 8BD0                                MOV     DX,AX                   ; SAVE PLANAR RAM SIZE IN DX
E3D4 8BD8                                MOV     BX,AX                   ;  AND IN BX

                                 ;----- DETERMINE IO CHANNEL RAM SIZE

E3D6 A11500                              MOV     AX,IO_RAM_SIZE          ; GET IO CHANNEL RAM SIZE
E3D9 83FB40                              CMP     BX,40H                  ; PLANAR RAM SIZE = 64K?
E3DC 7402                                JE      E20                     ; YES - ADD IO CHN RAM SIZE
E3DE 2BC0                                SUB     AX,AX                   ; NO - DON'T ADD ANY IO RAM
E3E0                             E20:                                    ; ADD_IO_SIZE:
E3E0 03C3                                ADD     AX,BX                   ; SUM TOTAL RAM SIZE
E3E2 A31300                              MOV     MEMORY_SIZE,AX          ; SETUP MEMORY SIZE PARM
E3E5 81FD3412                            CMP     BP,1234H                ; POD INITIATED BY KBD RESET?
E3E9 1E                                  PUSH    DS                      ; SAVE DATA SEGMENT
E3EA 744F                                JE      TST12                   ; YES - SKIP MEMORY TEST

                                 ;----- TEST ANY OTHER READ/WRITE STORAGE AVAILABLE

E3EC BB0004                              MOV     BX,400H
E3EF B91000                              MOV     CX,16
E3F2                             E21:
E3F2 3BD1                                CMP     DX,CX                   ; ANY MORE STG TO BE TESTED?
E3F4 762D                                JBE     E23                     ; NO - GO TO NEXT TEST
E3F6 8EDB                                MOV     DS,BX                   ; SETUP STG ADDR IN DS AND ES
E3F8 8EC3                                MOV     ES,BX
E3FA 83C110                              ADD     CX,16                   ; INCREMENT STG BYTE COUNTER
E3FD 81C30004                            ADD     BX,400H                 ; SET POINTER TO NEXT 16K BLK
E401 51                                  PUSH    CX                      ; SAVE REGS
E402 53                                  PUSH    BX
E403 52                                  PUSH    DX
E404 E811FC                              CALL    STGTST                  ; GO TEST A 16K BLK OF STG
E407 5A                                  POP     DX
```

```
LOC OBJ              SOURCE

E408 5B                      POP     BX                      ; RESTORE REGS
E409 59                      POP     CX
E40A 74E6                    JE      E21                     ; CHECK IF MORE STG TO TEST

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

E40C 8CDA                    MOV     DX,DS                   ; CONVERT FAILING HIGH-ORDER
E40E 8AE8                    MOV     CH,AL                   ; SAVE FAILING BIT PATTERN
E410 8AC6                    MOV     AL,DH                   ; GET FAILING ADDR
E412 E81002                  CALL    XPC_BYTE                ; CONVERT AND PRINT CODE
E415 8AC5                    MOV     AL,CH                   ; GET FAILING BIT PATTERN
E417 E80B02                  CALL    XPC_BYTE                ; CONVERT AND PRINT CODE
E41A BE67FA90                MOV     SI,OFFSET E1            ; SETUP ADDRESS OF ERROR MSG
E41E E89902                  CALL    P_MSG                   ; PRINT ERROR MSG
E421                 E22:
E421 EB18                    JMP     SHORT TST12             ; GO TO NEXT TEST
E423                 E23:                                    ; STG_TEST_DONE
E423 1F                      POP     DS                      ; POINT DS TO DATA SEGMENT
E424 1E                      PUSH    DS
E425 8B161500                MOV     DX,IO_RAM_SIZE          ; GET IO CHANNEL RAM SIZE
E429 0BD2                    OR      DX,DX                   ; SET FLAG RESULT
E42B 740E                    JZ      TST12                   ; NO IO RAM, GO TO NEXT TEST
E42D B90000                  MOV     CX,0
E430 81FB0010                CMP     BX,1000H                ; HAS IO RAM BEEN TESTED
E434 7705                    JA      TST12                   ; YES - GO TO NEXT TEST
E436 BB0010                  MOV     BX,1000H                ; SETUP BEG LOC FOR IO RAM
E439 EBB7                    JMP     E21                     ; GO TEST IO CHANNEL RAM
                     ;--------------------------------------------------------
                     ;       KEYBOARD TEST                                   :
                     ; DESCRIPTION                                           :
                     ;       RESET THE KEYBOARD AND CHECK THAT SCAN CODE     :
                     ;       'AA' IS RETURNED TO THE CPU.  CHECK FOR STUCK   :
                     ;        KEYS.                                          :
                     ;--------------------------------------------------------
                             ASSUME  DS:DATA
E43B                 TST12:
E43B 1F                      POP     DS
E43C 803E120001              CMP     MFG_TST,1               ; MANUFACTURING TEST MODE?
E441 742A                    JE      F7                      ; YES - SKIP KEYBOARD TEST
E443 E8FD01                  CALL    KBD_RESET               ; ISSUE SOFTWARE RESET TO KEYBRD
E446 E31E                    JCXZ    F6                      ; PRINT ERR MSG IF NO INTERRUPT
E448 B04D                    MOV     AL,4DH                  ; ENABLE KEYBOARD
E44A E661                    OUT     PORT_B,AL
E44C 80FBAA                  CMP     BL,0AAH                 ; SCAN CODE AS EXPECTED?
E44F 7515                    JNE     F6                      ; NO - DISPLAY ERROR MSG

                     ;----- CHECK FOR STUCK KEYS

E451 B0CC                    MOV     AL,0CCH                 ; CLR KBD, SET CLK LINE HIGH
E453 E661                    OUT     PORT_B,AL
E455 B04C                    MOV     AL,4CH                  ; ENABLE KBD,CLK IN NEXT BYTE
E457 E661                    OUT     PORT_B,AL
E459 2BC9                    SUB     CX,CX
E45B                 F5:                                     ; KBD_WAIT
E45B E2FE                    LOOP    F5                      ; DELAY FOR A WHILE
E45D E460                    IN      AL,KBD_IN               ; CHECK FOR STUCK KEYS
E45F 3C00                    CMP     AL,0                    ; SCAN CODE = 0?
E461 740A                    JE      F7                      ; YES - CONTINUE TESTING
E463 E8BF01                  CALL    XPC_BYTE                ; CONVERT AND PRINT
E466 BE33FF90        F6:     MOV     SI,OFFSET F1            ; GET MSG ADDR
E46A E84D02                  CALL    P_MSG                   ; PRINT MSG ON SCREEN

                     ;----- SETUP INTERRUPT VECTOR TABLE

E46D                 F7:                                     ; SETUP_INT_TABLE:
E46D 2BC0                    SUB     AX,AX
E46F 8EC0                    MOV     ES,AX
E471 B90800                  MOV     CX,8                    ; GET VECTOR CNT
E474 1E                      PUSH    DS                      ; SAVE DATA SEGMENT
E475 0E                      PUSH    CS                      ; SETUP DS SEG REG
E476 1F                      POP     DS
E477 BEF3FE90                MOV     SI,OFFSET VECTOR_TABLE
E47B BF2000                  MOV     DI,OFFSET INT_PTR
E47E                 F7A:
E47E A5                      MOVSW
E47F 47                      INC     DI                      ; SKIP OVER SEGMENT
E480 47                      INC     DI
```

```
E481 E2FB                   LOOP    F7A
                    ;-----------------------------------------------------
                    ;           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

E483                TST13:
E483 1F                     POP     DS
E484 1E                     PUSH    DS
E485 B04D                   MOV     AL,04DH                 ; SET TIMER 2 SPK OUT, AND CASST
E487 E661                   OUT     PORT_B,AL               ; OUT BITS ON, CASSETTE MOT OFF

                    ;----- WRITE A BIT

E489 B0FF                   MOV     AL,0FFH                 ; DISABLE TIMER INTERRUPTS
E48B E621                   OUT     INTA01,AL
E48D B0B6                   MOV     AL,0B6H                 ; SEL TIM 2, LSB, MSB, MD 3
E48F E643                   OUT     TIMER+3,AL              ; WRITE 8253 CMD/MODE REG
E491 B8D304                 MOV     AX,1235                 ; SET TIMER 2 CNT FOR 1000 USEC
E494 E642                   OUT     TIMER+2,AL              ; WRITE TIMER 2 COUNTER REG
E496 8AC4                   MOV     AL,AH                   ; WRITE MSB
E498 E642                   OUT     TIMER+2,AL

                    ;----- READ CASSETTE INPUT

E49A E462                   IN      AL,PORT_C               ; READ VALUE OF CASS IN BIT
E49C 2410                   AND     AL,10H                  ; ISOLATE FROM OTHER BITS
E49E A26B00                 MOV     LAST_VAL,AL
E4A1 E8D514                 CALL    READ_HALF_BIT
E4A4 E8D214                 CALL    READ_HALF_BIT
E4A7 E30C                   JCXZ    F8                      ; CAS_ERR
E4A9 81FB4005               CMP     BX,MAX_PERIOD
E4AD 7306                   JNC     F8                      ; CAS_ERR
E4AF 81FB1004               CMP     BX,MIN_PERIOD
E4B3 7307                   JNC     ROM_SCAN                ; GO TO NEXT TEST IF OK
E4B5                F8:                                     ; CAS_ERR
E4B5 BE39FF90               MOV     SI,OFFSET F2            ; CASSETTE WRAP FAILED
E4B9 E8FE01                 CALL    P_MSG                   ; GO PRINT ERROR MSG
                    ;--------------------------------------------------------------------
                    ;           CHECK FOR OPTIONAL ROM FROM C8000->F4000 IN 2K INCREMENTS    :
                    ;           (A VALID MODULE HAS '55AA' IN THE FIRST 2 LOCATIONS, LENGTH  :
                    ;           INDICATOR (LENGTH/512) IN THE 3RD LOCATION AND TEST/INIT.    :
                    ;           CODE STARTING IN THE 4TH LOCATION.)                          :
                    ;--------------------------------------------------------------------
E4BC                ROM_SCAN:
E4BC BA00C8                 MOV     DX,0C800H               ; SET BEGINNING ADDRESS
E4BF                ROM_SCAN_1:
E4BF 8EDA                   MOV     DS,DX
E4C1 2BDB                   SUB     BX,BX                   ; SET BX=0000
E4C3 8B07                   MOV     AX,[BX]                 ; GET 1ST WORD FROM MODULE
E4C5 3D55AA                 CMP     AX,0AA55H               ; = TO ID WORD?
E4C8 7505                   JNZ     NEXT_ROM                ; PROCEED TO NEXT ROM IF NOT
E4CA E8B701                 CALL    ROM_CHECK               ; GO DO CHECKSUM AND CALL
E4CD EB04                   JMP     SHORT ARE_WE_DONE       ; CHECK FOR END OF ROM SPACE
E4CF                NEXT_ROM:
E4CF 81C28000               ADD     DX,0080H                ; POINT TO NEXT 2K ADDRESS
E4D3                ARE_WE_DONE:
E4D3 81FA00F6               CMP     DX,0F600H               ; AT F6000 YET?
E4D7 7CE6                   JL      ROM_SCAN_1              ; GO CHECK ANOTHER ADD. IF NOT
E4D9 EB0190                 JMP     BASE_ROM_CHK            ; GO CHECK BASIC ROM
                    ;---------------------------------------------
                    ;           ROS CHECKSUM II                  :
                    ; DESCRIPTION                                :
                    ;           A CHECKSUM IS DONE FOR THE 4 ROS :
                    ;           MODULES CONTAINING BASIC CODE    :
                    ;---------------------------------------------
E4DC                BASE_ROM_CHK:
E4DC                E4:
E4DC 2BDB                   SUB     BX,BX                   ; SETUP STARTING ROS ADDR
E4DE 8EDA                   MOV     DS,DX
E4E0 E86907                 CALL    ROS_CHECKSUM            ; CHECK ROS
```

```
E4E3 7403                      JE      E5                      ; CONTINUE IF OK
E4E5 E82103                    CALL    ROM_ERR                 ; POST ERROR
E4E8                   E5:
E4E8 80C602                    ADD     DH,02H                  ; POINT TO NEXT 8K MOCDULE
E4EB 80FEFE                    CMP     DH,0FEH
E4EE 75EC                      JNZ     E4                      ; YES - CONTINUE
E4F0 1F                        POP     DS                      ; RECOVER DATA SEG PTR
                       ;----------------------------------------------------------------
                       ;       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.            :
                       ;----------------------------------------------------------------
E4F1                   F9:
E4F1 A01000                    MOV     AL,BYTE PTR EQUIP_FLAG  ; GET SENSE SWS INFO
E4F4 A801                      TEST    AL,01H                  ; IPL DISKETTE DRIVE ATTCH?
E4F6 750A                      JNZ     F10                     ; NO -SKIP THIS TEST
E4F8 803E120001                CMP     MFG_TST,1               ; MANUFACTURING TEST MODE?
E4FD 753D                      JNE     F15A                    ; NO - GO TO BOOT LOADER
E4FF E959FB                    JMP     START                   ; YES - LOOP POWER-ON-DIAGS
E502                   F10:
E502 E421                      IN      AL,INTA01               ; DISK_TEST
E504 24BF                      AND     AL,0BFH                 ; ENABLE DISKETTE INTERRUPTS
E506 E621                      OUT     INTA01,AL
E508 B400                      MOV     AH,0                    ; RESET NEC FDC
E50A 8AD4                      MOV     DL,AH                   ; (POINT TO DISKETTE)
E50C CD13                      INT     13H                     ; VERIFY STATUS AFTER RESET
E50E 7221                      JC      F13

                       ;----- TURN DRIVE 0 MOTOR ON

E510 BAF203                    MOV     DX,03F2H                ; GET ADDR OF FDC CARD
E513 52                        PUSH    DX                      ; SAVE IT
E514 B01C                      MOV     AL,1CH                  ; TURN MOTOR ON, EN DMA/INT
E516 EE                        OUT     DX,AL                   ; WRITE FDC CONTROL REG
E517 2BC9                      SUB     CX,CX
E519                   F11:                                    ; MOTOR_WAIT:
E519 E2FE                      LOOP    F11                     ; WAIT FOR 1 SECOND
E51B                   F12:                                    ; MOTOR_WAIT1:
E51B E2FE                      LOOP    F12
E51D 33D2                      XOR     DX,DX                   ; SELECT DRIVE 0
E51F B501                      MOV     CH,1                    ; SELECT TRACK 1
E521 88163E00                  MOV     SEEK_STATUS,DL
E525 E85509                    CALL    SEEK                    ; RECALIBRATE DISKETTE
E528 7207                      JC      F13                     ; GO TO ERR SUBROUTINE IF ERR
E52A B522                      MOV     CH,34                   ; SELECT TRACK 34
E52C E84E09                    CALL    SEEK                    ; SEEK TO TRACK 34
E52F 7307                      JNC     F14                     ; OK, TURN MOTOR OFF
E531                   F13:                                    ; DSK_ERR:
E531 BEEAFF90                  MOV     SI,OFFSET F3            ; GET ADDR OF MSG
E535 E88201                    CALL    P_MSG                   ; GO PRINT ERROR MSG

                       ;----- TURN DRIVE 0 MOTOR OFF

E538                   F14:                                    ; DR0_OFF:
E538 B00C                      MOV     AL,0CH                  ; TURN DRIVE 0 MOTOR OFF
E53A 5A                        POP     DX                      ; RECOVER FDC CTL ADDRESS
E53B EE                        OUT     DX,AL

                       ;----- SETUP PRINTER AND RS232 BASE ADDRESSES IF DEVICE ATTACHED

E53C                   F15A:
E53C BE1E00                    MOV     SI,OFFSET KB_BUFFER
E53F 89361A00                  MOV     BUFFER_HEAD,SI          ; SETUP KEYBOARD PARAMETERS
E543 89361C00                  MOV     BUFFER_TAIL,SI
E547 89368000                  MOV     BUFFER_START,SI         ; DEFAULT TO STANDARD BUFFER
E54B 83C620                    ADD     SI,32                   ; (32 BYTES LONG)
E54E 89368200                  MOV     BUFFER_END,SI
E552 E421                      IN      AL,INTA01
E554 24FC                      AND     AL,0FCH                 ; ENABLE TIMER AND KBD INTS
E556 E621                      OUT     INTA01,AL
E558 BD3DE690                  MOV     BP,OFFSET F4            ; PRT_SRC_TBL
E55C 2BF6                      SUB     SI,SI
E55E                   F16:                                    ; PRT_BASE:
E55E 2E8B5600                  MOV     DX,CS:[BP]              ; GET PRINTER BASE ADDR
```

```
LOC OBJ             SOURCE

E562 B0AA                   MOV     AL,0AAH                 ; WRITE DATA TO PORT A
E564 EE                     OUT     DX,AL
E565 52                     PUSH    DX
E566 EC                     IN      AL,DX                   ; READ PORT A
E567 5A                     POP     DX
E568 3CAA                   CMP     AL,0AAH                 ; DATA PATTERN SAME
E56A 7505                   JNE     F17                     ; NO - CHECK NEXT PRT CD
E56C 895408                 MOV     PRINTER_BASE[SI],DX     ; YES - STORE PRT BASE ADDR
E56F 46                     INC     SI                      ; INCREMENT TO NEXT WORD
E570 46                     INC     SI
E571                F17:                                    ; NO_STORE:
E571 45                     INC     BP                      ; POINT TO NEXT BASE ADDR
E572 45                     INC     BP
E573 81FD43E6               CMP     BP,OFFSET F4E           ; ALL POSSIBLE ADDRS CHECKED?
E577 75E5                   JNE     F16                     ; PRT_BASE
E579 2BDB                   SUB     BX,BX                   ; POINTER TO RS232 TABLE
E57B BAFA03                 MOV     DX,3FAH                 ; CHECK IF RS232 CD 1 ATTCH?
E57E EC                     IN      AL,DX                   ; READ INTR ID REG
E57F A8F8                   TEST    AL,0F8H
E581 7506                   JNZ     F18
E583 C707F803               MOV     RS232_BASE[BX],3F8H     ; SETUP RS232 CD #1 ADDR
E587 43                     INC     BX
E588 43                     INC     BX
E589                F18:
E589 B602                   MOV     DH,02H                  ; CHECK IF RS232 CD 2 ATTCH (AT 2FA)
E58B EC                     IN      AL,DX                   ; READ INTERRUPT ID REG
E58C A8F8                   TEST    AL,0F8H
E58E 7506                   JNZ     F19                     ; BASE_END
E590 C707F802               MOV     RS232_BASE[BX],2F8H     ; SETUP RS232 CD #2
E594 43                     INC     BX
E595 43                     INC     BX

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

E596                F19:                                    ; BASE_END:
E596 8BC6                   MOV     AX,SI                   ; SI HAS 2* NUMBER OF RS232
E598 B103                   MOV     CL,3                    ; SHIFT COUNT
E59A D2C8                   ROR     AL,CL                   ; ROTATE RIGHT 3 POSITIONS
E59C 0AC3                   OR      AL,BL                   ; OR IN THE PRINTER COUNT
E59E A21100                 MOV     BYTE PTR EQUIP_FLAG+1,AL ; STORE AS SECOND BYTE
E5A1 B201                   MOV     DL,01H                  ; DX=201
E5A3 EC                     IN      AL,DX
E5A4 A80F                   TEST    AL,0FH
E5A6 7505                   JNZ     F20                     ; NO_GAME_CARD
E5A8 800E110010             OR      BYTE PTR EQUIP_FLAG+1,16
E5AD                F20:

                    ;----- SET DEFAULT TIMEOUT VALUES FOR PRINTER AND RS232

E5AD 1E                     PUSH    DS
E5AE 07                     POP     ES
E5AF BF7800                 MOV     DI,OFFSET PRINT_TIM_OUT
E5B2 B81414                 MOV     AX,1414H                ; PRINTER DEFAULTS (COUNT=20)
E5B5 AB                     STOSW
E5B6 AB                     STOSW
E5B7 B80101                 MOV     AX,0101H                ; RS232 DEFAULTS=01
E5BA AB                     STOSW
E5BB AB                     STOSW

                    ;----- ENABLE NMI INTERRUPTS

E5BC B080                   MOV     AL,80H                  ; ENABLE NMI INTERRUPTS
E5BE E6A0                   OUT     0A0H,AL
E5C0 803E120001             CMP     MFG_TST,1               ; MFG MODE?
E5C5 7406                   JE      F21                     ; LOAD_BOOT_STRAP
E5C7 BA0100                 MOV     DX,1
E5CA E80200                 CALL    ERR_BEEP                ; BEEP 1 SHORT TONE

E5CD                F21:                                    ; LOAD_BOOT_STRAP:
E5CD CD19                   INT     19H                     ; BOOTSTRAP

                    ;-----------------------------------------------------
                    ;       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                                :
                    ;-------------------------------------------------------------------------
E5CF                ERR_BEEP  PROC  NEAR
E5CF 9C                     PUSHF                           ; SAVE FLAGS
E5D0 FA                     CLI                             ; DISABLE SYSTEM INTERRUPTS
E5D1 1E                     PUSH    DS                      ; SAVE DS REG CONTENTS
E5D2 E86919                 CALL    DDS
E5D5 0AF6                   OR      DH,DH                   ; ANY LONG ONES TO BEEP
E5D7 7418                   JZ      G3                      ; NO, DO THE SHORT ONES
E5D9                G1:                                     ; LONG_BEEP:
E5D9 B306                   MOV     BL,6                    ; COUNTER FOR BEEPS
E5DB E82500                 CALL    BEEP                    ; DO THE BEEP
E5DE E2FE           G2:     LOOP    G2                      ; DELAY BETWEEN BEEPS
E5E0 FECE                   DEC     DH                      ; ANY MORE TO DO
E5E2 75F5                   JNZ     G1                      ; DO IT
E5E4 803E120001             CMP     MFG_TST,1               ; MFG TEST MODE?
E5E9 7506                   JNE     G3                      ; YES - CONTINUE BEEPING SPEAKER
E5EB B0CD                   MOV     AL,0CDH                 ; STOP BLINKING LED
E5ED E661                   OUT     PORT_B,AL
E5EF EBE8                   JMP     SHORT G1
E5F1                G3:                                     ; SHORT_BEEP:
E5F1 B301                   MOV     BL,1                    ; COUNTER FOR A SHORT BEEP
E5F3 E80D00                 CALL    BEEP                    ; DO THE SOUND
E5F6                G4:
E5F6 E2FE                   LOOP    G4                      ; DELAY BETWEEN BEEPS
E5F8 FECA                   DEC     DL                      ; DONE WITH SHORTS
E5FA 75F5                   JNZ     G3                      ; DO SOME MORE
E5FC                G5:
E5FC E2FE                   LOOP    G5                      ; LONG DELAY BEFORE RETURN
E5FE                G6:
E5FE E2FE                   LOOP    G6
E600 1F                     POP     DS                      ; RESTORE ORIG CONTENTS OF DS
E601 9D                     POPF                            ; RESTORE FLAGS TO ORIG SETTINGS
E602 C3                     RET                             ; RETURN TO CALLER
                    ERR_BEEP        ENDP

                    ;----- ROUTINE TO SOUND BEEPER

E603                BEEP    PROC    NEAR
E603 B0B6                   MOV     AL,10110110B            ; SEL TIM 2,LSB,MSB,BINARY
E605 E643                   OUT     TIMER+3,AL              ; WRITE THE TIMER MODE REG
E607 B83305                 MOV     AX,533H                 ; DIVISOR FOR 1000 HZ
E60A E642                   OUT     TIMER+2,AL              ; WRITE TIMER 2 CNT - LSB
E60C 8AC4                   MOV     AL,AH
E60E E642                   OUT     TIMER+2,AL              ; WRITE TIMER 2 CNT - MSB
E610 E461                   IN      AL,PORT_B               ; GET CURRENT SETTING OF PORT
E612 8AE0                   MOV     AH,AL                   ; SAVE THAT SETTING
E614 0C03                   OR      AL,03                   ; TURN SPEAKER ON
E616 E661                   OUT     PORT_B,AL
E618 2BC9                   SUB     CX,CX                   ; SET CNT TO WAIT 500 MS
E61A                G7:
E61A E2FE                   LOOP    G7                      ; DELAY BEFORE TURNING OFF
E61C FECB                   DEC     BL                      ; DELAY CNT EXPIRED?
E61E 75FA                   JNZ     G7                      ; NO - CONTINUE BEEPING SPK
E620 8AC4                   MOV     AL,AH                   ; RECOVER VALUE OF PORT
E622 E661                   OUT     PORT_B,AL
E624 C3                     RET                             ; RETURN TO CALLER
                    BEEP    ENDP

                    ;----------------------------------------------
                    ; CONVERT AND PRINT ASCII CODE                 :
                    ;       AL MUST CONTAIN NUMBER TO BE CONVERTED. :
                    ;       AX AND BX DESTROYED.                   :
                    ;----------------------------------------------
E625                XPC_BYTE        PROC    NEAR
E625 50                     PUSH    AX                      ; RESAVE FOR LOW NIBBLE DISPLAY
E626 B104                   MOV     CL,4                    ; SHIFT COUNT
E628 D2E8                   SHR     AL,CL                   ; NIBBLE SWAP
E62A E80300                 CALL    XLAT_PR                 ; DO THE HIGH NIBBLE DISPLAY
E62D 58                     POP     AX                      ; RECOVER THE NIBBLE
E62E 240F                   AND     AL,0FH                  ; ISOLATE TO LOW NIBBLE
                                                            ; FALL INTO LOW NIBBLE CONVERSION
```

Appendix A

```
LOC OBJ            SOURCE

E630               XLAT_PR PROC    NEAR                  ; CONVERT 00-0F TO ASCII CHARACTER
E630 0490                  ADD     AL,090H               ; ADD FIRST CONVERSION FACTOR
E632 27                    DAA                           ; ADJUST FOR NUMERIC AND ALPHA RANGE
E633 1440                  ADC     AL,040H               ; ADD CONVERSION AND ADJUST LOW NIBBLE
E635 27                    DAA                           ; ADJUST HI NIBBLE TO ASCII RANGE
E636               PRT_HEX PROC    NEAR
E636 B40E                  MOV     AH,14                 ; DISPLAY CHAR. IN AL
E638 B700                  MOV     BH,0
E63A CD10                  INT     10H                   ; CALL VIDEO_IO
E63C C3                    RET
                   PRT_HEX ENDP
                   XLAT_PR ENDP
                   XPC_BYTE        ENDP

E63D               F4      LABEL   WORD                  ; PRINTER SOURCE TABLE
E63D BC03                  DW      3BCH
E63F 7803                  DW      378H
E641 7802                  DW      278H
E643               F4E     LABEL   WORD

                   ;---------------------------------------------------------------------
                   ;       THIS PROCEDURE WILL SEND A SOFTWARE RESET TO THE KEYBOARD.     :
                   ;       SCAN CODE 'AA' SHOULD BE RETURNED TO THE CPU.                  :
                   ;---------------------------------------------------------------------
E643               KBD_RESET       PROC    NEAR
E643 B00C                  MOV     AL,0CH                ; SET KBD CLK LINE LOW
E645 E661                  OUT     PORT_B,AL             ; WRITE 8255 PORT B
E647 B95629                MOV     CX,10582              ; HOLD KBD CLK LOW FOR 20 MS
E64A               G8:
E64A E2FE                  LOOP    G8                    ; LOOP FOR 20 MS
E64C B0CC                  MOV     AL,0CCH               ; SET CLK, ENABLE LINES HIGH
E64E E661                  OUT     PORT_B,AL
E650               SP_TEST:                              ; ENTRY FOR MANUFACTURING TEST 2
E650 B04C                  MOV     AL,4CH                ; SET KBD CLK HIGH, ENABLE LOW
E652 E661                  OUT     PORT_B,AL
E654 B0FD                  MOV     AL,0FDH               ; ENABLE KEYBOARD INTERRUPTS
E656 E621                  OUT     INTA01,AL             ; WRITE 8259 IMR
E658 FB                    STI                           ; ENABLE SYSTEM INTERRUPTS
E659 B400                  MOV     AH,0                  ; RESET INTERRUPT INDICATOR
E65B 2BC9                  SUB     CX,CX                 ; SETUP INTERRUPT TIMEOUT CNT
E65D               G9:
E65D F6C4FF                TEST    AH,0FFH               ; DID A KEYBOARD INTR OCCUR?
E660 7502                  JNZ     G10                   ; YES - READ SCAN CODE RETURNED
E662 E2F9                  LOOP    G9                    ; NO - LOOP TILL TIMEOUT
E664               G10:
E664 E460                  IN      AL,PORT_A             ; READ KEYBOARD SCAN CODE
E666 8AD8                  MOV     BL,AL                 ; SAVE SCAN CODE JUST READ
E668 B0CC                  MOV     AL,0CCH               ; CLEAR KEYBOARD
E66A E661                  OUT     PORT_B,AL
E66C C3                    RET                           ; RETURN TO CALLER
                   KBD_RESET       ENDP

                   ;---------------------------------------------------------------------
                   ;       BLINK LED PROCEDURE FOR MFG BURN-IN AND RUN-IN TESTS           :
                   ;       IF LED IS ON, TURN IT OFF. IF OFF, TURN ON.                    :
                   ;---------------------------------------------------------------------
E66D               BLINK_INT       PROC    NEAR
E66D FB                    STI
E66E 50                    PUSH    AX                    ; SAVE AX REG CONTENTS
E66F E461                  IN      AL,PORT_B             ; READ CURRENT VAL OF PORT B
E671 8AE0                  MOV     AH,AL
E673 F6D0                  NOT     AL                    ; FLIP ALL BITS
E675 2440                  AND     AL,01000000B          ; ISOLATE CONTROL BIT
E677 80E4BF                AND     AH,10111111B          ; MASK OUT OF ORIGINAL VAL
E67A 0AC4                  OR      AL,AH                 ; OR NEW CONTROL BIT IN
E67C E661                  OUT     PORT_B,AL
E67E B020                  MOV     AL,EOI
E680 E620                  OUT     INTA00,AL
E682 58                    POP     AX                    ; RESTORE AX REG
E683 CF                    IRET
                   BLINK_INT       ENDP

                   ;----- CHECKSUM AND CALL INIT CODE IN OPTIONAL ROMS

E684               ROM_CHECK       PROC    NEAR
E684 B84000                MOV     AX,DATA               ; SET ES=DATA
E687 8EC0                  MOV     ES,AX
```

```
LOC OBJ               SOURCE

E689 2AE4                     SUB    AH,AH                 ; ZERO OUT AH
E68B 8A4702                   MOV    AL,[BX+2]             ; GET LENGTH INDICATOR
E68E B109                     MOV    CL,09H                ; MULTIPLY BY 512
E690 D3E0                     SHL    AX,CL
E692 8BC8                     MOV    CX,AX                 ; SET COUNT
E694 51                       PUSH   CX
E695 B104                     MOV    CL,4
E697 D3E8                     SHR    AX,CL
E699 03D0                     ADD    DX,AX                 ; SET POINTER TO NEXT MODULE
E69B 59                       POP    CX

E69C E8B005                   CALL   ROS_CHECKSUM_CNT      ; DO CHECKSUM
E69F 7405                     JZ     ROM_CHECK_1
E6A1 E86501                   CALL   ROM_ERR               ; PRINT ERROR INFO
E6A4 EB13                     JMP    SHORT ROM_CHECK_END
E6A6                  ROM_CHECK_1:
E6A6 52                       PUSH   DX                    ; SAVE POINTER
E6A7 26C70600010300           MOV    ES:IO_ROM_INIT,0003H  ; LOAD OFFSET
E6AE 268C1E0201               MOV    ES:IO_ROM_SEG,DS      ; LOAD SEGMENT
E6B3 26FF1E0001               CALL   DWORD PTR ES:IO_ROM_INIT       ; CALL INIT RTN.
E6B8 5A                       POP    DX
E6B9                  ROM_CHECK_END:
E6B9 C3                       RET
                      ROM_CHECK      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        :
                      ;-----------------------------------------------------
E6BA                  P_MSG   PROC   NEAR
E6BA E88118                   CALL   DDS
E6BD 803E120001               CMP    MFG_TST,1             ; MFG TEST MODE?
E6C2 7505                     JNE    G12                   ; NO - DISPLAY ERROR MSG
E6C4 B601                     MOV    DH,1                  ; YES - SETUP TO BEEP SPEAKER
E6C6 E906FF                   JMP    ERR_BEEP              ; YES - BEEP SPEAKER
E6C9                  G12:                                 ; WRITE_MSG:
E6C9 2E8A04                   MOV    AL,CS:[SI]            ; PUT CHAR IN AL
E6CC 46                       INC    SI                    ; POINT TO NEXT CHAR
E6CD 50                       PUSH   AX                    ; SAVE PRINT CHAR
E6CE E865FF                   CALL   PRT_HEX               ; CALL VIDEO_IO
E6D1 58                       POP    AX                    ; RECOVER PRINT CHAR
E6D2 3C0A                     CMP    AL,10                 ; WAS IT LINE FEED
E6D4 75F3                     JNE    G12                   ; NO,KEEP PRINTING STRING
E6D6 C3                       RET
                      P_MSG   ENDP

E6D7 20524F4D         F3A     DB     ' ROM',13,10
E6DB 0D
E6DC 0A

E6DD                  D_EOI   PROC   NEAR
E6DD 50                       PUSH   AX
E6DE B020                     MOV    AL,20H
E6E0 E620                     OUT    20H,AL
E6E2 58                       POP    AX
E6E3 CF                       IRET
                      D_EOI   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 A     :
                      ;       HARDWARE ERROR CONTROL IS TRANSFERRED TO THE RESIDENT :
                      ;       BASIC ENTRY POINT.                                  :
                      ;                                                           :
                      ; IPL ASSUMPTIONS:                                          :
                      ;       8255 PORT 60H BIT 0 = 1 IF IPL FROM DISKETTE        :
                      ;-----------------------------------------------------------
                              ASSUME  CS:CODE,DS:ABS0
```

```

                    ;----- IPL WAS SUCCESSFUL

E6E4                H4:
E6E4 EA007C0000             JMP     BOOT_LOCN
E6F2                        ORG     0E6F2H
E6F2                BOOT_STRAP      PROC    NEAR
E6F2 FB                     STI                             ; ENABLE INTERRUPTS
E6F3 2BC0                   SUB     AX,AX
E6F5 8ED8                   MOV     DS,AX

                    ;----- RESET DISKETTE PARAMETER TABLE VECTOR

E6F7 C7067800C7EF           MOV     WORD PTR DISK_POINTER,OFFSET DISK_BASE
E6FD 8C0E7A00               MOV     WORD PTR DISK_POINTER+2,CS
E701 A11004                 MOV     AX,DATA_WORD[OFFSET EQUIP_FLAG] ; GET THE EQUIPMENT SWITCHES
E704 A801                   TEST    AL,1                    ; ISOLATE IPL SENSE SWITCH
E706 741E                   JZ      H3                      ; GO TO CASSETTE BASIC ENTRY POINT

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

E708 B90400                 MOV     CX,4                    ; SET RETRY COUNT
E70B                H1:                                     ; IPL_SYSTEM
E70B 51                     PUSH    CX                      ; SAVE RETRY COUNT
E70C B400                   MOV     AH,0                    ; RESET THE DISKETTE SYSTEM
E70E CD13                   INT     13H                     ; DISKETTE_IO
E710 720F                   JC      H2                      ; IF ERROR, TRY AGAIN
E712 B80102                 MOV     AX,201H                 ; READ IN THE SINGLE SECTOR
E715 2BD2                   SUB     DX,DX
E717 8EC2                   MOV     ES,DX
E719 BB007C                 MOV     BX,OFFSET BOOT_LOCN
E71C B90100                 MOV     CX,1                    ; SECTOR 1, TRACK 0
E71F CD13                   INT     13H                     ; DISKETTE_IO
E721 59             H2:     POP     CX                      ; RECOVER RETRY COUNT
E722 73C0                   JNC     H4                      ; CF SET BY UNSUCCESSFUL READ
E724 E2E5                   LOOP    H1                      ; DO IT FOR RETRY TIMES

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

E726                H3:                                     ; CASSETTE_JUMP:
E726 CD18                   INT     18H                     ; USE INTERRUPT VECTOR TO GET TO BASIC
                    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 PARAMETERS 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                                                       :
                    ;                                                                        :
                    ;       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 TRANSMIT THE BYTE OF DATA OVER THE LINE.      :
                    ;                       IF BIT 7 OF AH IS NOT SET, THE REMAINDER OF AH   :
                    ;                       IS SET AS IN A STATUS REQUEST, REFLECTING THE    :
                    ;                       CURRENT STATUS OF THE LINE.                      :
                    ;       (AH)=2  RECEIVE A CHARACTER IN (AL) FROM 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)           :
                    ;                       IF AH HAS BIT 7 ON (TIME OUT) THE REMAINING      :
                    ;                       BITS ARE NOT PREDICTABLE.                        :
                    ;                       THUS, AH IS NON ZERO ONLY WHEN AN ERROR          :
```

```
LOC OBJ          SOURCE

                 ;                       OCCURRED.                                :
                 ;       (AH)=3  RETURN THE COMMO PORT STATUS IN (AX)            :
                 ;               AH CONTAINS THE LINE 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 = RECEIVED 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  :
                 ;       DATA AREA LABEL RS232_TIM_OUT (BYTE) CONTAINS OUTER LOOP COUNT  :
                 ;       VALUE FOR TIMEOUT (DEFAULT=1)                            :
                 ; OUTPUT                                                         :
                 ;       AX MODIFIED ACCORDING TO PARMS OF CALL                   :
                 ;       ALL OTHERS UNCHANGED                                     :
                 ;--------------------------------------------------------------------
                         ASSUME  CS:CODE,DS:DATA
E729                     ORG     0E729H
E729             A1      LABEL   WORD                    ; TABLE OF INIT VALUE
E729 1704                DW      1047                    ; 110 BAUD
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 53                  PUSH    BX
E740 8BF2                MOV     SI,DX                   ; RS232 VALUE TO SI
E742 8BFA                MOV     DI,DX
E744 D1E6                SHL     SI,1                    ; WORD OFFSET
E746 E8F517              CALL    DDS
E749 8B14                MOV     DX,RS232_BASE[SI]       ; GET BASE ADDRESS
E74B 0BD2                OR      DX,DX                   ; TEST FOR 0 BASE ADDRESS
E74D 7413                JZ      A3                      ; RETURN
E74F 0AE4                OR      AH,AH                   ; TEST FOR (AH)=0
E751 7416                JZ      A4                      ; COMMUN INIT
E753 FECC                DEC     AH                      ; TEST FOR (AH)=1
E755 7445                JZ      A5                      ; SEND AL
E757 FECC                DEC     AH                      ; TEST FOR (AH)=2
E759 746A                JZ      A12                     ; RECEIVE INTO AL
E75B             A2:
E75B FECC                DEC     AH                      ; TEST FOR (AH)=3
E75D 7503                JNZ     A3
E75F E98300              JMP     A18                     ; COMMUNICATION STATUS
E762             A3:                                     ; RETURN FROM RS232
E762 5B                  POP     BX
E763 59                  POP     CX
E764 5F                  POP     DI
E765 5E                  POP     SI
E766 5A                  POP     DX
```

```
E767 1F                      POP     DS
E768 CF                      IRET                        ; RETURN TO CALLER, NO ACTION

                     ;----- INITIALIZE THE COMMUNICATIONS PORT

E769                 A4:
E769 8AE0                    MOV     AH,AL               ; SAVE INIT PARMS IN AH
E76B 83C203                  ADD     DX,3                ; POINT TO 8250 CONTROL REGISTER
E76E B080                    MOV     AL,80H
E770 EE                      OUT     DX,AL               ; SET DLAB=1

                     ;----- DETERMINE BAUD RATE DIVISOR

E771 8AD4                    MOV     DL,AH               ; GET PARMS TO DL
E773 B104                    MOV     CL,4
E775 D2C2                    ROL     DL,CL
E777 81E20E00                AND     DX,0EH              ; ISOLATE THEM
E77B BF29E7                  MOV     DI,OFFSET A1        ; BASE OF TABLE
E77E 03FA                    ADD     DI,DX               ; PUT INTO INDEX REGISTER
E780 8B14                    MOV     DX,RS232_BASE[SI]   ; POINT TO HIGH ORDER OF DIVISOR
E782 42                      INC     DX
E783 2E8A4501                MOV     AL,CS:[DI]+1        ; GET HIGH ORDER OF DIVISOR
E787 EE                      OUT     DX,AL               ; SET MS OF DIV TO 0
E788 4A                      DEC     DX
E789 2E8A05                  MOV     AL,CS:[DI]          ; GET LOW ORDER OF DIVISOR
E78C EE                      OUT     DX,AL               ; SET LOW OF DIVISOR
E78D 83C203                  ADD     DX,3
E790 8AC4                    MOV     AL,AH               ; GET PARMS BACK
E792 241F                    AND     AL,01FH             ; STRIP OFF THE BAUD BITS
E794 EE                      OUT     DX,AL               ; LINE CONTROL TO 8 BITS
E795 4A                      DEC     DX
E796 4A                      DEC     DX
E797 B000                    MOV     AL,0
E799 EE                      OUT     DX,AL               ; INTERRUPT ENABLES ALL OFF
E79A EB49                    JMP     SHORT A18           ; COM_STATUS

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

E79C                 A5:
E79C 50                      PUSH    AX                  ; SAVE CHAR TO SEND
E79D 83C204                  ADD     DX,4                ; MODEM CONTROL REGISTER
E7A0 B003                    MOV     AL,3                ; DTR AND RTS
E7A2 EE                      OUT     DX,AL               ; DATA TERMINAL READY, REQUEST TO SEND
E7A3 42                      INC     DX                  ; MODEM STATUS REGISTER
E7A4 42                      INC     DX
E7A5 B730                    MOV     BH,30H              ; DATA SET READY & CLEAR TO SEND
E7A7 E84800                  CALL    WAIT_FOR_STATUS     ; ARE BOTH TRUE
E7AA 7408                    JE      A9                  ; YES, READY TO TRANSMIT CHAR
E7AC                 A7:
E7AC 59                      POP     CX
E7AD 8AC1                    MOV     AL,CL               ; RELOAD DATA BYTE
E7AF                 A8:
E7AF 80CC80                  OR      AH,80H              ; INDICATE TIME OUT
E7B2 EBAE                    JMP     A3                  ; RETURN
E7B4                 A9:                                 ; CLEAR_TO_SEND
E7B4 4A                      DEC     DX                  ; LINE STATUS REGISTER
E7B5                 A10:                                ; WAIT_SEND
E7B5 B720                    MOV     BH,20H              ; IS TRANSMITTER READY
E7B7 E83800                  CALL    WAIT_FOR_STATUS     ; TEST FOR TRANSMITTER READY
E7BA 75F0                    JNZ     A7                  ; RETURN WITH TIME OUT SET
E7BC                 A11:                                ; OUT_CHAR
E7BC 83EA05                  SUB     DX,5                ; DATA PORT
E7BF 59                      POP     CX                  ; RECOVER IN CX TEMPORARILY
E7C0 8AC1                    MOV     AL,CL               ; MOVE CHAR TO AL FOR OUT, STATUS IN AH
E7C2 EE                      OUT     DX,AL               ; OUTPUT CHARACTER
E7C3 EB9D                    JMP     A3                  ; RETURN

                     ;----- RECEIVE CHARACTER FROM COMMO LINE

E7C5                 A12:
E7C5 83C204                  ADD     DX,4                ; MODEM CONTROL REGISTER
E7C8 B001                    MOV     AL,1                ; DATA TERMINAL READY
E7CA EE                      OUT     DX,AL
E7CB 42                      INC     DX                  ; MODEM STATUS REGISTER
E7CC 42                      INC     DX
E7CD                 A13:                                ; WAIT_DSR
E7CD B720                    MOV     BH,20H              ; DATA SET READY
```

```
LOC OBJ             SOURCE

E7CF E82000                 CALL    WAIT_FOR_STATUS         ; TEST FOR DSR
E7D2 750B                   JNZ     A8                      ; RETURN WITH ERROR
E7D4                A15:                                    ; WAIT_DSR_END
E7D4 4A                     DEC     DX                      ; LINE STATUS REGISTER
E7D5                A16:                                    ; WAIT_RECV
E7D5 B701                   MOV     BH,1                    ; RECEIVE BUFFER FULL
E7D7 E81800                 CALL    WAIT_FOR_STATUS         ; TEST FOR REC. BUFF. FULL
E7DA 75D3                   JNZ     A8                      ; SET TIME OUT ERROR
E7DC                A17:                                    ; GET_CHAR
E7DC 80E41E                 AND     AH,00011110B            ; TEST FOR ERR CONDITIONS ON RECV CHAR
E7DF 8B14                   MOV     DX,RS232_BASE[SI]       ; DATA PORT
E7E1 EC                     IN      AL,DX                   ; GET CHARACTER FROM LINE
E7E2 E97DFF                 JMP     A3                      ; RETURN

                    ;----- COMMO PORT STATUS ROUTINE

E7E5                A18:
E7E5 8B14                   MOV     DX,RS232_BASE[SI]
E7E7 83C205                 ADD     DX,5                    ; CONTROL PORT
E7EA EC                     IN      AL,DX                   ; GET LINE CONTROL STATUS
E7EB 8AE0                   MOV     AH,AL                   ; PUT IN AH FOR RETURN
E7ED 42                     INC     DX                      ; POINT TO MODEM STATUS REGISTER
E7EE EC                     IN      AL,DX                   ; GET MODEM CONTROL STATUS
E7EF E970FF                 JMP     A3                      ; RETURN
                    ;-----------------------------------------
                    ; WAIT FOR STATUS ROUTINE               :
                    ;                                       :
                    ; ENTRY:                                :
                    ;       BH=STATUS BIT(S) TO LOOK FOR,   :
                    ;       DX=ADDR. OF STATUS REG          :
                    ; EXIT:                                 :
                    ;       ZERO FLAG ON  = STATUS FOUND    :
                    ;       ZERO FLAG OFF = TIMEOUT.        :
                    ;       AH=LAST STATUS READ             :
                    ;-----------------------------------------
E7F2                WAIT_FOR_STATUS PROC    NEAR
E7F2 8A5D7C                 MOV     BL,RS232_TIM_OUT[DI]    ; LOAD OUTER LOOP COUNT
E7F5                WFS0:
E7F5 2BC9                   SUB     CX,CX
E7F7                WFS1:
E7F7 EC                     IN      AL,DX                   ; GET STATUS
E7F8 8AE0                   MOV     AH,AL                   ; MOVE TO AH
E7FA 22C7                   AND     AL,BH                   ; ISOLATE BITS TO TEST
E7FC 3AC7                   CMP     AL,BH                   ; EXACTLY = TO MASK
E7FE 7408                   JE      WFS_END                 ; RETURN WITH ZERO FLAG ON
E800 E2F5                   LOOP    WFS1                    ; TRY AGAIN
E802 FECB                   DEC     BL
E804 75EF                   JNZ     WFS0
E806 0AFF                   OR      BH,BH                   ; SET ZERO FLAG OFF
E808                WFS_END:
E808 C3                     RET
                    WAIT_FOR_STATUS ENDP
                    RS232_IO        ENDP

                    ;--------------------------------------------------------------------
                    ;       PRINT ADDRESS AND ERROR MESSAGE FOR ROM CHECKSUM ERRORS     :
                    ;--------------------------------------------------------------------
E809                ROM_ERR PROC    NEAR
E809 52                     PUSH    DX                      ; SAVE POINTER
E80A 50                     PUSH    AX
E80B 8CDA                   MOV     DX,DS                   ; GET ADDRESS POINTER
E80D 81FA00C8               CMP     DX,0C800H
E811 7E13                   JLE     ROM_ERR_BEEP            ; SPECIAL ERROR INDICATION
E813 8AC6                   MOV     AL,DH
E815 E80DFE                 CALL    XPC_BYTE                ; DISPLAY ADDRESS
E818 8AC2                   MOV     AL,DL
E81A E808FE                 CALL    XPC_BYTE
E81D BED7E6                 MOV     SI,OFFSET F3A           ; DISPLAY ERROR MSG
E820 E897FE                 CALL    P_MSG
E823                ROM_ERR_END:
E823 58                     POP     AX
E824 5A                     POP     DX
E825 C3                     RET
E826                ROM_ERR_BEEP:
E826 BA0201                 MOV     DX,0102H                ; BEEP 1 LONG, 2 SHORT
E829 E8A3FD                 CALL    ERR_BEEP
E82C EBF5                   JMP     SHORT ROM_ERR_END
```

```
                 ROM_ERR 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 PRESERVED                                            :
                 ;-------------------------------------------------------------------------
                         ASSUME  CS:CODE,DS:DATA
E82E                     ORG     0E82EH
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 E80A17              CALL    DDS
E834 0AE4                OR      AH,AH                   ; AH=0
E836 740A                JZ      K1                      ; ASCII_READ
E838 FECC                DEC     AH                      ; AH=1
E83A 741E                JZ      K2                      ; ASCII_STATUS
E83C FECC                DEC     AH                      ; AH=2
E83E 742B                JZ      K3                      ; SHIFT_STATUS
E840 EB2C                JMP     SHORT INT10_END         ; EXIT

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

E842             K1:                                     ; ASCII READ
E842 FB                  STI                             ; INTERRUPTS BACK ON DURING LOOP
E843 90                  NOP                             ; ALLOW AN INTERRUPT TO OCCUR
E844 FA                  CLI                             ; INTERRUPTS BACK OFF
E845 8B1E1A00            MOV     BX,BUFFER_HEAD          ; GET POINTER TO HEAD OF BUFFER
E849 3B1E1C00            CMP     BX,BUFFER_TAIL          ; TEST END OF BUFFER
E84D 74F3                JZ      K1                      ; LOOP UNTIL SOMETHING IN BUFFER
E84F 8B07                MOV     AX,[BX]                 ; GET SCAN CODE AND ASCII CODE
E851 E81D00              CALL    K4                      ; MOVE POINTER TO NEXT POSITION
E854 891E1A00            MOV     BUFFER_HEAD,BX          ; STORE VALUE IN VARIABLE
E858 EB14                JMP     SHORT INT10_END         ; RETURN

                 ;----- ASCII STATUS

E85A             K2:
E85A FA                  CLI                             ; INTERRUPTS OFF
E85B 8B1E1A00            MOV     BX,BUFFER_HEAD          ; GET HEAD POINTER
E85F 3B1E1C00            CMP     BX,BUFFER_TAIL          ; IF EQUAL (Z=1) THEN NOTHING THERE
E863 8B07                MOV     AX,[BX]
E865 FB                  STI                             ; INTERRUPTS BACK ON
E866 5B                  POP     BX                      ; RECOVER REGISTER
E867 1F                  POP     DS                      ; RECOVER SEGMENT
E868 CA0200              RET     2                       ; THROW AWAY FLAGS

                 ;----- SHIFT STATUS

E86B             K3:
E86B A01700              MOV     AL,KB_FLAG              ; GET THE SHIFT STATUS FLAGS
E86E             INT10_END:
E86E 5B                  POP     BX                      ; RECOVER REGISTER
E86F 1F                  POP     DS                      ; RECOVER REGISTERS
E870 CF                  IRET                            ; RETURN TO CALLER
                 KEYBOARD_IO     ENDP

                 ;----- INCREMENT A BUFFER POINTER

E871             K4      PROC    NEAR
E871 43                  INC     BX                      ; MOVE TO NEXT WORD IN LIST
E872 43                  INC     BX
```

```
LOC OBJ             LINE    SOURCE

E873 3B1E8200       1893            CMP     BX,BUFFER_END           ; AT END OF BUFFER?
E877 7504           1894            JNE     K5                      ; NO, CONTINUE
E879 8B1E8000       1895            MOV     BX,BUFFER_START         ; YES, RESET TO BUFFER BEGINNING
E87D                1896    K5:
E87D C3             1897            RET
                    1898    K4      ENDP
                    1899
                    1900    ;----- TABLE OF SHIFT KEYS AND MASK VALUES
                    1901
E87E                1902    K6      LABEL   BYTE
E87E 52             1903            DB      INS_KEY                 ; INSERT KEY
E87F 3A             1904            DB      CAPS_KEY,NUM_KEY,SCROLL_KEY,ALT_KEY,CTL_KEY
E880 45
E881 46
E882 38
E883 1D
E884 2A             1905            DB      LEFT_KEY,RIGHT_KEY
E885 36
  0008              1906    K6L     EQU     $-K6
                    1907
                    1908    ;----- SHIFT MASK TABLE
                    1909
E886                1910    K7      LABEL   BYTE
E886 80             1911            DB      INS_SHIFT               ; INSERT MODE SHIFT
E887 40             1912            DB      CAPS_SHIFT,NUM_SHIFT,SCROLL_SHIFT,ALT_SHIFT,CTL_SHIFT
E888 20
E889 10
E88A 08
E88B 04
E88C 02             1913            DB      LEFT_SHIFT,RIGHT_SHIFT
E88D 01
                    1914
                    1915    ;----- SCAN CODE TABLES
                    1916
E88E 1B             1917    K8              DB      27,-1,0,-1,-1,-1,30,-1
E88F FF
E890 00
E891 FF
E892 FF
E893 FF
E894 1E
E895 FF
E896 FF             1918                    DB      -1,-1,-1,31,-1,127,-1,17
E897 FF
E898 FF
E899 1F
E89A FF
E89B 7F
E89C FF
E89D 11
E89E 17             1919                    DB      23,5,18,20,25,21,9,15
E89F 05
E8A0 12
E8A1 14
E8A2 19
E8A3 15
E8A4 09
E8A5 0F
E8A6 10             1920                    DB      16,27,29,10,-1,1,19
E8A7 1B
E8A8 1D
E8A9 0A
E8AA FF
E8AB 01
E8AC 13
E8AD 04             1921                    DB      4,6,7,8,10,11,12,-1,-1
E8AE 06
E8AF 07
E8B0 08
E8B1 0A
E8B2 0B
E8B3 0C
E8B4 FF
E8B5 FF
E8B6 FF             1922                    DB      -1,-1,28,26,24,3,22,2
E8B7 FF
E8B8 1C
```

```
LOC OBJ                  LINE    SOURCE

E8B9 1A
E8BA 18
E8BB 03
E8BC 16
E8BD 02
E8BE 0E                  1923                    DB      14,13,-1,-1,-1,-1,-1,-1
E8BF 0D
E8C0 FF
E8C1 FF
E8C2 FF
E8C3 FF
E8C4 FF
E8C5 FF
E8C6 20                  1924                    DB      ' ',-1
E8C7 FF
                         1925    ;----- CTL TABLE SCAN
E8C8                     1926    K9      LABEL   BYTE
E8C8 5E                  1927                    DB      94,95,96,97,98,99,100,101
E8C9 5F
E8CA 60
E8CB 61
E8CC 62
E8CD 63
E8CE 64
E8CF 65
E8D0 66                  1928                    DB      102,103,-1,-1,119,-1,132,-1
E8D1 67
E8D2 FF
E8D3 FF
E8D4 77
E8D5 FF
E8D6 84
E8D7 FF
E8D8 73                  1929                    DB      115,-1,116,-1,117,-1,118,-1
E8D9 FF
E8DA 74
E8DB FF
E8DC 75
E8DD FF
E8DE 76
E8DF FF
E8E0 FF                  1930                    DB      -1
                         1931    ;----- LC TABLE
E8E1                     1932    K10     LABEL   BYTE
E8E1 1B                  1933                    DB      01BH,'1234567890-=',08H,09H
E8E2 31323334353637
     3839302D3D
E8EE 08
E8EF 09
E8F0 71776572747975      1934                    DB      'qwertyuiop[]',0DH,-1,'asdfghjkl;',027H
     696F705B5D
E8FC 0D
E8FD FF
E8FE 6173646667686A
     6B6C3B
E908 27
E909 60                  1935                    DB      60H,-1,5CH,'zxcvbnm,./',-1,'*',-1,' '
E90A FF
E90B 5C
E90C 7A786376626E6D
     2C2E2F
E916 FF
E917 2A
E918 FF
E919 20
E91A FF                  1936                    DB      -1
                         1937    ;----- UC TABLE
E91B                     1938    K11     LABEL   BYTE
E91B 1B                  1939                    DB      27,'!@#$',37,05EH,'&*()_+',08H,0
E91C 21402324
E920 25
E921 5E
E922 262A28295F2B
E928 08
E929 00
E92A 51574552545955      1940                    DB      'QWERTYUIOP{}',0DH,-1,'ASDFGHJKL:"'
     494F507B7D
```

```
LOC OBJ                 LINE    SOURCE

E936 0D
E937 FF
E938 4153444647484A
     4B4C3A22
E943 7E                 1941                    DB      07EH,-1,'|ZXCVBNM<>?',-1,0,-1,' ',-1
E944 FF
E945 7C5A584356424E
     4D3C3E3F
E950 FF
E951 00
E952 FF
E953 20
E954 FF
                        1942    ;----- UC TABLE SCAN
E955                    1943    K12     LABEL   BYTE
E955 54                 1944                    DB      84,85,86,87,88,89,90
E956 55
E957 56
E958 57
E959 58
E95A 59
E95B 5A
E95C 5B                 1945                    DB      91,92,93
E95D 5C
E95E 5D
                        1946    ;----- ALT TABLE SCAN
E95F                    1947    K13     LABEL   BYTE
E95F 68                 1948                    DB      104,105,106,107,108
E960 69
E961 6A
E962 6B
E963 6C
E964 6D                 1949                    DB      109,110,111,112,113
E965 6E
E966 6F
E967 70
E968 71
                        1950    ;----- NUM STATE TABLE
E969                    1951    K14     LABEL   BYTE
E969 3738392D343536     1952                    DB      '789-456+1230.'
     2B313233302E
                        1953    ;----- BASE CASE TABLE
E976                    1954    K15     LABEL   BYTE
E976 47                 1955                    DB      71,72,73,-1,75,-1,77
E977 48
E978 49
E979 FF
E97A 4B
E97B FF
E97C 4D
E97D FF                 1956                    DB      -1,79,80,81,82,83
E97E 4F
E97F 50
E980 51
E981 52
E982 53
                        1957
                        1958    ;----- KEYBOARD INTERRUPT ROUTINE
                        1959
E987                    1960            ORG     0E987H
E987                    1961    KB_INT  PROC    FAR
E987 FB                 1962            STI                             ; ALLOW FURTHER INTERRUPTS
E988 50                 1963            PUSH    AX
E989 53                 1964            PUSH    BX
E98A 51                 1965            PUSH    CX
E98B 52                 1966            PUSH    DX
E98C 56                 1967            PUSH    SI
E98D 57                 1968            PUSH    DI
E98E 1E                 1969            PUSH    DS
E98F 06                 1970            PUSH    ES
E990 FC                 1971            CLD                             ; FORWARD DIRECTION
E991 E8AA15             1972            CALL    DDS
E994 E460               1973            IN      AL,KB_DATA              ; READ IN THE CHARACTER
E996 50                 1974            PUSH    AX                      ; SAVE IT
E997 E461               1975            IN      AL,KB_CTL               ; GET THE CONTROL PORT
E999 8AE0               1976            MOV     AH,AL                   ; SAVE VALUE
E99B 0C80               1977            OR      AL,80H                  ; RESET BIT FOR KEYBOARD
```

```
LOC OBJ                  LINE    SOURCE

E99D E661                1978            OUT    KB_CTL,AL
E99F 86E0                1979            XCHG   AH,AL                   ; GET BACK ORIGINAL CONTROL
E9A1 E661                1980            OUT    KB_CTL,AL               ; KB HAS BEEN RESET
E9A3 58                  1981            POP    AX                      ; RECOVER SCAN CODE
E9A4 8AE0                1982            MOV    AH,AL                   ; SAVE SCAN CODE IN AH ALSO
                         1983
                         1984    ;----- TEST FOR OVERRUN SCAN CODE FROM KEYBOARD
                         1985
E9A6 3CFF                1986            CMP    AL,0FFH                 ; IS THIS AN OVERRUN CHAR
E9A8 7503                1987            JNZ    K16                     ; NO, TEST FOR SHIFT KEY
E9AA E97A02              1988            JMP    K62                     ; BUFFER_FULL_BEEP
                         1989
                         1990    ;----- TEST FOR SHIFT KEYS
                         1991
E9AD                     1992    K16:                                   ; TEST_SHIFT
E9AD 247F                1993            AND    AL,07FH                 ; TURN OFF THE BREAK BIT
E9AF 0E                  1994            PUSH   CS
E9B0 07                  1995            POP    ES                      ; ESTABLISH ADDRESS OF SHIFT TABLE
E9B1 BF7EE8              1996            MOV    DI,OFFSET K6            ; SHIFT KEY TABLE
E9B4 B90800              1997            MOV    CX,K6L                  ; LENGTH
E9B7 F2                  1998            REPNE  SCASB                   ; LOOK THROUGH THE TABLE FOR A MATCH
E9B8 AE
E9B9 8AC4                1999            MOV    AL,AH                   ; RECOVER SCAN CODE
E9BB 7403                2000            JE     K17                     ; JUMP IF MATCH FOUND
E9BD E98500              2001            JMP    K25                     ; IF NO MATCH, THEN SHIFT NOT FOUND
                         2002
                         2003    ;----- SHIFT KEY FOUND
                         2004
E9C0 81EF7FE8            2005    K17:    SUB    DI,OFFSET K6+1          ; ADJUST PTR TO SCAN CODE MTCH
E9C4 2E8AA586E8          2006            MOV    AH,CS:K7[DI]            ; GET MASK INTO AH
E9C9 A880                2007            TEST   AL,80H                  ; TEST FOR BREAK KEY
E9CB 7551                2008            JNZ    K23                     ; BREAK_SHIFT_FOUND
                         2009
                         2010    ;----- SHIFT MAKE FOUND, DETERMINE SET OR TOGGLE
                         2011
E9CD 80FC10              2012            CMP    AH,SCROLL_SHIFT
E9D0 7307                2013            JAE    K18                     ; IF SCROLL SHIFT OR ABOVE, TOGGLE KEY
                         2014
                         2015    ;----- PLAIN SHIFT KEY, SET SHIFT ON
                         2016
E9D2 08261700            2017            OR     KB_FLAG,AH              ; TURN ON SHIFT BIT
E9D6 E98000              2018            JMP    K26                     ; INTERRUPT_RETURN
                         2019
                         2020    ;----- TOGGLED SHIFT KEY, TEST FOR 1ST MAKE OR NOT
                         2021
E9D9                     2022    K18:                                   ; SHIFT-TOGGLE
E9D9 F606170004          2023            TEST   KB_FLAG, CTL_SHIFT      ; CHECK CTL SHIFT STATE
E9DE 7565                2024            JNZ    K25                     ; JUMP IF CTL STATE
E9E0 3C52                2025            CMP    AL, INS_KEY             ; CHECK FOR INSERT KEY
E9E2 7522                2026            JNZ    K22                     ; JUMP IF NOT INSERT KEY
E9E4 F606170008          2027            TEST   KB_FLAG, ALT_SHIFT      ; CHECK FOR ALTERNATE SHIFT
E9E9 755A                2028            JNZ    K25                     ; JUMP IF ALTERNATE SHIFT
E9EB F606170020          2029    K19:    TEST   KB_FLAG, NUM_STATE      ; CHECK FOR BASE STATE
E9F0 750D                2030            JNZ    K21                     ; JUMP IF NUM LOCK IS ON
E9F2 F606170003          2031            TEST   KB_FLAG, LEFT_SHIFT+ RIGHT_SHIFT
E9F7 7400                2032            JZ     K22                     ; JUMP IF BASE STATE
                         2033
E9F9                     2034    K20:                                   ; NUMERIC ZERO, NOT INSERT KEY
E9F9 B83052              2035            MOV    AX, 5230H               ; PUT OUT AN ASCII ZERO
E9FC E9D601              2036            JMP    K57                     ; BUFFER_FILL
E9FF                     2037    K21:                                   ; MIGHT BE NUMERIC
E9FF F606170003          2038            TEST   KB_FLAG, LEFT_SHIFT+ RIGHT_SHIFT
EA04 74F3                2039            JZ     K20                     ; JUMP NUMERIC, NOT INSERT
                         2040
EA06                     2041    K22:                                   ; SHIFT TOGGLE KEY HIT; PROCESS IT
EA06 84261800            2042            TEST   AH,KB_FLAG_1            ; IS KEY ALREADY DEPRESSED
EA0A 754D                2043            JNZ    K26                     ; JUMP IF KEY ALREADY DEPRESSED
EA0C 08261800            2044            OR     KB_FLAG_1,AH            ; INDICATE THAT THE KEY IS DEPRESSED
EA10 30261700            2045            XOR    KB_FLAG,AH              ; TOGGLE THE SHIFT STATE
EA14 3C52                2046            CMP    AL,INS_KEY              ; TEST FOR 1ST MAKE OF INSERT KEY
EA16 7541                2047            JNE    K26                     ; JUMP IF NOT INSERT KEY
EA18 B80052              2048            MOV    AX,INS_KEY*256          ; SET SCAN CODE INTO AH, 0 INTO AL
EA1B E9B701              2049            JMP    K57                     ; PUT INTO OUTPUT BUFFER
                         2050
                         2051    ;----- BREAK SHIFT FOUND
                         2052
EA1E                     2053    K23:                                   ; BREAK-SHIFT-FOUND
```

```
LOC OBJ                   LINE    SOURCE

EA1E 80FC10                                CMP     AH,SCROLL_SHIFT         ; IS THIS A TOGGLE KEY
EA21 731A                                  JAE     K24                     ; YES, HANDLE BREAK TOGGLE
EA23 F6D4                                  NOT     AH                      ; INVERT MASK
EA25 20261700                              AND     KB_FLAG,AH              ; TURN OFF SHIFT BIT
EA29 3CB8                                  CMP     AL,ALT_KEY+80H          ; IS THIS ALTERNATE SHIFT RELEASE
EA2B 752C                                  JNE     K26                     ; INTERRUPT_RETURN

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

EA2D A01900                                MOV     AL,ALT_INPUT
EA30 B400                                  MOV     AH,0                    ; SCAN CODE OF 0
EA32 88261900                              MOV     ALT_INPUT,AH            ; ZERO OUT THE FIELD
EA36 3C00                                  CMP     AL,0                    ; WAS THE INPUT=0
EA38 741F                                  JE      K26                     ; INTERRUPT_RETURN
EA3A E9A101                                JMP     K58                     ; IT WASN'T, SO PUT IN BUFFER
EA3D                              K24:                                     ; BREAK-TOGGLE
EA3D F6D4                                  NOT     AH                      ; INVERT MASK
EA3F 20261800                              AND     KB_FLAG_1,AH            ; INDICATE NO LONGER DEPRESSED
EA43 EB14                                  JMP     SHORT K26               ; INTERRUPT_RETURN

                                  ;----- TEST FOR HOLD STATE

EA45                              K25:                                     ; NO-SHIFT-FOUND
EA45 3C80                                  CMP     AL,80H                  ; TEST FOR BREAK KEY
EA47 7310                                  JAE     K26                     ; NOTHING FOR BREAK CHARS FROM HERE ON
EA49 F606180008                            TEST    KB_FLAG_1,HOLD_STATE    ; ARE WE IN HOLD STATE
EA4E 7417                                  JZ      K28                     ; BRANCH AROUND TEST IF NOT
EA50 3C45                                  CMP     AL,NUM_KEY
EA52 7405                                  JE      K26                     ; CAN'T END HOLD ON NUM_LOCK
EA54 80261800F7                            AND     KB_FLAG_1,NOT HOLD_STATE       ; TURN OFF THE HOLD STATE BIT
EA59                              K26:                                     ; INTERRUPT-RETURN
EA59 FA                                    CLI                             ; TURN OFF INTERRUPTS
EA5A B020                                  MOV     AL,EOI                  ; END OF INTERRUPT COMMAND
EA5C E620                                  OUT     020H,AL                 ; SEND COMMAND TO INT CONTROL PORT
EA5E                              K27:                                     ; INTERRUPT-RETURN-NO-EOI
EA5E 07                                    POP     ES
EA5F 1F                                    POP     DS
EA60 5F                                    POP     DI
EA61 5E                                    POP     SI
EA62 5A                                    POP     DX
EA63 59                                    POP     CX
EA64 5B                                    POP     BX
EA65 58                                    POP     AX                      ; RESTORE STATE
EA66 CF                                    IRET                            ; RETURN, INTERRUPTS BACK ON
                                                                           ;  WITH FLAG CHANGE

                                  ;----- NOT IN  HOLD STATE, TEST FOR SPECIAL CHARS

EA67                              K28:                                     ; NO-HOLD-STATE
EA67 F606170008                            TEST    KB_FLAG,ALT_SHIFT       ; ARE WE IN ALTERNATE SHIFT
EA6C 7503                                  JNZ     K29                     ; JUMP IF ALTERNATE SHIFT
EA6E E99100                                JMP     K38                     ; JUMP IF NOT ALTERNATE

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

EA71                              K29:                                     ; TEST-RESET
EA71 F606170004                            TEST    KB_FLAG,CTL_SHIFT       ; ARE WE IN CONTROL SHIFT ALSO
EA76 7433                                  JZ      K31                     ; NO_RESET
EA78 3C53                                  CMP     AL,DEL_KEY              ; SHIFT STATE IS THERE, TEST KEY
EA7A 752F                                  JNE     K31                     ; NO_RESET

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

EA7C C70672003412                          MOV     RESET_FLAG, 1234H       ; SET FLAG FOR RESET FUNCTION
EA82 EA5BE000F0                            JMP     RESET                   ; JUMP TO POWER ON DIAGNOSTICS

                                  ;----- ALT-INPUT-TABLE
EA87                              K30      LABEL   BYTE
EA87 52                                    DB      82,79,80,81,75,76,77
EA88 4F
EA89 50
EA8A 51
EA8B 4B
EA8C 4C
EA8D 4D
EA8E 47                                    DB      71,72,73                ; 10 NUMBERS ON KEYPAD
EA8F 48
```

```
EA90 49
                 ;----- SUPER-SHIFT-TABLE
EA91 10                   DB      16,17,18,19,20,21,22,23 ; A-Z TYPEWRITER CHARS
EA92 11
EA93 12
EA94 13
EA95 14
EA96 15
EA97 16
EA98 17
EA99 18                   DB      24,25,30,31,32,33,34,35
EA9A 19
EA9B 1E
EA9C 1F
EA9D 20
EA9E 21
EA9F 22
EAA0 23
EAA1 24                   DB      36,37,38,44,45,46,47,48
EAA2 25
EAA3 26
EAA4 2C
EAA5 2D
EAA6 2E
EAA7 2F
EAA8 30
EAA9 31                   DB      49,50
EAAA 32

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

EAAB             K31:                                     ; NO-RESET
EAAB 3C39                 CMP     AL,57                   ; TEST FOR SPACE KEY
EAAD 7505                 JNE     K32                     ; NOT THERE
EAAF B020                 MOV     AL,' '                  ; SET SPACE CHAR
EAB1 E92101               JMP     K57                     ; BUFFER_FILL

                 ;----- LOOK FOR KEY PAD ENTRY

EAB4             K32:                                     ; ALT-KEY-PAD
EAB4 BF87EA               MOV     DI,OFFSET K30           ; ALT-INPUT-TABLE
EAB7 B90A00               MOV     CX,10                   ; LOOK FOR ENTRY USING KEYPAD
EABA F2                   REPNE   SCASB                   ; LOOK FOR MATCH
EABB AE
EABC 7512                 JNE     K33                     ; NO_ALT_KEYPAD
EABE 81EF88EA             SUB     DI,OFFSET K30+1         ; DI NOW HAS ENTRY VALUE
EAC2 A01900               MOV     AL,ALT_INPUT            ; GET THE CURRENT BYTE
EAC5 B40A                 MOV     AH,10                   ; MULTIPLY BY 10
EAC7 F6E4                 MUL     AH
EAC9 03C7                 ADD     AX,DI                   ; ADD IN THE LATEST ENTRY
EACB A21900               MOV     ALT_INPUT,AL            ; STORE IT AWAY
EACE EB89                 JMP     K26                     ; THROW AWAY THAT KEYSTROKE

                 ;----- LOOK FOR SUPERSHIFT ENTRY

EAD0             K33:                                     ; NO-ALT-KEYPAD
EAD0 C606190000           MOV     ALT_INPUT,0             ; ZERO ANY PREVIOUS ENTRY INTO INPUT
EAD5 B91A00               MOV     CX,26                   ; DI,ES ALREADY POINTING
EAD8 F2                   REPNE   SCASB                   ; LOOK FOR MATCH IN ALPHABET
EAD9 AE
EADA 7505                 JNE     K34                     ; NOT FOUND, FUNCTION KEY OR OTHER
EADC B000                 MOV     AL,0                    ; ASCII CODE OF ZERO
EADE E9F400               JMP     K57                     ; PUT IT IN THE BUFFER

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

EAE1             K34:                                     ; ALT-TOP-ROW
EAE1 3C02                 CMP     AL,2                    ; KEY WITH '1' ON IT
EAE3 720C                 JB      K35                     ; NOT ONE OF INTERESTING KEYS
EAE5 3C0E                 CMP     AL,14                   ; IS IT IN THE REGION
EAE7 7308                 JAE     K35                     ; ALT-FUNCTION
EAE9 80C476               ADD     AH,118                  ; CONVERT PSUEDO SCAN CODE TO RANGE
EAEC B000                 MOV     AL,0                    ; INDICATE AS SUCH
EAEE E9E400               JMP     K57                     ; BUFFER_FILL

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

```

```
LOC OBJ               LINE    SOURCE

EAF1                          K35:                                  ; ALT-FUNCTION
EAF1 3C3B                             CMP    AL,59                  ; TEST FOR IN TABLE
EAF3 7303                             JAE    K37                    ; ALT-CONTINUE
EAF5                          K36:                                  ; CLOSE-RETURN
EAF5 E961FF                           JMP    K26                    ; IGNORE THE KEY
EAF8                          K37:                                  ; ALT-CONTINUE
EAF8 3C47                             CMP    AL,71                  ; IN KEYPAD REGION
EAFA 73F9                             JAE    K36                    ; IF SO, IGNORE
EAFC BB5FE9                           MOV    BX,OFFSET K13          ; ALT SHIFT PSEUDO SCAN TABLE
EAFF E91B01                           JMP    K63                    ; TRANSLATE THAT

                              ;----- NOT IN ALTERNATE SHIFT

EB02                          K38:                                  ; NOT-ALT-SHIFT
EB02 F606170004                       TEST   KB_FLAG,CTL_SHIFT      ; ARE WE IN CONTROL SHIFT
EB07 7458                             JZ     K44                    ; NOT-CTL-SHIFT

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

EB09 3C46                             CMP    AL,SCROLL_KEY          ; TEST FOR BREAK
EB0B 7518                             JNE    K39                    ; NO-BREAK
EB0D 8B1E8000                         MOV    BX,BUFFER_START        ; RESET BUFFER TO EMPTY
EB11 891E1A00                         MOV    BUFFER_HEAD,BX
EB15 891E1C00                         MOV    BUFFER_TAIL,BX
EB19 C606710080                       MOV    BIOS_BREAK,80H         ; TURN ON BIOS_BREAK BIT
EB1E CD1B                             INT    1BH                    ; BREAK INTERRUPT VECTOR
EB20 2BC0                             SUB    AX,AX                  ; PUT OUT DUMMY CHARACTER
EB22 E9B000                           JMP    K57                    ; BUFFER_FILL
EB25                          K39:                                  ; NO-BREAK
EB25 3C45                             CMP    AL,NUM_KEY             ; LOOK FOR PAUSE KEY
EB27 7521                             JNE    K41                    ; NO-PAUSE
EB29 800E180008                       OR     KB_FLAG_1,HOLD_STATE   ; TURN ON THE HOLD FLAG
EB2E B020                             MOV    AL,EOI                 ; END OF INTERRUPT TO CONTROL PORT
EB30 E620                             OUT    020H,AL                ; ALLOW FURTHER KEYSTROKE INTS

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

EB32 803E490007                       CMP    CRT_MODE,7             ; IS THIS BLACK AND WHITE CARD
EB37 7407                             JE     K40                    ; YES, NOTHING TO DO
EB39 BAD803                           MOV    DX,03D8H               ; PORT FOR COLOR CARD
EB3C A06500                           MOV    AL,CRT_MODE_SET        ; GET THE VALUE OF THE CURRENT MODE
EB3F EE                               OUT    DX,AL                  ; SET THE CRT MODE, SO THAT CRT IS ON
EB40                          K40:                                  ; PAUSE-LOOP
EB40 F606180008                       TEST   KB_FLAG_1,HOLD_STATE
EB45 75F9                             JNZ    K40                    ; LOOP UNTIL FLAG TURNED OFF
EB47 E914FF                           JMP    K27                    ; INTERRUPT_RETURN_NO_EOI
EB4A                          K41:                                  ; NO-PAUSE

                              ;----- TEST SPECIAL CASE KEY 55

EB4A 3C37                             CMP    AL,55
EB4C 7506                             JNE    K42                    ; NOT-KEY-55
EB4E B80072                           MOV    AX,114*256             ; START/STOP PRINTING SWITCH
EB51 E98100                           JMP    K57                    ; BUFFER_FILL

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

EB54                          K42:                                  ; NOT-KEY-55
EB54 BB8EE8                           MOV    BX,OFFSET K8           ; SET UP TO TRANSLATE CTL
EB57 3C3B                             CMP    AL,59                  ; IS IT IN TABLE
                                                                    ; CTL-TABLE-TRANSLATE
EB59 7276                             JB     K56                    ; YES, GO TRANSLATE CHAR
EB5B                          K43:                                  ; CTL-TABLE-TRANSLATE
EB5B BBC8E8                           MOV    BX,OFFSET K9           ; CTL TABLE SCAN
EB5E E9BC00                           JMP    K63                    ; TRANSLATE_SCAN

                              ;----- NOT IN CONTROL SHIFT

EB61                          K44:                                  ; NOT-CTL-SHIFT
EB61 3C47                             CMP    AL,71                  ; TEST FOR KEYPAD REGION
EB63 732C                             JAE    K48                    ; HANDLE KEYPAD REGION
EB65 F606170003                       TEST   KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT
EB6A 745A                             JZ     K54                    ; TEST FOR SHIFT STATE

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

```

```
LOC OBJ               SOURCE

EB6C 3C0F                     CMP     AL,15                   ; BACK TAB KEY
EB6E 7505                     JNE     K45                     ; NOT-BACK-TAB
EB70 B8000F                   MOV     AX,15*256               ; SET PSEUDO SCAN CODE
EB73 EB60                     JMP     SHORT K57               ; BUFFER_FILL
EB75                  K45:                                    ; NOT-BACK-TAB
EB75 3C37                     CMP     AL,55                   ; PRINT SCREEN KEY
EB77 7509                     JNE     K46                     ; NOT-PRINT-SCREEN

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

EB79 B020                     MOV     AL,EOI                  ; END OF CURRENT INTERRUPT
EB7B E620                     OUT     020H,AL                 ;  SO FURTHER THINGS CAN HAPPEN
EB7D CD05                     INT     5H                      ; ISSUE PRINT SCREEN INTERRUPT
EB7F E9DCFE                   JMP     K27                     ; GO BACK WITHOUT EOI OCCURRING
EB82                  K46:                                    ; NOT-PRINT-SCREEN
EB82 3C3B                     CMP     AL,59                   ; FUNCTION KEYS
EB84 7206                     JB      K47                     ; NOT-UPPER-FUNCTION
EB86 BB55E9                   MOV     BX,OFFSET K12           ; UPPER CASE PSEUDO SCAN CODES
EB89 E99100                   JMP     K63                     ; TRANSLATE_SCAN
EB8C                  K47:                                    ; NOT-UPPER-FUNCTION
EB8C BB1BE9                   MOV     BX,OFFSET K11           ; POINT TO UPPER CASE TABLE
EB8F EB40                     JMP     SHORT K56               ; OK, TRANSLATE THE CHAR

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

EB91                  K48:                                    ; KEYPAD-REGION
EB91 F606170020               TEST    KB_FLAG,NUM_STATE       ; ARE WE IN NUM_LOCK
EB96 7520                     JNZ     K52                     ; TEST FOR SURE
EB98 F606170003               TEST    KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT  ; ARE WE IN SHIFT STATE
EB9D 7520                     JNZ     K53                     ; IF SHIFTED, REALLY NUM STATE

                      ;----- BASE CASE FOR KEYPAD

EB9F                  K49:                                    ; BASE-CASE
EB9F 3C4A                     CMP     AL,74                   ; SPECIAL CASE FOR A COUPLE OF KEYS
EBA1 740B                     JE      K50                     ; MINUS
EBA3 3C4E                     CMP     AL,78
EBA5 740C                     JE      K51
EBA7 2C47                     SUB     AL,71                   ; CONVERT ORIGIN
EBA9 BB76E9                   MOV     BX,OFFSET K15           ; BASE CASE TABLE
EBAC EB71                     JMP     SHORT K64               ; CONVERT TO PSEUDO SCAN
EBAE                  K50:
EBAE B82D4A                   MOV     AX,74*256+'-'           ; MINUS
EBB1 EB22                     JMP     SHORT K57               ; BUFFER_FILL
EBB3                  K51:
EBB3 B82B4E                   MOV     AX,78*256+'+'           ; PLUS
EBB6 EB1D                     JMP     SHORT K57               ; BUFFER_FILL

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

EBB8                  K52:                                    ; ALMOST-NUM-STATE
EBB8 F606170003               TEST    KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT
EBBD 75E0                     JNZ     K49                     ; SHIFTED TEMP OUT OF NUM STATE
EBBF                  K53:                                    ; REALLY_NUM_STATE
EBBF 2C46                     SUB     AL,70                   ; CONVERT ORIGIN
EBC1 BB69E9                   MOV     BX,OFFSET K14           ; NUM STATE TABLE
EBC4 EB0B                     JMP     SHORT K56               ; TRANSLATE_CHAR

                      ;----- PLAIN OLD LOWER CASE

EBC6                  K54:                                    ; NOT-SHIFT
EBC6 3C3B                     CMP     AL,59                   ; TEST FOR FUNCTION KEYS
EBC8 7204                     JB      K55                     ; NOT-LOWER-FUNCTION
EBCA B000                     MOV     AL,0                    ; SCAN CODE IN AH ALREADY
EBCC EB07                     JMP     SHORT K57               ; BUFFER_FILL
EBCE                  K55:                                    ; NOT-LOWER-FUNCTION
EBCE BBE1E8                   MOV     BX,OFFSET K10           ; LC TABLE

                      ;----- TRANSLATE THE CHARACTER

EBD1                  K56:                                    ; TRANSLATE-CHAR
EBD1 FEC8                     DEC     AL                      ; CONVERT ORIGIN
EBD3 2ED7                     XLAT    CS:K11                  ; CONVERT THE SCAN CODE TO ASCII

                      ;----- PUT CHARACTER INTO BUFFER

EBD5                  K57:                                    ; BUFFER-FILL
```

```
LOC OBJ              SOURCE

EBD5 3CFF                    CMP     AL,-1                   ; IS THIS AN IGNORE CHAR
EBD7 741F                    JE      K59                     ; YES, DO NOTHING WITH IT
EBD9 80FCFF                  CMP     AH,-1                   ; LOOK FOR -1 PSEUDO SCAN
EBDC 741A                    JE      K59                     ; NEAR_INTERRUPT_RETURN

                     ;----- HANDLE THE CAPS LOCK PROBLEM

EBDE                 K58:                                    ; BUFFER-FILL-NOTEST
EBDE F606170040              TEST    KB_FLAG,CAPS_STATE      ; ARE WE IN CAPS LOCK STATE
EBE3 7420                    JZ      K61                     ; SKIP IF NOT

                     ;----- IN CAPS LOCK STATE

EBE5 F606170003              TEST    KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT  ; TEST FOR SHIFT STATE
EBEA 740F                    JZ      K60                     ; IF NOT SHIFT, CONVERT LOWER TO UPPER

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

EBEC 3C41                    CMP     AL,'A'                  ; FIND OUT IF ALPHABETIC
EBEE 7215                    JB      K61                     ; NOT_CAPS_STATE
EBF0 3C5A                    CMP     AL,'Z'
EBF2 7711                    JA      K61                     ; NOT_CAPS_STATE
EBF4 0420                    ADD     AL,'a'-'A'              ; CONVERT TO LOWER CASE
EBF6 EB0D                    JMP     SHORT K61               ; NOT_CAPS_STATE
EBF8                 K59:                                    ; NEAR-INTERRUPT-RETURN
EBF8 E95EFE                  JMP     K26                     ; INTERRUPT_RETURN

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

EBFB                 K60:                                    ; LOWER-TO-UPPER
EBFB 3C61                    CMP     AL,'a'                  ; FIND OUT IF ALPHABETIC
EBFD 7206                    JB      K61                     ; NOT_CAPS_STATE
EBFF 3C7A                    CMP     AL,'z'
EC01 7702                    JA      K61                     ; NOT_CAPS_STATE
EC03 2C20                    SUB     AL,'a'-'A'              ; CONVERT TO UPPER CASE
EC05                 K61:                                    ; NOT-CAPS-STATE
EC05 8B1E1C00                MOV     BX,BUFFER_TAIL          ; GET THE END POINTER TO THE BUFFER
EC09 8BF3                    MOV     SI,BX                   ; SAVE THE VALUE
EC0B E863FC                  CALL    K4                      ; ADVANCE THE TAIL
EC0E 3B1E1A00                CMP     BX,BUFFER_HEAD          ; HAS THE BUFFER WRAPPED AROUND
EC12 7413                    JE      K62                     ; BUFFER_FULL_BEEP
EC14 8904                    MOV     [SI],AX                 ; STORE THE VALUE
EC16 891E1C00                MOV     BUFFER_TAIL,BX          ; MOVE THE POINTER UP
EC1A E93CFE                  JMP     K26                     ; INTERRUPT_RETURN

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

EC1D                 K63:                                    ; TRANSLATE-SCAN
EC1D 2C3B                    SUB     AL,59                   ; CONVERT ORIGIN TO FUNCTION KEYS
EC1F                 K64:                                    ; TRANSLATE-SCAN-ORGD
EC1F 2ED7                    XLAT    CS:K9                   ; CTL TABLE SCAN
EC21 8AE0                    MOV     AH,AL                   ; PUT VALUE INTO AH
EC23 B000                    MOV     AL,0                    ; ZERO ASCII CODE
EC25 EBAE                    JMP     K57                     ; PUT IT INTO THE BUFFER

                     KB_INT  ENDP

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

EC27                 K62:                                    ; BUFFER-FULL-BEEP
EC27 B020                    MOV     AL,EOI                  ; END OF INTERRUPT COMMAND
EC29 E620                    OUT     20H,AL                  ; SEND COMMAND TO INT CONTROL PORT
EC2B BB8000                  MOV     BX,080H                 ; NUMBER OF CYCLES FOR 1/12 SECOND TONE
EC2E E461                    IN      AL,KB_CTL               ; GET CONTROL INFORMATION
EC30 50                      PUSH    AX                      ; SAVE
EC31                 K65:                                    ; BEEP-CYCLE
EC31 24FC                    AND     AL,0FCH                 ; TURN OFF TIMER GATE AND SPEAKER DATA
EC33 E661                    OUT     KB_CTL,AL               ; OUTPUT TO CONTROL
EC35 B94800                  MOV     CX,48H                  ; HALF CYCLE TIME FOR TONE
EC38                 K66:
EC38 E2FE                    LOOP    K66                     ; SPEAKER OFF
EC3A 0C02                    OR      AL,2                    ; TURN ON SPEAKER BIT
EC3C E661                    OUT     KB_CTL,AL               ; OUTPUT TO CONTROL
EC3E B94800                  MOV     CX,48H                  ; SET UP COUNT
EC41                 K67:
EC41 E2FE                    LOOP    K67                     ; ANOTHER HALF CYCLE
EC43 4B                      DEC     BX                      ; TOTAL TIME COUNT
```

```
EC44 75EB                JNZ     K65                     ; DO ANOTHER CYCLE
EC46 58                  POP     AX                      ; RECOVER CONTROL
EC47 E661                OUT     KB_CTL,AL               ; OUTPUT THE CONTROL
EC49 E912FE              JMP     K27
                ;----------------------------------------
                ;         ROS CHECKSUM SUBROUTINE        :
                ;----------------------------------------
EC4C            ROS_CHECKSUM    PROC    NEAR             ; NEXT_ROS_MODULE
EC4C B90020              MOV     CX,8192                 ; NUMBER OF BYTES TO ADD
EC4F            ROS_CHECKSUM_CNT:                        ; ENTRY FOR OPTIONAL ROS TEST
EC4F 32C0                XOR     AL,AL
EC51            C26:
EC51 0207                ADD     AL,DS:[BX]
EC53 43                  INC     BX                      ; POINT TO NEXT BYTE
EC54 E2FB                LOOP    C26                     ; ADD ALL BYTES IN ROS MODULE
EC56 0AC0                OR      AL,AL                   ; SUM = 0?
EC58 C3                  RET
                ROS_CHECKSUM    ENDP

                ;-- 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 REQUIRED :
                ;                ON ALL DRIVES                                      :
                ;        (AH)=1  READ THE STATUS OF THE SYSTEM INTO (AL)            :
                ;                DISKETTE_STATUS FROM LAST OPERATION 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,              :
                ;                                NOT USED FOR FORMAT)               :
                ;        (AL) - NUMBER OF SECTORS ( MAX = 8, NOT VALUE CHECKED, NOT USED :
                ;                                         FOR FORMAT)               :
                ;        (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                     ORG     0EC59H
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 E8D812                 CALL    DDS
EC66 E81C00                 CALL    J1                        ; CALL THE REST TO ENSURE DS RESTORED
EC69 BB0400                 MOV     BX,4                      ; GET THE MOTOR WAIT PARAMETER
EC6C E8FD01                 CALL    GET_PARM
EC6F 88264000               MOV     MOTOR_COUNT,AH            ; SET THE TIMER COUNT FOR THE MOTOR
EC73 8A264100               MOV     AH,DISKETTE_STATUS        ; GET STATUS OF OPERATION
EC77 80FC01                 CMP     AH,1                      ; SET THE CARRY FLAG TO INDICATE
EC7A F5                     CMC                               ;  SUCCESS OR FAILURE
EC7B 5A                     POP     DX                        ; RESTORE ALL REGISTERS
EC7C 5D                     POP     BP
EC7D 5F                     POP     DI
EC7E 5E                     POP     SI
EC7F 1F                     POP     DS
EC80 59                     POP     CX
EC81 5B                     POP     BX                        ; RECOVER ADDRESS
EC82 CA0200                 RET     2                         ; THROW AWAY SAVED FLAGS
                  DISKETTE_IO       ENDP

EC85              J1        PROC    NEAR
EC85 8AF0                   MOV     DH,AL                     ; SAVE # SECTORS IN DH
EC87 80263F007F             AND     MOTOR_STATUS,07FH         ; INDICATE A READ OPERATION
EC8C 0AE4                   OR      AH,AH                     ; AH=0
EC8E 7427                   JZ      DISK_RESET
EC90 FECC                   DEC     AH                        ; AH=1
EC92 7473                   JZ      DISK_STATUS
EC94 C606410000             MOV     DISKETTE_STATUS,0         ; RESET THE STATUS INDICATOR
EC99 80FA04                 CMP     DL,4                      ; TEST FOR DRIVE IN 0-3 RANGE
EC9C 7313                   JAE     J3                        ; ERROR IF ABOVE
EC9E FECC                   DEC     AH                        ; AH=2
ECA0 7469                   JZ      DISK_READ
ECA2 FECC                   DEC     AH                        ; AH=3
ECA4 7503                   JNZ     J2                        ; TEST_DISK_VERF
ECA6 E99500                 JMP     DISK_WRITE
ECA9              J2:                                         ; TEST_DISK_VERF
ECA9 FECC                   DEC     AH                        ; AH=4
ECAB 7467                   JZ      DISK_VERF
ECAD FECC                   DEC     AH                        ; AH=5
ECAF 7467                   JZ      DISK_FORMAT
ECB1              J3:                                         ; BAD_COMMAND
ECB1 C606410001             MOV     DISKETTE_STATUS,BAD_CMD   ; ERROR CODE, NO SECTORS TRANSFERRED
ECB6 C3                     RET                               ; UNDEFINED OPERATION
                  J1        ENDP

                  ;----- RESET THE DISKETTE SYSTEM

ECB7              DISK_RESET        PROC    NEAR
ECB7 BAF203                 MOV     DX,03F2H                  ; ADAPTER CONTROL PORT
ECBA FA                     CLI                               ; NO INTERRUPTS
ECBB A03F00                 MOV     AL,MOTOR_STATUS           ; WHICH MOTOR IS ON
ECBE B104                   MOV     CL,4                      ; SHIFT COUNT
ECC0 D2E0                   SAL     AL,CL                     ; MOVE MOTOR VALUE TO HIGH NYBBLE
ECC2 A820                   TEST    AL, 20H                   ; SELECT CORRESPONDING DRIVE
ECC4 750C                   JNZ     J5                        ; JUMP IF MOTOR ONE IS ON
ECC6 A840                   TEST    AL, 40H
ECC8 7506                   JNZ     J4                        ; JUMP IF MOTOR TWO IS ON
ECCA A880                   TEST    AL, 80H
ECCC 7406                   JZ      J6                        ; JUMP IF MOTOR ZERO IS ON
ECCE FEC0                   INC     AL
ECD0              J4:
ECD0 FEC0                   INC     AL
ECD2              J5:
ECD2 FEC0                   INC     AL
ECD4              J6:
ECD4 0C08                   OR      AL,8                      ; TURN ON INTERRUPT ENABLE
ECD6 EE                     OUT     DX,AL                     ; RESET THE ADAPTER
ECD7 C6063E0000             MOV     SEEK_STATUS,0             ; SET RECAL REQUIRED ON ALL DRIVES
ECDC C606410000             MOV     DISKETTE_STATUS,0         ; SET OK STATUS FOR DISKETTE
ECE1 0C04                   OR      AL,4                      ; TURN OFF RESET
ECE3 EE                     OUT     DX,AL                     ; TURN OFF THE RESET
```

```
LOC OBJ                 LINE    SOURCE

ECE4 FB                                 STI                                     ; REENABLE THE INTERRUPTS
ECE5 E82A02                             CALL    CHK_STAT_2                      ; DO SENSE INTERRUPT STATUS
                                                                                ;  FOLLOWING RESET
ECE8 A04200                             MOV     AL,NEC_STATUS                   ; IGNORE ERROR RETURN AND DO OWN TEST
ECEB 3CC0                               CMP     AL,0C0H                         ; TEST FOR DRIVE READY TRANSITION
ECED 7406                               JZ      J7                              ; EVERYTHING OK
ECEF 800E410020                         OR      DISKETTE_STATUS,BAD_NEC         ; SET ERROR CODE
ECF4 C3                                 RET

                                ;----- SEND SPECIFY COMMAND TO NEC

ECF5                            J7:                                             ; DRIVE_READY
ECF5 B403                               MOV     AH,03H                          ; SPECIFY COMMAND
ECF7 E84701                             CALL    NEC_OUTPUT                      ; OUTPUT THE COMMAND
ECFA BB0100                             MOV     BX,1                            ; FIRST BYTE PARM IN BLOCK
ECFD E86C01                             CALL    GET_PARM                        ;  TO THE NEC CONTROLLER
ED00 BB0300                             MOV     BX,3                            ; SECOND BYTE PARM IN BLOCK
ED03 E86601                             CALL    GET_PARM                        ;  TO THE NEC CONTROLLER
ED06                            J8:                                             ; RESET_RET
ED06 C3                                 RET                                     ; RETURN TO CALLER
                                DISK_RESET      ENDP

                                ;----- DISKETTE STATUS ROUTINE

ED07                            DISK_STATUS     PROC    NEAR
ED07 A04100                             MOV     AL,DISKETTE_STATUS
ED0A C3                                 RET
                                DISK_STATUS     ENDP

                                ;----- DISKETTE READ

ED0B                            DISK_READ       PROC    NEAR
ED0B B046                               MOV     AL,046H                         ; READ COMMAND FOR DMA
ED0D                            J9·                                             ; DISK_READ_CONT
ED0D E8B801                             CALL    DMA_SETUP                       ; SET UP THE DMA
ED10 B4E6                               MOV     AH,0E6H                         ; SET UP RD COMMAND FOR NEC CONTROLLER
ED12 EB36                               JMP     SHORT RW_OPN                    ; GO DO THE OPERATION
                                DISK_READ       ENDP

                                ;----- DISKETTE VERIFY

ED14                            DISK_VERF       PROC    NEAR
ED14 B042                               MOV     AL,042H                         ; VERIFY COMMAND FOR DMA
ED16 EBF5                               JMP     J9                              ; DO AS IF DISK READ
                                DISK_VERF       ENDP

                                ;----- DISKETTE FORMAT

ED18                            DISK_FORMAT     PROC    NEAR
ED18 800E3F0080                         OR      MOTOR_STATUS,80H                ; INDICATE WRITE OPERATION
ED1D B04A                               MOV     AL,04AH                         ; WILL WRITE TO THE DISKETTE
ED1F E8A601                             CALL    DMA_SETUP                       ; SET UP THE DMA
ED22 B44D                               MOV     AH,04DH                         ; ESTABLISH THE FORMAT COMMAND
ED24 EB24                               JMP     SHORT RW_OPN                    ; DO THE OPERATION
ED26                            J10:                                            ; CONTINUATION OF RW_OPN FOR FMT
ED26 BB0700                             MOV     BX,7                            ; GET THE
ED29 E84001                             CALL    GET_PARM                        ;  BYTES/SECTOR VALUE TO NEC
ED2C BB0900                             MOV     BX,9                            ; GET THE
ED2F E83A01                             CALL    GET_PARM                        ;  SECTORS/TRACK VALUE TO NEC
ED32 BB0F00                             MOV     BX,15                           ; GET THE
ED35 E83401                             CALL    GET_PARM                        ;  GAP LENGTH VALUE TO NEC
ED38 BB1100                             MOV     BX,17                           ; GET THE FILLER BYTE
ED3B E9AB00                             JMP     J16                             ;  TO THE CONTROLLER
                                DISK_FORMAT     ENDP

                                ;----- DISKETTE WRITE ROUTINE

ED3E                            DISK_WRITE      PROC    NEAR
ED3E 800E3F0080                         OR      MOTOR_STATUS,80H                ; INDICATE WRITE OPERATION
ED43 B04A                               MOV     AL,04AH                         ; DMA WRITE COMMAND
ED45 E88001                             CALL    DMA_SETUP
ED48 B4C5                               MOV     AH,0C5H                         ; NEC COMMAND TO WRITE TO DISKETTE
                                DISK_WRITE      ENDP

                                ;----- ALLOW WRITE ROUTINE TO FALL INTO RW_OPN

                                ;------------------------------------------------------------
```

```
LOC OBJ             LINE   SOURCE

                           ; RW_OPN                                                         :
                           ;        THIS ROUTINE PERFORMS THE READ/WRITE/VERIFY OPERATION  :
                           ;------------------------------------------------------------------
ED4A                       RW_OPN  PROC    NEAR
ED4A 7308                          JNC     J11                    ; TEST FOR DMA ERROR
ED4C C606410009                    MOV     DISKETTE_STATUS,DMA_BOUNDARY    ; SET ERROR
ED51 B000                          MOV     AL,0                   ; NO SECTORS TRANSFERRED
ED53 C3                            RET                            ; RETURN TO MAIN ROUTINE
ED54                       J11:                                   ; DO_RW_OPN
ED54 50                            PUSH    AX                     ; SAVE THE COMMAND

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

ED55 51                            PUSH    CX                     ; SAVE THE T/S PARMS
ED56 8ACA                          MOV     CL,DL                  ; GET DRIVE NUMBER AS SHIFT COUNT
ED58 B001                          MOV     AL,1                   ; MASK FOR DETERMINING MOTOR BIT
ED5A D2E0                          SAL     AL,CL                  ; SHIFT THE MASK BIT
ED5C FA                            CLI                            ; NO INTERRUPTS WHILE DETERMINING
                                                                  ;  MOTOR STATUS
ED5D C6064000FF                    MOV     MOTOR_COUNT,0FFH       ; SET LARGE COUNT DURING OPERATION
ED62 84063F00                      TEST    AL,MOTOR_STATUS        ; TEST THAT MOTOR FOR OPERATING
ED66 7531                          JNZ     J14                    ; IF RUNNING, SKIP THE WAIT
ED68 80263F00F0                    AND     MOTOR_STATUS,0F0H      ; TURN OFF ALL MOTOR BITS
ED6D 08063F00                      OR      MOTOR_STATUS,AL        ; TURN ON THE CURRENT MOTOR
ED71 FB                            STI                            ; INTERRUPTS BACK ON
ED72 B010                          MOV     AL,10H                 ; MASK BIT
ED74 D2E0                          SAL     AL,CL                  ; DEVELOP BIT MASK FOR MOTOR ENABLE
ED76 0AC2                          OR      AL,DL                  ; GET DRIVE SELECT BITS IN
ED78 0C0C                          OR      AL,0CH                 ; NO RESET, ENABLE DMA/INT
ED7A 52                            PUSH    DX                     ; SAVE REG
ED7B BAF203                        MOV     DX,03F2H               ; CONTROL PORT ADDRESS
ED7E EE                            OUT     DX,AL
ED7F 5A                            POP     DX                     ; RECOVER REGISTERS

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

ED80 F6063F0080                    TEST    MOTOR_STATUS,80H       ; IS THIS A WRITE
ED85 7412                          JZ      J14                    ; NO, CONTINUE WITHOUT WAIT
ED87 BB1400                        MOV     BX,20                  ; GET THE MOTOR WAIT
ED8A E8DF00                        CALL    GET_PARM               ;  PARAMETER
ED8D 0AE4                          OR      AH,AH                  ; TEST FOR NO WAIT
ED8F                       J12:                                   ; TEST_WAIT_TIME
ED8F 7408                          JZ      J14                    ; EXIT WITH TIME EXPIRED
ED91 2BC9                          SUB     CX,CX                  ; SET UP 1/8 SECOND LOOP TIME
ED93                       J13:
ED93 E2FE                          LOOP    J13                    ; WAIT FOR THE REQUIRED TIME
ED95 FECC                          DEC     AH                     ; DECREMENT TIME VALUE
ED97 EBF6                          JMP     J12                    ; ARE WE DONE YET
ED99                       J14:                                   ; MOTOR_RUNNING
ED99 FB                            STI                            ; INTERRUPTS BACK ON FOR BYPASS WAIT
ED9A 59                            POP     CX

                           ;----- DO THE SEEK OPERATION

ED9B E8DF00                        CALL    SEEK                   ; MOVE TO CORRECT TRACK
ED9E 58                            POP     AX                     ; RECOVER COMMAND
ED9F 8AFC                          MOV     BH,AH                  ; SAVE COMMAND IN BH
EDA1 B600                          MOV     DH,0                   ; SET NO SECTORS READ IN CASE OF ERROR
EDA3 724B                          JC      J17                    ; IF ERROR, THEN EXIT AFTER MOTOR OFF
EDA5 BEF0ED90                      MOV     SI,OFFSET J17          ; DUMMY RETURN ON STACK FOR NEC_OUTPUT
EDA9 56                            PUSH    SI                     ;  SO THAT IT WILL RETURN TO MOTOR OFF
                                                                  ;  LOCATION

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

EDAA E89400                        CALL    NEC_OUTPUT             ; OUTPUT THE OPERATION COMMAND
EDAD 8A6601                        MOV     AH,[BP+1]              ; GET THE CURRENT HEAD NUMBER
EDB0 D0E4                          SAL     AH,1                   ; MOVE IT TO BIT 2
EDB2 D0E4                          SAL     AH,1
EDB4 80E404                        AND     AH,4                   ; ISOLATE THAT BIT
EDB7 0AE2                          OR      AH,DL                  ; OR IN THE DRIVE NUMBER
EDB9 E88500                        CALL    NEC_OUTPUT

                           ;----- TEST FOR FORMAT COMMAND

EDBC 80FF4D                        CMP     BH,04DH                ; IS THIS A FORMAT OPERATION
EDBF 7503                          JNE     J15                    ; NO. CONTINUE WITH R/W/V
```

```
LOC OBJ              SOURCE

EDC1 E962FF                JMP     J10                 ; IF SO, HANDLE SPECIAL
EDC4                 J15:
EDC4 8AE5                  MOV     AH,CH               ; CYLINDER NUMBER
EDC6 E87800                CALL    NEC_OUTPUT
EDC9 8A6601                MOV     AH,[BP+1]           ; HEAD NUMBER FROM STACK
EDCC E87200                CALL    NEC_OUTPUT
EDCF 8AE1                  MOV     AH,CL               ; SECTOR NUMBER
EDD1 E86D00                CALL    NEC_OUTPUT
EDD4 BB0700                MOV     BX,7                ; BYTES/SECTOR PARM FROM BLOCK
EDD7 E89200                CALL    GET_PARM            ;  TO THE NEC
EDDA BB0900                MOV     BX,9                ; EOT PARM FROM BLOCK
EDDD E88C00                CALL    GET_PARM            ;  TO THE NEC
EDE0 BB0B00                MOV     BX,11               ; GAP LENGTH PARM FROM BLOCK
EDE3 E88600                CALL    GET_PARM            ;  TO THE NEC
EDE6 BB0D00                MOV     BX,13               ; DTL PARM FROM BLOCK
EDE9                 J16:                              ; RW_OPN_FINISH
EDE9 E88000                CALL    GET_PARM            ;  TO THE NEC
EDEC 5E                    POP     SI                  ; CAN NOW DISCARD THAT DUMMY
                                                       ;  RETURN ADDRESS

                     ;----- LET THE OPERATION HAPPEN

EDED E84301                CALL    WAIT_INT            ; WAIT FOR THE INTERRUPT
EDF0                 J17:                              ; MOTOR_OFF
EDF0 7245                  JC      J21                 ; LOOK FOR ERROR
EDF2 E87401                CALL    RESULTS             ; GET THE NEC STATUS
EDF5 723F                  JC      J20                 ; LOOK FOR ERROR

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

EDF7 FC                    CLD                         ; SET THE CORRECT DIRECTION
EDF8 BE4200                MOV     SI,OFFSET NEC_STATUS ; POINT TO STATUS FIELD
EDFB AC                    LODS    NEC_STATUS          ; GET ST0
EDFC 24C0                  AND     AL,0C0H             ; TEST FOR NORMAL TERMINATION
EDFE 743B                  JZ      J22                 ; OPN_OK
EE00 3C40                  CMP     AL,040H             ; TEST FOR ABNORMAL TERMINATION
EE02 7529                  JNZ     J18                 ; NOT ABNORMAL, BAD NEC

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

EE04 AC                    LODS    NEC_STATUS          ; GET ST1
EE05 D0E0                  SAL     AL,1                ; TEST FOR EOT FOUND
EE07 B404                  MOV     AH,RECORD_NOT_FND
EE09 7224                  JC      J19                 ; RW_FAIL
EE0B D0E0                  SAL     AL,1
EE0D D0E0                  SAL     AL,1                ; TEST FOR CRC ERROR
EE0F B410                  MOV     AH,BAD_CRC
EE11 721C                  JC      J19                 ; RW_FAIL
EE13 D0E0                  SAL     AL,1                ; TEST FOR DMA OVERRUN
EE15 B408                  MOV     AH,BAD_DMA
EE17 7216                  JC      J19                 ; RW_FAIL
EE19 D0E0                  SAL     AL,1
EE1B D0E0                  SAL     AL,1                ; TEST FOR RECORD NOT FOUND
EE1D B404                  MOV     AH,RECORD_NOT_FND
EE1F 720E                  JC      J19                 ; RW_FAIL
EE21 D0E0                  SAL     AL,1
EE23 B403                  MOV     AH,WRITE_PROTECT    ; TEST FOR WRITE_PROTECT
EE25 7208                  JC      J19                 ; RW_FAIL
EE27 D0E0                  SAL     AL,1                ; TEST MISSING ADDRESS MARK
EE29 B402                  MOV     AH,BAD_ADDR_MARK
EE2B 7202                  JC      J19                 ; RW_FAIL

                     ;----- NEC MUST HAVE FAILED

EE2D                 J18:                              ; RW-NEC-FAIL
EE2D B420                  MOV     AH,BAD_NEC
EE2F                 J19:                              ; RW-FAIL
EE2F 08264100              OR      DISKETTE_STATUS,AH
EE33 E87801                CALL    NUM_TRANS           ; HOW MANY WERE REALLY TRANSFERRED
EE36                 J20:                              ; RW_ERR
EE36 C3                    RET                         ; RETURN TO CALLER
EE37                 J21:                              ; RW_ERR_RES
EE37 E82F01                CALL    RESULTS             ; FLUSH THE RESULTS BUFFER
EE3A C3                    RET

                     ;----- OPERATION WAS SUCCESSFUL

```

```
LOC OBJ             SOURCE

EE3B                J22:                                    ; OPN_OK
EE3B E87001                 CALL    NUM_TRANS               ; HOW MANY GOT MOVED
EE3E 32E4                   XOR     AH,AH                   ; NO ERRORS
EE40 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                                                :
                    ;-------------------------------------------------------------------
EE41                NEC_OUTPUT      PROC    NEAR
EE41 52                     PUSH    DX                      ; SAVE REGISTERS
EE42 51                     PUSH    CX
EE43 BAF403                 MOV     DX,03F4H                ; STATUS PORT
EE46 33C9                   XOR     CX,CX                   ; COUNT FOR TIME OUT
EE48                J23:
EE48 EC                     IN      AL,DX                   ; GET STATUS
EE49 A840                   TEST    AL,040H                 ; TEST DIRECTION BIT
EE4B 740C                   JZ      J25                     ; DIRECTION OK
EE4D E2F9                   LOOP    J23
EE4F                J24:                                    ; TIME_ERROR
EE4F 800E410080             OR      DISKETTE_STATUS,TIME_OUT
EE54 59                     POP     CX
EE55 5A                     POP     DX                      ; SET ERROR CODE AND RESTORE REGS
EE56 58                     POP     AX                      ; DISCARD THE RETURN ADDRESS
EE57 F9                     STC                             ; INDICATE ERROR TO CALLER
EE58 C3                     RET
EE59                J25:
EE59 33C9                   XOR     CX,CX                   ; RESET THE COUNT
EE5B                J26:
EE5B EC                     IN      AL,DX                   ; GET THE STATUS
EE5C A880                   TEST    AL,080H                 ; IS IT READY
EE5E 7504                   JNZ     J27                     ; YES, GO OUTPUT
EE60 E2F9                   LOOP    J26                     ; COUNT DOWN AND TRY AGAIN
EE62 EBEB                   JMP     J24                     ; ERROR CONDITION
EE64                J27:                                    ; OUTPUT
EE64 8AC4                   MOV     AL,AH                   ; GET BYTE TO OUTPUT
EE66 B2F5                   MOV     DL,0F5H                 ; DATA PORT (3F5)
EE68 EE                     OUT     DX,AL                   ; OUTPUT THE BYTE
EE69 59                     POP     CX                      ; RECOVER REGISTERS
EE6A 5A                     POP     DX
EE6B 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                                        :
                    ;-------------------------------------------------------------------
EE6C                GET_PARM        PROC    NEAR
EE6C 1E                     PUSH    DS                      ; SAVE SEGMENT
EE6D 2BC0                   SUB     AX,AX                   ; ZERO TO AX
EE6F 8ED8                   MOV     DS,AX
                            ASSUME  DS:ABS0
EE71 C5367800               LDS     SI,DISK_POINTER         ; POINT TO BLOCK
EE75 D1EB                   SHR     BX,1                    ; DIVIDE BX BY 2, AND SET FLAG
                                                            ;  FOR EXIT
EE77 8A20                   MOV     AH,[SI+BX]              ; GET THE WORD
```

```
LOC OBJ            LINE    SOURCE

EE79 1F                            POP     DS                      ; RESTORE SEGMENT
                                   ASSUME  DS:DATA
EE7A 72C5                          JC      NEC_OUTPUT              ; IF FLAG SET, OUTPUT TO CONTROLLER
EE7C 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                                            :
                           ;-----------------------------------------------------------------
EE7D                       SEEK    PROC    NEAR
EE7D B001                          MOV     AL,1                    ; ESTABLISH MASK FOR RECAL TEST
EE7F 51                            PUSH    CX                      ; SAVE INPUT VALUES
EE80 8ACA                          MOV     CL,DL                   ; GET DRIVE VALUE INTO CL
EE82 D2C0                          ROL     AL,CL                   ; SHIFT IT BY THE DRIVE VALUE
EE84 59                            POP     CX                      ; RECOVER TRACK VALUE
EE85 84063E00                      TEST    AL,SEEK_STATUS          ; TEST FOR RECAL REQUIRED
EE89 7513                          JNZ     J28                     ; NO_RECAL
EE8B 08063E00                      OR      SEEK_STATUS,AL          ; TURN ON THE NO RECAL BIT IN FLAG
EE8F B407                          MOV     AH,07H                  ; RECALIBRATE COMMAND
EE91 E8ADFF                        CALL    NEC_OUTPUT
EE94 8AE2                          MOV     AH,DL
EE96 E8A8FF                        CALL    NEC_OUTPUT              ; OUTPUT THE DRIVE NUMBER
EE99 E87600                        CALL    CHK_STAT_2              ; GET THE INTERUPT AND SENSE INT STATUS
EE9C 7229                          JC      J32                     ; SEEK_ERROR

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

EE9E                       J28:
EE9E B40F                          MOV     AH,0FH                  ; SEEK COMMAND TO NEC
EEA0 E89EFF                        CALL    NEC_OUTPUT
EEA3 8AE2                          MOV     AH,DL                   ; DRIVE NUMBER
EEA5 E899FF                        CALL    NEC_OUTPUT
EEA8 8AE5                          MOV     AH,CH                   ; TRACK NUMBER
EEAA E894FF                        CALL    NEC_OUTPUT
EEAD E86200                        CALL    CHK_STAT_2              ; GET ENDING INTERRUPT AND
                                                                   ;  SENSE STATUS

                           ;----- WAIT FOR HEAD SETTLE

EEB0 9C                            PUSHF                           ; SAVE STATUS FLAGS
EEB1 BB1200                        MOV     BX,18                   ; GET HEAD SETTLE PARAMETER
EEB4 E8B5FF                        CALL    GET_PARM
EEB7 51                            PUSH    CX                      ; SAVE REGISTER
EEB8                       J29:                                    ; HEAD_SETTLE
EEB8 B92602                        MOV     CX,550                  ; 1 MS LOOP
EEBB 0AE4                          OR      AH,AH                   ; TEST FOR TIME EXPIRED
EEBD 7406                          JZ      J31
EEBF                       J30:
EEBF E2FE                          LOOP    J30                     ; DELAY FOR 1 MS
EEC1 FECC                          DEC     AH                      ; DECREMENT THE COUNT
EEC3 EBF3                          JMP     J29                     ; DO IT SOME MORE
EEC5                       J31:
EEC5 59                            POP     CX                      ; RECOVER STATE
EEC6 9D                            POPF
EEC7                       J32:                                    ; SEEK_ERROR
EEC7 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                                            :
                           ;-----------------------------------------------------------------
EEC8                       DMA_SETUP       PROC    NEAR
EEC8 51                            PUSH    CX                      ; SAVE THE REGISTER
```

```
LOC OBJ                 SOURCE

EEC9 FA                         CLI                             ; NO MORE INTERRUPTS
EECA E60C                       OUT     DMA+12,AL               ; SET THE FIRST/LAST F/F
EECC 50                         PUSH    AX
EECD 58                         POP     AX
EECE E60B                       OUT     DMA+11,AL               ; OUTPUT THE MODE BYTE
EED0 8CC0                       MOV     AX,ES                   ; GET THE ES VALUE
EED2 B104                       MOV     CL,4                    ; SHIFT COUNT
EED4 D3C0                       ROL     AX,CL                   ; ROTATE LEFT
EED6 8AE8                       MOV     CH,AL                   ; GET HIGHEST NYBLE OF ES TO CH
EED8 24F0                       AND     AL,0F0H                 ; ZERO THE LOW NYBBLE FROM SEGMENT
EEDA 03C3                       ADD     AX,BX                   ; TEST FOR CARRY FROM ADDITION
EEDC 7302                       JNC     J33
EEDE FEC5                       INC     CH                      ; CARRY MEANS HIGH 4 BITS MUST BE INC
EEE0                    J33:
EEE0 50                         PUSH    AX                      ; SAVE START ADDRESS
EEE1 E604                       OUT     DMA+4,AL                ; OUTPUT LOW ADDRESS
EEE3 8AC4                       MOV     AL,AH
EEE5 E604                       OUT     DMA+4,AL                ; OUTPUT HIGH ADDRESS
EEE7 8AC5                       MOV     AL,CH                   ; GET HIGH 4 BITS
EEE9 240F                       AND     AL,0FH
EEEB E681                       OUT     081H,AL                 ; OUTPUT THE HIGH 4 BITS TO
                                                                ;  THE PAGE REGISTER

                        ;----- DETERMINE COUNT

EEED 8AE6                       MOV     AH,DH                   ; NUMBER OF SECTORS
EEEF 2AC0                       SUB     AL,AL                   ;  TIMES 256 INTO AX
EEF1 D1E8                       SHR     AX,1                    ; SECTORS * 128 INTO AX
EEF3 50                         PUSH    AX
EEF4 BB0600                     MOV     BX,6                    ; GET THE BYTES/SECTOR PARM
EEF7 E872FF                     CALL    GET_PARM
EEFA 8ACC                       MOV     CL,AH                   ; USE AS SHIFT COUNT (0=128, 1=256 ETC)
EEFC 58                         POP     AX
EEFD D3E0                       SHL     AX,CL                   ; MULTIPLY BY CORRECT AMOUNT
EEFF 48                         DEC     AX                      ; -1 FOR DMA VALUE
EF00 50                         PUSH    AX                      ; SAVE COUNT VALUE
EF01 E605                       OUT     DMA+5,AL                ; LOW BYTE OF COUNT
EF03 8AC4                       MOV     AL,AH
EF05 E605                       OUT     DMA+5,AL                ; HIGH BYTE OF COUNT
EF07 FB                         STI                             ; INTERRUPTS BACK ON
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 E825FF                     CALL    NEC_OUTPUT
EF1C E84A00                     CALL    RESULTS                 ; READ IN THE RESULTS
EF1F 720A                       JC      J34                     ; CHK2_RETURN
EF21 A04200                     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
```

```
LOC OBJ               SOURCE

EF2C 800E410040              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              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              OR      DISKETTE_STATUS,TIME_OUT        ; NOTHING HAPPENED
EF4C F9                      STC                             ; ERROR RETURN
EF4D                  J37:
EF4D 9C                      PUSHF                           ; SAVE CURRENT CARRY
EF4E 80263E007F              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
                      ;-------------------------------------------------------
                      ; DISK_INT                                             :
                      ;       THIS ROUTINE HANDLES THE DISKETTE INTERRUPT    :
                      ;INPUT                                                 :
                      ;       NONE                                           :
                      ; OUTPUT                                               :
                      ;       THE INTERRUPT FLAG IS SET IS SEEK_STATUS       :
                      ;-------------------------------------------------------
EF57                         ORG     0EF57H
EF57                  DISK_INT        PROC    FAR
EF57 FB                      STI                             ; RE ENABLE INTERRUPTS
EF58 1E                      PUSH    DS
EF59 50                      PUSH    AX
EF5A E8E10F                  CALL    DDS
EF5D 800E3E0080              OR      SEEK_STATUS,INT_FLAG
EF62 B020                    MOV     AL,20H                  ; END OF INTERRUPT MARKER
EF64 E620                    OUT     20H,AL                  ; INTERRUPT CONTROL PORT
EF66 58                      POP     AX
EF67 1F                      POP     DS                      ; RECOVER SYSTEM
EF68 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                                                :
                      ;----------------------------------------------------------------------
EF69                  RESULTS PROC    NEAR
EF69 FC                      CLD
EF6A BF4200                  MOV     DI,OFFSET NEC_STATUS    ; POINTER TO DATA AREA
EF6D 51                      PUSH    CX                      ; SAVE COUNTER
EF6E 52                      PUSH    DX
```

```
LOC OBJ            LINE    SOURCE

EF6F 53                           PUSH    BX
EF70 B307                         MOV     BL,7                    ; MAX STATUS BYTES

                           ;----- WAIT FOR REQUEST FOR MASTER

EF72                       J38:                                   ; INPUT_LOOP
EF72 33C9                         XOR     CX,CX                   ; COUNTER
EF74 BAF403                       MOV     DX,03F4H                ; STATUS PORT
EF77                       J39:                                   ; WAIT FOR MASTER
EF77 EC                           IN      AL,DX                   ; GET STATUS
EF78 A880                         TEST    AL,080H                 ; MASTER READY
EF7A 750C                         JNZ     J40A                    ; TEST_DIR
EF7C E2F9                         LOOP    J39                     ; WAIT_MASTER
EF7E 800E410080                   OR      DISKETTE_STATUS,TIME_OUT
EF83                       J40:                                   ; RESULTS_ERROR
EF83 F9                           STC                             ; SET ERROR RETURN
EF84 5B                           POP     BX
EF85 5A                           POP     DX
EF86 59                           POP     CX
EF87 C3                           RET

                           ;----- TEST THE DIRECTION BIT

EF88                       J40A:
EF88 EC                           IN      AL,DX                   ; GET STATUS REG AGAIN
EF89 A840                         TEST    AL,040H                 ; TEST DIRECTION BIT
EF8B 7507                         JNZ     J42                     ; OK TO READ STATUS
EF8D                       J41:                                   ; NEC_FAIL
EF8D 800E410020                   OR      DISKETTE_STATUS,BAD_NEC
EF92 EBEF                         JMP     J40                     ; RESULTS_ERROR

                           ;----- READ IN THE STATUS

EF94                       J42:                                   ; INPUT_STAT
EF94 42                           INC     DX                      ; POINT AT DATA PORT
EF95 EC                           IN      AL,DX                   ; GET THE DATA
EF96 8805                         MOV     [DI],AL                 ; STORE THE BYTE
EF98 47                           INC     DI                      ; INCREMENT THE POINTER
EF99 B90A00                       MOV     CX,10                   ; LOOP TO KILL TIME FOR NEC
EF9C E2FE                  J43:   LOOP    J43
EF9E 4A                           DEC     DX                      ; POINT AT STATUS PORT
EF9F EC                           IN      AL,DX                   ; GET STATUS
EFA0 A810                         TEST    AL,010H                 ; TEST FOR NEC STILL BUSY
EFA2 7406                         JZ      J44                     ; RESULTS DONE
EFA4 FECB                         DEC     BL                      ; DECREMENT THE STATUS COUNTER
EFA6 75CA                         JNZ     J38                     ; GO BACK FOR MORE
EFA8 EBE3                         JMP     J41                     ; CHIP HAS FAILED

                           ;----- RESULT OPERATION IS DONE

EFAA                       J44:
EFAA 5B                           POP     BX
EFAB 5A                           POP     DX
EFAC 59                           POP     CX                      ; RECOVER REGISTERS
EFAD 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                          :
                           ;-------------------------------------------------------------
EFAE                       NUM_TRANS      PROC    NEAR
EFAE A04500                       MOV     AL,NEC_STATUS+3         ; GET CYLINDER ENDED UP ON
EFB1 3AC5                         CMP     AL,CH                   ; SAME AS WE STARTED
EFB3 A04700                       MOV     AL,NEC_STATUS+5         ; GET ENDING SECTOR
EFB6 740A                         JZ      J45                     ; IF ON SAME CYL, THEN NO ADJUST
EFB8 BB0800                       MOV     BX,8
EFBB E8AEFE                       CALL    GET_PARM                ; GET EOT VALUE
EFBE 8AC4                         MOV     AL,AH                   ;  INTO AL
EFC0 FEC0                         INC     AL                      ; USE EOT+1 FOR CALCULATION
EFC2                       J45:
EFC2 2AC1                         SUB     AL,CL                   ; SUBTRACT START FROM END
```

```
LOC OBJ              SOURCE

EFC4 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  :
                     ;       DISK_POINTER TO IT.                                             :
                     ;-----------------------------------------------------------------------
EFC7                         ORG     0EFC7H
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)

                     ;--- INT 17 ------------------------------------------------------------
                     ; PRINTER_IO                                                            :
                     ;       THIS ROUTINE PROVIDES COMMUNICATION WITH THE PRINTER            :
                     ; INPUT                                                                 :
                     ;       (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 = NOT 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)                                               :
                     ;                                                                       :
                     ; DATA AREA PRINT_TIM_OUT (BYTE) MAY BE CHANGED TO CAUSE DIFFERENT      :
                     ; TIME-OUT WAITS. DEFAULT=20                                            :
                     ;                                                                       :
                     ; REGISTERS     AH IS MODIFIED                                          :
                     ;               ALL OTHERS UNCHANGED                                    :
                     ;-----------------------------------------------------------------------
                             ASSUME  CS:CODE,DS:DATA
EFD2                         ORG     0EFD2H
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 E8630F                  CALL    DDS
EFDB 8BF2                    MOV     SI,DX                   ; GET PRINTER PARM
EFDD 8A5C78                  MOV     BL,PRINT TIM OUT[SI]    ; LOAD TIME-OUT PARM
EFE0 D1E6                    SHL     SI,1                    ; WORD OFFSET INTO TABLE
EFE2 8B5408                  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 743F                    JZ      B8                      ; INIT_PRT
```

```
LOC OBJ          SOURCE

EFF1 FECC                 DEC     AH                    ; TEST FOR (AH)=2
EFF3 7428                 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 EE                   OUT     DX,AL                 ; OUTPUT CHAR TO PORT
EFFD 42                   INC     DX                    ; POINT TO STATUS PORT
EFFE             B3:
EFFE 2BC9                 SUB     CX,CX                 ; WAIT_BUSY
F000             B3_1:
F000 EC                   IN      AL,DX                 ; GET STATUS
F001 8AE0                 MOV     AH,AL                 ; STATUS TO AH ALSO
F003 A880                 TEST    AL,80H                ; IS THE PRINTER CURRENTLY BUSY
F005 750E                 JNZ     B4                    ; OUT_STROBE
F007 E2F7                 LOOP    B3_1                  ; TRY AGAIN
F009 FECB                 DEC     BL                    ; DROP LOOP COUNT
F00B 75F1                 JNZ     B3                    ; GO TILL TIMEOUT ENDS
F00D 80CC01               OR      AH,1                  ; SET ERROR FLAG
F010 80E4F9               AND     AH,0F9H               ; TURN OFF THE OTHER BITS
F013 EB13                 JMP     SHORT B7              ; RETURN WITH ERROR FLAG SET
F015             B4:                                    ; OUT_STROBE
F015 B00D                 MOV     AL,0DH                ; SET THE STROBE HIGH
F017 42                   INC     DX                    ; STROBE IS BIT 0 OF PORT C OF 8255
F018 EE                   OUT     DX,AL
F019 B00C                 MOV     AL,0CH                ; SET THE STROBE LOW
F01B EE                   OUT     DX,AL
F01C 58                   POP     AX                    ; RECOVER THE OUTPUT CHAR

                 ;------ PRINTER STATUS

F01D             B5:
F01D 50                   PUSH    AX                    ; SAVE AL REG
F01E             B6:
F01E 8B5408               MOV     DX,PRINTER_BASE[SI]
F021 42                   INC     DX
F022 EC                   IN      AL,DX                 ; GET PRINTER STATUS
F023 8AE0                 MOV     AH,AL
F025 80E4F8               AND     AH,0F8H               ; TURN OFF UNUSED BITS
F028             B7:                                    ; STATUS_SET
F028 5A                   POP     DX                    ; RECOVER AL REG
F029 8AC2                 MOV     AL,DL                 ; GET CHARACTER INTO AL
F02B 80F448               XOR     AH,48H                ; FLIP A COUPLE OF BITS
F02E EBC5                 JMP     B1                    ; RETURN FROM ROUTINE

                 ;------ INITIALIZE THE PRINTER PORT

F030             B8:
F030 50                   PUSH    AX                    ; SAVE AL
F031 42                   INC     DX                    ; POINT TO OUTPUT PORT
F032 42                   INC     DX
F033 B008                 MOV     AL,8                  ; SET INIT LINE LOW
F035 EE                   OUT     DX,AL
F036 B8E803               MOV     AX,1000
F039             B9:                                    ; INIT_LOOP
F039 48                   DEC     AX                    ; LOOP FOR RESET TO TAKE
F03A 75FD                 JNZ     B9                    ; INIT_LOOP
F03C B00C                 MOV     AL,0CH                ; NO INTERRUPTS, NON AUTO LF,
                                                        ;  INIT HIGH
F03E EE                   OUT     DX,AL
F03F EBDD                 JMP     B6                    ; PRT_STATUS_1
                 PRINTER_IO       ENDP

F041 62E1        C2       DW      C24                   ; RETURN ADDRESS FOR DUMMY STACK

                 ;--- 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 BLIN              :
;                       ** 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 VAL (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) = PALETTE COLOR ID BEING SET (0-127)               :
                ;               (BL) = COLOR VALUE TO BE USED WITH THAT COLOR ID        :
                ;                  NOTE: FOR THE CURRENT COLOR CARD, THIS ENTRY POINT   :
                ;                        HAS MEANING ONLY FOR 320X200 GRAPHICS.         :
                ;                       COLOR ID = 0 SELECTS THE BACKGROUND COLOR (0-15):
                ;                       COLOR ID = 1 SELECTS THE PALETTE TO BE USED:    :
                ;                               0 = GREEN(1)/RED(2)/YELLOW(3)           :
                ;                               1 = CYAN(1)/MAGENTA(2)/WHITE(3)         :
                ;                       IN 40X25 OR 80X25 ALPHA MODES, THE VALUE SET    :
                ;                               FOR PALETTE COLOR 0 INDICATES THE       :
                ;                               BORDER COLOR TO BE USED (VALUES 0-31,   :
                ;                               WHERE 16-31 SELECT THE HIGH INTENSITY   :
                ;                               BACKGROUND SET.                         :
                ;       (AH) = 12 WRITE DOT                                             :
                ;               (DX) = ROW NUMBER                                       :
                ;               (CX) = COLUMN NUMBER                                    :
                ;               (AL) = COLOR VALUE                                      :
                ;                       IF BIT 7 OF AL = 1, THEN THE COLOR VALUE IS     :
                ;                       EXCLUSIVE OR'D WITH THE CURRENT CONTENTS OF     :
                ;                       THE DOT                                         :
                ;       (AH) = 13 READ DOT                                              :
                ;               (DX) = ROW NUMBER                                       :
                ;               (CX) = COLUMN NUMBER                                    :
                ;               (AL) RETURNS THE DOT READ                               :
                ;                                                                       :
                ; ASCII TELETYPE ROUTINE FOR OUTPUT                                     :
                ;                                                                       :
                ;       (AH) = 14 WRITE TELETYPE TO ACTIVE PAGE                         :
                ;               (AL) = CHAR TO WRITE                                    :
                ;               (BL) = FOREGROUND COLOR IN GRAPHICS MODE                :
                ;               NOTE -- SCREEN WIDTH IS CONTROLLED BY PREVIOUS MODE SET :
                ;                                                                       :
                ;       (AH) = 15 CURRENT VIDEO STATE                                   :
                ;               RETURNS THE CURRENT VIDEO STATE                         :
                ;               (AL) = MODE CURRENTLY SET ( SEE AH=0 FOR EXPLANATION)   :
                ;               (AH) = NUMBER OF CHARACTER COLUMNS ON SCREEN            :
                ;               (BH) = CURRENT ACTIVE DISPLAY PAGE                      :
                ;                                                                       :
                ;       CS,SS,DS,ES,BX,CX,DX PRESERVED DURING CALL                      :
                ;       ALL OTHERS DESTROYED                                            :
                ;-----------------------------------------------------------------------
                        ASSUME  CS:CODE,DS:DATA,ES:VIDEO_RAM
F045                    ORG     0F045H
F045            M1      LABEL   WORD                    ; TABLE OF ROUTINES WITHIN VIDEO I/O
F045 FCF0               DW      OFFSET  SET_MODE
F047 CDF1               DW      OFFSET  SET_CTYPE
F049 EEF1               DW      OFFSET  SET_CPOS
F04B 39F2               DW      OFFSET  READ_CURSOR
F04D 9CF7               DW      OFFSET  READ_LPEN
F04F 17F2               DW      OFFSET  ACT_DISP_PAGE
F051 96F2               DW      OFFSET  SCROLL_UP
F053 38F3               DW      OFFSET  SCROLL_DOWN
F055 74F3               DW      OFFSET  READ_AC_CURRENT
F057 B9F3               DW      OFFSET  WRITE_AC_CURRENT
F059 ECF3               DW      OFFSET  WRITE_C_CURRENT
F05B 4EF2               DW      OFFSET  SET_COLOR
F05D 2FF4               DW      OFFSET  WRITE_DOT
F05F 1EF4               DW      OFFSET  READ_DOT
F061 18F7               DW      OFFSET  WRITE_TTY
F063 74F2               DW      OFFSET  VIDEO_STATE
  0020          M1L     EQU     $-M1

F065                    ORG     0F065H
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
```

```
LOC OBJ                LINE   SOURCE

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 E94501                           JMP     VIDEO_RETURN              ; DO NOTHING IF NOT IN RANGE
F080                          M2:
F080 E8BB0E                           CALL    DDS
F083 B800B8                           MOV     AX,0B800H                 ; SEGMENT FOR COLOR CARD
F086 8B3E1000                         MOV     DI,EQUIP_FLAG             ; GET EQUIPMENT SETTING
F08A 81E73000                         AND     DI,30H                    ; ISOLATE CRT SWITCHES
F08E 83FF30                           CMP     DI,30H                    ; IS SETTING FOR BW CARD?
F091 7502                             JNE     M3
F093 B4B0                             MOV     AH,0B0H                   ; SEGMENT FOR BW CARD
F095                          M3:
F095 8EC0                             MOV     ES,AX                     ; SET UP TO POINT AT VIDEO RAM AREAS
F097 58                               POP     AX                        ; RECOVER VALUE
F098 8A264900                         MOV     AH,CRT_MODE               ; GET CURRENT MODE INTO AH
F09C 2EFFA445F0                       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                                  ORG     0F0A4H
F0A4                          VIDEO_PARMS     LABEL   BYTE
                              ;----- INIT_TABLE
F0A4 38                               DB      38H,28H,2DH,0AH,1FH,6,19H        ; SET UP FOR 40X25
F0A5 28
F0A6 2D
F0A7 0A
F0A8 1F
F0A9 06
F0AA 19
F0AB 1C                               DB      1CH,2,7,6,7
F0AC 02
F0AD 07
F0AE 06
F0AF 07
F0B0 00                               DB      0,0,0,0
F0B1 00
F0B2 00
F0B3 00
  0010                        M4      EQU     $-VIDEO_PARMS

F0B4 71                               DB      71H,50H,5AH,0AH,1FH,6,19H        ; SET UP FOR 80X25
F0B5 50
F0B6 5A
F0B7 0A
F0B8 1F
F0B9 06
F0BA 19
F0BB 1C                               DB      1CH,2,7,6,7
F0BC 02
F0BD 07
F0BE 06
F0BF 07
F0C0 00                               DB      0,0,0,0
F0C1 00
F0C2 00
F0C3 00

F0C4 38                               DB      38H,28H,2DH,0AH,7FH,6,64H        ; SET UP FOR GRAPHICS
F0C5 28
```

```
LOC OBJ              LINE    SOURCE

F0C6 2D
F0C7 0A
F0C8 7F
F0C9 06
F0CA 64
F0CB 70              3535            DB      70H,2,1,6,7
F0CC 02
F0CD 01
F0CE 06
F0CF 07
F0D0 00              3536            DB      0,0,0,0
F0D1 00
F0D2 00
F0D3 00
                     3537
F0D4 61              3538            DB      61H,50H,52H,0FH,19H,6,19H        ; SET UP FOR 80X25 B&W CARD
F0D5 50
F0D6 52
F0D7 0F
F0D8 19
F0D9 06
F0DA 19
F0DB 19              3539            DB      19H,2,0DH,0BH,0CH
F0DC 02
F0DD 0D
F0DE 0B
F0DF 0C
F0E0 00              3540            DB      0,0,0,0
F0E1 00
F0E2 00
F0E3 00
                     3541
F0E4                 3542    M5      LABEL   WORD                    ; TABLE OF REGEN LENGTHS
F0E4 0008            3543            DW      2048                    ; 40X25
F0E6 0010            3544            DW      4096                    ; 80X25
F0E8 0040            3545            DW      16384                   ; GRAPHICS
F0EA 0040            3546            DW      16384
                     3547
                     3548    ;----- COLUMNS
                     3549
F0EC                 3550    M6      LABEL   BYTE
F0EC 28              3551            DB      40,40,80,80,40,40,80,80
F0ED 28
F0EE 50
F0EF 50
F0F0 28
F0F1 28
F0F2 50
F0F3 50
                     3552
                     3553    ;----- C_REG_TAB
                     3554
F0F4                 3555    M7      LABEL   BYTE                    ; TABLE OF MODE SETS
F0F4 2C              3556            DB      2CH,28H,2DH,29H,2AH,2EH,1EH,29H
F0F5 28
F0F6 2D
F0F7 29
F0F8 2A
F0F9 2E
F0FA 1E
F0FB 29
                     3557
F0FC                 3558    SET_MODE        PROC    NEAR
F0FC BAD403          3559            MOV     DX,03D4H                ; ADDRESS OF COLOR CARD
F0FF B300            3560            MOV     BL,0                    ; MODE SET FOR COLOR CARD
F101 83FF30          3561            CMP     DI,30H                  ; IS BW CARD INSTALLED
F104 7506            3562            JNE     M8                      ; OK WITH COLOR
F106 B007            3563            MOV     AL,7                    ; INDICATE BW CARD MODE
F108 B2B4            3564            MOV     DL,0B4H                 ; ADDRESS OF BW CARD (3B4)
F10A FEC3            3565            INC     BL                      ; MODE SET FOR BW CARD
F10C                 3566    M8:
F10C 8AE0            3567            MOV     AH,AL                   ; SAVE MODE IN AH
F10E A24900          3568            MOV     CRT_MODE,AL             ; SAVE IN GLOBAL VARIABLE
F111 89166300        3569            MOV     ADDR_6845,DX            ; SAVE ADDRESS OF BASE
F115 1E              3570            PUSH    DS                      ; SAVE POINTER TO DATA SEGMENT
F116 50              3571            PUSH    AX                      ; SAVE MODE
F117 52              3572            PUSH    DX                      ; SAVE OUTPUT PORT VALUE
```

```
LOC OBJ              SOURCE

F118 83C204                  ADD     DX,4                    ; POINT TO CONTROL REGISTER
F11B 8AC3                    MOV     AL,BL                   ; GET MODE SET FOR CARD
F11D EE                      OUT     DX,AL                   ; RESET VIDEO
F11E 5A                      POP     DX                      ; BACK TO BASE REGISTER
F11F 2BC0                    SUB     AX,AX                   ; SET UP FOR ABS0 SEGMENT
F121 8ED8                    MOV     DS,AX                   ; ESTABLISH VECTOR TABLE ADDRESSING
                             ASSUME  DS:ABS0
F123 C51E7400                LDS     BX,PARM_PTR             ; GET POINTER TO VIDEO PARMS
F127 58                      POP     AX                      ; RECOVER PARMS
                             ASSUME  DS:CODE
F128 B91000                  MOV     CX,M4                   ; LENGTH OF EACH ROW OF TABLE
F12B 80FC02                  CMP     AH,2                    ; DETERMINE WHICH ONE TO USE
F12E 7210                    JC      M9                      ; MODE IS 0 OR 1
F130 03D9                    ADD     BX,CX                   ; MOVE TO NEXT ROW OF INIT TABLE
F132 80FC04                  CMP     AH,4
F135 7209                    JC      M9                      ; MODE IS 2 OR 3
F137 03D9                    ADD     BX,CX                   ; MOVE TO GRAPHICS ROW OF INIT_TABLE
F139 80FC07                  CMP     AH,7
F13C 7202                    JC      M9                      ; MODE IS 4,5, OR 6
F13E 03D9                    ADD     BX,CX                   ; MOVE TO BW CARD ROW OF INIT_TABLE

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

F140                 M9:                                     ; OUT_INIT
F140 50                      PUSH    AX                      ; SAVE MODE IN AH
F141 32E4                    XOR     AH,AH                   ; AH WILL SERVE AS REGISTER
                                                             ;  NUMBER DURING LOOP

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

F143                 M10:                                    ; INIT LOOP
F143 8AC4                    MOV     AL,AH                   ; GET 6845 REGISTER NUMBER
F145 EE                      OUT     DX,AL
F146 42                      INC     DX                      ; POINT TO DATA PORT
F147 FEC4                    INC     AH                      ; NEXT REGISTER VALUE
F149 8A07                    MOV     AL,[BX]                 ; GET TABLE VALUE
F14B EE                      OUT     DX,AL                   ; OUT TO CHIP
F14C 43                      INC     BX                      ; NEXT IN TABLE
F14D 4A                      DEC     DX                      ; BACK TO POINTER REGISTER
F14E E2F3                    LOOP    M10                     ; DO THE WHOLE TABLE
F150 58                      POP     AX                      ; GET MODE BACK
F151 1F                      POP     DS                      ; RECOVER SEGMENT VALUE
                             ASSUME  DS:DATA

                     ;----- FILL REGEN AREA WITH BLANK

F152 33FF                    XOR     DI,DI                   ; SET UP POINTER FOR REGEN
F154 893E4E00                MOV     CRT_START,DI            ; START ADDRESS SAVED IN GLOBAL
F158 C606620000              MOV     ACTIVE_PAGE,0           ; SET PAGE VALUE
F15D B90020                  MOV     CX,8192                 ; NUMBER OF WORDS IN COLOR CARD
F160 80FC04                  CMP     AH,4                    ; TEST FOR GRAPHICS
F163 720B                    JC      M12                     ; NO_GRAPHICS_INIT
F165 80FC07                  CMP     AH,7                    ; TEST FOR BW CARD
F168 7404                    JE      M11                     ; BW_CARD_INIT
F16A 33C0                    XOR     AX,AX                   ; FILL FOR GRAPHICS MODE
F16C EB05                    JMP     SHORT M13               ; CLEAR_BUFFER
F16E                 M11:                                    ; BW_CARD_INIT
F16E B508                    MOV     CH,08H                  ; BUFFER SIZE ON BW CARD
F170                 M12:                                    ; NO_GRAPHICS_INIT
F170 B82007                  MOV     AX,' '+7*256            ; FILL CHAR FOR ALPHA
F173                 M13:                                    ; CLEAR_BUFFER
F173 F3                      REP     STOSW                   ; FILL THE REGEN BUFFER WITH BLANKS
F174 AB

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

F175 C70660000706            MOV     CURSOR_MODE,607H        ; SET CURRENT CURSOR MODE
F17B A04900                  MOV     AL,CRT_MODE             ; GET THE MODE
F17E 32E4                    XOR     AH,AH                   ;  INTO AX REGISTER
F180 8BF0                    MOV     SI,AX                   ; TABLE POINTER, INDEXED BY MODE
F182 8B166300                MOV     DX,ADDR_6845            ; PREPARE TO OUTPUT TO
                                                             ;  VIDEO ENABLE PORT
F186 83C204                  ADD     DX,4
F189 2E8A84F4F0              MOV     AL,CS:[SI+OFFSET M7]
F18E EE                      OUT     DX,AL                   ; SET VIDEO ENABLE PORT
F18F A26500                  MOV     CRT_MODE_SET,AL         ; SAVE THAT VALUE
```

```
LOC OBJ                 SOURCE

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

F192 2E8A84ECF0                MOV     AL,CS:[SI + OFFSET M6]
F197 32E4                      XOR     AH,AH
F199 A34A00                    MOV     CRT_COLS,AX             ; NUMBER OF COLUMNS IN THIS SCREEN

                        ;----- SET CURSOR POSITIONS

F19C 81E60E00                  AND     SI,0EH                  ; WORD OFFSET INTO CLEAR LENGTH TABLE
F1A0 2E8B8CE4F0                MOV     CX,CS:[SI + OFFSET M5]  ; LENGTH TO CLEAR
F1A5 890E4C00                  MOV     CRT_LEN,CX              ; SAVE LENGTH OF CRT -- NOT USED FOR BW
F1A9 B90800                    MOV     CX,8                    ; CLEAR ALL CURSOR POSITIONS
F1AC BF5000                    MOV     DI,OFFSET CURSOR_POSN
F1AF 1E                        PUSH    DS                      ; ESTABLISH SEGMENT
F1B0 07                        POP     ES                      ;   ADDRESSING
F1B1 33C0                      XOR     AX,AX
F1B3 F3                        REP     STOSW                   ; FILL WITH ZEROES
F1B4 AB

                        ;----- SET UP OVERSCAN REGISTER

F1B5 42                        INC     DX                      ; SET OVERSCAN PORT TO A DEFAULT
F1B6 B030                      MOV     AL,30H                  ; VALUE OF 30H FOR ALL MODES
                                                               ;  EXCEPT 640X200
F1B8 803E490006                CMP     CRT_MODE,6              ; SEE IF THE MODE IS 640X200 BW
F1BD 7502                      JNZ     M14                     ; IF IT ISNT 640X200, THEN GOTO REGULAR
F1BF B03F                      MOV     AL,3FH                  ; IF IT IS 640X200, THEN PUT IN 3FH
F1C1                    M14:
F1C1 EE                        OUT     DX,AL                   ; OUTPUT THE CORRECT VALUE TO 3D9 PORT
F1C2 A26600                    MOV     CRT_PALETTE,AL          ; SAVE THE VALUE FOR FUTURE USE

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

F1C5                    VIDEO_RETURN:
F1C5 5F                        POP     DI
F1C6 5E                        POP     SI
F1C7 5B                        POP     BX
F1C8                    M15:                                   ; VIDEO_RETURN_C
F1C8 59                        POP     CX
F1C9 5A                        POP     DX
F1CA 1F                        POP     DS
F1CB 07                        POP     ES                      ; RECOVER SEGMENTS
F1CC 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                                                  :
                        ;--------------------------------------------------------------
F1CD                    SET_CTYPE       PROC    NEAR
F1CD B40A                      MOV     AH,10                   ; 6845 REGISTER FOR CURSOR SET
F1CF 890E6000                  MOV     CURSOR_MODE,CX          ; SAVE IN DATA AREA
F1D3 E80200                    CALL    M16                     ; OUTPUT CX REG
F1D6 EBED                      JMP     VIDEO_RETURN

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

F1D8                    M16:
F1D8 8B166300                  MOV     DX,ADDR_6845            ; ADDRESS REGISTER
F1DC 8AC4                      MOV     AL,AH                   ; GET VALUE
F1DE EE                        OUT     DX,AL                   ; REGISTER SET
F1DF 42                        INC     DX                      ; DATA REGISTER
F1E0 8AC5                      MOV     AL,CH                   ; DATA
F1E2 EE                        OUT     DX,AL
F1E3 4A                        DEC     DX
F1E4 8AC4                      MOV     AL,AH
F1E6 FEC0                      INC     AL                      ; POINT TO OTHER DATA REGISTER
F1E8 EE                        OUT     DX,AL                   ; SET FOR SECOND REGISTER
F1E9 42                        INC     DX
F1EA 8AC1                      MOV     AL,CL                   ; SECOND DATA VALUE
F1EC EE                        OUT     DX,AL
F1ED 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                      :
                    ;-----------------------------------------------
F1EE                SET_CPOS        PROC    NEAR
F1EE 8ACF                   MOV     CL,BH
F1F0 32ED                   XOR     CH,CH                   ; ESTABLISH LOOP COUNT
F1F2 D1E1                   SAL     CX,1                    ; WORD OFFSET
F1F4 8BF1                   MOV     SI,CX                   ; USE INDEX REGISTER
F1F6 895450                 MOV     [SI+OFFSET CURSOR_POSN],DX      ; SAVE THE POINTER
F1F9 383E6200               CMP     ACTIVE_PAGE,BH
F1FD 7505                   JNZ     M17                     ; SET_CPOS_RETURN
F1FF 8BC2                   MOV     AX,DX                   ; GET ROW/COLUMN TO AX
F201 E80200                 CALL    M18                     ; CURSOR_SET
F204                M17:                                    ; SET_CPOS_RETURN
F204 EBBF                   JMP     VIDEO_RETURN
                    SET_CPOS        ENDP

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

F206                M18     PROC    NEAR
F206 E87C00                 CALL    POSITION                ; DETERMINE LOCATION IN REGEN BUFFER
F209 8BC8                   MOV     CX,AX
F20B 030E4E00               ADD     CX,CRT_START            ; ADD IN THE START ADDR FOR THIS PAGE
F20F D1F9                   SAR     CX,1                    ; DIVIDE BY 2 FOR CHAR ONLY COUNT
F211 B40E                   MOV     AH,14                   ; REGISTER NUMBER FOR CURSOR
F213 E8C2FF                 CALL    M16                     ; OUTPUT THE VALUE TO THE 6845
F216 C3                     RET
                    M18     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                 :
                    ;--------------------------------------------------------------
F217                ACT_DISP_PAGE   PROC    NEAR
F217 A26200                 MOV     ACTIVE_PAGE,AL          ; SAVE ACTIVE PAGE VALUE
F21A 8B0E4C00               MOV     CX,CRT_LEN              ; GET SAVED LENGTH OF REGEN BUFFER
F21E 98                     CBW                             ; CONVERT AL TO WORD
F21F 50                     PUSH    AX                      ; SAVE PAGE VALUE
F220 F7E1                   MUL     CX                      ; DISPLAY PAGE TIMES REGEN LENGTH
F222 A34E00                 MOV     CRT_START,AX            ; SAVE START ADDRESS FOR
                                                            ;  LATER REQUIREMENTS
F225 8BC8                   MOV     CX,AX                   ; START ADDRESS TO CX
F227 D1F9                   SAR     CX,1                    ; DIVIDE BY 2 FOR 6845 HANDLING
F229 B40C                   MOV     AH,12                   ; 6845 REGISTER FOR START ADDRESS
F22B E8AAFF                 CALL    M16
F22E 5B                     POP     BX                      ; RECOVER PAGE VALUE
F22F D1E3                   SAL     BX,1                    ; *2 FOR WORD OFFSET
F231 8B4750                 MOV     AX,[BX + OFFSET CURSOR_POSN]    ; GET CURSOR FOR THIS PAGE
F234 E8CFFF                 CALL    M18                     ; SET THE CURSOR POSITION
F237 EB8C                   JMP     SHORT VIDEO_RETURN
                    ACT_DISP_PAGE   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                               :
                    ;--------------------------------------------------------------
F239                READ_CURSOR     PROC    NEAR
F239 8ADF                   MOV     BL,BH
F23B 32FF                   XOR     BH,BH
F23D D1E3                   SAL     BX,1                    ; WORD OFFSET
```

```
LOC OBJ                 SOURCE

F23F 8B5750                     MOV     DX,[BX+OFFSET CURSOR_POSN]
F242 8B0E6000                   MOV     CX,CURSOR_MODE
F246 5F                         POP     DI
F247 5E                         POP     SI
F248 5B                         POP     BX
F249 58                         POP     AX                      ; DISCARD SAVED CX AND DX
F24A 58                         POP     AX
F24B 1F                         POP     DS
F24C 07                         POP     ES
F24D CF                         IRET
                        READ_CURSOR     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 PALETTE 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                               :
                        ;---------------------------------------------------------------------
F24E                    SET_COLOR       PROC    NEAR
F24E 8B166300                   MOV     DX,ADDR_6845            ; I/O PORT FOR PALETTE
F252 83C205                     ADD     DX,5                    ; OVERSCAN PORT
F255 A06600                     MOV     AL,CRT_PALETTE          ; GET THE CURRENT PALETTE VALUE
F258 0AFF                       OR      BH,BH                   ; IS THIS COLOR 0?
F25A 750E                       JNZ     M20                     ; OUTPUT COLOR 1

                        ;----- HANDLE COLOR 0 BY SETTING THE BACKGROUND COLOR

F25C 24E0                       AND     AL,0E0H                 ; TURN OFF LOW 5 BITS OF CURRENT
F25E 80E31F                     AND     BL,01FH                 ; TURN OFF HIGH 3 BITS OF INPUT VALUE
F261 0AC3                       OR      AL,BL                   ; PUT VALUE INTO REGISTER
F263                    M19:                                    ; OUTPUT THE PALETTE
F263 EE                         OUT     DX,AL                   ; OUTPUT COLOR SELECTION TO 3D9 PORT
F264 A26600                     MOV     CRT_PALETTE,AL          ; SAVE THE COLOR VALUE
F267 E95BFF                     JMP     VIDEO_RETURN

                        ;----- HANDLE COLOR 1 BY SELECTING THE PALETTE TO BE USED

F26A                    M20:
F26A 24DF                       AND     AL,0DFH                 ; TURN OFF PALETTE SELECT BIT
F26C D0EB                       SHR     BL,1                    ; TEST THE LOW ORDER BIT OF BL
F26E 73F3                       JNC     M19                     ; ALREADY DONE
F270 0C20                       OR      AL,20H                  ; TURN ON PALETTE SELECT BIT
F272 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                     :
                        ;-----------------------------------------------
F274                    VIDEO_STATE     PROC    NEAR
F274 8A264A00                   MOV     AH,BYTE PTR CRT_COLS    ; GET NUMBER OF COLUMNS
F278 A04900                     MOV     AL,CRT_MODE             ; CURRENT MODE
F27B 8A3E6200                   MOV     BH,ACTIVE_PAGE          ; GET CURRENT ACTIVE PAGE
F27F 5F                         POP     DI                      ; RECOVER REGISTERS
F280 5E                         POP     SI
F281 59                         POP     CX                      ; DISCARD SAVED BX
F282 E943FF                     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    :
                     ;-----------------------------------------------------
F285                 POSITION        PROC    NEAR
F285 53                      PUSH    BX                      ; SAVE REGISTER
F286 8BD8                    MOV     BX,AX
F288 8AC4                    MOV     AL,AH                   ; ROWS TO AL
F28A F6264A00                MUL     BYTE PTR CRT_COLS       ; DETERMINE BYTES TO ROW
F28E 32FF                    XOR     BH,BH
F290 03C3                    ADD     AX,BX                   ; ADD IN COLUMN VALUE
F292 D1E0                    SAL     AX,1                    ; * 2 FOR ATTRIBUTE BYTES
F294 5B                      POP     BX
F295 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
F296                 SCROLL_UP       PROC    NEAR
F296 8AD8                    MOV     BL,AL                   ; SAVE LINE COUNT IN BL
F298 80FC04                  CMP     AH,4                    ; TEST FOR GRAPHICS MODE
F29B 7208                    JC      N1                      ; HANDLE SEPARATELY
F29D 80FC07                  CMP     AH,7                    ; TEST FOR BW CARD
F2A0 7403                    JE      N1
F2A2 E9F001                  JMP     GRAPHICS_UP
F2A5                 N1:                                     ; UP_CONTINUE
F2A5 53                      PUSH    BX                      ; SAVE FILL ATTRIBUTE IN BH
F2A6 8BC1                    MOV     AX,CX                   ; UPPER LEFT POSITION
F2A8 E83700                  CALL    SCROLL_POSITION         ; DO SETUP FOR SCROLL
F2AB 7431                    JZ      N7                      ; BLANK_FIELD
F2AD 03F0                    ADD     SI,AX                   ; FROM ADDRESS
F2AF 8AE6                    MOV     AH,DH                   ; # ROWS IN BLOCK
F2B1 2AE3                    SUB     AH,BL                   ; # ROWS TO BE MOVED
F2B3                 N2:                                     ; ROW_LOOP
F2B3 E87200                  CALL    N10                     ; MOVE ONE ROW
F2B6 03F5                    ADD     SI,BP
F2B8 03FD                    ADD     DI,BP                   ; POINT TO NEXT LINE IN BLOCK
F2BA FECC                    DEC     AH                      ; COUNT OF LINES TO MOVE
F2BC 75F5                    JNZ     N2                      ; ROW_LOOP
F2BE                 N3:                                     ; CLEAR_ENTRY
F2BE 58                      POP     AX                      ; RECOVER ATTRIBUTE IN AH
F2BF B020                    MOV     AL,' '                  ; FILL WITH BLANKS
F2C1                 N4:                                     ; CLEAR_LOOP
F2C1 E86D00                  CALL    N11                     ; CLEAR THE ROW
F2C4 03FD                    ADD     DI,BP                   ; POINT TO NEXT LINE
F2C6 FECB                    DEC     BL                      ; COUNTER OF LINES TO SCROLL
F2C8 75F7                    JNZ     N4                      ; CLEAR_LOOP
F2CA                 N5:                                     ; SCROLL_END
F2CA E8710C                  CALL    DDS
F2CD 803E490007              CMP     CRT_MODE,7              ; IS THIS THE BLACK AND WHITE CARD
F2D2 7407                    JE      N6                      ; IF SO, SKIP THE MODE RESET
F2D4 A06500                  MOV     AL,CRT_MODE_SET         ; GET THE VALUE OF THE MODE SET
F2D7 BAD803                  MOV     DX,03D8H                ; ALWAYS SET COLOR CARD PORT
F2DA EE                      OUT     DX,AL
F2DB                 N6:                                     ; VIDEO_RET_HERE
F2DB E9E7FE                  JMP     VIDEO_RETURN
F2DE                 N7:                                     ; BLANK_FIELD
F2DE 8ADE                    MOV     BL,DH                   ; GET ROW COUNT
F2E0 EBDC                    JMP     N3                      ; GO CLEAR THAT AREA
                     SCROLL_UP       ENDP

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

F2E2                 SCROLL_POSITION PROC    NEAR
F2E2 803E490002              CMP     CRT_MODE,2              ; TEST FOR SPECIAL CASE HERE
F2E7 7218                    JB      N9                      ; HAVE TO HANDLE 80X25 SEPARATELY
```

```
LOC OBJ              SOURCE

F2E9 803E490003              CMP     CRT_MODE,3
F2EE 7711                    JA      N9

                     ;----- 80X25 COLOR CARD SCROLL

F2F0 52                      PUSH    DX
F2F1 BADA03                  MOV     DX,3DAH                ; GUARANTEED TO BE COLOR CARD HERE
F2F4 50                      PUSH    AX
F2F5                 N8:                                    ; WAIT_DISP_ENABLE
F2F5 EC                      IN      AL,DX                  ; GET PORT
F2F6 A808                    TEST    AL,8                   ; WAIT FOR VERTICAL RETRACE
F2F8 74FB                    JZ      N8                     ; WAIT_DISP_ENABLE
F2FA B025                    MOV     AL,25H
F2FC B2D8                    MOV     DL,0D8H                ; DX=3D8
F2FE EE                      OUT     DX,AL                  ; TURN OFF VIDEO
F2FF 58                      POP     AX                     ; DURING VERTICAL RETRACE
F300 5A                      POP     DX
F301                 N9:
F301 E881FF                  CALL    POSITION               ; CONVERT TO REGEN POINTER
F304 03064E00                ADD     AX,CRT_START           ; OFFSET OF ACTIVE PAGE
F308 8BF8                    MOV     DI,AX                  ; TO ADDRESS FOR SCROLL
F30A 8BF0                    MOV     SI,AX                  ; FROM ADDRESS FOR SCROLL
F30C 2BD1                    SUB     DX,CX                  ; DX = #ROWS, #COLS IN BLOCK
F30E FEC6                    INC     DH
F310 FEC2                    INC     DL                     ; INCREMENT FOR 0 ORIGIN
F312 32ED                    XOR     CH,CH                  ; SET HIGH BYTE OF COUNT TO ZERO
F314 8B2E4A00                MOV     BP,CRT_COLS            ; GET NUMBER OF COLUMNS IN DISPLAY
F318 03ED                    ADD     BP,BP                  ; TIMES 2 FOR ATTRIBUTE BYTE
F31A 8AC3                    MOV     AL,BL                  ; GET LINE COUNT
F31C F6264A00                MUL     BYTE PTR CRT_COLS      ; DETERMINE OFFSET TO FROM ADDRESS
F320 03C0                    ADD     AX,AX                  ; *2 FOR ATTRIBUTE BYTE
F322 06                      PUSH    ES                     ; ESTABLISH ADDRESSING TO REGEN BUFFER
F323 1F                      POP     DS                     ;  FOR BOTH POINTERS
F324 80FB00                  CMP     BL,0                   ; 0 SCROLL MEANS BLANK FIELD
F327 C3                      RET                            ; RETURN WITH FLAGS SET
                     SCROLL_POSITION ENDP

                     ;----- MOVE_ROW

F328                 N10     PROC    NEAR
F328 8ACA                    MOV     CL,DL                  ; GET # OF COLS TO MOVE
F32A 56                      PUSH    SI
F32B 57                      PUSH    DI                     ; SAVE START ADDRESS
F32C F3                      REP     MOVSW                  ; MOVE THAT LINE ON SCREEN
F32D A5
F32E 5F                      POP     DI
F32F 5E                      POP     SI                     ; RECOVER ADDRESSES
F330 C3                      RET
                     N10     ENDP

                     ;----- CLEAR_ROW

F331                 N11     PROC    NEAR
F331 8ACA                    MOV     CL,DL                  ; GET # COLUMNS TO CLEAR
F333 57                      PUSH    DI
F334 F3                      REP     STOSW                  ; STORE THE FILL CHARACTER
F335 AB
F336 5F                      POP     DI
F337 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                             :
                     ; OUTPUT                                                 :
                     ;       NONE -- SCREEN IS SCROLLED                       :
                     ;--------------------------------------------------------
F338                 SCROLL_DOWN     PROC    NEAR
```

```
LOC OBJ             SOURCE

F338 FD                     STD                             ; DIRECTION FOR SCROLL DOWN
F339 8AD8                   MOV     BL,AL                   ; LINE COUNT TO BL
F33B 80FC04                 CMP     AH,4                    ; TEST FOR GRAPHICS
F33E 7208                   JC      N12
F340 80FC07                 CMP     AH,7                    ; TEST FOR BW CARD
F343 7403                   JE      N12
F345 E9A601                 JMP     GRAPHICS_DOWN
F348                N12:                                    ; CONTINUE DOWN
F348 53                     PUSH    BX                      ; SAVE ATTRIBUTE IN BH
F349 8BC2                   MOV     AX,DX                   ; LOWER RIGHT CORNER
F34B E894FF                 CALL    SCROLL_POSITION         ; GET REGEN LOCATION
F34E 7420                   JZ      N16
F350 2BF0                   SUB     SI,AX                   ; SI IS FROM ADDRESS
F352 8AE6                   MOV     AH,DH                   ; GET TOTAL # ROWS
F354 2AE3                   SUB     AH,BL                   ; COUNT TO MOVE IN SCROLL
F356                N13:
F356 E8CFFF                 CALL    N10                     ; MOVE ONE ROW
F359 2BF5                   SUB     SI,BP
F35B 2BFD                   SUB     DI,BP
F35D FECC                   DEC     AH
F35F 75F5                   JNZ     N13
F361                N14:
F361 58                     POP     AX                      ; RECOVER ATTRIBUTE IN AH
F362 B020                   MOV     AL,' '
F364                N15:
F364 E8CAFF                 CALL    N11                     ; CLEAR ONE ROW
F367 2BFD                   SUB     DI,BP                   ; GO TO NEXT ROW
F369 FECB                   DEC     BL
F36B 75F7                   JNZ     N15
F36D E95AFF                 JMP     N5                      ; SCROLL_END
F370                N16:
F370 8ADE                   MOV     BL,DH
F372 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
F374                READ_AC_CURRENT PROC    NEAR
F374 80FC04                 CMP     AH,4                    ; IS THIS GRAPHICS
F377 7208                   JC      P1
F379 80FC07                 CMP     AH,7                    ; IS THIS BW CARD
F37C 7403                   JE      P1
F37E E9A802                 JMP     GRAPHICS_READ
F381                P1:                                     ; READ_AC_CONTINUE
F381 E81A00                 CALL    FIND_POSITION
F384 8BF3                   MOV     SI,BX                   ; ESTABLISH ADDRESSING IN SI

                    ;----- WAIT FOR HORIZONTAL RETRACE

F386 8B166300               MOV     DX,ADDR_6845            ; GET BASE ADDRESS
F38A 83C206                 ADD     DX,6                    ; POINT AT STATUS PORT
F38D 06                     PUSH    ES
F38E 1F                     POP     DS                      ; GET SEGMENT FOR QUICK ACCESS
F38F                P2:                                     ; WAIT FOR RETRACE LOW
F38F EC                     IN      AL,DX                   ; GET STATUS
F390 A801                   TEST    AL,1                    ; IS HORZ RETRACE LOW
F392 75FB                   JNZ     P2                      ; WAIT UNTIL IT IS
F394 FA                     CLI                             ; NO MORE INTERRUPTS
F395                P3:                                     ; WAIT FOR RETRACE HIGH
F395 EC                     IN      AL,DX                   ; GET STATUS
F396 A801                   TEST    AL,1                    ; IS IT HIGH
F398 74FB                   JZ      P3                      ; WAIT UNTIL IT IS
F39A AD                     LODSW                           ; GET THE CHAR/ATTR
F39B E927FE                 JMP     VIDEO_RETURN
                    READ_AC_CURRENT ENDP
```

```

F39E                FIND_POSITION   PROC    NEAR
F39E 8ACF                   MOV     CL,BH                   ; DISPLAY PAGE TO CX
F3A0 32ED                   XOR     CH,CH
F3A2 8BF1                   MOV     SI,CX                   ; MOVE TO SI FOR INDEX
F3A4 D1E6                   SAL     SI,1                    ; * 2 FOR WORD OFFSET
F3A6 8B4450                 MOV     AX,[SI+ OFFSET CURSOR_POSN]     ; GET ROW/COLUMN OF THAT PAGE
F3A9 33DB                   XOR     BX,BX                   ; SET START ADDRESS TO ZERO
F3AB E306                   JCXZ    P5                      ; NO_PAGE
F3AD                P4:                                     ; PAGE_LOOP
F3AD 031E4C00               ADD     BX,CRT_LEN              ; LENGTH OF BUFFER
F3B1 E2FA                   LOOP    P4
F3B3                P5:                                     ; NO_PAGE
F3B3 E8CFFE                 CALL    POSITION                ; DETERMINE LOCATION IN REGEN
F3B6 03D8                   ADD     BX,AX                   ; ADD TO START OF REGEN
F3B8 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                                    :
                    ;------------------------------------------------
F3B9                WRITE_AC_CURRENT        PROC    NEAR
F3B9 80FC04                 CMP     AH,4                    ; IS THIS GRAPHICS
F3BC 7208                   JC      P6
F3BE 80FC07                 CMP     AH,7                    ; IS THIS BW CARD
F3C1 7403                   JE      P6
F3C3 E9B201                 JMP     GRAPHICS_WRITE
F3C6                P6:                                     ; WRITE_AC_CONTINUE
F3C6 8AE3                   MOV     AH,BL                   ; GET ATTRIBUTE TO AH
F3C8 50                     PUSH    AX                      ; SAVE ON STACK
F3C9 51                     PUSH    CX                      ; SAVE WRITE COUNT
F3CA E8D1FF                 CALL    FIND_POSITION
F3CD 8BFB                   MOV     DI,BX                   ; ADDRESS TO DI REGISTER
F3CF 59                     POP     CX                      ; WRITE COUNT
F3D0 5B                     POP     BX                      ; CHARACTER IN BX REG
F3D1                P7:                                     ; WRITE_LOOP

                    ;----- WAIT FOR HORIZONTAL RETRACE

F3D1 8B166300               MOV     DX,ADDR_6845            ; GET BASE ADDRESS
F3D5 83C206                 ADD     DX,6                    ; POINT AT STATUS PORT
F3D8                P8:
F3D8 EC                     IN      AL,DX                   ; GET STATUS
F3D9 A801                   TEST    AL,1                    ; IS IT LOW
F3DB 75FB                   JNZ     P8                      ; WAIT UNTIL IT IS
F3DD FA                     CLI                             ; NO MORE INTERRUPTS
F3DE                P9:
F3DE EC                     IN      AL,DX                   ; GET STATUS
F3DF A801                   TEST    AL,1                    ; IS IT HIGH
F3E1 74FB                   JZ      P9                      ; WAIT UNTIL IT IS
F3E3 8BC3                   MOV     AX,BX                   ; RECOVER THE CHAR/ATTR
F3E5 AB                     STOSW                           ; PUT THE CHAR/ATTR
F3E6 FB                     STI                             ; INTERRUPTS BACK ON
F3E7 E2E8                   LOOP    P7                      ;  AS MANY TIMES AS REQUESTED
F3E9 E9D9FD                 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     :
```

Appendix A

```
LOC OBJ               SOURCE

                      ;       (AL) = CHAR TO WRITE                    :
                      ;       (DS) = DATA SEGMENT                     :
                      ;       (ES) = REGEN SEGMENT                    :
                      ; OUTPUT                                        :
                      ;       NONE                                    :
                      ;-----------------------------------------------
F3EC                  WRITE_C_CURRENT PROC    NEAR
F3EC 80FC04                   CMP     AH,4                    ; IS THIS GRAPHICS
F3EF 7208                     JC      P10
F3F1 80FC07                   CMP     AH,7                    ; IS THIS BW CARD
F3F4 7403                     JE      P10
F3F6 E97F01                   JMP     GRAPHICS_WRITE
F3F9                  P10:
F3F9 50                       PUSH    AX                      ; SAVE ON STACK
F3FA 51                       PUSH    CX                      ; SAVE WRITE COUNT
F3FB E8A0FF                   CALL    FIND_POSITION
F3FE 8BFB                     MOV     DI,BX                   ; ADDRESS TO DI
F400 59                       POP     CX                      ; WRITE COUNT
F401 5B                       POP     BX                      ; BL HAS CHAR TO WRITE
F402                  P11:                                    ; WRITE_LOOP

                      ;----- WAIT FOR HORIZONTAL RETRACE

F402 8B166300                 MOV     DX,ADDR_6845            ; GET BASE ADDRESS
F406 83C206                   ADD     DX,6                    ; POINT AT STATUS PORT
F409                  P12:
F409 EC                       IN      AL,DX                   ; GET STATUS
F40A A801                     TEST    AL,1                    ; IS IT LOW
F40C 75FB                     JNZ     P12                     ; WAIT UNTIL IT IS
F40E FA                       CLI                             ; NO MORE INTERRUPTS
F40F                  P13:
F40F EC                       IN      AL,DX                   ; GET STATUS
F410 A801                     TEST    AL,1                    ; IS IT HIGH
F412 74FB                     JZ      P13                     ; WAIT UNTIL IT IS
F414 8AC3                     MOV     AL,BL                   ; RECOVER CHAR
F416 AA                       STOSB                           ; PUT THE CHAR/ATTR
F417 FB                       STI                             ; INTERRUPTS BACK ON
F418 47                       INC     DI                      ; BUMP POINTER PAST ATTRIBUTE
F419 E2E7                     LOOP    P11                     ;  AS MANY TIMES AS REQUESTED
F41B E9A7FD                   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
F41E                  READ_DOT        PROC    NEAR
F41E E83100                   CALL    R3                      ; DETERMINE BYTE POSITION OF DOT
F421 268A04                   MOV     AL,ES:[SI]              ; GET THE BYTE
F424 22C4                     AND     AL,AH                   ; MASK OFF THE OTHER BITS IN THE BYTE
F426 D2E0                     SHL     AL,CL                   ; LEFT JUSTIFY THE VALUE
F428 8ACE                     MOV     CL,DH                   ; GET NUMBER OF BITS IN RESULT
F42A D2C0                     ROL     AL,CL                   ; RIGHT JUSTIFY THE RESULT
F42C E996FD                   JMP     VIDEO_RETURN            ; RETURN FROM VIDEO IO
                      READ_DOT        ENDP

F42F                  WRITE_DOT       PROC    NEAR
F42F 50                       PUSH    AX                      ; SAVE DOT VALUE
F430 50                       PUSH    AX                      ;  TWICE
F431 E81E00                   CALL    R3                      ; DETERMINE BYTE POSITION OF THE DOT
F434 D2E8                     SHR     AL,CL                   ; SHIFT TO SET UP THE BITS FOR OUTPUT
F436 22C4                     AND     AL,AH                   ; STRIP OFF THE OTHER BITS
F438 268A0C                   MOV     CL,ES:[SI]              ; GET THE CURRENT BYTE
F43B 5B                       POP     BX                      ; RECOVER XOR FLAG
F43C F6C380                   TEST    BL,80H                  ; IS IT ON
```

```
LOC OBJ             SOURCE

F43F 750D                   JNZ     R2                      ; YES, XOR THE DOT
F441 F6D4                   NOT     AH                      ; SET THE MASK TO REMOVE THE
F443 22CC                   AND     CL,AH                   ;  INDICATED BITS
F445 0AC1                   OR      AL,CL                   ; OR IN THE NEW VALUE OF THOSE BITS
F447                R1:                                     ; FINISH_DOT
F447 268804                 MOV     ES:[SI],AL              ; RESTORE THE BYTE IN MEMORY
F44A 58                     POP     AX
F44B E977FD                 JMP     VIDEO_RETURN            ; RETURN FROM VIDEO IO
F44E                R2:                                     ; XOR_DOT
F44E 32C1                   XOR     AL,CL                   ; EXCLUSIVE OR THE DOTS
F450 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                               :
                    ;------------------------------------------------------
F452                R3      PROC    NEAR
F452 53                     PUSH    BX                      ; SAVE BX DURING OPERATION
F453 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

F454 B028                   MOV     AL,40
F456 52                     PUSH    DX                      ; SAVE ROW VALUE
F457 80E2FE                 AND     DL,0FEH                 ; STRIP OFF ODD/EVEN BIT
F45A F6E2                   MUL     DL                      ; AX HAS ADDRESS OF 1ST BYTE
                                                            ;  OF INDICATED ROW
F45C 5A                     POP     DX                      ; RECOVER IT
F45D F6C201                 TEST    DL,1                    ; TEST FOR EVEN/ODD
F460 7403                   JZ      R4                      ; JUMP IF EVEN ROW
F462 050020                 ADD     AX,2000H                ; OFFSET TO LOCATION OF ODD ROWS
F465                R4:                                     ; EVEN_ROW
F465 8BF0                   MOV     SI,AX                   ; MOVE POINTER TO SI
F467 58                     POP     AX                      ; RECOVER AL VALUE
F468 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)   :
                    ;--------------------------------------------------------------

F46A BBC002                 MOV     BX,2C0H
F46D B90203                 MOV     CX,302H                 ; SET PARMS FOR MED RES
F470 803E490006             CMP     CRT_MODE,6
F475 7206                   JC      R5                      ; HANDLE IF MED ARES
F477 BB8001                 MOV     BX,180H
F47A B90307                 MOV     CX,703H                 ; SET PARMS FOR HIGH RES

                    ;----- DETERMINE BIT OFFSET IN BYTE FROM COLUMN MASK

F47D                R5:
F47D 22EA                   AND     CH,DL                   ; ADDRESS OF PEL WITHIN BYTE TO CH

                    ;----- DETERMINE BYTE OFFSET FOR THIS LOCATION IN COLUMN

F47F D3EA                   SHR     DX,CL                   ; SHIFT BY CORRECT AMOUNT
F481 03F2                   ADD     SI,DX                   ; INCREMENT THE POINTER
F483 8AF7                   MOV     DH,BH                   ; GET THE # OF BITS IN RESULT TO DH

                    ;----- MULTIPLY BH (VALID BITS IN BYTE) BY CH (BIT OFFSET)

F485 2AC9                   SUB     CL,CL                   ; ZERO INTO STORAGE LOCATION
F487                R6:
```

```
LOC OBJ             SOURCE

F487 D0C8                   ROR     AL,1                    ; LEFT JUSTIFY THE VALUE
                                                            ;  IN AL (FOR WRITE)
F489 02CD                   ADD     CL,CH                   ; ADD IN THE BIT OFFSET VALUE
F48B FECF                   DEC     BH                      ; LOOP CONTROL
F48D 75F8                   JNZ     R6                      ; ON EXIT, CL HAS SHIFT COUNT
                                                            ;  TO RESTORE BITS
F48F 8AE3                   MOV     AH,BL                   ; GET MASK TO AH
F491 D2EC                   SHR     AH,CL                   ; MOVE THE MASK TO CORRECT LOCATION
F493 5B                     POP     BX                      ; RECOVER REG
F494 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                       :
                    ;--------------------------------------------------------------
F495                GRAPHICS_UP     PROC    NEAR
F495 8AD8                   MOV     BL,AL                   ; SAVE LINE COUNT IN BL
F497 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

F499 E86902                 CALL    GRAPH_POSN
F49C 8BF8                   MOV     DI,AX                   ; SAVE RESULT AS DESTINATION ADDRESS

                    ;----- DETERMINE SIZE OF WINDOW

F49E 2BD1                   SUB     DX,CX
F4A0 81C20101               ADD     DX,101H                 ; ADJUST VALUES
F4A4 D0E6                   SAL     DH,1                    ; MULTIPLY # ROWS BY 4
                                                            ;  SINCE 8 VERT DOTS/CHAR
F4A6 D0E6                   SAL     DH,1                    ;  AND EVEN/ODD ROWS

                    ;----- DETERMINE CRT MODE

F4A8 803E490006             CMP     CRT_MODE,6              ; TEST FOR MEDIUM RES
F4AD 7304                   JNC     R7                      ; FIND_SOURCE

                    ;----- MEDIUM RES UP

F4AF D0E2                   SAL     DL,1                    ; # COLUMNS * 2, SINCE 2 BYTES/CHAR
F4B1 D1E7                   SAL     DI,1                    ; OFFSET *2 SINCE 2 BYTES/CHAR

                    ;----- DETERMINE THE SOURCE ADDRESS IN THE BUFFER

F4B3                R7:                                     ; FIND_SOURCE
F4B3 06                     PUSH    ES                      ; GET SEGMENTS BOTH POINTING TO REGEN
F4B4 1F                     POP     DS
F4B5 2AED                   SUB     CH,CH                   ; ZERO TO HIGH OF COUNT REG
F4B7 D0E3                   SAL     BL,1                    ; MULTIPLY NUMBER OF LINES BY 4
F4B9 D0E3                   SAL     BL,1
F4BB 742D                   JZ      R11                     ; IF ZERO, THEN BLANK ENTIRE FIELD
F4BD 8AC3                   MOV     AL,BL                   ; GET NUMBER OF LINES IN AL
F4BF B450                   MOV     AH,80                   ; 80 BYTES/ROW
F4C1 F6E4                   MUL     AH                      ; DETERMINE OFFSET TO SOURCE
F4C3 8BF7                   MOV     SI,DI                   ; SET UP SOURCE
F4C5 03F0                   ADD     SI,AX                   ;  ADD IN OFFSET TO IT
F4C7 8AE6                   MOV     AH,DH                   ; NUMBER OF ROWS IN FIELD
F4C9 2AE3                   SUB     AH,BL                   ; DETERMINE NUMBER TO MOVE

                    ;----- LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD FIELDS

F4CB                R8:                                     ; ROW_LOOP
F4CB E88000                 CALL    R17                     ; MOVE ONE ROW
F4CE 81EEB01F               SUB     SI,2000H-80             ; MOVE TO NEXT ROW
F4D2 81EFB01F               SUB     DI,2000H-80
```

```
LOC OBJ            LINE    SOURCE

F4D6 FECC                          DEC     AH                      ; NUMBER OF ROWS TO MOVE
F4D8 75F1                          JNZ     R8                      ; CONTINUE TILL ALL MOVED

                           ;----- FILL IN THE VACATED LINE(S)

F4DA                       R9:                                     ; CLEAR_ENTRY
F4DA 8AC7                          MOV     AL,BH                   ; ATTRIBUTE TO FILL WITH
F4DC                       R10:
F4DC E88800                        CALL    R18                     ; CLEAR THAT ROW
F4DF 81EFB01F                      SUB     DI,2000H-80             ; POINT TO NEXT LINE
F4E3 FECB                          DEC     BL                      ; NUMBER OF LINES TO FILL
F4E5 75F5                          JNZ     R10                     ; CLEAR_LOOP
F4E7 E9D8FC                        JMP     VIDEO_RETURN            ; EVERYTHING DONE
F4EA                       R11:                                    ; BLANK_FIELD
F4EA 8ADE                          MOV     BL,DH                   ; SET BLANK COUNT TO
                                                                   ;  EVERYTHING IN FIELD
F4EC 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                       :
                           ;--------------------------------------------------------------
F4EE                       GRAPHICS_DOWN   PROC    NEAR
F4EE FD                            STD                             ; SET DIRECTION
F4EF 8AD8                          MOV     BL,AL                   ; SAVE LINE COUNT IN BL
F4F1 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

F4F3 E80F02                        CALL    GRAPH_POSN
F4F6 8BF8                          MOV     DI,AX                   ; SAVE RESULT AS DESTINATION ADDRESS

                           ;----- DETERMINE SIZE OF WINDOW

F4F8 2BD1                          SUB     DX,CX
F4FA 81C20101                      ADD     DX,101H                 ; ADJUST VALUES
F4FE D0E6                          SAL     DH,1                    ; MULTIPLY # ROWS BY 4
                                                                   ;  SINCE 8 VERT DOTS/CHAR
F500 D0E6                          SAL     DH,1                    ;  AND EVEN/ODD ROWS

                           ;----- DETERMINE CRT MODE

F502 803E490006                    CMP     CRT_MODE,6              ; TEST FOR MEDIUM RES
F507 7305                          JNC     R12                     ; FIND_SOURCE_DOWN

                           ;----- MEDIUM RES DOWN

F509 D0E2                          SAL     DL,1                    ; # COLUMNS * 2, SINCE
                                                                   ;  2 BYTES/CHAR (OFFSET OK)
F50B D1E7                          SAL     DI,1                    ; OFFSET *2 SINCE 2 BYTES/CHAR
F50D 47                            INC     DI                      ; POINT TO LAST BYTE

                           ;----- DETERMINE THE SOURCE ADDRESS IN THE BUFFER

F50E                       R12:                                    ; FIND_SOURCE_DOWN
F50E 06                            PUSH    ES                      ; BOTH SEGMENTS TO REGEN
F50F 1F                            POP     DS
F510 2AED                          SUB     CH,CH                   ; ZERO TO HIGH OF COUNT REG
F512 81C7F000                      ADD     DI,240                  ; POINT TO LAST ROW OF PIXELS
F516 D0E3                          SAL     BL,1                    ; MULTIPLY NUMBER OF LINES BY 4
F518 D0E3                          SAL     BL,1
F51A 742E                          JZ      R16                     ; IF ZERO, THEN BLANK ENTIRE FIELD
F51C 8AC3                          MOV     AL,BL                   ; GET NUMBER OF LINES IN AL
F51E B450                          MOV     AH,80                   ; 80 BYTES/ROW
F520 F6E4                          MUL     AH                      ; DETERMINE OFFSET TO SOURCE
```

```
LOC OBJ                SOURCE

F522 8BF7                     MOV     SI,DI                   ; SET UP SOURCE
F524 2BF0                     SUB     SI,AX                   ;  SUBTRACT THE OFFSET
F526 8AE6                     MOV     AH,DH                   ; NUMBER OF ROWS IN FIELD
F528 2AE3                     SUB     AH,BL                   ; DETERMINE NUMBER TO MOVE

                       ;----- LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD FIELDS

F52A                   R13:                                   ; ROW_LOOP_DOWN
F52A E82100                   CALL    R17                     ; MOVE ONE ROW
F52D 81EE5020                 SUB     SI,2000H+80             ; MOVE TO NEXT ROW
F531 81EF5020                 SUB     DI,2000H+80
F535 FECC                     DEC     AH                      ; NUMBER OF ROWS TO MOVE
F537 75F1                     JNZ     R13                     ; CONTINUE TILL ALL MOVED

                       ;----- FILL IN THE VACATED LINE(S)

F539                   R14:                                   ; CLEAR_ENTRY_DOWN
F539 8AC7                     MOV     AL,BH                   ; ATTRIBUTE TO FILL WITH
F53B                   R15:                                   ; CLEAR_LOOP_DOWN
F53B E82900                   CALL    R18                     ; CLEAR A ROW
F53E 81EF5020                 SUB     DI,2000H+80             ; POINT TO NEXT LINE
F542 FECB                     DEC     BL                      ; NUMBER OF LINES TO FILL
F544 75F5                     JNZ     R15                     ; CLEAR_LOOP_DOWN
F546 FC                       CLD                             ; RESET THE DIRECTION FLAG
F547 E97BFC                   JMP     VIDEO_RETURN            ; EVERYTHING DONE
F54A                   R16:                                   ; BLANK_FIELD_DOWN
F54A 8ADE                     MOV     BL,DH                   ; SET BLANK COUNT TO
                                                              ;  EVERYTHING IN FIELD
F54C EBEB                     JMP     R14                     ; CLEAR THE FIELD
                       GRAPHICS_DOWN  ENDP

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

F54E                   R17    PROC    NEAR
F54E 8ACA                     MOV     CL,DL                   ; NUMBER OF BYTES IN THE ROW
F550 56                       PUSH    SI
F551 57                       PUSH    DI                      ; SAVE POINTERS
F552 F3                       REP     MOVSB                   ; MOVE THE EVEN FIELD
F553 A4
F554 5F                       POP     DI
F555 5E                       POP     SI
F556 81C60020                 ADD     SI,2000H
F55A 81C70020                 ADD     DI,2000H                ; POINT TO THE ODD FIELD
F55E 56                       PUSH    SI
F55F 57                       PUSH    DI                      ; SAVE THE POINTERS
F560 8ACA                     MOV     CL,DL                   ; COUNT BACK
F562 F3                       REP     MOVSB                   ; MOVE THE ODD FIELD
F563 A4
F564 5F                       POP     DI
F565 5E                       POP     SI                      ; POINTERS BACK
F566 C3                       RET                             ; RETURN TO CALLER
                       R17    ENDP

                       ;----- CLEAR A SINGLE ROW

F567                   R18    PROC    NEAR
F567 8ACA                     MOV     CL,DL                   ; NUMBER OF BYTES IN FIELD
F569 57                       PUSH    DI                      ; SAVE POINTER
F56A F3                       REP     STOSB                   ; STORE THE NEW VALUE
F56B AA
F56C 5F                       POP     DI                      ; POINTER BACK
F56D 81C70020                 ADD     DI,2000H                ; POINT TO ODD FIELD
F571 57                       PUSH    DI
F572 8ACA                     MOV     CL,DL
F574 F3                       REP     STOSB                   ; FILL THE ODD FILELD
F575 AA
F576 5F                       POP     DI
F577 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
F578                GRAPHICS_WRITE  PROC    NEAR
F578 B400                   MOV     AH,0                    ; ZERO TO HIGH OF CODE POINT
F57A 50                     PUSH    AX                      ; SAVE CODE POINT VALUE

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

F57B E88401                 CALL    S26                     ; FIND LOCATION IN REGEN BUFFER
F57E 8BF8                   MOV     DI,AX                   ; REGEN POINTER IN DI

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

F580 58                     POP     AX                      ; RECOVER CODE POINT
F581 3C80                   CMP     AL,80H                  ; IS IT IN SECOND HALF
F583 7306                   JAE     S1                      ; YES

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

F585 BE6EFA                 MOV     SI,0FA6EH               ; CRT_CHAR_GEN (OFFSET OF IMAGES)
F588 0E                     PUSH    CS                      ; SAVE SEGMENT ON STACK
F589 EB0F                   JMP     SHORT S2                ; DETERMINE_MODE

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

F58B                S1:                                     ; EXTEND_CHAR
F58B 2C80                   SUB     AL,80H                  ; ZERO ORIGIN FOR SECOND HALF
F58D 1E                     PUSH    DS                      ; SAVE DATA POINTER
F58E 2BF6                   SUB     SI,SI
F590 8EDE                   MOV     DS,SI                   ; ESTABLISH VECTOR ADDRESSING
                            ASSUME  DS:ABS0
F592 C5367C00               LDS     SI,EXT_PTR              ; GET THE OFFSET OF THE TABLE
F596 8CDA                   MOV     DX,DS                   ; GET THE SEGMENT OF THE TABLE
                            ASSUME  DS:DATA
F598 1F                     POP     DS                      ; RECOVER DATA SEGMENT
F599 52                     PUSH    DX                      ; SAVE TABLE SEGMENT ON STACK

                    ;----- DETERMINE GRAPHICS MODE IN OPERATION

F59A                S2:                                     ; DETERMINE_MODE
F59A D1E0                   SAL     AX,1                    ; MULTIPLY CODE POINT
F59C D1E0                   SAL     AX,1                    ;  VALUE BY 8
F59E D1E0                   SAL     AX,1
F5A0 03F0                   ADD     SI,AX                   ; SI HAS OFFSET OF DESIRED CODES
F5A2 803E490006             CMP     CRT_MODE,6
F5A7 1F                     POP     DS                      ; RECOVER TABLE POINTER SEGMENT
F5A8 722C                   JC      S7                      ; TEST FOR MEDIUM RESOLUTION MODE

                    ;----- HIGH RESOLUTION MODE

F5AA                S3:                                     ; HIGH_CHAR
F5AA 57                     PUSH    DI                      ; SAVE REGEN POINTER
F5AB 56                     PUSH    SI                      ; SAVE CODE POINTER
F5AC B604                   MOV     DH,4                    ; NUMBER OF TIMES THROUGH LOOP
```

```
LOC OBJ            LINE    SOURCE

F5AE                       S4:
F5AE AC                            LODSB                          ; GET BYTE FROM CODE POINTS
F5AF F6C380                        TEST    BL,80H                 ; SHOULD WE USE THE FUNCTION
F5B2 7516                          JNZ     S6                     ;  TO PUT CHAR IN
F5B4 AA                            STOSB                          ; STORE IN REGEN BUFFER
F5B5 AC                            LODSB
F5B6                       S5:
F5B6 268885FF1F                    MOV     ES:[DI+2000H-1],AL     ; STORE IN SECOND HALF
F5BB 83C74F                        ADD     DI,79                  ; MOVE TO NEXT ROW IN REGEN
F5BE FECE                          DEC     DH                     ; DONE WITH LOOP
F5C0 75EC                          JNZ     S4
F5C2 5E                            POP     SI
F5C3 5F                            POP     DI                     ; RECOVER REGEN POINTER
F5C4 47                            INC     DI                     ; POINT TO NEXT CHAR POSITION
F5C5 E2E3                          LOOP    S3                     ; MORE CHARS TO WRITE
F5C7 E9FBFB                        JMP     VIDEO_RETURN
F5CA                       S6:
F5CA 263205                        XOR     AL,ES:[DI]             ; EXCLUSIVE OR WITH CURRENT
F5CD AA                            STOSB                          ; STORE THE CODE POINT
F5CE AC                            LODSB                          ; AGAIN FOR ODD FIELD
F5CF 263285FF1F                    XOR     AL,ES:[DI+2000H-1]
F5D4 EBE0                          JMP     S5                     ; BACK TO MAINSTREAM

                           ;----- MEDIUM RESOLUTION WRITE

F5D6                       S7:                                    ; MED_RES_WRITE
F5D6 8AD3                          MOV     DL,BL                  ; SAVE HIGH COLOR BIT
F5D8 D1E7                          SAL     DI,1                   ; OFFSET*2 SINCE 2 BYTES/CHAR
F5DA E80100                        CALL    S19                    ; EXPAND BL TO FULL WORD OF COLOR
F5DD                       S8:                                    ; MED_CHAR
F5DD 57                            PUSH    DI                     ; SAVE REGEN POINTER
F5DE 56                            PUSH    SI                     ; SAVE THE CODE POINTER
F5DF B604                          MOV     DH,4                   ; NUMBER OF LOOPS
F5E1                       S9:
F5E1 AC                            LODSB                          ; GET CODE POINT
F5E2 E80E00                        CALL    S21                    ; DOUBLE UP ALL THE BITS
F5E5 23C3                          AND     AX,BX                  ; CONVERT THEM TO FOREGROUND
                                                                  ;  COLOR ( 0 BACK )
F5E7 F6C280                        TEST    DL,80H                 ; IS THIS XOR FUNCTION
F5EA 7407                          JZ      S10                    ; NO, STORE IT IN AS IT IS
F5EC 263225                        XOR     AH,ES:[DI]             ; DO FUNCTION WITH HALF
F5EF 26324501                      XOR     AL,ES:[DI+1]           ;  AND WITH OTHER HALF
F5F3                       S10:
F5F3 268825                        MOV     ES:[DI],AH             ; STORE FIRST BYTE
F5F6 26884501                      MOV     ES:[DI+1],AL           ; STORE SECOND BYTE
F5FA AC                            LODSB                          ; GET CODE POINT
F5FB E8C500                        CALL    S21
F5FE 23C3                          AND     AX,BX                  ; CONVERT TO COLOR
F600 F6C280                        TEST    DL,80H                 ; AGAIN, IS THIS XOR FUNCTION
F603 740A                          JZ      S11                    ; NO, JUST STORE THE VALUES
F605 2632A50020                    XOR     AH,ES:[DI+2000H]       ; FUNCTION WITH FIRST HALF
F60A 2632850120                    XOR     AL,ES:[DI+2001H]       ; AND WITH SECOND HALF
F60F                       S11:
F60F 2688A50020                    MOV     ES:[DI+2000H],AH
F614 2688850120                    MOV     ES:[DI+2000H+1],AL     ; STORE IN SECOND PORTION OF BUFFER
F619 83C750                        ADD     DI,80                  ; POINT TO NEXT LOCATION
F61C FECE                          DEC     DH
F61E 75C1                          JNZ     S9                     ; KEEP GOING
F620 5E                            POP     SI                     ; RECOVER CODE PONTER
F621 5F                            POP     DI                     ; RECOVER REGEN POINTER
F622 47                            INC     DI                     ; POINT TO NEXT CHAR POSITION
F623 47                            INC     DI
F624 E2B7                          LOOP    S8                     ; MORE TO WRITE
F626 E99CFB                        JMP     VIDEO_RETURN
                           GRAPHICS_WRITE  ENDP
                           ;------------------------
                           ; GRAPHICS READ          :
                           ;------------------------
F629                       GRAPHICS_READ   PROC    NEAR
F629 E80600                        CALL    S26                    ; CONVERTED TO OFFSET IN REGEN
F62C 8BF0                          MOV     SI,AX                  ; SAVE IN SI
F62E 83EC08                        SUB     SP,8                   ; ALLOCATE SPACE TO SAVE THE
                                                                  ;  READ CODE POINT
F631 8BEC                          MOV     BP,SP                  ; POINTER TO SAVE AREA

                           ;----- DETERMINE GRAPHICS MODES

```

```
F633 803E490006            CMP     CRT_MODE,6
F638 06                    PUSH    ES
F639 1F                    POP     DS                      ; POINT TO REGEN SEGMENT
F63A 721A                  JC      S13                     ; MEDIUM RESOLUTION

                  ;----- HIGH RESOLUTION READ

                  ;----- GET VALUES FROM REGEN BUFFER AND CONVERT TO CODE POINT

F63C B604                  MOV     DH,4                    ; NUMBER OF PASSES
F63E              S12:
F63E 8A04                  MOV     AL,[SI]                 ; GET FIRST BYTE
F640 884600                MOV     [BP],AL                 ; SAVE IN STORAGE AREA
F643 45                    INC     BP                      ; NEXT LOCATION
F644 8A840020              MOV     AL,[SI+2000H]           ; GET LOWER REGION BYTE
F648 884600                MOV     [BP],AL                 ; ADJUST AND STORE
F64B 45                    INC     BP
F64C 83C650                ADD     SI,80                   ; POINTER INTO REGEN
F64F FECE                  DEC     DH                      ; LOOP CONTROL
F651 75EB                  JNZ     S12                     ; DO IT SOME MORE
F653 EB1790                JMP     S15                     ; GO MATCH THE SAVED CODE POINTS

                  ;----- MEDIUM RESOLUTION READ

F656              S13:                                     ; MED_RES_READ
F656 D1E6                  SAL     SI,1                    ; OFFSET*2 SINCE 2 BYTES/CHAR
F658 B604                  MOV     DH,4                    ; NUMBER OF PASSES
F65A              S14:
F65A E88800                CALL    S23                     ; GET PAIR BYTES FROM REGEN
                                                           ;  INTO SINGLE SAVE
F65D 81C60020              ADD     SI,2000H                ; GO TO LOWER REGION
F661 E88100                CALL    S23                     ; GET THIS PAIR INTO SAVE
F664 81EEB01F              SUB     SI,2000H-80             ; ADJUST POINTER BACK INTO UPPER
F668 FECE                  DEC     DH
F66A 75EE                  JNZ     S14                     ; KEEP GOING UNTIL ALL 8 DONE

                  ;----- SAVE AREA HAS CHARACTER IN IT, MATCH IT

F66C              S15:                                     ; FIND_CHAR
F66C BF6EFA90              MOV     DI,OFFSET CRT_CHAR_GEN  ; ESTABLISH ADDRESSING
F670 0E                    PUSH    CS
F671 07                    POP     ES                      ; CODE POINTS IN CS
F672 83ED08                SUB     BP,8                    ; ADJUST POINTER TO BEGINNING
                                                           ;  OF SAVE AREA
F675 8BF5                  MOV     SI,BP
F677 FC                    CLD                             ; ENSURE DIRECTION
F678 B000                  MOV     AL,0                    ; CURRENT CODE POINT BEING MATCHED
F67A              S16:
F67A 16                    PUSH    SS                      ; ESTABLISH ADDRESSING TO STACK
F67B 1F                    POP     DS                      ; FOR THE STRING COMPARE
F67C BA8000                MOV     DX,128                  ; NUMBER TO TEST AGAINST
F67F              S17:
F67F 56                    PUSH    SI                      ; SAVE SAVE AREA POINTER
F680 57                    PUSH    DI                      ; SAVE CODE POINTER
F681 B90800                MOV     CX,8                    ; NUMBER OF BYTES TO MATCH
F684 F3                    REPE    CMPSB                   ; COMPARE THE 8 BYTES
F685 A6
F686 5F                    POP     DI                      ; RECOVER THE POINTERS
F687 5E                    POP     SI
F688 741E                  JZ      S18                     ; IF ZERO FLAG SET, THEN MATCH OCCURRED
F68A FEC0                  INC     AL                      ; NO MATCH, MOVE ON TO NEXT
F68C 83C708                ADD     DI,8                    ; NEXT CODE POINT
F68F 4A                    DEC     DX                      ; LOOP CONTROL
F690 75ED                  JNZ     S17                     ; DO ALL OF THEM

                  ;----- CHAR NOT MATCHED, MIGHT BE IN USER SUPPLIED SECOND HALF

F692 3C00                  CMP     AL,0                    ; AL <> 0 IF ONLY 1ST HALF SCANNED
F694 7412                  JE      S18                     ; IF = 0, THEN ALL HAS BEEN SCANNED
F696 2BC0                  SUB     AX,AX
F698 8ED8                  MOV     DS,AX                   ; ESTABLISH ADDRESSING TO VECTOR
                           ASSUME  DS:ABS0
F69A C43E7C00              LES     DI,EXT_PTR              ; GET POINTER
F69E 8CC0                  MOV     AX,ES                   ; SEE IF THE POINTER REALLY EXISTS
F6A0 0BC7                  OR      AX,DI                   ; IF ALL 0, THEN DOESN'T EXIST
F6A2 7404                  JZ      S18                     ; NO SENSE LOOKING
F6A4 B080                  MOV     AL,128                  ; ORIGIN FOR SECOND HALF
```

```
LOC OBJ              SOURCE

F6A6 EBD2                    JMP     S16                     ; GO BACK AND TRY FOR IT
                             ASSUME  DS:DATA

                     ;----- CHARACTER IS FOUND ( AL=0 IF NOT FOUND )

F6A8                 S18:
F6A8 83C408                  ADD     SP,8                    ; READJUST THE STACK, THROW AWAY SAVE
F6AB E917F8                  JMP     VIDEO_RETURN            ; ALL DONE
                     GRAPHICS_READ   ENDP
                     ;--------------------------------------------------------
                     ; EXPAND_MED_COLOR                                       :
                     ;       THIS ROUTINE EXPANDS THE LOW 2 BITS IN BL TO     :
                     ;       FILL THE ENTIRE BX REGISTER                      :
                     ; ENTRY                                                  :
                     ;       BL = COLOR TO BE USED ( LOW 2 BITS )             :
                     ; EXIT                                                   :
                     ;       BX = COLOR TO BE USED ( 8 REPLICATIONS OF THE    :
                     ;       2 COLOR BITS )                                   :
                     ;--------------------------------------------------------
F6AE                 S19     PROC    NEAR
F6AE 80E303                  AND     BL,3                    ; ISOLATE THE COLOR BITS
F6B1 8AC3                    MOV     AL,BL                   ; COPY TO AL
F6B3 51                      PUSH    CX                      ; SAVE REGISTER
F6B4 B90300                  MOV     CX,3                    ; NUMBER OF TIMES TO DO THIS
F6B7                 S20:
F6B7 D0E0                    SAL     AL,1
F6B9 D0E0                    SAL     AL,1                    ; LEFT SHIFT BY 2
F6BB 0AD8                    OR      BL,AL                   ; ANOTHER COLOR VERSION INTO BL
F6BD E2F8                    LOOP    S20                     ; FILL ALL OF BL
F6BF 8AFB                    MOV     BH,BL                   ; FILL UPPER PORTION
F6C1 59                      POP     CX                      ; REGISTER BACK
F6C2 C3                      RET                             ; ALL DONE
                     S19     ENDP
                     ;--------------------------------------------------------
                     ; EXPAND_BYTE                                            :
                     ;       THIS ROUTINE TAKES THE BYTE IN AL AND DOUBLES    :
                     ;       ALL OF THE BITS, TURNING THE 8 BITS INTO         :
                     ;       16 BITS. THE RESULT IS LEFT IN AX                :
                     ;--------------------------------------------------------
F6C3                 S21     PROC    NEAR
F6C3 52                      PUSH    DX                      ; SAVE REGISTERS
F6C4 51                      PUSH    CX
F6C5 53                      PUSH    BX
F6C6 2BD2                    SUB     DX,DX                   ; RESULT REGISTER
F6C8 B90100                  MOV     CX,1                    ; MASK REGISTER
F6CB                 S22:
F6CB 8BD8                    MOV     BX,AX                   ; BASE INTO TEMP
F6CD 23D9                    AND     BX,CX                   ; USE MASK TO EXTRACT A BIT
F6CF 0BD3                    OR      DX,BX                   ; PUT INTO RESULT REGISTER
F6D1 D1E0                    SHL     AX,1
F6D3 D1E1                    SHL     CX,1                    ; SHIFT BASE AND MASK BY 1
F6D5 8BD8                    MOV     BX,AX                   ; BASE TO TEMP
F6D7 23D9                    AND     BX,CX                   ; EXTRACT THE SAME BIT
F6D9 0BD3                    OR      DX,BX                   ; PUT INTO RESULT
F6DB D1E1                    SHL     CX,1                    ; SHIFT ONLY MASK NOW,
                                                             ;  MOVING TO NEXT BASE
F6DD 73EC                    JNC     S22                     ; USE MASK BIT COMING OUT TO TERMINATE
F6DF 8BC2                    MOV     AX,DX                   ; RESULT TO PARM REGISTER
F6E1 5B                      POP     BX
F6E2 59                      POP     CX                      ; RECOVER REGISTERS
F6E3 5A                      POP     DX
F6E4 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                       :
                     ;--------------------------------------------------------
```

```
LOC OBJ                 SOURCE

F6E5                    S23     PROC    NEAR
F6E5 8A24                       MOV     AH,[SI]                 ; GET FIRST BYTE
F6E7 8A4401                     MOV     AL,[SI+1]               ; GET SECOND BYTE
F6EA B900C0                     MOV     CX,0C000H               ; 2 BIT MASK TO TEST THE ENTRIES
F6ED B200                       MOV     DL,0                    ; RESULT REGISTER
F6EF                    S24:
F6EF 85C1                       TEST    AX,CX                   ; IS THIS SECTION BACKGROUND?
F6F1 F8                         CLC                             ; CLEAR CARRY IN HOPES THAT IT IS
F6F2 7401                       JZ      S25                     ; IF ZERO, IT IS BACKGROUND
F6F4 F9                         STC                             ; WASN'T, SO SET CARRY
F6F5 D0D2               S25:    RCL     DL,1                    ; MOVE THAT BIT INTO THE RESULT
F6F7 D1E9                       SHR     CX,1
F6F9 D1E9                       SHR     CX,1                    ; MOVE THE MASK TO THE RIGHT BY 2 BITS
F6FB 73F2                       JNC     S24                     ; DO IT AGAIN IF MASK DIDN'T FALL OUT
F6FD 885600                     MOV     [BP],DL                 ; STORE RESULT IN SAVE AREA
F700 45                         INC     BP                      ; ADJUST POINTER
F701 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   :
                        ;---------------------------------------------
F702                    S26     PROC    NEAR
F702 A15000                     MOV     AX,CURSOR_POSN          ; GET CURRENT CURSOR
F705                    GRAPH_POSN      LABEL   NEAR
F705 53                         PUSH    BX                      ; SAVE REGISTER
F706 8BD8                       MOV     BX,AX                   ; SAVE A COPY OF CURRENT CURSOR
F708 8AC4                       MOV     AL,AH                   ; GET ROWS TO AL
F70A F6264A00                   MUL     BYTE PTR CRT_COLS       ; MULTIPLY BY BYTES/COLUMN
F70E D1E0                       SHL     AX,1                    ; MULTIPLY * 4 SINCE 4 ROWS/BYTE
F710 D1E0                       SHL     AX,1
F712 2AFF                       SUB     BH,BH                   ; ISOLATE COLUMN VALUE
F714 03C3                       ADD     AX,BX                   ; DETERMINE OFFSET
F716 5B                         POP     BX                      ; RECOVER POINTER
F717 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 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,   :
                        ;       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
F718                    WRITE_TTY       PROC    NEAR
F718 50                         PUSH    AX                      ; SAVE REGISTERS
F719 50                         PUSH    AX                      ; SAVE CHAR TO WRITE
F71A B403                       MOV     AH,3
F71C 8A3E6200                   MOV     BH,ACTIVE_PAGE          ; GET THE CURRENT ACTIVE PAGE
F720 CD10                       INT     10H                     ; READ THE CURRENT CURSOR POSITION
F722 58                         POP     AX                      ; RECOVER CHAR
```

```
LOC OBJ                 SOURCE

                        ;----- DX NOW HAS THE CURRENT CURSOR POSITION

F723 3C08                       CMP     AL,8                    ; IS IT A BACKSPACE
F725 7452                       JE      U8                      ; BACK_SPACE
F727 3C0D                       CMP     AL,0DH                  ; IS IT CARRIAGE RETURN
F729 7457                       JE      U9                      ; CAR_RET
F72B 3C0A                       CMP     AL,0AH                  ; IS IT A LINE FEED
F72D 7457                       JE      U10                     ; LINE_FEED
F72F 3C07                       CMP     AL,07H                  ; IS IT A BELL
F731 745A                       JE      U11                     ; BELL

                        ;----- WRITE THE CHAR TO THE SCREEN


F733 B40A                       MOV     AH,10                   ; WRITE CHAR ONLY
F735 B90100                     MOV     CX,1                    ; ONLY ONE CHAR
F738 CD10                       INT     10H                     ; WRITE THE CHAR

                        ;----- POSITION THE CURSOR FOR NEXT CHAR

F73A FEC2                       INC     DL
F73C 3A164A00                   CMP     DL,BYTE PTR CRT_COLS    ; TEST FOR COLUMN OVERFLOW
F740 7533                       JNZ     U7                      ; SET_CURSOR
F742 B200                       MOV     DL,0                    ; COLUMN FOR CURSOR
F744 80FE18                     CMP     DH,24
F747 752A                       JNZ     U6                      ; SET_CURSOR_INC

                        ;----- SCROLL REQUIRED

F749                    U1:
F749 B402                       MOV     AH,2
F74B CD10                       INT     10H                     ; SET THE CURSOR

                        ;----- DETERMINE VALUE TO FILL WITH DURING SCROLL

F74D A04900                     MOV     AL,CRT_MODE             ; GET THE CURRENT MODE
F750 3C04                       CMP     AL,4
F752 7206                       JC      U2                      ; READ-CURSOR
F754 3C07                       CMP     AL,7
F756 B700                       MOV     BH,0                    ; FILL WITH BACKGROUND
F758 7506                       JNE     U3                      ; SCROLL-UP
F75A                    U2:                                     ; READ-CURSOR
F75A B408                       MOV     AH,8
F75C CD10                       INT     10H                     ; READ CHAR/ATTR AT CURRENT CURSOR
F75E 8AFC                       MOV     BH,AH                   ; STORE IN BH
F760                    U3:                                     ; SCROLL-UP
F760 B80106                     MOV     AX,601H                 ; SCROLL ONE LINE
F763 2BC9                       SUB     CX,CX                   ; UPPER LEFT CORNER
F765 B618                       MOV     DH,24                   ; LOWER RIGHT ROW
F767 8A164A00                   MOV     DL,BYTE PTR CRT_COLS    ; LOWER RIGHT COLUMN
F76B FECA                       DEC     DL
F76D                    U4:                                     ; VIDEO-CALL-RETURN
F76D CD10                       INT     10H                     ; SCROLL UP THE SCREEN
F76F                    U5:                                     ; TTY-RETURN
F76F 58                         POP     AX                      ; RESTORE THE CHARACTER
F770 E952FA                     JMP     VIDEO_RETURN            ; RETURN TO CALLER
F773                    U6:                                     ; SET-CURSOR-INC
F773 FEC6                       INC     DH                      ; NEXT ROW
F775                    U7:                                     ; SET-CURSOR
F775 B402                       MOV     AH,2
F777 EBF4                       JMP     U4                      ; ESTABLISH THE NEW CURSOR

                        ;----- BACK SPACE FOUND

F779                    U8:
F779 80FA00                     CMP     DL,0                    ; ALREADY AT END OF LINE
F77C 74F7                       JE      U7                      ; SET_CURSOR
F77E FECA                       DEC     DL                      ; NO -- JUST MOVE IT BACK
F780 EBF3                       JMP     U7                      ; SET_CURSOR

                        ;----- CARRIAGE RETURN FOUND

F782                    U9:
F782 B200                       MOV     DL,0                    ; MOVE TO FIRST COLUMN
F784 EBEF                       JMP     U7                      ; SET_CURSOR

```

```
                  ;----- LINE FEED FOUND

F786              U10:
F786 80FE18               CMP     DH,24                   ; BOTTOM OF SCREEN
F789 75E8                 JNE     U6                      ; YES, SCROLL THE SCREEN
F78B EBBC                 JMP     U1                      ; NO, JUST SET THE CURSOR

                  ;----- BELL FOUND

F78D              U11:
F78D B302                 MOV     BL,2                    ; SET UP COUNT FOR BEEP
F78F E871EE               CALL    BEEP                    ; SOUND THE POD BELL
F792 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
F794              V1      LABEL   BYTE
F794 03                   DB      3,3,5,5,3,3,3,4 ;
F795 03
F796 05
F797 05
F798 03
F799 03
F79A 03
F79B 04
F79C              READ_LPEN       PROC    NEAR

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

F79C B400                 MOV     AH,0                    ; SET NO LIGHT PEN RETURN CODE
F79E 8B166300             MOV     DX,ADDR_6845            ; GET BASE ADDRESS OF 6845
F7A2 83C206               ADD     DX,6                    ; POINT TO STATUS REGISTER
F7A5 EC                   IN      AL,DX                   ; GET STATUS REGISTER
F7A6 A804                 TEST    AL,4                    ; TEST LIGHT PEN SWITCH
F7A8 757E                 JNZ     V6                      ; NOT SET, RETURN

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

F7AA A802                 TEST    AL,2                    ; TEST LIGHT PEN TRIGGER
F7AC 7503                 JNZ     V7A                     ; RETURN WITHOUT RESETTING TRIGGER
F7AE E98100               JMP     V7

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

F7B1              V7A:
F7B1 B410                 MOV     AH,16                   ; LIGHT PEN REGISTERS ON 6845

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

F7B3 8B166300             MOV     DX,ADDR_6845            ; ADDRESS REGISTER FOR 6845
F7B7 8AC4                 MOV     AL,AH                   ; REGISTER TO READ
F7B9 EE                   OUT     DX,AL                   ; SET IT UP
F7BA 42                   INC     DX                      ; DATA REGISTER
F7BB EC                   IN      AL,DX                   ; GET THE VALUE
F7BC 8AE8                 MOV     CH,AL                   ; SAVE IN CX
F7BE 4A                   DEC     DX                      ; ADDRESS REGISTER
F7BF FEC4                 INC     AH
F7C1 8AC4                 MOV     AL,AH                   ; SECOND DATA REGISTER
F7C3 EE                   OUT     DX,AL
F7C4 42                   INC     DX                      ; POINT TO DATA REGISTER
F7C5 EC                   IN      AL,DX                   ; GET SECOND DATA VALUE
F7C6 8AE5                 MOV     AH,CH                   ; AX HAS INPUT VALUE
```

```

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

F7C8 8A1E4900                 MOV     BL,CRT_MODE
F7CC 2AFF                     SUB     BH,BH                   ; MODE VALUE TO BX
F7CE 2E8A9F94F7               MOV     BL,CS:V1[BX]            ; DETERMINE AMOUNT TO SUBTRACT
F7D3 2BC3                     SUB     AX,BX                   ; TAKE IT AWAY
F7D5 8B1E4E00                 MOV     BX,CRT_START
F7D9 D1EB                     SHR     BX,1
F7DB 2BC3                     SUB     AX,BX
F7DD 7902                     JNS     V2                      ; IF POSITIVE, DETERMINE MODE
F7DF 2BC0                     SUB     AX,AX                   ; <0 PLAYS AS 0

                      ;----- DETERMINE MODE OF OPERATION

F7E1                  V2:                                     ; DETERMINE_MODE
F7E1 B103                     MOV     CL,3                    ; SET *8 SHIFT COUNT
F7E3 803E490004               CMP     CRT_MODE,4              ; DETERMINE IF GRAPHICS OR ALPHA
F7E8 722A                     JB      V4                      ; ALPHA_PEN
F7EA 803E490007               CMP     CRT_MODE,7
F7EF 7423                     JE      V4                      ; ALPHA_PEN

                      ;----- GRAPHICS MODE

F7F1 B228                     MOV     DL,40                   ; DIVISOR FOR GRAPHICS
F7F3 F6F2                     DIV     DL                      ; DETERMINE ROW(AL) AND COLUMN(AH)
                                                              ;  AL RANGE 0-99, AH RANGE 0-39

                      ;----- DETERMINE GRAPHIC ROW POSITION

F7F5 8AE8                     MOV     CH,AL                   ; SAVE ROW VALUE IN CH
F7F7 02ED                     ADD     CH,CH                   ; *2 FOR EVEN/ODD FIELD
F7F9 8ADC                     MOV     BL,AH                   ; COLUMN VALUE TO BX
F7FB 2AFF                     SUB     BH,BH                   ; MULTIPLY BY 8 FOR MEDIUM RES
F7FD 803E490006               CMP     CRT_MODE,6              ; DETERMINE MEDIUM OR HIGH RES
F802 7504                     JNE     V3                      ; NOT_HIGH_RES
F804 B104                     MOV     CL,4                    ; SHIFT VALUE FOR HIGH RES
F806 D0E4                     SAL     AH,1                    ; COLUMN VALUE TIMES 2 FOR HIGH RES
F808                  V3:                                     ; NOT_HIGH_RES
F808 D3E3                     SHL     BX,CL                   ; MULTIPLY *16 FOR HIGH RES

                      ;----- DETERMINE ALPHA CHAR POSITION

F80A 8AD4                     MOV     DL,AH                   ; COLUMN VALUE FOR RETURN
F80C 8AF0                     MOV     DH,AL                   ; ROW VALUE
F80E D0EE                     SHR     DH,1                    ; DIVIDE BY 4
F810 D0EE                     SHR     DH,1                    ;  FOR VALUE IN 0-24 RANGE
F812 EB12                     JMP     SHORT V5                ; LIGHT_PEN_RETURN_SET

                      ;----- ALPHA MODE ON LIGHT PEN

F814                  V4:                                     ; ALPHA_PEN
F814 F6364A00                 DIV     BYTE PTR CRT_COLS       ; DETERMINE ROW,COLUMN VALUE
F818 8AF0                     MOV     DH,AL                   ; ROWS TO DH
F81A 8AD4                     MOV     DL,AH                   ; COLS TO DL
F81C D2E0                     SAL     AL,CL                   ; MULTIPLY ROWS * 8
F81E 8AE8                     MOV     CH,AL                   ; GET RASTER VALUE TO RETURN REG
F820 8ADC                     MOV     BL,AH                   ; COLUMN VALUE
F822 32FF                     XOR     BH,BH                   ;  TO BX
F824 D3E3                     SAL     BX,CL
F826                  V5:                                     ; LIGHT_PEN_RETURN_SET
F826 B401                     MOV     AH,1                    ; INDICATE EVERTHING SET
F828                  V6:                                     ; LIGHT_PEN_RETURN
F828 52                       PUSH    DX                      ; SAVE RETURN VALUE (IN CASE)
F829 8B166300                 MOV     DX,ADDR_6845            ; GET BASE ADDRESS
F82D 83C207                   ADD     DX,7                    ; POINT TO RESET PARM
F830 EE                       OUT     DX,AL                   ; ADDRESS, NOT DATA, IS IMPORTANT
F831 5A                       POP     DX                      ; RECOVER VALUE
F832                  V7:                                     ; RETURN_NO_RESET
F832 5F                       POP     DI
F833 5E                       POP     SI
F834 1F                       POP     DS                      ; DISCARD SAVED BX,CX,DX
F835 1F                       POP     DS
F836 1F                       POP     DS

F837 1F                       POP     DS
F838 07                       POP     ES
```

```
F839 CF                     IRET
                    READ_LPEN       ENDP

                    ;--- INT 12 ----------------------------------------------------------
                    ; MEMORY_SIZE_DET                                                    :
                    ;       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                        ORG     0F841H
F841                MEMORY_SIZE_DET PROC    FAR
F841 FB                     STI                             ; INTERRUPTS BACK ON
F842 1E                     PUSH    DS                      ; SAVE SEGMENT
F843 E8F806                 CALL    DDS
F846 A11300                 MOV     AX,MEMORY_SIZE          ; GET VALUE
F849 1F                     POP     DS                      ; RECOVER SEGMENT
F84A CF                     IRET                            ; RETURN TO CALLER
                    MEMORY_SIZE_DET  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
F84D                        ORG     0F84DH
F84D                EQUIPMENT       PROC    FAR
F84D FB                     STI                             ; INTERRUPTS BACK ON
F84E 1E                     PUSH    DS                      ; SAVE SEGMENT REGISTER
F84F E8EC06                 CALL    DDS
F852 A11000                 MOV     AX,EQUIP_FLAG           ; GET THE CURRENT SETTINGS
F855 1F                     POP     DS                      ; RECOVER SEGMENT
F856 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                       ORG     0F859H
F859               CASSETTE_IO     PROC    FAR
F859 FB                    STI                             ; INTERRUPTS BACK ON
F85A 1E                    PUSH    DS                      ; ESTABLISH ADDRESSING TO DATA
F85B E8E006                CALL    DDS
F85E 802671007F            AND     BIOS_BREAK, 7FH         ; MAKE SURE BREAK FLAG IS OFF
F863 E80400                CALL    W1                      ; CASSETTE_IO_CONT
F866 1F                    POP     DS
F867 CA0200                RET     2                       ; INTERRUPT RETURN
                   CASSETTE_IO     ENDP
F86A               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                 :
                   ;-----------------------------------------------------
F86A 0AE4                  OR      AH,AH                   ; TURN ON MOTOR?
F86C 7413                  JZ      MOTOR_ON                ; YES, DO IT
F86E FECC                  DEC     AH                      ; TURN OFF MOTOR?
F870 7418                  JZ      MOTOR_OFF               ; YES, DO IT
F872 FECC                  DEC     AH                      ; READ CASSETTE BLOCK?
F874 741A                  JZ      READ_BLOCK              ; YES, DO IT
F876 FECC                  DEC     AH                      ; WRITE CASSETTE BLOCK?
F878 7503                  JNZ     W2                      ;  NOT_DEFINED
F87A E92401                JMP     WRITE_BLOCK             ; YES, DO IT
F87D               W2:                                     ; COMMAND NOT DEFINED
F87D B480                  MOV     AH,080H                 ; ERROR, UNDEFINED OPERATION
F87F F9                    STC                             ; ERROR FLAG
F880 C3                    RET
                   W1      ENDP
F881               MOTOR_ON        PROC    NEAR
                   ;---------------------------------------
                   ; PURPOSE:                              :
                   ;       TO TURN ON CASSETTE MOTOR       :
                   ;---------------------------------------
F881 E461                  IN      AL,PORT_B               ; READ CASSETTE OUTPUT
F883 24F7                  AND     AL,NOT 08H              ; CLEAR BIT TO TURN ON MOTOR
F885               W3:
F885 E661                  OUT     PORT_B,AL               ; WRITE IT OUT
F887 2AE4                  SUB     AH,AH                   ; CLEAR AH
F889 C3                    RET
                   MOTOR_ON        ENDP
F88A               MOTOR_OFF       PROC    NEAR
```

```
LOC OBJ              SOURCE

                     ;----------------------------------------
                     ; PURPOSE:                               :
                     ;       TO TURN CASSETTE MOTOR OFF       :
                     ;----------------------------------------
F88A E461                    IN      AL,PORT_B               ; READ CASSETTE OUTPUT
F88C 0C08                    OR      AL,08H                  ; SET BIT TO TURN OFF
F88E EBF5                    JMP     W3                      ; WRITE IT, CLEAR ERROR, RETURN
                     MOTOR_OFF       ENDP
F890                 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                   :
                     ;-----------------------------------------------------------------
F890 53                      PUSH    BX                      ; SAVE BX
F891 51                      PUSH    CX                      ; SAVE CX
F892 56                      PUSH    SI                      ; SAVE SI
F893 BE0700                  MOV     SI, 7                   ; SET UP RETRY COUNT FOR LEADER
F896 E8BF01                  CALL    BEGIN_OP                ; BEGIN BY STARTING MOTOR
F899                 W4:                                     ; SEARCH FOR LEADER
F899 E462                    IN      AL,PORT_C               ; GET INTIAL VALUE
F89B 2410                    AND     AL,010H                 ; MASK OFF EXTRANEOUS BITS
F89D A26B00                  MOV     LAST_VAL,AL             ; SAVE IN LOC LAST_VAL
F8A0 BA7A3F                  MOV     DX,16250                ; # OF TRANSITIONS TO LOOK FOR
F8A3                 W5:                                     ; WAIT_FOR_EDGE
F8A3 F606710080              TEST    BIOS_BREAK, 80H         ; CHECK FOR BREAK KEY
F8A8 7503                    JNZ     W6A                     ; JUMP IF NO BREAK KEY
                                                             ; JUMP IF BREAK KEY HIT
F8AA                 W6:
F8AA 4A                      DEC     DX
F8AB 7503                    JNZ     W7                      ; JUMP IF BEGINNING OF LEADER
F8AD                 W6A:
F8AD E98400                  JMP     W17                     ; JUMP IF NO LEADER FOUND
F8B0                 W7:
F8B0 E8C600                  CALL    READ_HALF_BIT           ; IGNORE FIRST EDGE
F8B3 E3EE                    JCXZ    W5                      ; JUMP IF NO EDGE DETECTED
F8B5 BA7803                  MOV     DX,0378H                ; CHECK FOR HALF BITS
F8B8 B90002                  MOV     CX,200H                 ; MUST HAVE AT LEAST THIS MANY ONE SIZE
                                                             ; PULSES BEFORE CHCKNG FOR SYNC BIT (0)
F8BB E421                    IN      AL, 021H                ; INTERRUPT MASK REGISTER
F8BD 0C01                    OR      AL,1                    ; DISABLE TIMER INTERRUPTS
F8BF E621                    OUT     021H, AL
F8C1                 W8:                                     ; SEARCH-LDR
F8C1 F606710080              TEST    BIOS_BREAK, 80H         ; CHECK FOR BREAK KEY
F8C6 756C                    JNZ     W17                     ; JUMP IF BREAK KEY HIT
F8C8 51                      PUSH    CX                      ; SAVE REG CX
F8C9 E8AD00                  CALL    READ_HALF_BIT           ; GET PULSE WIDTH
F8CC 0BC9                    OR      CX, CX                  ; CHECK FOR TRANSITION
F8CE 59                      POP     CX                      ; RESTORE ONE BIT COUNTER
F8CF 74C8                    JZ      W4                      ; JUMP IF NO TRANSITION
F8D1 3BD3                    CMP     DX,BX                   ; CHECK PULSE WIDTH
F8D3 E304                    JCXZ    W9                      ; IF CX=0 THEN WE CAN LOOK
                                                             ; FOR SYNC BIT (0)
F8D5 73C2                    JNC     W4                      ; JUMP IF ZERO BIT (NOT GOOD LEADER)
F8D7 E2E8                    LOOP    W8                      ; DEC CX AND READ ANOTHER HALF ONE BIT
F8D9                 W9:                                     ; FIND-SYNC
F8D9 72E6                    JC      W8                      ; JUMP IF ONE BIT (STILL LEADER)

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

F8DB E89B00                  CALL    READ_HALF_BIT           ; SKIP OTHER HALF OF SYNC BIT (0)
F8DE E86A00                  CALL    READ_BYTE               ; READ SYN BYTE
F8E1 3C16                    CMP     AL, 16H                 ; SYNCHRONIZATION CHARACTER
F8E3 7549                    JNE     W16                     ; JUMP IF BAD LEADER FOUND.

                     ;----- GOOD CRC SO READ DATA BLOCK(S)
```

```
LOC OBJ               SOURCE

F8E5 5E                        POP    SI                      ; RESTORE REGS
F8E6 59                        POP    CX
F8E7 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             :
                      ;-------------------------------------------------------------
F8E8 51                        PUSH   CX                      ; SAVE BYTE COUNT
F8E9                  W10:                                    ; COME HERE BEFORE EACH
                                                              ; 256 BYTE BLOCK IS READ
F8E9 C7066900FFFF              MOV    CRC_REG,0FFFFH          ; INIT CRC REG
F8EF BA0001                    MOV    DX,256                  ; SET DX TO DATA BLOCK SIZE
F8F2                  W11:                                    ; RD_BLK
F8F2 F606710080                TEST   BIOS_BREAK, 80H         ; CHECK FOR BREAK KEY
F8F7 7523                      JNZ    W13                     ; JUMP IF BREAK KEY HIT
F8F9 E84F00                    CALL   READ_BYTE               ; READ BYTE FROM CASSETTE
F8FC 721E                      JC     W13                     ; CY SET INDICATES NO DATA TRANSITIONS
F8FE E305                      JCXZ   W12                     ; IF WE'VE ALREADY REACHED
                                                              ; END OF MEMORY BUFFER
                                                              ; SKIP REST OF BLOCK
F900 268807                    MOV    ES:[BX],AL              ; STORE DATA BYTE AT BYTE PTR
F903 43                        INC    BX                      ; INC BUFFER PTR
F904 49                        DEC    CX                      ; DEC BYTE COUNTER
F905                  W12:                                    ; LOOP UNTIL DATA BLOCK HAS BEEN
                                                              ;  READ FROM CASSETTE.
F905 4A                        DEC    DX                      ; DEC BLOCK CNT
F906 7FEA                      JG     W11                     ; RD_BLK
F908 E84000                    CALL   READ_BYTE               ; NOW READ TWO CRC BYTES
F90B E83D00                    CALL   READ_BYTE
F90E 2AE4                      SUB    AH,AH                   ; CLEAR AH
F910 813E69000F1D              CMP    CRC_REG,1D0FH           ; IS THE CRC CORRECT
F916 7506                      JNE    W14                     ; IF NOT EQUAL CRC IS BAD
F918 E306                      JCXZ   W15                     ; IF BYTE COUNT IS ZERO
                                                              ; THEN WE HAVE READ ENOUGH
                                                              ; SO WE WILL EXIT
F91A EBCD                      JMP    W10                     ; STILL MORE, SO READ ANOTHER BLOCK
F91C                  W13:                                    ; MISSING-DATA
                                                              ; NO DATA TRANSITIONS SO
F91C B401                      MOV    AH,01H                  ; SET AH=02 TO INDICATE
                                                              ; DATA TIMEOUT
F91E                  W14:                                    ; BAD-CRC
F91E FEC4                      INC    AH                      ; EXIT EARLY ON ERROR
                                                              ; SET AH=01 TO INDICATE CRC ERROR
F920                  W15:                                    ; RD-BLK-EX
F920 5A                        POP    DX                      ; CALCULATE COUNT OF
F921 2BD1                      SUB    DX,CX                   ; DATA BYTES ACTUALLY READ
                                                              ; RETURN COUNT IN REG DX
F923 50                        PUSH   AX                      ; SAVE AX (RET CODE)
F924 F6C490                    TEST   AH, 90H                 ; CHECK FOR ERRORS
F927 7513                      JNZ    W18                     ; JUMP IF ERROR DETECTED
F929 E81F00                    CALL   READ_BYTE               ; READ TRAILER
F92C EB0E                      JMP    SHORT W18               ; SKIP TO TURN OFF MOTOR
F92E                  W16:                                    ; BAD-LEADER
F92E 4E                        DEC    SI                      ; CHECK RETRIES
F92F 7403                      JZ     W17                     ; JUMP IF TOO MANY RETRIES
F931 E965FF                    JMP    W4                      ; JUMP IF NOT TOO MANY RETRIES
F934                  W17:                                    ; NO VALID DATA FOUND

                      ;----- NO DATA FROM CASSETTE ERROR, I.E. TIMEOUT

F934 5E                        POP    SI                      ; RESTORE REGS
F935 59                        POP    CX                      ; RESTORE REGS
F936 5B                        POP    BX
F937 2BD2                      SUB    DX,DX                   ; ZERO NUMBER OF BYTES READ
F939 B404                      MOV    AH,04H                  ; TIME OUT ERROR (NO LEADER)
F93B 50                        PUSH   AX
F93C                  W18:                                    ; MOT-OFF
```

```
F93C E421                   IN      AL, 021H                ; RE_ENABLE INTERRUPTS
F93E 24FE                   AND     AL, 0FFH- 1
F940 E621                   OUT     021H, AL
F942 E845FF                 CALL    MOTOR_OFF               ; TURN OFF MOTOR
F945 58                     POP     AX                      ; RESTORE RETURN CODE
F946 80FC01                 CMP     AH,01H                  ; SET CARRY IF ERROR (AH>0)
F949 F5                     CMC
F94A C3                     RET                             ; FINISHED
                    READ_BLOCK      ENDP
                    ;----------------------------------------
                    ; PURPOSE:                              :
                    ;       TO READ A BYTE FROM CASSETTE    :
                    ; ON EXIT                               :
                    ;       REG AL CONTAINS READ DATA BYTE  :
                    ;----------------------------------------
F94B                READ_BYTE       PROC    NEAR
F94B 53                     PUSH    BX                      ; SAVE REGS BX,CX
F94C 51                     PUSH    CX
F94D B108                   MOV     CL,8H                   ; SET BIT COUNTER FOR 8 BITS
F94F                W19:                                    ; BYTE-ASM
F94F 51                     PUSH    CX                      ; SAVE CX
                    ;--------------------------------
                    ; READ DATA BIT FROM CASSETTE   :
                    ;--------------------------------
F950 E82600                 CALL    READ_HALF_BIT           ; READ ONE PULSE
F953 E320                   JCXZ    W21                     ; IF CX=0 THEN TIMEOUT
                                                            ; BECAUSE OF NO DATA TRANSITIONS
F955 53                     PUSH    BX                      ; SAVE 1ST HALF BIT'S
                                                            ; PULSE WIDTH (IN BX)
F956 E82000                 CALL    READ_HALF_BIT           ; READ COMPLEMENTARY PULSE
F959 58                     POP     AX                      ; COMPUTE DATA BIT
F95A E319                   JCXZ    W21                     ; IF CX=0 THEN TIMEOUT DUE TO
                                                            ; NO DATA TRANSITIONS
F95C 03D8                   ADD     BX,AX                   ; PERIOD
F95E 81FBF006               CMP     BX, 06F0H               ; CHECK FOR ZERO BIT
F962 F5                     CMC                             ; CARRY IS SET IF ONE BIT
F963 9F                     LAHF                            ; SAVE CARRY IN AH
F964 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
F965 D0D5                   RCL     CH,1                    ; ROTATE REG CH LEFT WITH CARRY TO
                                                            ;   LS BIT OF REG CH
F967 9E                     SAHF                            ; RESTORE CARRY FOR CRC ROUTINE
F968 E8D900                 CALL    CRC_GEN                 ; GENERATE CRC FOR BIT
F96B FEC9                   DEC     CL                      ; LOOP TILL ALL 8 BITS OF DATA
                                                            ; ASSEMBLED IN REG CH
F96D 75E0                   JNZ     W19                     ; BYTE_ASM
F96F 8AC5                   MOV     AL,CH                   ; RETURN DATA BYTE IN REG AL
F971 F8                     CLC
F972                W20:                                    ; RD-BYT-EX
F972 59                     POP     CX                      ; RESTORE REGS CX,BX
F973 5B                     POP     BX
F974 C3                     RET                             ; FINISHED
F975                W21:                                    ; NO-DATA
F975 59                     POP     CX                      ; RESTORE CX
F976 F9                     STC                             ; INDICATE ERROR
F977 EBF9                   JMP     W20                     ; RD_BYT_EX
                    READ_BYTE       ENDP
                    ;----------------------------------------------
                    ; 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)    :
                    ;----------------------------------------------
F979                READ_HALF_BIT   PROC    NEAR
F979 B96400                 MOV     CX, 100                 ; SET TIME TO WAIT FOR BIT
F97C 8A266B00               MOV     AH,LAST_VAL             ; GET PRESENT INPUT VALUE
```

```
F980                 W22:                                       ; RD-H-BIT
F980 E462                    IN      AL,PORT_C                  ; INPUT DATA BIT
F982 2410                    AND     AL,010H                    ; MASK OFF EXTRANEOUS BITS
F984 3AC4                    CMP     AL,AH                      ; SAME AS BEFORE?
F986 E1F8                    LOOPE   W22                        ; LOOP TILL IT CHANGES
F988 A26B00                  MOV     LAST_VAL,AL                ; UPDATE LAST_VAL WITH NEW VALUE
F98B B000                    MOV     AL,0                       ; READ TIMER'S COUNTER COMMAND
F98D E643                    OUT     TIM_CTL,AL                 ; LATCH COUNTER
F98F 8B1E6700                MOV     BX,EDGE_CNT                ; BX GETS LAST EDGE COUNT
F993 E440                    IN      AL,TIMER0                  ; GET LS BYTE
F995 8AE0                    MOV     AH,AL                      ; SAVE IN AH
F997 E440                    IN      AL,TIMER0                  ; GET MS BYTE
F999 86C4                    XCHG    AL,AH                      ; XCHG AL,AH
F99B 2BD8                    SUB     BX,AX                      ; SET BX EQUAL TO HALF BIT PERIOD
F99D A36700                  MOV     EDGE_CNT,AX                ; UPDATE EDGE COUNT;
F9A0 C3                      RET
                     READ_HALF_BIT   ENDP
                     ;--------------------------------------------------------------
                     ; PURPOSE                                                     :
                     ;       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                                            :
                     ;--------------------------------------------------------------
F9A1                 WRITE_BLOCK     PROC    NEAR
F9A1 53                      PUSH    BX
F9A2 51                      PUSH    CX
F9A3 E461                    IN      AL,PORT_B                  ; DISABLE SPEAKER
F9A5 24FD                    AND     AL,NOT 02H
F9A7 0C01                    OR      AL, 01H                    ; ENABLE TIMER
F9A9 E661                    OUT     PORT_B,AL
F9AB B0B6                    MOV     AL,0B6H                    ; SET UP TIMER -- MODE 3 SQUARE WAVE
F9AD E643                    OUT     TIM_CTL,AL
F9AF E8A600                  CALL    BEGIN_OP                   ; START MOTOR AND DELAY
F9B2 B8A004                  MOV     AX,1184                    ; SET NORMAL BIT SIZE
F9B5 E88500                  CALL    W31                        ; SET_TIMER
F9B8 B90008                  MOV     CX,0800H                   ; SET CX FOR LEADER BYTE COUNT
F9BB                 W23:                                       ; WRITE LEADER
F9BB F9                      STC                                ; WRITE ONE BITS
F9BC E86800                  CALL    WRITE_BIT
F9BF E2FA                    LOOP    W23                        ; LOOP 'TIL LEADER IS WRITTEN
F9C1 F8                      CLC                                ; WRITE SYNC BIT (0)
F9C2 E86200                  CALL    WRITE_BIT
F9C5 59                      POP     CX                         ; RESTORE REGS CX,BX
F9C6 5B                      POP     BX
F9C7 B016                    MOV     AL, 16H                    ; WRITE SYN CHARACTER
F9C9 E84400                  CALL    WRITE_BYTE
                     ;--------------------------------------------------------------
                     ; PURPOSE                                                     :
                     ;       WRITE 1 OR MORE 256 BYTE BLOCKS TO CASSETTE           :
                     ; ON ENTRY:                                                   :
                     ;       BX POINTS TO MEMORY BUFFER ADDRESS                    :
                     ;       CONTAINS NUMBER OF BYTES TO WRITE                     :
                     ; ON EXIT:                                                    :
                     ;       BX POINTS 1 BYTE PAST LAST BYTE WRITTEN TO CASSETTE   :
                     ;       CX IS ZERO                                            :
                     ;--------------------------------------------------------------
F9CC                 WR_BLOCK:
F9CC C7066900FFFF            MOV     CRC_REG,0FFFFH             ; INIT CRC
F9D2 BA0001                  MOV     DX,256                     ; FOR 256 BYTES
F9D5                 W24:                                       ; WR-BLK
F9D5 268A07                  MOV     AL,ES:[BX]                 ; READ BYTE FROM MEM
F9D8 E83500                  CALL    WRITE_BYTE                 ; WRITE IT TO CASSETTE
F9DB E302                    JCXZ    W25                        ; UNLESS CX=0, ADVANCE PTRS & DEC COUNT
F9DD 43                      INC     BX                         ; INC BUFFER POINTER
F9DE 49                      DEC     CX                         ; DEC BYTE COUNTER
F9DF                 W25:                                       ; SKIP-ADV
F9DF 4A                      DEC     DX                         ; DEC BLOCK CNT
F9E0 7FF3                    JG      W24                        ; LOOP TILL 256 BYTE BLOCK
                                                                ;  IS WRITTEN TO TAPE
                     ;--------------------------------------------------------------
                     ; WRITE CRC                                                   :
                     ;       WRITE 1'S COMPLEMENT OF CRC REG TO CASSETTE           :
```

```
LOC OBJ             SOURCE

                    ;         WHICH IS CHECKED FOR CORRECTNESS WHEN THE BLOCK IS READ :
                    ; REG AX IS MODIFIED                                             :
                    ;-----------------------------------------------------------------
F9E2 A16900                 MOV     AX,CRC_REG              ; WRITE THE ONE'S COMPLEMENT OF THE
                                                            ;  TWO BYTE CRC TO TAPE
F9E5 F7D0                   NOT     AX                      ; FOR 1'S COMPLEMENT
F9E7 50                     PUSH    AX                      ; SAVE IT
F9E8 86E0                   XCHG    AH,AL                   ; WRITE MS BYTE FIRST
F9EA E82300                 CALL    WRITE_BYTE              ; WRITE IT
F9ED 58                     POP     AX                      ; GET IT BACK
F9EE E81F00                 CALL    WRITE_BYTE              ; NOW WRITE LS BYTE
F9F1 0BC9                   OR      CX,CX                   ; IS BYTE COUNT EXHAUSTED?
F9F3 75D7                   JNZ     WR_BLOCK                ; JUMP IF NOT DONE YET
F9F5 51                     PUSH    CX                      ; SAVE REG CX
F9F6 B92000                 MOV     CX, 32                  ; WRITE OUT TRAILER BITS
F9F9                W26:                                    ; TRAIL-LOOP
F9F9 F9                     STC
F9FA E82A00                 CALL    WRITE_BIT
F9FD E2FA                   LOOP    W26                     ; WRITE UNTIL TRAILER WRITTEN
F9FF 59                     POP     CX                      ; RESTORE REG CX
FA00 B0B0                   MOV     AL, 0B0H                ; TURN TIMER2 OFF
FA02 E643                   OUT     TIM_CTL, AL
FA04 B80100                 MOV     AX, 1
FA07 E83300                 CALL    W31                     ; SET_TIMER
FA0A E87DFE                 CALL    MOTOR_OFF               ; TURN MOTOR OFF
FA0D 2BC0                   SUB     AX,AX                   ; NO ERRORS REPORTED ON WRITE OP
FA0F C3                     RET                             ; FINISHED
                    WRITE_BLOCK     ENDP
                    ;--------------------------------
                    ; WRITE A BYTE TO CASSETTE.     :
                    ; BYTE TO WRITE IS IN REG AL.   :
                    ;--------------------------------
FA10                WRITE_BYTE      PROC    NEAR
FA10 51                     PUSH    CX                      ; SAVE REGS CX,AX
FA11 50                     PUSH    AX
FA12 8AE8                   MOV     CH,AL                   ; AL=BYTE TO WRITE.
                                                            ;   (MS BIT WRITTEN FIRST)
FA14 B108                   MOV     CL,8                    ; FOR 8 DATA BITS IN BYTE.
                                                            ;   NOTE: TWO EDGES PER BIT
FA16                W27:                                    ; DISASSEMBLE THE DATA BIT
FA16 D0D5                   RCL     CH,1                    ; ROTATE MS BIT INTO CARRY
FA18 9C                     PUSHF                           ; SAVE FLAGS.
                                                            ;   NOTE: DATA BIT IS IN CARRY
FA19 E80B00                 CALL    WRITE_BIT               ; WRITE DATA BIT
FA1C 9D                     POPF                            ; RESTORE CARRY FOR CRC CALC
FA1D E82400                 CALL    CRC_GEN                 ; COMPUTE CRC ON DATA BIT
FA20 FEC9                   DEC     CL                      ; LOOP TILL ALL 8 BITS DONE
FA22 75F2                   JNZ     W27                     ; JUMP IF NOT DONE YET
FA24 58                     POP     AX                      ; RESTORE REGS AX,CX
FA25 59                     POP     CX
FA26 C3                     RET                             ; WE ARE FINISHED
                    WRITE_BYTE      ENDP
                    ;------------------------------------------------------
                    ; 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                               :
                    ;------------------------------------------------------
FA27                WRITE_BIT       PROC    NEAR
                                                            ; ASSUME IT'S A '1'
FA27 B8A004                 MOV     AX,1184                 ; SET AX TO NOMINAL ONE SIZE
FA2A 7203                   JC      W28                     ; JUMP IF ONE BIT
FA2C B85002                 MOV     AX,592                  ; NO, SET TO NOMINAL ZERO SIZE
FA2F                W28:                                    ; WRITE-BIT-AX
FA2F 50                     PUSH    AX                      ; WRITE BIT WITH PERIOD EQ TO VALUE AX
FA30                W29:
FA30 E462                   IN      AL,PORT_C               ; INPUT TIMER_0 OUTPUT
FA32 2420                   AND     AL,020H
```

```
FA34 74FA                       JZ      W29                     ; LOOP TILL HIGH
FA36                    W30:
FA36 E462                       IN      AL,PORT_C               ; NOW WAIT TILL TIMER'S OUTPUT IS LOW
FA38 2420                       AND     AL,020H
FA3A 75FA                       JNZ     W30
                                                                ; RELOAD TIMER WITH PERIOD
                                                                ; FOR NEXT DATA BIT
FA3C 58                         POP     AX                      ; RESTORE PERIOD COUNT
FA3D                    W31:                                    ; SET TIMER
FA3D E642                       OUT     042H, AL                ; SET LOW BYTE OF TIMER 2
FA3F 8AC4                       MOV     AL, AH
FA41 E642                       OUT     042H, AL                ; SET HIGH BYTE OF TIMER 2
FA43 C3                         RET
                        WRITE_BIT       ENDP
                        ;----------------------------------------------
                        ;  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                          :
                        ;----------------------------------------------
FA44                    CRC_GEN         PROC    NEAR
FA44 A16900                     MOV     AX,CRC_REG
                                                                ; THE FOLLOWING INSTUCTIONS
                                                                ; WILL SET THE OVERFLOW FLAG
                                                                ; IF CARRY AND MS BIT OF CRC
                                                                ; ARE UNEQUAL
FA47 D1D8                       RCR     AX,1
FA49 D1D0                       RCL     AX,1
FA4B F8                         CLC                             ; CLEAR CARRY
FA4C 7104                       JNO     W32                     ; SKIP IF NO OVERFLOW
                                                                ; IF DATA BIT XORED WITH
                                                                ;  CRC REG BIT 15 IS ONE
FA4E 351008                     XOR     AX,0810H                ; THEN XOR CRC REG WITH 0801H
FA51 F9                         STC                             ; SET CARRY
FA52                    W32:
FA52 D1D0                       RCL     AX,1                    ; ROTATE CARRY (DATA BIT)
                                                                ; INTO CRC REG
FA54 A36900                     MOV     CRC_REG,AX              ; UPDATE CRC_REG
FA57 C3                         RET                             ; FINISHED
                        CRC_GEN         ENDP

FA58                    BEGIN_OP        PROC    NEAR            ; START TAPE AND DELAY
FA58 E826FE                     CALL    MOTOR_ON                ;TURN ON MOTOR
FA5B B342                       MOV     BL,42H                  ;DELAY FOR TAPE DRIVE
                                                                ;TO GET UP TO SPEED  (1/2 SEC)
FA5D                    W33:
FA5D B90007                     MOV     CX,700H                 ;INNER LOOP= APPROX. 10 MILLISEC
FA60 E2FE               W34:    LOOP    W34
FA62 FECB                       DEC     BL
FA64 75F7                       JNZ     W33
FA66 C3                         RET
                        BEGIN_OP        ENDP

FA67 20323031           E1      DB      ' 201',13,10
FA6B 0D
FA6C 0A

                        ;-------------------------------------------------------------------
                        ;       CHARACTER GENERATOR GRAPHICS FOR 320X200 AND 640X200 GRAPHICS
                        ;-------------------------------------------------------------------
FA6E                            ORG     0FA6EH
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 10387CFE7C381000           DB      010H,038H,07CH,0FEH,07CH,038H,010H,000H ; 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
```

```
LOC OBJ                 LINE    SOURCE

FAD6 3F333F303070F0E0   5787           DB   03FH,033H,03FH,030H,030H,070H,0F0H,0E0H ; D_0D
FADE 7F637F636367E6C0   5788           DB   07FH,063H,07FH,063H,063H,067H,0E6H,0C0H ; D_0E
FAE6 995A3CE7E73C5A99   5789           DB   099H,05AH,03CH,0E7H,0E7H,03CH,05AH,099H ; D_0F
FAEE 80E0F8FEF8E08000   5790           DB   080H,0E0H,0F8H,0FEH,0F8H,0E0H,080H,000H ; D_10
FAF6 020E3EFE3E0E0200   5791           DB   002H,00EH,03EH,0FEH,03EH,00EH,002H,000H ; D_11
FAFE 183C7E18187E3C18   5792           DB   018H,03CH,07EH,018H,018H,07EH,03CH,018H ; D_12
FB06 6666666666006600   5793           DB   066H,066H,066H,066H,066H,000H,066H,000H ; D_13
FB0E 7FDBDB7B1B1B1B00   5794           DB   07FH,0DBH,0DBH,07BH,01BH,01BH,01BH,000H ; D_14
FB16 3E63386C6C38CC78   5795           DB   03EH,063H,038H,06CH,06CH,038H,0CCH,078H ; D_15
FB1E 000000007E7E7E00   5796           DB   000H,000H,000H,000H,07EH,07EH,07EH,000H ; D_16
FB26 183C7E187E3C18FF   5797           DB   018H,03CH,07EH,018H,07EH,03CH,018H,0FFH ; D_17
FB2E 183C7E1818181800   5798           DB   018H,03CH,07EH,018H,018H,018H,018H,000H ; D_18
FB36 181818187E3C1800   5799           DB   018H,018H,018H,018H,07EH,03CH,018H,000H ; D_19
FB3E 00180CFE0C180000   5800           DB   000H,018H,00CH,0FEH,00CH,018H,000H,000H ; D_1A
FB46 003060FE60300000   5801           DB   000H,030H,060H,0FEH,060H,030H,000H,000H ; D_1B
FB4E 0000C0C0C0FE0000   5802           DB   000H,000H,0C0H,0C0H,0C0H,0FEH,000H,000H ; D_1C
FB56 002466FF66240000   5803           DB   000H,024H,066H,0FFH,066H,024H,000H,000H ; D_1D
FB5E 00183C7EFFFF0000   5804           DB   000H,018H,03CH,07EH,0FFH,0FFH,000H,000H ; D_1E
FB66 00FFFF7E3C180000   5805           DB   000H,0FFH,0FFH,07EH,03CH,018H,000H,000H ; D_1F
FB6E 0000000000000000   5806           DB   000H,000H,000H,000H,000H,000H,000H,000H ; SP D_20
FB76 3078783030003000   5807           DB   030H,078H,078H,030H,030H,000H,030H,000H ; ! D_21
FB7E 6C6C6C0000000000   5808           DB   06CH,06CH,06CH,000H,000H,000H,000H,000H ; " D_22
FB86 6C6CFE6CFE6C6C00   5809           DB   06CH,06CH,0FEH,06CH,0FEH,06CH,06CH,000H ; # D_23
FB8E 307CC0780CF83000   5810           DB   030H,07CH,0C0H,078H,00CH,0F8H,030H,000H ; $ D_24
FB96 00C6CC183066C600   5811           DB   000H,0C6H,0CCH,018H,030H,066H,0C6H,000H ; PER CENT D_25
FB9E 386C3876DCCC7600   5812           DB   038H,06CH,038H,076H,0DCH,0CCH,076H,000H ; & D_26
FBA6 6060C00000000000   5813           DB   060H,060H,0C0H,000H,000H,000H,000H,000H ; ' D_27
FBAE 1830606060301800   5814           DB   018H,030H,060H,060H,060H,030H,018H,000H ; ( D_28
FBB6 6030181818306000   5815           DB   060H,030H,018H,018H,018H,030H,060H,000H ; ) D_29
FBBE 00663CFF3C660000   5816           DB   000H,066H,03CH,0FFH,03CH,066H,000H,000H ; * D_2A
FBC6 003030FC30300000   5817           DB   000H,030H,030H,0FCH,030H,030H,000H,000H ; + D_2B
FBCE 0000000000303060   5818           DB   000H,000H,000H,000H,000H,030H,030H,060H ; , D_2C
FBD6 000000FC00000000   5819           DB   000H,000H,000H,0FCH,000H,000H,000H,000H ; - D_2D
FBDE 0000000000303000   5820           DB   000H,000H,000H,000H,000H,030H,030H,000H ; . D_2E
FBE6 060C183060C08000   5821           DB   006H,00CH,018H,030H,060H,0C0H,080H,000H ; / D_2F
FBEE 7CC6CEDEF6E67C00   5822           DB   07CH,0C6H,0CEH,0DEH,0F6H,0E6H,07CH,000H ; 0 D_30
FBF6 307030303030FC00   5823           DB   030H,070H,030H,030H,030H,030H,0FCH,000H ; 1 D_31
FBFE 78CC0C3860CCFC00   5824           DB   078H,0CCH,00CH,038H,060H,0CCH,0FCH,000H ; 2 D_32
FC06 78CC0C380CCC7800   5825           DB   078H,0CCH,00CH,038H,00CH,0CCH,078H,000H ; 3 D_33
FC0E 1C3C6CCCFE0C1E00   5826           DB   01CH,03CH,06CH,0CCH,0FEH,00CH,01EH,000H ; 4 D_34
FC16 FCC0F80C0CCC7800   5827           DB   0FCH,0C0H,0F8H,00CH,00CH,0CCH,078H,000H ; 5 D_35
FC1E 3860C0F8CCCC7800   5828           DB   038H,060H,0C0H,0F8H,0CCH,0CCH,078H,000H ; 6 D_36
FC26 FCCC0C1830303000   5829           DB   0FCH,0CCH,00CH,018H,030H,030H,030H,000H ; 7 D_37
FC2E 78CCCC78CCCC7800   5830           DB   078H,0CCH,0CCH,078H,0CCH,0CCH,078H,000H ; 8 D_38
FC36 78CCCC7C0C187000   5831           DB   078H,0CCH,0CCH,07CH,00CH,018H,070H,000H ; 9 D_39
FC3E 0030300000303000   5832           DB   000H,030H,030H,000H,000H,030H,030H,000H ; : D_3A
FC46 0030300000303060   5833           DB   000H,030H,030H,000H,000H,030H,030H,060H ; ; D_3B
FC4E 183060C060301800   5834           DB   018H,030H,060H,0C0H,060H,030H,018H,000H ; < D_3C
FC56 0000FC0000FC0000   5835           DB   000H,000H,0FCH,000H,000H,0FCH,000H,000H ; = D_3D
FC5E 6030180C18306000   5836           DB   060H,030H,018H,00CH,018H,030H,060H,000H ; > D_3E
FC66 78CC0C1830003000   5837           DB   078H,0CCH,00CH,018H,030H,000H,030H,000H ; ? D_3F
FC6E 7CC6DEDEDEC07800   5838           DB   07CH,0C6H,0DEH,0DEH,0DEH,0C0H,078H,000H ; @ D_40
FC76 3078CCCCFCCCCC00   5839           DB   030H,078H,0CCH,0CCH,0FCH,0CCH,0CCH,000H ; A D_41
FC7E FC66667C6666FC00   5840           DB   0FCH,066H,066H,07CH,066H,066H,0FCH,000H ; B D_42
FC86 3C66C0C0C0663C00   5841           DB   03CH,066H,0C0H,0C0H,0C0H,066H,03CH,000H ; C D_43
FC8E F86C6666666CF800   5842           DB   0F8H,06CH,066H,066H,066H,06CH,0F8H,000H ; D D_44
FC96 FE6268786862FE00   5843           DB   0FEH,062H,068H,078H,068H,062H,0FEH,000H ; E D_45
FC9E FE6268786860F000   5844           DB   0FEH,062H,068H,078H,068H,060H,0F0H,000H ; F D_46
FCA6 3C66C0C0CE663E00   5845           DB   03CH,066H,0C0H,0C0H,0CEH,066H,03EH,000H ; G D_47
FCAE CCCCCCFCCCCCCC00   5846           DB   0CCH,0CCH,0CCH,0FCH,0CCH,0CCH,0CCH,000H ; H D_48
FCB6 7830303030307800   5847           DB   078H,030H,030H,030H,030H,030H,078H,000H ; I D_49
FCBE 1E0C0C0CCCCC7800   5848           DB   01EH,00CH,00CH,00CH,0CCH,0CCH,078H,000H ; J D_4A
FCC6 E6666C786C66E600   5849           DB   0E6H,066H,06CH,078H,06CH,066H,0E6H,000H ; K D_4B
FCCE F06060606266FE00   5850           DB   0F0H,060H,060H,060H,062H,066H,0FEH,000H ; L D_4C
FCD6 C6EEFEFED6C6C600   5851           DB   0C6H,0EEH,0FEH,0FEH,0D6H,0C6H,0C6H,000H ; M D_4D
FCDE C6E6F6DECEC6C600   5852           DB   0C6H,0E6H,0F6H,0DEH,0CEH,0C6H,0C6H,000H ; N D_4E
FCE6 386CC6C6C66C3800   5853           DB   038H,06CH,0C6H,0C6H,0C6H,06CH,038H,000H ; O D_4F
FCEE FC66667C6060F000   5854           DB   0FCH,066H,066H,07CH,060H,060H,0F0H,000H ; P D_50
FCF6 78CCCCCCDC781C00   5855           DB   078H,0CCH,0CCH,0CCH,0DCH,078H,01CH,000H ; Q D_51
FCFE FC66667C6C66E600   5856           DB   0FCH,066H,066H,07CH,06CH,066H,0E6H,00CH ; R D_52
FD06 78CCE0701CCC7800   5857           DB   078H,0CCH,0E0H,070H,01CH,0CCH,078H,000H ; S D_53
FD0E FCB4303030307800   5858           DB   0FCH,0B4H,030H,030H,030H,030H,078H,000H ; T D_54
FD16 CCCCCCCCCCCCFC00   5859           DB   0CCH,0CCH,0CCH,0CCH,0CCH,0CCH,0FCH,000H ; U D_55
FD1E CCCCCCCCCC783000   5860           DB   0CCH,0CCH,0CCH,0CCH,0CCH,078H,030H,000H ; V D_56
FD26 C6C6C6D6FEEEC600   5861           DB   0C6H,0C6H,0C6H,0D6H,0FEH,0EEH,0C6H,000H ; W D_57
FD2E C6C66C38386CC600   5862           DB   0C6H,0C6H,06CH,038H,038H,06CH,0C6H,000H ; X D_58
FD36 CCCCCC7830307800   5863           DB   0CCH,0CCH,0CCH,078H,030H,030H,078H,000H ; Y D_59
```

```
LOC OBJ                      LINE    SOURCE

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 ; TILDE D_7E
FE66 0010386CC6C6FE00               DB      000H,010H,038H,06CH,0C6H,0C6H,0FEH,000H ; DELTA D_7F

                            ;--- 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                                ORG     0FE6EH
FE6E                        TIME_OF_DAY     PROC    FAR
FE6E FB                             STI                             ; INTERRUPTS BACK ON
FE6F 1E                             PUSH    DS                      ; SAVE SEGMENT
FE70 E8CB00                         CALL    DDS
FE73 0AE4                           OR      AH,AH                   ; AH=0
FE75 7407                           JZ      T2                      ; READ_TIME
FE77 FECC                           DEC     AH                      ; AH=1
FE79 7416                           JZ      T3                      ; SET_TIME
FE7B                        T1:                                     ; TOD_RETURN
FE7B FB                             STI                             ; INTERRUPTS BACK ON
FE7C 1F                             POP     DS                      ; RECOVER SEGMENT
FE7D CF                             IRET                            ; RETURN TO CALLER
FE7E                        T2:                                     ; READ_TIME
FE7E FA                             CLI                             ; NO TIMER INTERRUPTS WHILE READING
FE7F A07000                         MOV     AL,TIMER_OFL
FE82 C606700000                     MOV     TIMER_OFL,0             ; GET OVERFLOW, AND RESET THE FLAG
FE87 8B0E6E00                       MOV     CX,TIMER_HIGH
FE8B 8B166C00                       MOV     DX,TIMER_LOW
```

```
FE8F EBEA                   JMP     T1                      ; TOD_RETURN
FE91                T3:                                     ; SET_TIME
FE91 FA                     CLI                             ; NO INTERRUPTS WHILE WRITING
FE92 89166C00               MOV     TIMER_LOW,DX
FE96 890E6E00               MOV     TIMER_HIGH,CX           ; SET THE TIME
FE9A C606700000             MOV     TIMER_OFL,0             ; RESET OVERFLOW
FE9F 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                        ORG     0FEA5H
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 E89200                 CALL    DDS
FEAC FF066C00               INC     TIMER_LOW               ; INCREMENT TIME
FEB0 7504                   JNZ     T4                      ; TEST_DAY
FEB2 FF066E00               INC     TIMER_HIGH              ; INCREMENT HIGH WORD OF TIME
FEB6                T4:                                     ; TEST_DAY
FEB6 833E6E0018             CMP     TIMER_HIGH,018H         ; TEST FOR COUNT EQUALING 24 HOURS
FEBB 7515                   JNZ     T5                      ; DISKETTE_CTL
FEBD 813E6C00B000           CMP     TIMER_LOW,0B0H
FEC3 750D                   JNZ     T5                      ; DISKETTE_CTL

                    ;------ TIMER HAS GONE 24 HOURS

FEC5 2BC0                   SUB     AX,AX
FEC7 A36E00                 MOV     TIMER_HIGH,AX
FECA A36C00                 MOV     TIMER_LOW,AX
FECD C606700001             MOV     TIMER_OFL,1

                    ;------ TEST FOR DISKETTE TIME OUT

FED2                T5:                                     ; DISKETTE_CTL
FED2 FE0E4000               DEC     MOTOR_COUNT
FED6 750B                   JNZ     T6                      ; RETURN IF COUNT NOT OUT
FED8 80263F00F0             AND     MOTOR_STATUS,0F0H       ; TURN OFF MOTOR RUNNING BITS
FEDD B00C                   MOV     AL,0CH
FEDF BAF203                 MOV     DX,03F2H                ; FDC CTL PORT
FEE2 EE                     OUT     DX,AL                   ; TURN OFF THE MOTOR
FEE3                T6:                                     ; TIMER_RET:
FEE3 CD1C                   INT     1CH                     ; TRANSFER CONTROL TO A USER ROUTINE
FEE5 B020                   MOV     AL,EOI
FEE7 E620                   OUT     020H,AL                 ; END OF INTERRUPT TO 8259
FEE9 5A                     POP     DX
FEEA 58                     POP     AX
FEEB 1F                     POP     DS                      ; RESET MACHINE STATE
FEEC CF                     IRET                            ; RETURN FROM INTERRUPT
                    TIMER_INT       ENDP

FEED 31383031       F3B     DB      '1801',13,10
FEF1 0D
FEF2 0A

                    ;--------------------------------------------------------------
                    ;       THESE ARE THE VECTORS WHICH ARE MOVED INTO            :
                    ;       THE 8086 INTERRUPT AREA DURING POWER ON.              :
                    ;       ONLY THE OFFSETS ARE DISPLAYED HERE, CODE SEGMENT     :
                         ;       WILL BE ADDED FOR ALL OF THEM, EXCEPT WHERE NOTED       :
                         ;--------------------------------------------------------------
                                 ASSUME  CS:CODE
FEF3                             ORG     0FEF3H
FEF3                     VECTOR_TABLE    LABEL   WORD           ; VECTOR TABLE FOR MOVE TO INTERRUPTS
FEF3 A5FE                        DW      OFFSET TIMER_INT        ; INTERRUPT 8
FEF5 87E9                        DW      OFFSET  KB_INT          ; INTERRUPT 9
FEF7 DDE6                        DW      OFFSET D_EOI            ; INTERRUPT A
FEF9 DDE6                        DW      OFFSET D_EOI            ; INTERRUPT B
FEFB DDE6                        DW      OFFSET D_EOI            ; INTERRUPT C
FEFD DDE6                        DW      OFFSET D_EOI            ; INTERRUPT D
FEFF 57EF                        DW      OFFSET DISK_INT         ; INTERRUPT E
FF01 DDE6                        DW      OFFSET D_EOI            ; INTERRUPT F
FF03 65F0                        DW      OFFSET VIDEO_IO         ; INTERRUPT 10H
FF05 4DF8                        DW      OFFSET EQUIPMENT        ; INTERRUPT 11H
FF07 41F8                        DW      OFFSET MEMORY_SIZE_DET  ; INTERRUPT 12H
FF09 59EC                        DW      OFFSET DISKETTE_IO      ; INTERRUPT 13H
FF0B 39E7                        DW      OFFSET RS232_IO         ; INTERRUPT 14H
FF0D 59F8                        DW      OFFSET CASSETTE_IO      ; INTERRUPT 15H
FF0F 2EE8                        DW      OFFSET KEYBOARD_IO      ; INTERRUPT 16H
FF11 D2EF                        DW      OFFSET PRINTER_IO       ; INTERRUPT 17H

FF13 0000                        DW      00000H                  ; INTERRUPT 18H
                         ;       DW      0F600H                  ;  MUST BE INSERTED INTO TABLE LATER

FF15 F2E6                        DW      OFFSET BOOT_STRAP       ; INTERRUPT 19H
FF17 6EFE                        DW      TIME_OF_DAY             ; INTERRUPT 1AH -- TIME OF DAY
FF19 53FF                        DW      DUMMY_RETURN            ; INTERRUPT 1BH -- KEYBOARD BREAK ADDR
FF1B 53FF                        DW      DUMMY_RETURN            ; INTERRUPT 1C -- TIMER BREAK ADDR
FF1D A4F0                        DW      VIDEO_PARMS             ; INTERRUPT 1D -- VIDEO PARAMETERS
FF1F C7EF                        DW      OFFSET DISK_BASE        ; INTERRUPT 1E -- DISK PARMS
FF21 0000                        DW      0                       ; INTERRUPT 1F -- POINTER TO VIDEO EXT

FF23 50415249545920      D2      DB      'PARITY CHECK 1',13,10
     434845434B2031
FF31 0D
FF32 0A
FF33 20333031            F1      DB      ' 301',13,10
FF37 0D
FF38 0A
FF39 313331              F2      DB      '131',13,10
FF3C 0D
FF3D 0A

FF3E                     DDS     PROC    NEAR
FF3E 50                          PUSH    AX                      ; SAVE AX
FF3F B84000                      MOV     AX,DATA
FF42 8ED8                        MOV     DS,AX                   ; SET DATA SEGMENT
FF44 58                          POP     AX                      ; RESTORE AX
FF45 C3                          RET
                         DDS     ENDP

                         ;-------------------------------------------------------
                         ;       TEMPORARY INTERRUPT SERVICE ROUTINE            :
                         ;-------------------------------------------------------
FF47                             ORG     0FF47H
FF47                     D11     PROC    NEAR
FF47 B401                        MOV     AH,1
FF49 50                          PUSH    AX                      ; SAVE REG AX CONTENTS
FF4A B0FF                        MOV     AL,0FFH                 ; MASK ALL INTERRUPTS OFF
FF4C E621                        OUT     INTA01,AL
FF4E B020                        MOV     AL,EOI
FF50 E620                        OUT     INTA00,AL
FF52 58                          POP     AX                      ; RESTORE REG AX CONTENTS
FF53                     DUMMY_RETURN:                           ; NEED IRET FOR VECTOR TABLE
FF53 CF                          IRET
                         D11     ENDP

                         ;-- 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:          :
                         ;                                                                      :
```

```
LOC OBJ               SOURCE

                      ;       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                     :
                      ;               =255    ERROR ENCOUNTERED DURING PRINTING               :
                      ;---------------------------------------------------------------------
                              ASSUME  CS:CODE,DS:XXDATA
FF54                          ORG     0FF54H
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]
                      ;-------------------------------------------------------------
                      ;       THE LOOP FROM PRI10 TO THE INSTRUCTION PRIOR TO PRI20   :
                      ;       IS THE LOOP TO READ EACH CURSOR POSITION FROM THE       :
                      ;       SCREEN AND PRINT.                                       :
                      ;-------------------------------------------------------------
FF7F                  PRI10:
FF7F B402                     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                  PRI20:
FFAF 5A                       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                  ERR10:
```

```
LOC OBJ                 LINE    SOURCE

FFBB 5A                                 POP     DX                      ; GET CURSOR POSITION
FFBC B402                               MOV     AH,2                    ; TO REQUEST CURSOR SET
FFBE CD10                               INT     10H                     ; CURSOR POSITION RESTORED
FFC0                            ERR20:
FFC0 C6060000FF                         MOV     STATUS_BYTE,0FFH        ; INDICATE ERROR
FFC5                            EXIT:
FFC5 5A                                 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

FFDA 50415249545920             D1      DB      'PARITY CHECK 2',13,10
     434845434B2032
FFE8 0D
FFE9 0A
FFEA 363031                     F3      DB      '601',13,10
FFED 0D
FFEE 0A

----                            CODE    ENDS

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

                                ;----- POWER ON RESET

0000 EA5BE000F0                         JMP     RESET

0005 31302F32372F38                     DB      '10/27/82'              ; RELEASE MARKER
     32
----                            VECTOR  ENDS
                                        END
```

```
LOC OBJ                 LINE    SOURCE

$TITLE(FIXED DISK BIOS FOR IBM DISK CONTROLLER)

;-- INT 13 -----------------------------------------------------
;                                                              :
; FIXED DISK I/O INTERFACE                                     :
;                                                              :
;       THIS INTERFACE PROVIDES ACCESS TO 5 1/4" FIXED DISKS   :
;       THROUGH THE IBM FIXED DISK CONTROLLER.                 :
;                                                              :
;--------------------------------------------------------------

;--------------------------------------------------------------
;       THE  BIOS ROUTINES ARE MEANT TO BE ACCESSED THROUGH    :
;       SOFTWARE INTERRUPTS ONLY.  ANY ADDRESSES PRESENT IN    :
;       THE LISTINGS  ARE INCLUDED  ONLY FOR  COMPLETENESS,    :
;       NOT FOR  REFERENCE.   APPLICATIONS WHICH  REFERENCE    :
;       ABSOLUTE   ADDRESSES   WITHIN   THE   CODE  SEGMENT    :
;       VIOLATE THE STRUCTURE AND DESIGN OF BIOS.              :
;--------------------------------------------------------------
;
; INPUT    (AH = HEX VALUE)
;
;       (AH)=00 RESET DISK (DL = 80H,81H) / DISKETTE
;       (AH)=01 READ THE STATUS OF THE LAST DISK OPERATION INTO (AL)
;               NOTE: DL < 80H - DISKETTE
;                     DL > 80H - DISK
;       (AH)=02 READ THE DESIRED SECTORS INTO MEMORY
;       (AH)=03 WRITE THE DESIRED SECTORS FROM MEMORY
;       (AH)=04 VERIFY THE DESIRED SECTORS
;       (AH)=05 FORMAT THE DESIRED TRACK
;       (AH)=06 FORMAT THE DESIRED TRACK AND SET BAD SECTOR FLAGS
;       (AH)=07 FORMAT THE DRIVE STARTING AT THE DESIRED TRACK
;       (AH)=08 RETURN THE CURRENT DRIVE PARAMETERS
;
;       (AH)=09 INITIALIZE DRIVE PAIR CHARACTERISTICS
;               INTERRUPT 41 POINTS TO DATA BLOCK
;       (AH)=0A READ LONG
;       (AH)=0B WRITE LONG
;       NOTE: READ AND WRITE LONG ENCOMPASS 512 + 4 BYTES ECC
;       (AH)=0C SEEK
;       (AH)=0D ALTERNATE DISK RESET (SEE DL)
;       (AH)=0E READ SECTOR BUFFER
;       (AH)=0F WRITE SECTOR BUFFER,
;               (RECOMMENDED PRACTICE BEFORE FORMATTING)
;       (AH)=10 TEST DRIVE READY
;       (AH)=11 RECALIBRATE
;       (AH)=12 CONTROLLER RAM DIAGNOSTIC
;       (AH)=13 DRIVE DIAGNOSTIC
;       (AH)=14 CONTROLLER INTERNAL DIAGNOSTIC
;
;               REGISTERS USED FOR FIXED DISK OPERATIONS
;
;               (DL)    -  DRIVE NUMBER     (80H-87H FOR DISK, VALUE CHECKED)
;               (DH)    -  HEAD NUMBER      (0-7 ALLOWED, NOT VALUE CHECKED)
;               (CH)    -  CYLINDER NUMBER  (0-1023, NOT VALUE CHECKED)(SEE CL)
;               (CL)    -  SECTOR NUMBER    (1-17, NOT VALUE CHECKED)
;
;                          NOTE: HIGH 2 BITS OF CYLINDER NUMBER ARE PLACED
;                                IN THE HIGH 2 BITS OF THE CL REGISTER
;                                (10 BITS TOTAL)
;               (AL)    -  NUMBER OF SECTORS (MAXIMUM POSSIBLE RANGE 1-80H,
;                                             FOR READ/WRITE LONG 1-79H)
;                          (INTERLEAVE VALUE FOR FORMAT 1-16D)
;               (ES:BX) -  ADDRESS OF BUFFER FOR READS AND WRITES,
;                          (NOT REQUIRED FOR VERIFY)
;
; OUTPUT
;      AH = STATUS OF CURRENT OPERATION
;           STATUS BITS ARE DEFINED IN THE EQUATES BELOW
;      CY = 0  SUCCESSFUL OPERATION (AH=0 ON RETURN)
;      CY = 1  FAILED OPERATION (AH HAS ERROR REASON)
;
;      NOTE:   ERROR 11H  INDICATES THAT THE DATA READ HAD A RECOVERABLE
;              ERROR WHICH WAS CORRECTED BY THE ECC ALGORITHM.  THE DATA
;              IS PROBABLY GOOD,   HOWEVER THE BIOS ROUTINE INDICATES AN
;              ERROR TO ALLOW THE CONTROLLING PROGRAM A CHANCE TO DECIDE
;              FOR  ITSELF.  THE  ERROR  MAY  NOT  RECUR  IF THE DATA IS
```

```
                ;               REWRITTEN. (AL) CONTAINS THE BURST LENGTH.
                ;
                ;       IF DRIVE PARAMETERS WERE REQUESTED,
                ;
                ;       DL = NUMBER OF CONSECUTIVE ACKNOWLEDGING DRIVES ATTACHED (0-2)
                ;               (CONTROLLER CARD ZERO TALLY ONLY)
                ;       DH = MAXIMUM USEABLE VALUE FOR HEAD NUMBER
                ;       CH = MAXIMUM USEABLE VALUE FOR CYLINDER NUMBER
                ;       CL = MAXIMUM USEABLE VALUE FOR SECTOR NUMBER
                ;            AND CYLINDER NUMBER HIGH BITS
                ;
                ;       REGISTERS WILL BE PRESERVED EXCEPT WHEN THEY ARE USED TO RETURN
                ;       INFORMATION.
                ;
                ;       NOTE: IF AN ERROR IS REPORTED BY THE DISK CODE, THE APPROPRIATE
                ;               ACTION IS TO RESET THE DISK, THEN RETRY THE OPERATION.
                ;
                ;----------------------------------------------------------------------

 00FF           SENSE_FAIL      EQU     0FFH            ; SENSE OPERATION FAILED
 00BB           UNDEF_ERR       EQU     0BBH            ; UNDEFINED ERROR OCCURRED
 0080           TIME_OUT        EQU     80H             ; ATTACHMENT FAILED TO RESPOND
 0040           BAD_SEEK        EQU     40H             ; SEEK OPERATION FAILED
 0020           BAD_CNTLR       EQU     20H             ; CONTROLLER HAS FAILED
 0011           DATA_CORRECTED  EQU     11H             ; ECC CORRECTED DATA ERROR
 0010           BAD_ECC         EQU     10H             ; BAD ECC ON DISK READ
 000B           BAD_TRACK       EQU     0BH             ; BAD TRACK FLAG DETECTED
 0009           DMA_BOUNDARY    EQU'    09H             ; ATTEMPT TO DMA ACROSS 64K BOUNDARY
 0007           INIT_FAIL       EQU     07H             ; DRIVE PARAMETER ACTIVITY FAILED
 0005           BAD_RESET       EQU     05H             ; RESET FAILED
 0004           RECORD_NOT_FND  EQU     04H             ; REQUESTED SECTOR NOT FOUND
 0002           BAD_ADDR_MARK   EQU     02H             ; ADDRESS MARK NOT FOUND
 0001           BAD_CMD         EQU     01H             ; BAD COMMAND PASSED TO DISK I/O

                ;----------------------------------------
                ;       INTERRUPT AND STATUS AREAS      :
                ;----------------------------------------

----            DUMMY   SEGMENT AT 0
0034                    ORG     0DH*4                   ; FIXED DISK INTERRUPT VECTOR
0034            HDISK_INT       LABEL   DWORD
004C                    ORG     13H*4                   ; DISK INTERRUPT VECTOR
004C            ORG_VECTOR      LABEL   DWORD
0064                    ORG     19H*4                   ; BOOTSTRAP INTERRUPT VECTOR
0064            BOOT_VEC        LABEL   DWORD
0078                    ORG     1EH*4                   ; DISKETTE PARAMETERS
0078            DISKETTE_PARM   LABEL   DWORD
0100                    ORG     040H*4                  ; NEW DISKETTE INTERRUPT VECTOR
0100            DISK_VECTOR     LABEL   DWORD
0104                    ORG     041H*4                  ; FIXED DISK PARAMETER VECTOR
0104            HF_TBL_VEC      LABEL   DWORD
7C00                    ORG     7C00H                   ; BOOTSTRAP LOADER VECTOR
7C00            BOOT_LOCN       LABEL   FAR
----            DUMMY   ENDS

----            DATA    SEGMENT AT 40H
0042                    ORG     42H
0042            CMD_BLOCK       LABEL   BYTE
0042 (7 ??)     HD_ERROR        DB      7 DUP(?)        ; OVERLAYS DISKETTE STATUS
006C                    ORG     06CH
006C ????       TIMER_LOW       DW      ?               ; TIMER LOW WORD
0072                    ORG     72H
0072 ????       RESET_FLAG      DW      ?               ; 1234H IF KEYBOARD RESET UNDERWAY
0074                    ORG     74H
0074 ??         DISK_STATUS     DB      ?               ; FIXED DISK STATUS BYTE
0075 ??         HF_NUM          DB      ?               ; COUNT OF FIXED DISK DRIVES
0076 ??         CONTROL_BYTE    DB      ?               ; CONTROL BYTE DRIVE OPTIONS
0077 ??         PORT_OFF        DB      ?               ; PORT OFFSET
----            DATA    ENDS

----            CODE    SEGMENT

                ;----------------------------------------------------------
                ; HARDWARE SPECIFIC VALUES                                :
                ;                                                         :
                ;  -  CONTROLLER I/O PORT                                 :
                ;      > WHEN READ FROM:                                  :
                        ;       HF_PORT+0 - READ DATA (FROM CONTROLLER TO CPU)  :
                        ;       HF_PORT+1 - READ CONTROLLER HARDWARE STATUS     :
                        ;                   (CONTROLLER TO CPU)                 :
                        ;       HF_PORT+2 - READ CONFIGURATION SWITCHES         :
                        ;       HF_PORT+3 - NOT USED                            :
                        ;     > WHEN WRITTEN TO:                                :
                        ;       HF_PORT+0 - WRITE DATA (FROM CPU TO CONTROLLER) :
                        ;       HF_PORT+1 - CONTROLLER RESET                    :
                        ;       HF_PORT+2 - GENERATE CONTROLLER SELECT PULSE    :
                        ;       HF_PORT+3 - WRITE PATTERN TO DMA AND INTERRUPT  :
                        ;                   MASK REGISTER                       :
                        ;                                                       :
                        ;--------------------------------------------------------

 0320                   HF_PORT         EQU     0320H           ; DISK PORT
 0008                   R1_BUSY         EQU     00001000B       ; DISK PORT 1 BUSY BIT
 0004                   R1_BUS          EQU     00000100B       ;               COMMAND/DATA BIT
 0002                   R1_IOMODE       EQU     00000010B       ;               MODE BIT
 0001                   R1_REQ          EQU     00000001B       ;               REQUEST BIT

 0047                   DMA_READ        EQU     01000111B       ; CHANNEL 3 (047H)
 004B                   DMA_WRITE       EQU     01001011B       ; CHANNEL 3 (04BH)
 0000                   DMA             EQU     0               ; DMA ADDRESS
 0082                   DMA_HIGH        EQU     082H            ; PORT FOR HIGH 4 BITS OF DMA

 0000                   TST_RDY_CMD     EQU     00000000B       ; CNTLR READY (00H)
 0001                   RECAL_CMD       EQU     00000001B       ;       RECAL (01H)
 0003                   SENSE_CMD       EQU     00000011B       ;       SENSE (03H)
 0004                   FMTDRV_CMD      EQU     00000100B       ;       DRIVE (04H)
 0005                   CHK_TRK_CMD     EQU     00000101B       ;       T CHK (05H)
 0006                   FMTTRK_CMD      EQU     00000110B       ;       TRACK (06H)
 0007                   FMTBAD_CMD      EQU     00000111B       ;       BAD   (07H)
 0008                   READ_CMD        EQU     00001000B       ;       READ  (08H)
 000A                   WRITE_CMD       EQU     00001010B       ;       WRITE (0AH)
 000B                   SEEK_CMD        EQU     00001011B       ;       SEEK  (0BH)
 000C                   INIT_DRV_CMD    EQU     00001100B       ;       INIT  (0CH)
 000D                   RD_ECC_CMD      EQU     00001101B       ;       BURST (0DH)
 000E                   RD_BUFF_CMD     EQU     00001110B       ;       BUFFR (0EH)
 000F                   WR_BUFF_CMD     EQU     00001111B       ;       BUFFR (0FH)
 00E0                   RAM_DIAG_CMD    EQU     11100000B       ;       RAM   (E0H)
 00E3                   CHK_DRV_CMD     EQU     11100011B       ;       DRV   (E3H)
 00E4                   CNTLR_DIAG_CMD  EQU     11100100B       ;       CNTLR (E4H)
 00E5                   RD_LONG_CMD     EQU     11100101B       ;       RLONG (E5H)
 00E6                   WR_LONG_CMD     EQU     11100110B       ;       WLONG (E6H)

 0020                   INT_CTL_PORT    EQU     20H             ; 8259 CONTROL PORT
 0020                   EOI             EQU     20H             ; END OF INTERRUPT COMMAND

 0008                   MAX_FILE        EQU     8
 0002                   S_MAX_FILE      EQU     2

                                ASSUME  CS:CODE
0000                            ORG     0H
0000 55                         DB      055H                    ; GENERIC BIOS HEADER
0001 AA                         DB      0AAH
0002 10                         DB      16D

                        ;--------------------------------------------------------------
                        ; FIXED DISK I/O SETUP                                        :
                        ;                                                             :
                        ;  -  ESTABLISH TRANSFER VECTORS FOR THE FIXED DISK           :
                        ;  -  PERFORM POWER ON DIAGNOSTICS                            :
                        ;     SHOULD AN ERROR OCCUR A "1701" MESSAGE IS DISPLAYED     :
                        ;                                                             :
                        ;--------------------------------------------------------------

0003                    DISK_SETUP      PROC    FAR
0003 EB1E                       JMP     SHORT   L3
0005 35303030303539             DB      '5000059 (C)COPYRIGHT  IBM 1982'        ; COPYRIGHT NOTICE
     20284329434F50
     59524947485420
     2049424D203139
     3832
0023                     L3:
                                ASSUME  DS:DUMMY
0023 2BC0                       SUB     AX,AX                           ; ZERO
0025 8ED8                       MOV     DS,AX
```

```
LOC OBJ              LINE     SOURCE

0027 FA                               CLI
0028 A14C00                           MOV    AX,WORD PTR ORG_VECTOR              ; GET DISKETTE VECTOR
002B A30001                           MOV    WORD PTR DISK_VECTOR,AX             ;  INTO INT 40H
002E A14E00                           MOV    AX,WORD PTR ORG_VECTOR+2
0031 A30201                           MOV    WORD PTR DISK_VECTOR+2,AX
0034 C7064C005602                     MOV    WORD PTR ORG_VECTOR, OFFSET DISK_IO ; HDISK HANDLER
003A 8C0E4E00                         MOV    WORD PTR ORG_VECTOR+2,CS
003E B86007                           MOV    AX, OFFSET HD_INT                   ; HDISK INTERRUPT
0041 A33400                           MOV    WORD PTR HDISK_INT,AX
0044 8C0E3600                         MOV    WORD PTR HDISK_INT+2,CS
0048 C70664008601                     MOV    WORD PTR BOOT_VEC,OFFSET BOOT_STRAP ; BOOTSTRAP
004E 8C0E6600                         MOV    WORD PTR BOOT_VEC+2,CS
0052 C7060401E703                     MOV    WORD PTR HF_TBL_VEC,OFFSET FD_TBL   ; PARAMETER TBL
0058 8C0E0601                         MOV    WORD PTR HF_TBL_VEC+2,CS
005C FB                               STI

                                      ASSUME DS:DATA
005D B84000                           MOV    AX,DATA                  ; ESTABLISH SEGMENT
0060 8ED8                             MOV    DS,AX
0062 C606740000                       MOV    DISK_STATUS,0            ; RESET THE STATUS INDICATOR
0067 C606750000                       MOV    HF_NUM,0                 ; ZERO COUNT OF DRIVES
006C C606430000                       MOV    CMD_BLOCK+1,0            ; DRIVE ZERO, SET VALUE IN BLOCK
0071 C606770000                       MOV    PORT_OFF,0               ; ZERO CARD OFFSET

0076 B92500                           MOV    CX,25H                   ; RETRY COUNT
0079                           L4:
0079 E8F200                           CALL   HD_RESET_1               ; RESET CONTROLLER
007C 7305                             JNC    L7
007E E2F9                             LOOP   L4                       ; TRY RESET AGAIN
0080 E9BF00                           JMP    ERROR_EX
0083                           L7:
0083 B90100                           MOV    CX,1
0086 BA8000                           MOV    DX,80H

0089 B80012                           MOV    AX,1200H                 ; CONTROLLER DIAGNOSTICS
008C CD13                             INT    13H
008E 7303                             JNC    P7
0090 E9AF00                           JMP    ERROR_EX
0093                           P7:
0093 B80014                           MOV    AX,1400H                 ; CONTROLLER DIAGNOSTICS
0096 CD13                             INT    13H
0098 7303                             JNC    P9
009A E9A500                           JMP    ERROR_EX
009D                           P9:
009D C7066C000000                     MOV    TIMER_LOW,0              ; ZERO TIMER
00A3 A17200                           MOV    AX,RESET_FLAG
00A6 3D3412                           CMP    AX,1234H                 ; KEYBOARD RESET
00A9 7506                             JNE    P8
00AB C7066C009A01                     MOV    TIMER_LOW,410D           ; SKIP WAIT ON RESET
00B1                           P8:
00B1 E421                             IN     AL,021H                  ; TIMER
00B3 24FE                             AND    AL,0FEH                  ; ENABLE TIMER
00B5 E621                             OUT    021H,AL                  ; START TIMER
00B7                           P4:
00B7 E8B400                           CALL   HD_RESET_1               ; RESET CONTROLLER
00BA 7207                             JC     P10
00BC B80010                           MOV    AX,1000H                 ; READY
00BF CD13                             INT    13H
00C1 730B                             JNC    P2
00C3                           P10:
00C3 A16C00                           MOV    AX,TIMER_LOW
00C6 3DBE01                           CMP    AX,446D                  ; 25 SECONDS
00C9 72EC                             JB     P4
00CB EB7590                           JMP    ERROR_EX
00CE                           P2:
00CE B90100                           MOV    CX,1
00D1 BA8000                           MOV    DX,80H

00D4 B80011                           MOV    AX,1100H                 ; RECALIBRATE
00D7 CD13                             INT    13H
00D9 7267                             JC     ERROR_EX

00DB B80009                           MOV    AX,0900H                 ; SET DRIVE PARAMETERS
00DE CD13                             INT    13H
00E0 7260                             JC     ERROR_EX

00E2 B800C8                           MOV    AX,0C800H                ; DMA TO BUFFER
```

```
LOC OBJ                 LINE    SOURCE

00E5 8EC0                                MOV     ES,AX                   ; SET SEGMENT
00E7 2BDB                                SUB     BX,BX
00E9 B8000F                              MOV     AX,0F00H                ; WRITE SECTOR BUFFER
00EC CD13                                INT     13H
00EE 7252                                JC      ERROR_EX

00F0 FE067500                            INC     HF_NUM                  ; DRIVE ZERO RESPONDED

00F4 BA1302                              MOV     DX,213H                 ; EXPANSION BOX
00F7 B000                                MOV     AL,0
00F9 EE                                  OUT     DX,AL                   ; TURN BOX OFF
00FA BA2103                              MOV     DX,321H                 ; TEST IF CONTROLLER
00FD EC                                  IN      AL,DX                   ; ... IS IN THE SYSTEM UNIT
00FE 240F                                AND     AL,0FH
0100 3C0F                                CMP     AL,0FH
0102 7406                                JE      BOX_ON
0104 C7066C00A401                        MOV     TIMER_LOW,420D          ; CONTROLLER IS IN SYSTEM UNIT
010A                            BOX_ON:
010A BA1302                              MOV     DX,213H                 ; EXPANSION BOX
010D B0FF                                MOV     AL,0FFH
010F EE                                  OUT     DX,AL                   ; TURN BOX ON

0110 B90100                              MOV     CX,1                    ; ATTEMPT NEXT DRIVES
0113 BA8100                              MOV     DX,081H
0116                            P3:
0116 2BC0                                SUB     AX,AX                   ; RESET
0118 CD13                                INT     13H
011A 7240                                JC      POD_DONE
011C B80011                              MOV     AX,01100H               ; RECAL
011F CD13                                INT     13H
0121 730B                                JNC     P5
0123 A16C00                              MOV     AX,TIMER_LOW
0126 3DBE01                              CMP     AX,446D                 ; 25 SECONDS
0129 72EB                                JB      P3
012B EB2F90                              JMP     POD_DONE
012E                            P5:
012E B80009                              MOV     AX,0900H                ; INITIALIZE CHARACTERISTICS
0131 CD13                                INT     13H
0133 7227                                JC      POD_DONE
0135 FE067500                            INC     HF_NUM                  ; TALLY ANOTHER DRIVE
0139 81FA8100                            CMP     DX,(80H + S_MAX_FILE - 1)
013D 731D                                JAE     POD_DONE
013F 42                                  INC     DX
0140 EBD4                                JMP     P3

                                ;----- POD ERROR

0142                            ERROR_EX:
0142 BD0F00                              MOV     BP,0FH                  ; POD ERROR FLAG
0145 2BC0                                SUB     AX,AX
0147 8BF0                                MOV     SI,AX
0149 B9060090                            MOV     CX,F17L                 ; MESSAGE CHARACTER COUNT
014D B700                                MOV     BH,0                    ; PAGE ZERO
014F                            OUT_CH:
014F 2E8A846801                          MOV     AL,CS:F17[SI]           ; GET BYTE
0154 B40E                                MOV     AH,14D                  ; VIDEO OUT
0156 CD10                                INT     10H                     ; DISPLAY CHARACTER
0158 46                                  INC     SI                      ; NEXT CHAR
0159 E2F4                                LOOP    OUT_CH                  ; DO MORE
015B F9                                  STC
015C                            POD_DONE:
015C FA                                  CLI
015D E421                                IN      AL,021H                 ; BE SURE TIMER IS DISABLED
015F 0C01                                OR      AL,01H
0161 E621                                OUT     021H,AL
0163 FB                                  STI
0164 E8A500                              CALL    DSBL
0167 CB                                  RET

0168 31373031                   F17      DB      '1701',0DH,0AH
```

```
LOC OBJ                 SOURCE

016C 0D
016D 0A
 0006                   F17L     EQU      $-F17

016E                    HD_RESET_1        PROC     NEAR
016E 51                          PUSH     CX                      ; SAVE REGISTER
016F 52                          PUSH     DX
0170 F8                          CLC                              ; CLEAR CARRY
0171 B90001                      MOV      CX,0100H                ; RETRY COUNT
0174                    L6:
0174 E80706                      CALL     PORT_1
0177 EE                          OUT      DX,AL                   ; RESET CARD
0178 E80306                      CALL     PORT_1
017B EC                          IN       AL,DX                   ; CHECK STATUS
017C 2402                        AND      AL,2                    ; ERROR BIT
017E 7403                        JZ       R3
0180 E2F2                        LOOP     L6
0182 F9                          STC
0183                    R3:
0183 5A                          POP      DX                      ; RESTORE REGISTER
0184 59                          POP      CX
0185 C3                          RET
                        HD_RESET_1        ENDP

                        DISK_SETUP        ENDP

                        ;----- INT 19 -----------------------------------------------
                        ;                                                          :
                        ; INTERRUPT 19 BOOT STRAP LOADER                           :
                        ;                                                          :
                        ;  -  THE FIXED DISK BIOS REPLACES THE INTERRUPT 19        :
                        ;     BOOT STRAP VECTOR WITH A POINTER TO THIS BOOT ROUTINE :
                        ;  -  RESET THE DEFAULT DISK AND DISKETTE PARAMETER VECTORS :
                        ;  -  THE BOOT BLOCK TO BE READ IN WILL BE ATTEMPTED FROM  :
                        ;     CYLINDER 0 SECTOR 1 OF THE DEVICE.                   :
                        ;  -  THE BOOTSTRAP SEQUENCE IS:                           :
                        ;      > ATTEMPT TO LOAD FROM THE DISKETTE INTO THE BOOT   :
                        ;        LOCATION (0000:7C00) AND TRANSFER CONTROL THERE   :
                        ;      > IF THE DISKETTE FAILS THE FIXED DISK IS TRIED FOR A :
                        ;        VALID BOOTSTRAP BLOCK. A VALID BOOT BLOCK ON THE  :
                        ;        FIXED DISK CONSISTS OF THE BYTES  055H 0AAH  AS THE :
                        ;        LAST TWO BYTES OF THE BLOCK                       :
                        ;      > IF THE ABOVE FAILS CONTROL IS PASSED TO RESIDENT BASIC :
                        ;                                                          :
                        ;-----------------------------------------------------------

0186                    BOOT_STRAP:
                                 ASSUME   DS:DUMMY,ES:DUMMY
0186 2BC0                        SUB      AX,AX
0188 8ED8                        MOV      DS,AX                   ; ESTABLISH SEGMENT

                        ;----- RESET PARAMETER VECTORS

018A FA                          CLI
018B C7060401E703                MOV      WORD PTR HF_TBL_VEC, OFFSET FD_TBL
0191 8C0E0601                    MOV      WORD PTR HF_TBL_VEC+2, CS
0195 C70678000102                MOV      WORD PTR DISKETTE_PARM, OFFSET DISKETTE_TBL
019B 8C0E7A00                    MOV      WORD PTR DISKETTE_PARM+2, CS
019F FB                          STI

                        ;----- ATTEMPT BOOTSTRAP FROM DISKETTE

01A0 B90300                      MOV      CX,3                    ; SET RETRY COUNT
01A3                    H1:                                       ; IPL_SYSTEM
01A3 51                          PUSH     CX                      ; SAVE RETRY COUNT
01A4 2BD2                        SUB      DX,DX                   ; DRIVE ZERO
01A6 2BC0                        SUB      AX,AX                   ; RESET THE DISKETTE
01A8 CD13                        INT      13H                     ; FILE IO CALL
01AA 720F                        JC       H2                      ; IF ERROR, TRY AGAIN
01AC B80102                      MOV      AX,0201H                ; READ IN THE SINGLE SECTOR

01AF 2BD2                        SUB      DX,DX
01B1 8EC2                        MOV      ES,DX                   ; ESTABLISH SEGMENT
01B3 BB007C                      MOV      BX,OFFSET BOOT_LOCN

01B6 B90100                      MOV      CX,1                    ; SECTOR 1, TRACK 0
01B9 CD13                        INT      13H                     ; FILE IO CALL
```

```
LOC OBJ                 SOURCE

01BB 59                 H2:     POP     CX                      ; RECOVER RETRY COUNT
01BC 730A                       JNC     H4                      ; CF SET BY UNSUCCESSFUL READ
01BE 80FC80                     CMP     AH,80H                  ; IF TIME OUT, NO RETRY
01C1 740A                       JZ      H5                      ; TRY FIXED DISK
01C3 E2DE                       LOOP    H1                      ; DO IT FOR RETRY TIMES
01C5 EB0690                     JMP     H5                      ; UNABLE TO IPL FROM THE DISKETTE
01C8                    H4:                                     ; IPL WAS SUCCESSFUL
01C8 EA007C0000                 JMP     BOOT_LOCN

                        ;----- ATTEMPT BOOTSTRAP FROM FIXED DISK

01CD                    H5:
01CD 2BC0                       SUB     AX,AX                   ; RESET DISKETTE
01CF 2BD2                       SUB     DX,DX
01D1 CD13                       INT     13H
01D3 B90300                     MOV     CX,3                    ; SET RETRY COUNT
01D6                    H6:                                     ; IPL_SYSTEM
01D6 51                         PUSH    CX                      ; SAVE RETRY COUNT
01D7 BA8000                     MOV     DX,0080H                ; FIXED DISK ZERO
01DA 2BC0                       SUB     AX,AX                   ; RESET THE FIXED DISK
01DC CD13                       INT     13H                     ; FILE IO CALL
01DE 7212                       JC      H7                      ; IF ERROR, TRY AGAIN
01E0 B80102                     MOV     AX,0201H                ; READ IN THE SINGLE SECTOR
01E3 2BDB                       SUB     BX,BX
01E5 8EC3                       MOV     ES,BX
01E7 BB007C                     MOV     BX,OFFSET BOOT_LOCN     ; TO THE BOOT LOCATION
01EA BA8000                     MOV     DX,80H                  ; DRIVE NUMBER
01ED B90100                     MOV     CX,1                    ; SECTOR 1, TRACK 0
01F0 CD13                       INT     13H                     ; FILE IO CALL
01F2 59                 H7:     POP     CX                      ; RECOVER RETRY COUNT
01F3 7208                       JC      H8
01F5 A1FE7D                     MOV     AX,WORD PTR BOOT_LOCN+510D
01F8 3D55AA                     CMP     AX,0AA55H               ; TEST FOR GENERIC BOOT BLOCK
01FB 74CB                       JZ      H4
01FD                    H8:
01FD E2D7                       LOOP    H6                      ; DO IT FOR RETRY TIMES

                        ;------ UNABLE TO IPL FROM THE DISKETTE OR FIXED DISK

01FF CD18                       INT     18H                     ; RESIDENT BASIC

0201                    DISKETTE_TBL:

0201 CF                         DB      11001111B               ; SRT=C, HD UNLOAD=0F - 1ST SPEC BYTE
0202 02                         DB      2                       ; HD LOAD=1, MODE=DMA - 2ND SPEC BYTE
0203 25                         DB      25H                     ; WAIT AFTER OPN TIL MOTOR OFF
0204 02                         DB      2                       ; 512 BYTES PER SECTOR
0205 08                         DB      8                       ; EOT (LAST SECTOR ON TRACK)
0206 2A                         DB      02AH                    ; GAP LENGTH
0207 FF                         DB      0FFH                    ; DTL
0208 50                         DB      050H                    ; GAP LENGTH FOR FORMAT
0209 F6                         DB      0F6H                    ; FILL BYTE FOR FORMAT
020A 19                         DB      25                      ; HEAD SETTLE TIME (MILLISECONDS)
020B 04                         DB      4                       ; MOTOR START TIME (1/8 SECOND)

                        ;----- MAKE SURE THAT ALL HOUSEKEEPING IS DONE BEFORE EXIT

020C                    DSBL    PROC    NEAR
                                ASSUME  DS:DATA
020C 1E                         PUSH    DS                      ; SAVE SEGMENT
020D B84000                     MOV     AX,DATA
0210 8ED8                       MOV     DS,AX

0212 8A267700                   MOV     AH,PORT_OFF
0216 50                         PUSH    AX                      ; SAVE OFFSET

0217 C606770000                 MOV     PORT_OFF,0H
021C E86905                     CALL    PORT_3
021F 2AC0                       SUB     AL,AL
0221 EE                         OUT     DX,AL                   ; RESET INT/DMA MASK
0222 C606770004                 MOV     PORT_OFF,4H
0227 E85E05                     CALL    PORT_3
022A 2AC0                       SUB     AL,AL
022C EE                         OUT     DX,AL                   ; RESET INT/DMA MASK
022D C606770008                 MOV     PORT_OFF,8H
0232 E85305                     CALL    PORT_3
0235 2AC0                       SUB     AL,AL
```

```
LOC OBJ                 SOURCE

0237 EE                           OUT    DX,AL                     ; RESET INT/DMA MASK
0238 C60677000C                   MOV    PORT_OFF,0CH
023D E84805                       CALL   PORT_3
0240 2AC0                         SUB    AL,AL
0242 EE                           OUT    DX,AL                     ; RESET INT/DMA MASK
0243 B007                         MOV    AL,07H
0245 E60A                         OUT    DMA+10,AL                 ; SET DMA MODE TO DISABLE
0247 FA                           CLI                              ; DISABLE INTERRUPTS
0248 E421                         IN     AL,021H
024A 0C20                         OR     AL,020H
024C E621                         OUT    021H,AL                   ; DISABLE INTERRUPT 5
024E FB                           STI                              ; ENABLE INTERRUPTS
024F 58                           POP    AX                        ; RESTORE OFFSET
0250 88267700                     MOV    PORT_OFF,AH
0254 1F                           POP    DS                        ; RESTORE SEGMENT
0255 C3                           RET
                        DSBL      ENDP

                        ;----------------------------------------
                        ;         FIXED DISK BIOS ENTRY POINT     :
                        ;----------------------------------------

0256                    DISK_IO PROC     FAR
                                  ASSUME DS:NOTHING,ES:NOTHING
0256 80FA80                       CMP    DL,80H                    ; TEST FOR FIXED DISK DRIVE
0259 7305                         JAE    HARD_DISK                 ; YES, HANDLE HERE
025B CD40                         INT    40H                       ; DISKETTE HANDLER
025D                    RET_2:
025D CA0200                       RET    2                         ; BACK TO CALLER
0260                    HARD_DISK:
                                  ASSUME DS:DATA
0260 FB                           STI                              ; ENABLE INTERRUPTS
0261 0AE4                         OR     AH,AH
0263 7509                         JNZ    A3
0265 CD40                         INT    40H                       ; RESET NEC WHEN AH=0
0267 2AE4                         SUB    AH,AH
0269 80FA81                       CMP    DL,(80H + S_MAX_FILE - 1)
026C 77EF                         JA     RET_2
026E                    A3:
026E 80FC08                       CMP    AH,08                     ; GET PARAMETERS IS A SPECIAL CASE
0271 7503                         JNZ    A2
0273 E91A01                       JMP    GET_PARM_N
0276                    A2:
0276 53                           PUSH   BX                        ; SAVE REGISTERS DURING OPERATION
0277 51                           PUSH   CX
0278 52                           PUSH   DX
0279 1E                           PUSH   DS
027A 06                           PUSH   ES
027B 56                           PUSH   SI
027C 57                           PUSH   DI

027D E86A00                       CALL   DISK_IO_CONT              ; PERFORM THE OPERATION

0280 50                           PUSH   AX
0281 E888FF                       CALL   DSBL                      ; BE SURE DISABLES OCCURRED
0284 B84000                       MOV    AX,DATA
0287 8ED8                         MOV    DS,AX                     ; ESTABLISH SEGMENT
0289 58                           POP    AX
028A 8A267400                     MOV    AH,DISK_STATUS            ; GET STATUS FROM OPERATION
028E 80FC01                       CMP    AH,1                      ; SET THE CARRY FLAG TO INDICATE
0291 F5                           CMC                              ;  SUCCESS OR FAILURE
0292 5F                           POP    DI                        ; RESTORE REGISTERS
0293 5E                           POP    SI
0294 07                           POP    ES
0295 1F                           POP    DS
0296 5A                           POP    DX
0297 59                           POP    CX
0298 5B                           POP    BX
0299 CA0200                       RET    2                         ; THROW AWAY SAVED FLAGS
                        DISK_IO ENDP

029C                    M1        LABEL  WORD                      ; FUNCTION TRANSFER TABLE
029C 3803                         DW     DISK_RESET                ; 000H
029E 4D03                         DW     RETURN_STATUS             ; 001H
02A0 5603                         DW     DISK_READ                 ; 002H
02A2 6003                         DW     DISK_WRITE                ; 003H
02A4 6A03                         DW     DISK_VERF                 ; 004H
```

```
LOC OBJ                SOURCE

02A6 7203                      DW      FMT_TRK                 ; 005H
02A8 7903                      DW      FMT_BAD                 ; 006H
02AA 8003                      DW      FMT_DRV                 ; 007H
02AC 3003                      DW      BAD_COMMAND             ; 008H
02AE 2704                      DW      INIT_DRV                ; 009H
02B0 CF04                      DW      RD_LONG                 ; 00AH
02B2 DD04                      DW      WR_LONG                 ; 00BH
02B4 F204                      DW      DISK_SEEK               ; 00CH
02B6 3803                      DW      DISK_RESET              ; 00DH
02B8 F904                      DW      RD_BUFF                 ; 00EH
02BA 0705                      DW      WR_BUFF                 ; 00FH
02BC 1505                      DW      TST_RDY                 ; 010H
02BE 1C05                      DW      HDISK_RECAL             ; 011H
02C0 2305                      DW      RAM_DIAG                ; 012H
02C2 2A05                      DW      CHK_DRV                 ; 013H
02C4 3105                      DW      CNTLR_DIAG              ; 014H
  002A                 M1L     EQU     $-M1

02C6                   SETUP_A PROC    NEAR

02C6 C606740000                MOV     DISK_STATUS,0           ; RESET THE STATUS INDICATOR
02CB 51                        PUSH    CX                      ; SAVE CX

                       ;----- CALCULATE THE PORT OFFSET

02CC 8AEA                      MOV     CH,DL                   ; SAVE DL
02CE 80CA01                    OR      DL,1
02D1 FECA                      DEC     DL
02D3 D0E2                      SHL     DL,1                    ; GENERATE OFFSET
02D5 88167700                  MOV     PORT_OFF,DL             ; STORE OFFSET
02D9 8AD5                      MOV     DL,CH                   ; RESTORE DL
02DB 80E201                    AND     DL,1

02DE B105                      MOV     CL,5                    ; SHIFT COUNT
02E0 D2E2                      SHL     DL,CL                   ; DRIVE NUMBER (0,1)
02E2 0AD6                      OR      DL,DH                   ; HEAD NUMBER
02E4 88164300                  MOV     CMD_BLOCK+1,DL
02E8 59                        POP     CX
02E9 C3                        RET
                       SETUP_A ENDP

02EA                   DISK_IO_CONT    PROC    NEAR
02EA 50                        PUSH    AX
02EB B84000                    MOV     AX,DATA
02EE 8ED8                      MOV     DS,AX                   ; ESTABLISH SEGMENT
02F0 58                        POP     AX
02F1 80FC01                    CMP     AH,01H                  ; RETURN STATUS
02F4 7503                      JNZ     A4
02F6 EB5590                    JMP     RETURN_STATUS
02F9                   A4:
02F9 80EA80                    SUB     DL,80H                  ; CONVERT DRIVE NUMBER TO 0 BASED RANGE
02FC 80FA08                    CMP     DL,MAX_FILE             ; LEGAL DRIVE TEST
02FF 732F                      JAE     BAD_COMMAND

0301 E8C2FF                    CALL    SETUP_A

                       ;----- SET UP COMMAND BLOCK

0304 FEC9                      DEC     CL                      ; SECTORS 0-16 FOR CONTROLLER
0306 C606420000                MOV     CMD_BLOCK+0,0
030B 880E4400                  MOV     CMD_BLOCK+2,CL          ; SECTOR AND HIGH 2 BITS CYLINDER
030F 882E4500                  MOV     CMD_BLOCK+3,CH          ; CYLINDER
0313 A24600                    MOV     CMD_BLOCK+4,AL          ; INTERLEAVE / BLOCK COUNT
0316 A07600                    MOV     AL,CONTROL_BYTE         ; CONTROL BYTE (STEP OPTION)
0319 A24700                    MOV     CMD_BLOCK+5,AL
031C 50                        PUSH    AX                      ; SAVE AX
031D 8AC4                      MOV     AL,AH                   ; GET INTO LOW BYTE
031F 32E4                      XOR     AH,AH                   ; ZERO HIGH BYTE
0321 D1E0                      SAL     AX,1                    ; *2 FOR TABLE LOOKUP
0323 8BF0                      MOV     SI,AX                   ; PUT INTO SI FOR BRANCH
0325 3D2A00                    CMP     AX,M1L                  ; TEST WITHIN RANGE
0328 58                        POP     AX                      ; RESTORE AX
0329 7305                      JNB     BAD_COMMAND
032B 2EFFA49C02                JMP     WORD PTR CS:[SI + OFFSET M1]
0330                   BAD_COMMAND:
0330 C606740001                MOV     DISK_STATUS,BAD_CMD     ; COMMAND ERROR
0335 B000                      MOV     AL,0
```

```
LOC OBJ            LINE   SOURCE

0337 C3                           RET
                          DISK_IO_CONT    ENDP

                          ;-----------------------------------------------
                          ;        RESET THE DISK SYSTEM  (AH = 000H)      :
                          ;-----------------------------------------------

0338                      DISK_RESET      PROC    NEAR
0338 E84304                       CALL    PORT_1                  ; RESET PORT
033B EE                           OUT     DX,AL                   ; ISSUE RESET
033C E83F04                       CALL    PORT_1                  ; CONTROLLER HARDWARE STATUS
033F EC                           IN      AL,DX                   ; GET STATUS
0340 2402                         AND     AL,2                    ; ERROR BIT
0342 7406                         JZ      DR1
0344 C606740005                   MOV     DISK_STATUS,BAD_RESET
0349 C3                           RET
034A                      DR1:
034A E9DA00                       JMP     INIT_DRV                ; SET THE DRIVE PARAMETERS
                          DISK_RESET      ENDP

                          ;-----------------------------------------------
                          ;        DISK STATUS ROUTINE   (AH = 001H)       :
                          ;-----------------------------------------------

034D                      RETURN_STATUS   PROC    NEAR
034D A07400                       MOV     AL,DISK_STATUS          ; OBTAIN PREVIOUS STATUS
0350 C606740000                   MOV     DISK_STATUS,0           ; RESET STATUS
0355 C3                           RET
                          RETURN_STATUS   ENDP

                          ;-----------------------------------------------
                          ;        DISK READ ROUTINE   (AH = 002H)         :
                          ;-----------------------------------------------

0356                      DISK_READ       PROC    NEAR
0356 B047                         MOV     AL,DMA_READ             ; MODE BYTE FOR DMA READ
0358 C606420008                   MOV     CMD_BLOCK+0,READ_CMD
035D E9E501                       JMP     DMA_OPN
                          DISK_READ       ENDP

                          ;-----------------------------------------------
                          ;        DISK WRITE ROUTINE   (AH = 003H)        :
                          ;-----------------------------------------------

0360                      DISK_WRITE      PROC    NEAR
0360 B04B                         MOV     AL,DMA_WRITE            ; MODE BYTE FOR DMA WRITE
0362 C60642000A                   MOV     CMD_BLOCK+0,WRITE_CMD
0367 E9DB01                       JMP     DMA_OPN
                          DISK_WRITE      ENDP

                          ;-----------------------------------------------
                          ;        DISK VERIFY   (AH = 004H)               :
                          ;-----------------------------------------------

036A                      DISK_VERF       PROC    NEAR
036A C606420005                   MOV     CMD_BLOCK+0,CHK_TRK_CMD
036F E9C401                       JMP     NDMA_OPN
                          DISK_VERF       ENDP

                          ;-----------------------------------------------
                          ;        FORMATTING   (AH = 005H 006H 007H)      :
                          ;-----------------------------------------------

0372                      FMT_TRK PROC    NEAR                    ; FORMAT TRACK  (AH = 005H)
0372 C606420006                   MOV     CMD_BLOCK,FMTTRK_CMD
0377 EB0C                         JMP     SHORT   FMT_CONT
                          FMT_TRK ENDP

0379                      FMT_BAD PROC    NEAR                    ; FORMAT BAD TRACK  (AH = 006H)
0379 C606420007                   MOV     CMD_BLOCK,FMTBAD_CMD
037E EB05                         JMP     SHORT   FMT_CONT
                          FMT_BAD ENDP

0380                      FMT_DRV PROC    NEAR                    ; FORMAT DRIVE  (AH = 007H)
0380 C606420004                   MOV     CMD_BLOCK,FMTDRV_CMD
                          FMT_DRV ENDP
```

```
LOC OBJ                  SOURCE

0385                     FMT_CONT:
0385 A04400                      MOV     AL,CMD_BLOCK+2          ; ZERO OUT SECTOR FIELD
0388 24C0                        AND     AL,11000000B
038A A24400                      MOV     CMD_BLOCK+2,AL
038D E9A601                      JMP     NODMA_OPN

                         ;----------------------------------------------
                         ;       GET PARAMETERS   (AH = 8)              :
                         ;----------------------------------------------

0390                     GET_PARM_N      LABEL   NEAR
0390                     GET_PARM        PROC    FAR             ; GET DRIVE PARAMETERS
0390 1E                          PUSH    DS                      ; SAVE REGISTERS
0391 06                          PUSH    ES
0392 53                          PUSH    BX

                                 ASSUME  DS:DUMMY
0393 2BC0                        SUB     AX,AX                   ; ESTABLISH ADDRESSING
0395 8ED8                        MOV     DS,AX
0397 C41E0401                    LES     BX,HF_TBL_VEC
                                 ASSUME  DS:DATA
039B B84000                      MOV     AX,DATA
039E 8ED8                        MOV     DS,AX                   ; ESTABLISH SEGMENT

03A0 80EA80                      SUB     DL,80H
03A3 80FA08                      CMP     DL,MAX_FILE             ; TEST WITHIN RANGE
03A6 732F                        JAE     G4

03A8 E81BFF                      CALL    SETUP_A

03AB E8DF03                      CALL    SW2_OFFS
03AE 7227                        JC      G4
03B0 03D8                        ADD     BX,AX

03B2 268B07                      MOV     AX,ES:[BX]              ; MAX NUMBER OF CYLINDERS
03B5 2D0200                      SUB     AX,2                    ; ADJUST FOR 0-N
                                                                 ; AND RESERVE LAST TRACK
03B8 8AE8                        MOV     CH,AL
03BA 250003                      AND     AX,0300H                ; HIGH TWO BITS OF CYL
03BD D1E8                        SHR     AX,1
03BF D1E8                        SHR     AX,1
03C1 0C11                        OR      AL,011H                 ; SECTORS
03C3 8AC8                        MOV     CL,AL

03C5 268A7702                    MOV     DH,ES:[BX][2]           ; HEADS
03C9 FECE                        DEC     DH                      ; 0-N RANGE
03CB 8A167500                    MOV     DL,HF_NUM               ; DRIVE COUNT
03CF 2BC0                        SUB     AX,AX
03D1                     G5:
03D1 5B                          POP     BX                      ; RESTORE REGISTERS
03D2 07                          POP     ES
03D3 1F                          POP     DS
03D4 CA0200                      RET     2
03D7                     G4:
03D7 C606740007                  MOV     DISK_STATUS,INIT_FAIL   ; OPERATION FAILED
03DC B407                        MOV     AH,INIT_FAIL
03DE 2AC0                        SUB     AL,AL
03E0 2BD2                        SUB     DX,DX
03E2 2BC9                        SUB     CX,CX
03E4 F9                          STC                             ; SET ERROR FLAG
03E5 EBEA                        JMP     G5
                         GET_PARM        ENDP

                         ;-------------------------------------------------------
                         ; INITIALIZE DRIVE CHARACTERISTICS                      :
                         ;                                                       :
                         ; FIXED DISK PARAMETER TABLE                            :
                         ;                                                       :
                         ;  -  THE TABLE IS COMPOSED OF A BLOCK DEFINED AS:      :
                         ;                                                       :
                         ;       (1 WORD) - MAXIMUM NUMBER OF CYLINDERS          :
                         ;       (1 BYTE) - MAXIMUM NUMBER OF HEADS              :
                         ;       (1 WORD) - STARTING REDUCED WRITE CURRENT CYL   :
                         ;       (1 WORD) - STARTING WRITE PRECOMPENSATION CYL   :
                         ;       (1 BYTE) - MAXIMUM ECC DATA BURST LENGTH        :
                         ;       (1 BYTE) - CONTROL BYTE (DRIVE STEP OPTION)     :
                         ;                  BIT    7 DISABLE DISK-ACCESS RETRIES :
                         ;                  BIT    6 DISABLE ECC RETRIES         :
```

```
                   ;                BITS 5-3 ZERO                         :
                   ;                BITS 2-0 DRIVE OPTION                 :
                   ;        (1 BYTE) - STANDARD TIME OUT VALUE (SEE BELOW) :
                   ;        (1 BYTE) - TIME OUT VALUE FOR FORMAT DRIVE    :
                   ;        (1 BYTE) - TIME OUT VALUE FOR CHECK DRIVE     :
                   ;        (4 BYTES)                                     :
                   ;                 - RESERVED FOR FUTURE USE            :
                   ;                                                      :
                   ;         - TO DYNAMICALLY DEFINE A SET OF PARAMETERS  :
                   ;           BUILD A TABLE OF VALUES AND PLACE THE      :
                   ;           CORRESPONDING VECTOR INTO INTERRUPT 41.    :
                   ;                                                      :
                   ;        NOTE:                                         :
                   ;                THE DEFAULT TABLE IS VECTORED IN FOR  :
                   ;                AN INTERRUPT 19H (BOOTSTRAP)          :
                   ;                                                      :
                   ;                                                      :
                   ; ON THE CARD SWITCH SETTINGS                          :
                   ;                                                      :
                   ;                 DRIVE 0    DRIVE 1                   :
                   ;               -----------------------                :
                   ;          ON    :           /         :               :
                   ;                : -1-  -2- / -3-  -4- :               :
                   ;          OFF   :           /         :               :
                   ;               -----------------------                :
                   ;                                                      :
                   ;                                                      :
                   ;        TRANSLATION TABLE                             :
                   ;                                                      :
                   ;        1/3  :  2/4  :  TABLE ENTRY                   :
                   ;        ---------------------------                   :
                   ;         ON  :   ON  :       0                        :
                   ;         ON  :  OFF  :       1                        :
                   ;        OFF  :   ON  :       2                        :
                   ;        OFF  :  OFF  :       3                        :
                   ;                                                      :
                   ;-------------------------------------------------------

03E7               FD_TBL:

                   ;----- DRIVE TYPE 00

03E7 3201                  DW      0306D
03E9 02                    DB      02D
03EA 3201                  DW      0306D
03EC 0000                  DW      0000D
03EE 0B                    DB      0BH
03EF 00                    DB      00H
03F0 0C                    DB      0CH                     ; STANDARD
03F1 B4                    DB      0B4H                    ; FORMAT DRIVE
03F2 28                    DB      028H                    ; CHECK DRIVE
03F3 00000000              DB      0,0,0,0

                   ;----- DRIVE TYPE 01

03F7 7701                  DW      0375D
03F9 08                    DB      08D
03FA 7701                  DW      0375D
03FC 0000                  DW      0000D
03FE 0B                    DB      0BH
03FF 05                    DB      05H
0400 0C                    DB      0CH                     ; STANDARD
0401 B4                    DB      0B4H                    ; FORMAT DRIVE
0402 28                    DB      028H                    ; CHECK DRIVE
0403 00000000              DB      0,0,0,0

                   ;----- DRIVE TYPE 02

0407 3201                  DW      0306D
0409 06                    DB      06D
040A 8000                  DW      0128D
040C 0001                  DW      0256D
040E 0B                    DB      0BH
040F 05                    DB      05H
0410 0C                    DB      0CH                     ; STANDARD
0411 B4                    DB      0B4H                    ; FORMAT DRIVE
```

```
0412 28                              DB      028H                    ; CHECK DRIVE
0413 00000000                        DB      0,0,0,0

                         ;----- DRIVE TYPE 03

0417 3201                            DW      0306D
0419 04                              DB      04D
041A 3201                            DW      0306D
041C 0000                            DW      0000D
041E 0B                              DB      0BH
041F 05                              DB      05H
0420 0C                              DB      0CH                     ; STANDARD
0421 B4                              DB      0B4H                    ; FORMAT DRIVE
0422 28                              DB      028H                    ; CHECK DRIVE
0423 00000000                        DB      0,0,0,0

0427                     INIT_DRV            PROC    NEAR

                         ;----- DO DRIVE ZERO

0427 C60642000C                      MOV     CMD_BLOCK+0,INIT_DRV_CMD
042C C606430000                      MOV     CMD_BLOCK+1,0
0431 E81000                          CALL    INIT_DRV_R
0434 720D                            JC      INIT_DRV_OUT

                         ;----- DO DRIVE ONE

0436 C60642000C                      MOV     CMD_BLOCK+0,INIT_DRV_CMD
043B C606430020                      MOV     CMD_BLOCK+1,00100000B
0440 E80100                          CALL    INIT_DRV_R
0443                     INIT_DRV_OUT:
0443 C3                              RET
                         INIT_DRV            ENDP

0444                     INIT_DRV_R          PROC    NEAR
                                     ASSUME  ES:CODE
0444 2AC0                            SUB     AL,AL
0446 E81901                          CALL    COMMAND                 ; ISSUE THE COMMAND
0449 7301                            JNC     B1
044B C3                              RET
044C                     B1:
044C 1E                              PUSH    DS                      ; SAVE SEGMENT
                                     ASSUME  DS:DUMMY
044D 2BC0                            SUB     AX,AX
044F 8ED8                            MOV     DS,AX                   ; ESTABLISH SEGMENT
0451 C41E0401                        LES     BX,HF_TBL_VEC
0455 1F                              POP     DS                      ; RESTORE SEGMENT
                                     ASSUME  DS:DATA
0456 E83403                          CALL    SW2_OFFS
0459 7257                            JC      B3
045B 03D8                            ADD     BX,AX

                         ;----- SEND DRIVE PARAMETERS MOST SIGNIFICANT BYTE FIRST

045D BF0100                          MOV     DI,1
0460 E85F00                          CALL    INIT_DRV_S
0463 724D                            JC      B3

0465 BF0000                          MOV     DI,0
0468 E85700                          CALL    INIT_DRV_S
046B 7245                            JC      B3

046D BF0200                          MOV     DI,2
0470 E84F00                          CALL    INIT_DRV_S
0473 723D                            JC      B3

0475 BF0400                          MOV     DI,4
0478 E84700                          CALL    INIT_DRV_S
047B 7235                            JC      B3

047D BF0300                          MOV     DI,3
0480 E83F00                          CALL    INIT_DRV_S
0483 722D                            JC      B3

0485 BF0600                          MOV     DI,6
0488 E83700                          CALL    INIT_DRV_S
048B 7225                            JC      B3
```

```
LOC OBJ               SOURCE

0480 BF0500                    MOV     DI,5
0490 E82F00                    CALL    INIT_DRV_S
0493 721D                      JC      B3

0495 BF0700                    MOV     DI,7
0498 E82700                    CALL    INIT_DRV_S
049B 7215                      JC      B3

049D BF0800                    MOV     DI,8                    ; DRIVE STEP OPTION
04A0 268A01                    MOV     AL,ES:[BX + DI]
04A3 A27600                    MOV     CONTROL_BYTE,AL

04A6 2BC9                      SUB     CX,CX
04A8                  B5:
04A8 E80302                    CALL    PORT_1
04AB EC                        IN      AL,DX
04AC A802                      TEST    AL,R1_IOMODE            ; STATUS INPUT MODE
04AE 7509                      JNZ     B6
04B0 E2F6                      LOOP    B5
04B2                  B3:
04B2 C606740007                MOV     DISK_STATUS,INIT_FAIL   ; OPERATION FAILED
04B7 F9                        STC
04B8 C3                        RET

04B9                  B6:
04B9 E8B502                    CALL    PORT_0
04BC EC                        IN      AL,DX
04BD 2402                      AND     AL,2                    ; MASK ERROR BIT
04BF 75F1                      JNZ     B3
04C1 C3                        RET
                               ASSUME  ES:NOTHING
                      INIT_DRV_R       ENDP

                      ;----- SEND THE BYTE OUT TO THE CONTROLLER

04C2                  INIT_DRV_S       PROC    NEAR
04C2 E8C501                    CALL    HD_WAIT_REQ
04C5 7207                      JC      D1
04C7 E8A702                    CALL    PORT_0
04CA 268A01                    MOV     AL,ES:[BX + DI]
04CD EE                        OUT     DX,AL
04CE                  D1:
04CE C3                        RET
                      INIT_DRV_S       ENDP

                      ;----------------------------------------
                      ;        READ LONG  (AH = 0AH)          :
                      ;----------------------------------------

04CF                  RD_LONG          PROC    NEAR
04CF E81900                    CALL    CHK_LONG
04D2 726B                      JC      G8
04D4 C6064200E5                MOV     CMD_BLOCK+0,RD_LONG_CMD
04D9 B047                      MOV     AL,DMA_READ
04DB EB68                      JMP     SHORT   DMA_OPN
                      RD_LONG          ENDP

                      ;----------------------------------------
                      ;        WRITE LONG  (AH = 0BH)         :
                      ;----------------------------------------

04DD                  WR_LONG          PROC    NEAR
04DD E80B00                    CALL    CHK_LONG
04E0 725D                      JC      G8
04E2 C6064200E6                MOV     CMD_BLOCK+0,WR_LONG_CMD
04E7 B04B                      MOV     AL,DMA_WRITE
04E9 EB5A                      JMP     SHORT   DMA_OPN
                      WR_LONG          ENDP

04EB                  CHK_LONG         PROC    NEAR
04EB A04600                    MOV     AL,CMD_BLOCK+4
04EE 3C80                      CMP     AL,080H
04F0 F5                        CMC
04F1 C3                        RET
                      CHK_LONG         ENDP

                      ;----------------------------------------
                      ;       SEEK    (AH = 0CH)              :
                      ;----------------------------------------

04F2                  DISK_SEEK       PROC    NEAR
04F2 C60642000B               MOV     CMD_BLOCK,SEEK_CMD
04F7 EB30                     JMP     SHORT   NDMA_OPN
                      DISK_SEEK       ENDP

                      ;---------------------------------------------------
                      ;       READ SECTOR BUFFER    (AH = 0EH)          :
                      ;---------------------------------------------------

04F9                  RD_BUFF PROC    NEAR
04F9 C60642000E               MOV     CMD_BLOCK+0,RD_BUFF_CMD
04FE C606460001               MOV     CMD_BLOCK+4,1           ; ONLY ONE BLOCK
0503 B047                     MOV     AL,DMA_READ
0505 EB3E                     JMP     SHORT   DMA_OPN
                      RD_BUFF ENDP

                      ;---------------------------------------------------
                      ;       WRITE SECTOR BUFFER    (AH = 0FH)         :
                      ;---------------------------------------------------

0507                  WR_BUFF PROC    NEAR
0507 C60642000F               MOV     CMD_BLOCK+0,WR_BUFF_CMD
050C C606460001               MOV     CMD_BLOCK+4,1           ; ONLY ONE BLOCK
0511 B04B                     MOV     AL,DMA_WRITE
0513 EB30                     JMP     SHORT   DMA_OPN
                      WR_BUFF ENDP

                      ;---------------------------------------------------
                      ;       TEST DISK READY    (AH = 010H)            :
                      ;---------------------------------------------------

0515                  TST_RDY PROC    NEAR
0515 C606420000               MOV     CMD_BLOCK+0,TST_RDY_CMD
051A EB1A                     JMP     SHORT   NDMA_OPN
                      TST_RDY ENDP

                      ;---------------------------------------------------
                      ;       RECALIBRATE    (AH = 011H)                :
                      ;---------------------------------------------------

051C                  HDISK_RECAL     PROC    NEAR
051C C606420001               MOV     CMD_BLOCK,RECAL_CMD
0521 EB13                     JMP     SHORT   NDMA_OPN
                      HDISK_RECAL     ENDP

                      ;------------------------------------------------------------
                      ;       CONTROLLER RAM DIAGNOSTICS    (AH = 012H)          :
                      ;------------------------------------------------------------

0523                  RAM_DIAG        PROC    NEAR
0523 C6064200E0               MOV     CMD_BLOCK+0,RAM_DIAG_CMD
0528 EB0C                     JMP     SHORT   NDMA_OPN
                      RAM_DIAG        ENDP

                      ;---------------------------------------------------
                      ;       DRIVE DIAGNOSTICS    (AH = 013H)          :
                      ;---------------------------------------------------

052A                  CHK_DRV PROC    NEAR
052A C6064200E3               MOV     CMD_BLOCK+0,CHK_DRV_CMD
052F EB05                     JMP     SHORT   NDMA_OPN
                      CHK_DRV ENDP

                      ;----------------------------------------------------------------
                      ;      CONTROLLER INTERNAL DIAGNOSTICS    (AH = 014H)          :
                      ;----------------------------------------------------------------

0531                  CNTLR_DIAG      PROC    NEAR
0531 C6064200E4               MOV     CMD_BLOCK+0,CNTLR_DIAG_CMD
                      CNTLR_DIAG      ENDP

```

```
LOC OBJ          LINE    SOURCE

                         ;-------------------------------------------------------
                         ;                  SUPPORT ROUTINES                     :
                         ;-------------------------------------------------------

0536                     NDMA_OPN:
0536 B002                        MOV     AL,02H
0538 E82700                      CALL    COMMAND                 ; ISSUE THE COMMAND
053B 7221                        JC      G11
053D EB16                        JMP     SHORT   G3
053F                     G8:
053F C606740009                  MOV     DISK_STATUS,DMA_BOUNDARY
0544 C3                          RET
0545                     DMA_OPN:
0545 E85701                      CALL    DMA_SETUP               ; SET UP FOR DMA OPERATION
0548 72F5                        JC      G8
054A B003                        MOV     AL,03H
054C E81300                      CALL    COMMAND                 ; ISSUE THE COMMAND
054F 720D                        JC      G11
0551 B003                        MOV     AL,03H
0553 E60A                        OUT     DMA+10,AL               ; INITIALIZE THE DISK CHANNEL
0555                     G3:
0555 E421                        IN      AL,021H
0557 24DF                        AND     AL,0DFH
0559 E621                        OUT     021H,AL
055B E8AA01                      CALL    WAIT_INT
055E                     G11:
055E E83B00                      CALL    ERROR_CHK
0561 C3                          RET

                         ;-------------------------------------------------------
                         ; COMMAND                                               :
                         ;       THIS ROUTINE OUTPUTS THE COMMAND BLOCK          :
                         ; INPUT                                                 :
                         ;       AL = CONTROLLER DMA/INTERRUPT REGISTER MASK     :
                         ;                                                       :
                         ;-------------------------------------------------------

0562                     COMMAND PROC    NEAR
0562 BE4200                      MOV     SI,OFFSET CMD_BLOCK
0565 E81B02                      CALL    PORT_2
0568 EE                          OUT     DX,AL                   ; CONTROLLER SELECT PULSE
0569 E81C02                      CALL    PORT_3
056C EE                          OUT     DX,AL
056D 2BC9                        SUB     CX,CX                   ; WAIT COUNT
056F E80C02                      CALL    PORT_1
0572                     WAIT_BUSY:
0572 EC                          IN      AL,DX                   ; GET STATUS
0573 240F                        AND     AL,0FH
0575 3C0D                        CMP     AL,R1_BUSY OR R1_BUS OR R1_REQ
0577 7409                        JE      C1
0579 E2F7                        LOOP    WAIT_BUSY
057B C606740080                  MOV     DISK_STATUS,TIME_OUT
0580 F9                          STC
0581 C3                          RET                             ; ERROR RETURN
0582                     C1:
0582 FC                          CLD
0583 B90600                      MOV     CX,6                    ; BYTE COUNT
0586                     CM3:
0586 E8E801                      CALL    PORT_0
0589 AC                          LODSB                           ; GET THE NEXT COMMAND BYTE
058A EE                          OUT     DX,AL                   ; OUT IT GOES
058B E2F9                        LOOP    CM3                     ; DO MORE

058D E8EE01                      CALL    PORT_1                  ; STATUS
0590 EC                          IN      AL,DX
0591 A801                        TEST    AL,R1_REQ
0593 7406                        JZ      CM7
0595 C606740020                  MOV     DISK_STATUS,BAD_CNTLR
059A F9                          STC
059B                     CM7:
059B C3                          RET
                         COMMAND ENDP

                         ;---------------------------------------------
                         ;               SENSE STATUS BYTES            :
                         ;                                             :
                         ; BYTE 0                                      :
```

```
LOC OBJ               SOURCE

                      ;       BIT    7   ADDRESS VALID, WHEN SET      :
                      ;       BIT    6   SPARE, SET TO ZERO           :
                      ;       BITS 5-4   ERROR TYPE                   :
                      ;       BITS 3-0   ERROR CODE                   :
                      ;                                               :
                      ; BYTE 1                                        :
                      ;       BITS 7-6   ZERO                         :
                      ;       BIT    5   DRIVE (0-1)                  :
                      ;       BITS 4-0   HEAD NUMBER                  :
                      ;                                               :
                      ; BYTE 2                                        :
                      ;       BITS 7-5   CYLINDER HIGH                :
                      ;       BITS 4-0   SECTOR NUMBER                :
                      ;                                               :
                      ; BYTE 3                                        :
                      ;       BITS 7-0   CYLINDER LOW                 :
                      ;                                               :
                      ;------------------------------------------------

059C                  ERROR_CHK       PROC    NEAR
                              ASSUME  ES:DATA
059C A07400                   MOV     AL,DISK_STATUS          ; CHECK IF THERE WAS AN ERROR
059F 0AC0                     OR      AL,AL
05A1 7501                     JNZ     G21
05A3 C3                       RET

                      ;----- PERFORM SENSE STATUS

05A4                  G21:
05A4 B84000                   MOV     AX,DATA
05A7 8EC0                     MOV     ES,AX                   ; ESTABLISH SEGMENT
05A9 2BC0                     SUB     AX,AX
05AB 8BF8                     MOV     DI,AX
05AD C606420003               MOV     CMD_BLOCK+0,SENSE_CMD
05B2 2AC0                     SUB     AL,AL
05B4 E8ABFF                   CALL    COMMAND                 ; ISSUE SENSE STATUS COMMAND
05B7 7223                     JC      SENSE_ABORT             ; CANNOT RECOVER
05B9 B90400                   MOV     CX,4
05BC                  G22:
05BC E8CB00                   CALL    HD_WAIT_REQ
05BF 7220                     JC      G24
05C1 E8AD01                   CALL    PORT_0
05C4 EC                       IN      AL,DX
05C5 26884542                 MOV     ES:HD_ERROR[DI],AL      ; STORE AWAY SENSE BYTES
05C9 47                       INC     DI
05CA E8B101                   CALL    PORT_1
05CD E2ED                     LOOP    G22
05CF E8B800                   CALL    HD_WAIT_REQ
05D2 720D                     JC      G24
05D4 E89A01                   CALL    PORT_0
05D7 EC                       IN      AL,DX
05D8 A802                     TEST    AL,2
05DA 740F                     JZ      STAT_ERR
05DC                  SENSE_ABORT:
05DC C6067400FF               MOV     DISK_STATUS,SENSE_FAIL
05E1                  G24:
05E1 F9                       STC
05E2 C3                       RET
                      ERROR_CHK       ENDP

05E3 1A06             T_0     DW      TYPE_0
05E5 2706             T_1     DW      TYPE_1
05E7 6A06             T_2     DW      TYPE_2
05E9 7706             T_3     DW      TYPE_3

05EB                  STAT_ERR:
05EB 268A1E4200               MOV     BL,ES:HD_ERROR          ; GET ERROR BYTE
05F0 8AC3                     MOV     AL,BL
05F2 240F                     AND     AL,0FH
05F4 80E330                   AND     BL,00110000B            ; ISOLATE TYPE
05F7 2AFF                     SUB     BH,BH
05F9 B103                     MOV     CL,3
05FB D3EB                     SHR     BX,CL                   ; ADJUST
05FD 2EFFA7E305               JMP     WORD PTR CS:[BX + OFFSET T_0]
                              ASSUME  ES:NOTHING

0602                  TYPE0_TABLE     LABEL   BYTE
```

```
LOC OBJ                 SOURCE

0602 00204020800020             DB      0,BAD_CNTLR,BAD_SEEK,BAD_CNTLR,TIME_OUT,0,BAD_CNTLR
0609 0040                       DB      0,BAD_SEEK
  0009                  TYPE0_LEN       EQU     $-TYPE0_TABLE
060B                    TYPE1_TABLE     LABEL   BYTE
060B 1010020004                 DB      BAD_ECC,BAD_ECC,BAD_ADDR_MARK,0,RECORD_NOT_FND
0610 400000110B                 DB      BAD_SEEK,0,0,DATA_CORRECTED,BAD_TRACK
  000A                  TYPE1_LEN       EQU     $-TYPE1_TABLE
0615                    TYPE2_TABLE     LABEL   BYTE
0615 0102                       DB      BAD_CMD,BAD_ADDR_MARK
  0002                  TYPE2_LEN       EQU     $-TYPE2_TABLE
0617                    TYPE3_TABLE     LABEL   BYTE
0617 202010                     DB      BAD_CNTLR,BAD_CNTLR,BAD_ECC
  0003                  TYPE3_LEN       EQU     $-TYPE3_TABLE

                        ;----- TYPE 0 ERROR

061A                    TYPE_0:
061A BB0206                     MOV     BX,OFFSET TYPE0_TABLE
061D 3C09                       CMP     AL,TYPE0_LEN            ; CHECK IF ERROR IS DEFINED
061F 7363                       JAE     UNDEF_ERR_L
0621 2ED7                       XLAT    CS:TYPE0_TABLE          ; TABLE LOOKUP
0623 A27400                     MOV     DISK_STATUS,AL          ; SET ERROR CODE
0626 C3                         RET

                        ;------ TYPE 1 ERROR

0627                    TYPE_1:
0627 BB0B06                     MOV     BX,OFFSET TYPE1_TABLE
062A 8BC8                       MOV     CX,AX
062C 3C0A                       CMP     AL,TYPE1_LEN            ; CHECK IF ERROR IS DEFINED
062E 7354                       JAE     UNDEF_ERR_L
0630 2ED7                       XLAT    CS:TYPE1_TABLE          ; TABLE LOOKUP
0632 A27400                     MOV     DISK_STATUS,AL          ; SET ERROR CODE
0635 80E108                     AND     CL,08H                  ; CORRECTED ECC
0638 80F908                     CMP     CL,08H
063B 752A                       JNZ     G30

                        ;----- OBTAIN ECC ERROR BURST LENGTH

063D C60642000D                 MOV     CMD_BLOCK+0,RD_ECC_CMD
0642 2AC0                       SUB     AL,AL
0644 E81BFF                     CALL    COMMAND
0647 721E                       JC      G30
0649 E83E00                     CALL    HD_WAIT_REQ
064C 7219                       JC      G30
064E E82001                     CALL    PORT_0
0651 EC                         IN      AL,DX
0652 8AC8                       MOV     CL,AL
0654 E83300                     CALL    HD_WAIT_REQ
0657 720E                       JC      G30
0659 E81501                     CALL    PORT_0
065C EC                         IN      AL,DX
065D A801                       TEST    AL,01H
065F 7406                       JZ      G30
0661 C606740020                 MOV     DISK_STATUS,BAD_CNTLR
0666 F9                         STC
0667                    G30:
0667 8AC1                       MOV     AL,CL
0669 C3                         RET

                        ;----- TYPE 2 ERROR

066A                    TYPE_2:
066A BB1506                     MOV     BX,OFFSET TYPE2_TABLE
066D 3C02                       CMP     AL,TYPE2_LEN            ; CHECK IF ERROR IS DEFINED
066F 7313                       JAE     UNDEF_ERR_L
0671 2ED7                       XLAT    CS:TYPE1_TABLE          ; TABLE LOOKUP
0673 A27400                     MOV     DISK_STATUS,AL          ; SET ERROR CODE
0676 C3                         RET

                        ;----- TYPE 3 ERROR

0677                    TYPE_3:
0677 BB1706                     MOV     BX,OFFSET TYPE3_TABLE
067A 3C03                       CMP     AL,TYPE3_LEN
067C 7306                       JAE     UNDEF_ERR_L
067E 2ED7                       XLAT    CS:TYPE3_TABLE
```

```
LOC OBJ                     SOURCE

0680 A27400                         MOV     DISK_STATUS,AL
0683 C3                             RET

0684                        UNDEF_ERR_L:
0684 C6067400BB                     MOV     DISK_STATUS,UNDEF_ERR
0689 C3                             RET

068A                        HD_WAIT_REQ     PROC    NEAR
068A 51                             PUSH    CX
068B 2BC9                           SUB     CX,CX
068D E8EE00                         CALL    PORT_1
0690                        L1:
0690 EC                             IN      AL,DX
0691 A801                           TEST    AL,R1_REQ
0693 7508                           JNZ     L2
0695 E2F9                           LOOP    L1
0697 C606740080                     MOV     DISK_STATUS,TIME_OUT
069C F9                             STC
069D                        L2:
069D 59                             POP     CX
069E C3                             RET
                            HD_WAIT_REQ     ENDP

                            ;------------------------------------------------------
                            ; DMA_SETUP                                          :
                            ;       THIS ROUTINE SETS UP FOR DMA OPERATIONS.     :
                            ; INPUT                                              :
                            ;       (AL) = MODE BYTE FOR THE DMA                 :
                            ;       (ES:BX) = ADDRESS TO READ/WRITE THE DATA     :
                            ; OUTPUT                                             :
                            ;       (AX) DESTROYED                               :
                            ;------------------------------------------------------
069F                        DMA_SETUP       PROC    NEAR
069F 50                             PUSH    AX
06A0 A04600                         MOV     AL,CMD_BLOCK+4
06A3 3C81                           CMP     AL,81H                  ; BLOCK COUNT OUT OF RANGE
06A5 58                             POP     AX
06A6 7202                           JB      J1
06A8 F9                             STC
06A9 C3                             RET
06AA                        J1:
06AA 51                             PUSH    CX                      ; SAVE THE REGISTER
06AB FA                             CLI                             ; NO MORE INTERRUPTS
06AC E60C                           OUT     DMA+12,AL               ; SET THE FIRST/LAST F/F
06AE 50                             PUSH    AX
06AF 58                             POP     AX
06B0 E60B                           OUT     DMA+11,AL               ; OUTPUT THE MODE BYTE
06B2 8CC0                           MOV     AX,ES                   ; GET THE ES VALUE
06B4 B104                           MOV     CL,4                    ; SHIFT COUNT
06B6 D3C0                           ROL     AX,CL                   ; ROTATE LEFT
06B8 8AE8                           MOV     CH,AL                   ; GET HIGHEST NYBBLE OF ES TO CH
06BA 24F0                           AND     AL,0F0H                 ; ZERO THE LOW NYBBLE FROM SEGMENT
06BC 03C3                           ADD     AX,BX                   ; TEST FOR CARRY FROM ADDITION
06BE 7302                           JNC     J33
06C0 FEC5                           INC     CH                      ; CARRY MEANS HIGH 4 BITS MUST BE INC
06C2                        J33:
06C2 50                             PUSH    AX                      ; SAVE START ADDRESS
06C3 E606                           OUT     DMA+6,AL                ; OUTPUT LOW ADDRESS
06C5 8AC4                           MOV     AL,AH
06C7 E606                           OUT     DMA+6,AL                ; OUTPUT HIGH ADDRESS
06C9 8AC5                           MOV     AL,CH                   ; GET HIGH 4 BITS
06CB 240F                           AND     AL,0FH
06CD E682                           OUT     DMA_HIGH,AL             ; OUTPUT THE HIGH 4 BITS TO PAGE REG

                            ;------ DETERMINE COUNT

06CF A04600                         MOV     AL,CMD_BLOCK+4          ; RECOVER BLOCK COUNT
06D2 D0E0                           SHL     AL,1                    ; MULTIPLY BY 512 BYTES PER SECTOR
06D4 FEC8                           DEC     AL                      ;   AND DECREMENT VALUE BY ONE
06D6 8AE0                           MOV     AH,AL
06D8 B0FF                           MOV     AL,0FFH

                            ;----- HANDLE READ AND WRITE LONG (516D BYTE BLOCKS)

06DA 50                             PUSH    AX                      ; SAVE REGISTER
06DB A04200                         MOV     AL,CMD_BLOCK+0          ; GET COMMAND
06DE 3CE5                           CMP     AL,RD_LONG_CMD
```

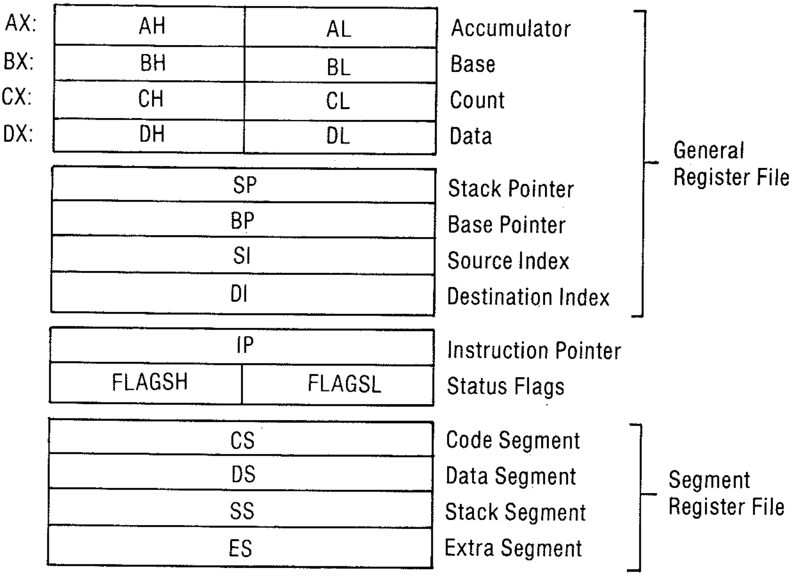
```
06E0 7407                      JE      ADD4
06E2 3CE6                      CMP     AL,WR_LONG_CMD
06E4 7403                      JE      ADD4
06E6 58                        POP     AX                      ; RESTORE REGISTER
06E7 EB11                      JMP     SHORT   J20
06E9                   ADD4:
06E9 58                        POP     AX                      ; RESTORE REGISTER
06EA B80402                    MOV     AX,516D                 ; ONE BLOCK (512) PLUS 4 BYTES ECC
06ED 53                        PUSH    BX
06EE 2AFF                      SUB     BH,BH
06F0 8A1E4600                  MOV     BL,CMD_BLOCK+4
06F4 52                        PUSH    DX
06F5 F7E3                      MUL     BX                      ; BLOCK COUNT TIMES 516
06F7 5A                        POP     DX
06F8 5B                        POP     BX
06F9 48                        DEC     AX                      ; ADJUST
06FA                   J20:

06FA 50                        PUSH    AX                      ; SAVE COUNT VALUE
06FB E607                      OUT     DMA+7,AL                ; LOW BYTE OF COUNT
06FD 8AC4                      MOV     AL,AH
06FF E607                      OUT     DMA+7,AL                ; HIGH BYTE OF COUNT
0701 FB                        STI                             ; INTERRUPTS BACK ON
0702 59                        POP     CX                      ; RECOVER COUNT VALUE
0703 58                        POP     AX                      ; RECOVER ADDRESS VALUE
0704 03C1                      ADD     AX,CX                   ; ADD, TEST FOR 64K OVERFLOW
0706 59                        POP     CX                      ; RECOVER REGISTER
0707 C3                        RET                     ; RETURN TO CALLER, CFL SET BY ABOVE IF ERROR
                       DMA_SETUP       ENDP

                       ;----------------------------------------------
                       ; WAIT_INT                                     :
                       ;       THIS ROUTINE WAITS FOR THE FIXED DISK  :
                       ;       CONTROLLER TO SIGNAL THAT AN INTERRUPT :
                       ;       HAS OCCURRED.                          :
                       ;----------------------------------------------
0708                   WAIT_INT        PROC    NEAR
0708 FB                        STI                             ; TURN ON INTERRUPTS
0709 53                        PUSH    BX                      ; PRESERVE REGISTERS
070A 51                        PUSH    CX
070B 06                        PUSH    ES
070C 56                        PUSH    SI
070D 1E                        PUSH    DS
                               ASSUME  DS:DUMMY
070E 2BC0                      SUB     AX,AX
0710 8ED8                      MOV     DS,AX                   ; ESTABLISH SEGMENT
0712 C4360401                  LES     SI,HF_TBL_VEC
                               ASSUME  DS:DATA
0716 1F                        POP     DS

                       ;----- SET TIMEOUT VALUES

0717 2AFF                      SUB     BH,BH
0719 268A5C09                  MOV     BL,BYTE PTR ES:[SI][9]          ; STANDARD TIME OUT
071D 8A264200                  MOV     [illegible]
0721 80FC04                    CMP     AH,FMTDRV_CMD
0724 7506                      JNZ     W5
0726 268A5C0A                  MOV     BL,BYTE PTR ES:[SI][0AH]        ; FORMAT DRIVE
072A EB09                      JMP     SHORT   W4
072C 80FCE3            W5:     CMP     AH,CHK_DRV_CMD
072F 7504                      JNZ     W4
0731 268A5C0B                  MOV     BL,BYTE PTR ES:[SI][0BH]        ; CHECK DRIVE
0735                   W4:
0735 2BC9                      SUB     CX,CX

                       ;----- WAIT FOR INTERRUPT

0737                   W1:
0737 E84400                    CALL    PORT_1
073A EC                        IN      AL,DX
073B 2420                      AND     AL,020H
073D 3C20                      CMP     AL,020H                 ; DID INTERRUPT OCCUR
073F 740A                      JZ      W2
0741 E2F4                      LOOP    W1                      ; INNER LOOP
0743 4B                        DEC     BX
0744 75F1                      JNZ     W1                      ; OUTER LOOP
0746 C606740080                MOV     DISK_STATUS,TIME_OUT
074B                   W2:
```

```
LOC OBJ            SOURCE

074B E82300                CALL    PORT_0
074E EC                    IN      AL,DX
074F 2402                  AND     AL,2                    ; ERROR BIT
0751 08067400              OR      DISK_STATUS,AL          ; SAVE
0755 E83000                CALL    PORT_3                  ; INTERRUPT MASK REGISTER
0758 32C0                  XOR     AL,AL                   ; ZERO
075A EE                    OUT     DX,AL                   ; RESET MASK
075B 5E                    POP     SI                      ; RESTORE REGISTERS
075C 07                    POP     ES
075D 59                    POP     CX
075E 5B                    POP     BX
075F C3                    RET
                   WAIT_INT        ENDP

0760               HD_INT  PROC    NEAR
0760 50                    PUSH    AX
0761 B020                  MOV     AL,EOI                  ; END OF INTERRUPT
0763 E620                  OUT     INT_CTL_PORT,AL
0765 B007                  MOV     AL,07H                  ; SET DMA MODE TO DISABLE
0767 E60A                  OUT     DMA+10,AL
0769 E421                  IN      AL,021H
076B 0C20                  OR      AL,020H
076D E621                  OUT     021H,AL
076F 58                    POP     AX
0770 CF                    IRET
                   HD_INT  ENDP

                   ;----------------------------------------
                   ; PORTS                                  :
                   ;       GENERATE PROPER PORT VALUE       :
                   ;       BASED ON THE PORT OFFSET         :
                   ;----------------------------------------

0771               PORT_0  PROC    NEAR
0771 BA2003                MOV     DX,HF_PORT              ; BASE VALUE
0774 50                    PUSH    AX
0775 2AE4                  SUB     AH,AH
0777 A07700                MOV     AL,PORT_OFF             ; ADD IN THE OFFSET
077A 03D0                  ADD     DX,AX
077C 58                    POP     AX
077D C3                    RET
                   PORT_0  ENDP

077E               PORT_1  PROC    NEAR
077E E8F0FF                CALL    PORT_0
0781 42                    INC     DX                      ; INCREMENT TO PORT ONE
0782 C3                    RET
                   PORT_1  ENDP

0783               PORT_2  PROC    NEAR
0783 E8F8FF                CALL    PORT_1
0786 42                    INC     DX                      ; INCREMENT TO PORT TWO
0787 C3                    RET
                   PORT_2  ENDP

0788               PORT_3  PROC    NEAR
0788 E8F8FF                CALL    PORT_2
078B 42                    INC     DX                      ; INCREMENT TO PORT THREE
078C C3                    RET
                   PORT_3  ENDP

                   ;------------------------------------------------
                   ; SW2_OFFS                                       :
                   ;       DETERMINE PARAMETER TABLE OFFSET         :
                   ;       USING CONTROLLER PORT TWO AND            :
                   ;       DRIVE NUMBER SPECIFIER (0-1)             :
                   ;------------------------------------------------

078D               SW2_OFFS        PROC    NEAR
078D E8F3FF                CALL    PORT_2
0790 EC                    IN      AL,DX                   ; READ PORT 2
0791 50                    PUSH    AX
0792 E8E9FF                CALL    PORT_1
0795 EC                    IN      AL,DX
0796 2402                  AND     AL,2                    ; CHECK FOR ERROR
0798 58                    POP     AX
0799 7516                  JNZ     SW2_OFFS_ERR
079B 8A264300              MOV     AH,CMD_BLOCK+1
```

```
LOC OBJ              SOURCE

079F 80E420                   AND     AH,00100000B            ; DRIVE 0 OR 1
07A2 7504                     JNZ     SW2_AND
07A4 D0E8                     SHR     AL,1                    ; ADJUST
07A6 D0E8                     SHR     AL,1
07A8                 SW2_AND:
07A8 2403                     AND     AL,011B                 ; ISOLATE
07AA B104                     MOV     CL,4
07AC D2E0                     SHL     AL,CL                   ; ADJUST
07AE 2AE4                     SUB     AH,AH
07B0 C3                       RET
07B1                 SW2_OFFS_ERR:
07B1 F9                       STC
07B2 C3                       RET
                     SW2_OFFS         ENDP

07B3 30382F31362F38           DB      '08/16/82'              ; RELEASE MARKER
     32

07BB                 END_ADDRESS      LABEL   BYTE
----                 CODE     ENDS
                              END
```

APPENDIX B: 8088 ASSEMBLY INSTRUCTION SET REFERENCE

8088
Register Model



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	

Second Instruction Byte Summary

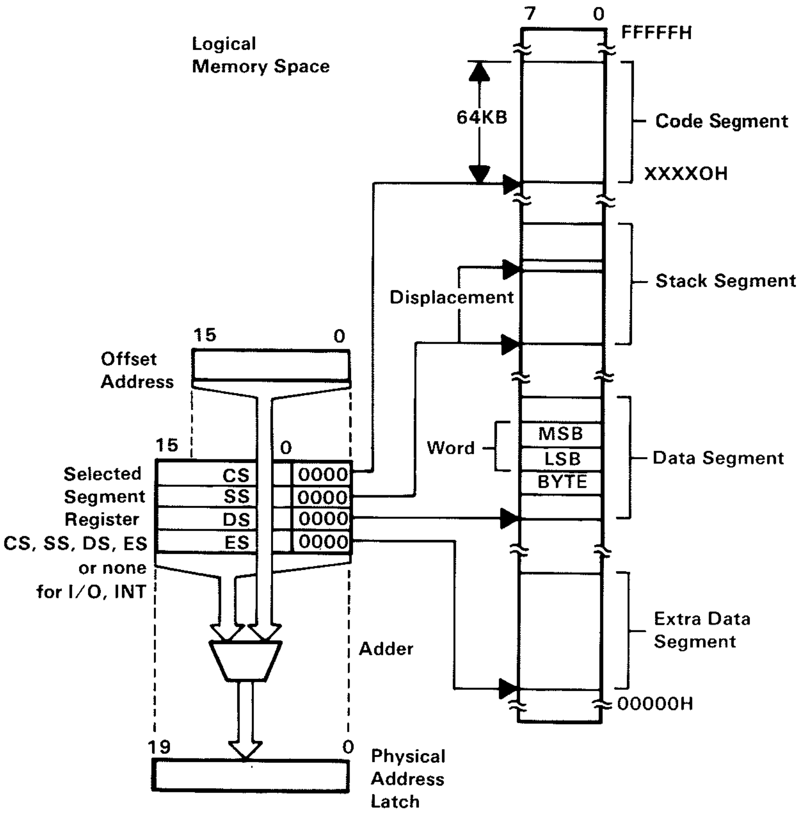
mod	xxx	r/m

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 including strings)	DS	ES, SS, or CS
SI (Implicit Source Address for Strings)	DS	ES, SS, or CS
DI (Implicit Destination Address 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	0	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
Register/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
Register/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
Register/memory and register to either

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

Immediate from register/memory

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

Immediate from accumulator

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

SBB = Subtract with borrow
Register/memory and register to either

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
Register/memory and register to either

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

Immediate to register/memory

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

Immediate to accumulator

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

TEST = And function to flags, no result

Register/memory and register

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

Immediate data and register/memory

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

Immediate data and accumulator

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

OR = OR

Register/memory and register to either

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

Immediate to register/memory

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

Immediate to accumulator

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

XOR = Exclusive or

Register/memory and register to either

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

Immediate to register/memory

1 0 0 0 0 0 0 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 segment adding immediate to SP

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

Intersegment

1 1 0 0 1 0 1 1

Intersegment, adding immediate to SP

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

JE/JZ = Jump on equal/zero

0 1 1 1 0 1 0 0	disp

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

0 1 1 1 1 1 0 0	disp

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

0 1 1 1 1 1 1 0	disp

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

0 1 1 1 0 0 1 0	disp

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

0 1 1 1 0 1 1 0	disp

JP/JPE = Jump on parity/parity even

0 1 1 1 1 0 1 0	disp

JO = Jump on overflow

0 1 1 1 0 0 0 0	disp

JS = Jump on sign

0 1 1 1 1 0 0 0	disp

JNE/JNZ = Jump on not equal/not zero

0 1 1 1 0 1 0 1	disp

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

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	((SF xor OF) or ZF) = 1	"less or equal" or "not greater"
JB or JNAE or JC	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 or JNC	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	0

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	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

LO HI	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 AL	MOV AL m	MOV AL 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	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= CS	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	110	111
Immed	ADD	OR	ADC	SBB	AND	SUB	XOR	CMP
Shift	ROL	ROR	RCL	RCR	SHL/SAL	SHR	—	SAR
Grp 1	TEST	—	NOT	NEG	MUL	IMUL	DIV	IDIV
Grp 2	INC	DEC	CALL id	CALL l,id	JMP id	JMP l,id	PUSH	—

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
28	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	f	f	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	Δ	Ctrl ←		White	White	High Intensity

Value		As Characters			As Text Attributes		
					Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground	
* * * * 80 to 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 Underline
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	Pt	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	░	Alt 176	Note 6	Cyan	Black	Normal
B1	177	▒	Alt 177	Note 6	Cyan	Blue	Underline
B2	178	▓	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	ε	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	η	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 $\boxed{\underset{*}{\text{Prt Sc}}}$ key or 2) in shift mode hit the $\boxed{\overset{*}{8}}$ key.

NOTE 2 Period (.) can easily be keyed using two methods: 1) hit the $\boxed{\underset{\bullet}{>}}$ key or 2) in shift or Num Lock mode hit the $\boxed{\overset{\bullet}{\text{Del}}}$ key.

NOTE 3 Numeric characters (0—9) can easily be keyed using two methods: 1) hit the numeric keys on the top row of the typewriter portion of the keyboard or 2) in shift or Num Lock mode hit the numeric keys in the 10—key pad portion of the keyboard.

NOTE 4 Upper case alphabetic characters (A—Z) can easily be keyed in two modes: 1) in shift mode the appropriate alphabetic key or 2) in Caps Lock mode hit the appropriate alphabetic key.

NOTE 5 Lower case alphabetic characters (a—z) can easily be keyed in two modes: 1) in "normal" mode hit the appropriate key or 2) in Caps Lock combined with shift mode hit the appropriate alphabetic key.

NOTE 6 The 3 digits after the Alt key must be typed from the numeric key pad (keys 71—73, 75—77, 79—82). Character codes 000 through 255 can be entered in this fashion. (With Caps Lock activated, Character codes 97 through 122 will display upper case rather than lower case alphabetic characters.)

Character Set (00-7F) Quick Reference

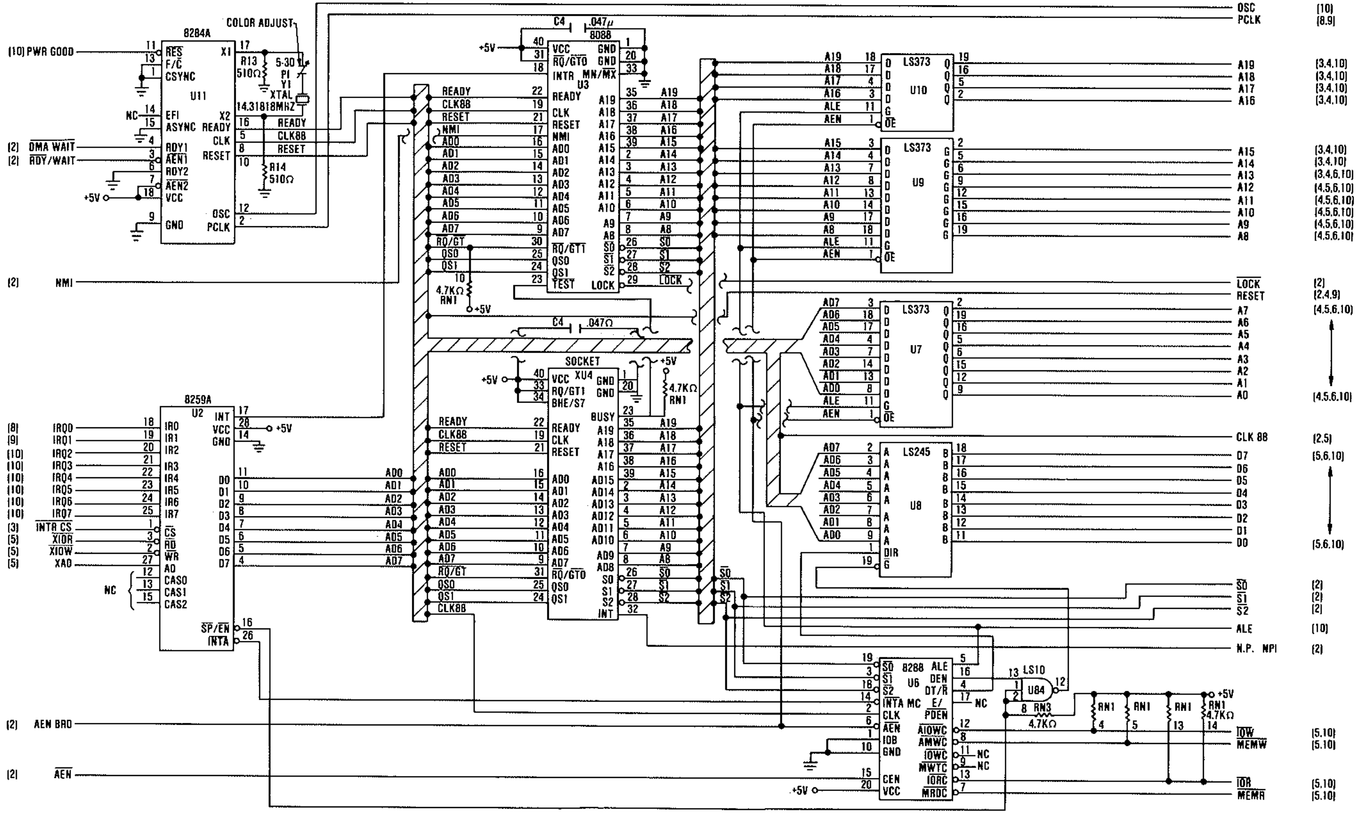
DECIMAL VALUE	➡	0	16	32	48	64	80	96	112
⬇	HEXA-DECIMAL VALUE	0	1	2	3	4	5	6	7
0	0	BLANK (NULL)	►	BLANK (SPACE)	0	@	P	‘	p
1	1	☺	◄	!	1	A	Q	a	q
2	2	☻	↕	"	2	B	R	b	r
3	3	♥	‼	#	3	C	S	c	s
4	4	♦	¶	$	4	D	T	d	t
5	5	♣	§	%	5	E	U	e	u
6	6	♠	▬	&	6	F	V	f	v
7	7	•	↨	'	7	G	W	g	w
8	8	◘	↑	(	8	H	X	h	x
9	9	○	↓	)	9	I	Y	i	y
10	A	◙	→	*	:	J	Z	j	z
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

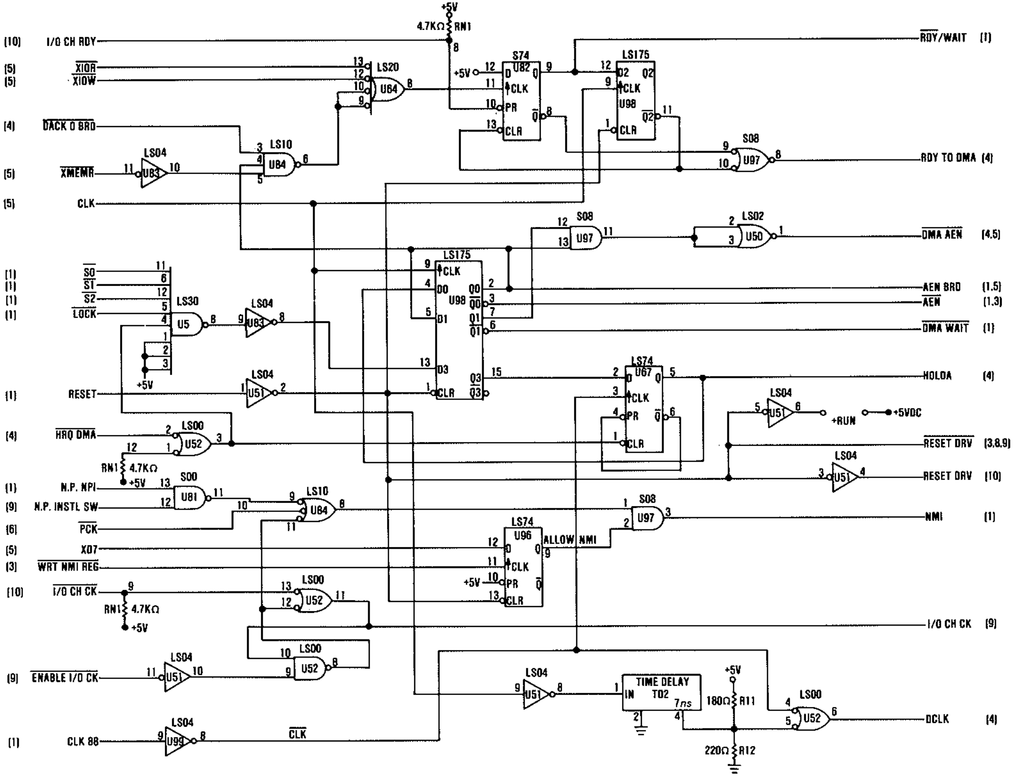
DECIMAL VALUE	➡	128	144	160	176	192	208	224	240
⬇	HEXA-DECIMAL VALUE	8	9	A	B	C	D	E	F
0	0	Ç	É	á	░	└	╨	∝	≡
1	1	ü	æ	í	▒	┴	╤	β	±
2	2	é	Æ	ó	▓	┬	╥	Γ	≥
3	3	â	ô	ú	│	├	╙	π	≤
4	4	ä	ö	ñ	┤	─	╘	Σ	⌠
5	5	à	ò	Ñ	╡	┼	╒	σ	⌡
6	6	å	û	ª	╢	╞	╓	µ	÷
7	7	ç	ù	º	╖	╟	╫	τ	≈
8	8	ê	ÿ	¿	╕	╚	╪	Φ	°
9	9	ë	Ö	⌐	╣	╔	┘	Θ	•
10	A	è	Ü	¬	║	╩	┌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

Notes:

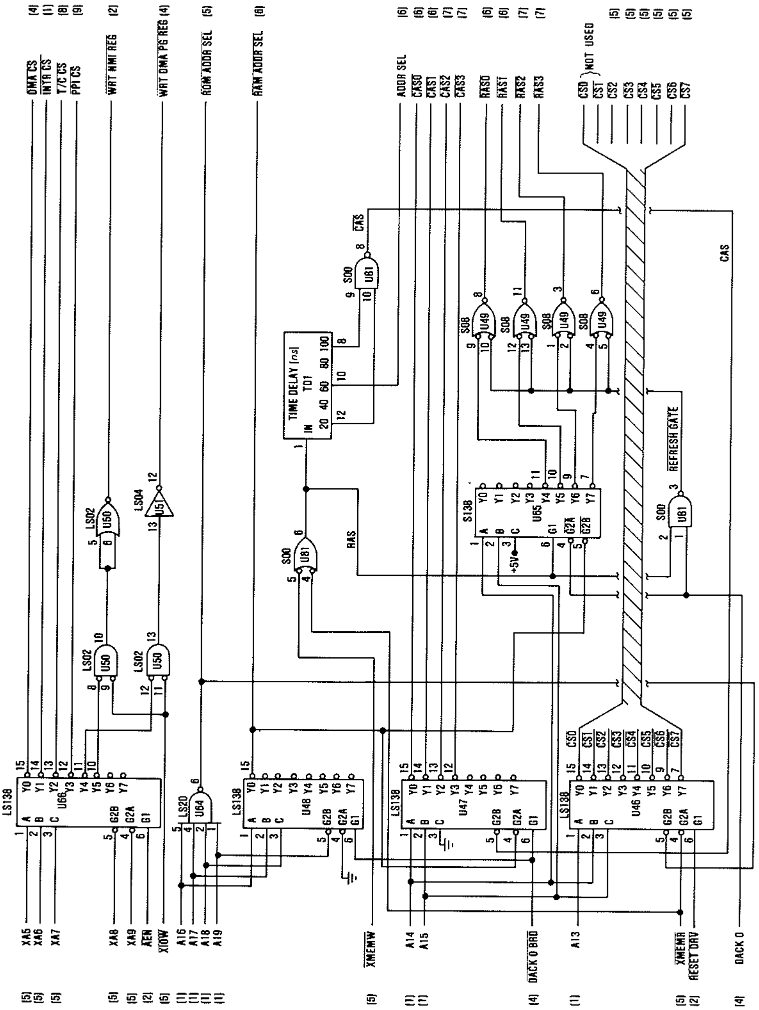
APPENDIX D: LOGIC DIAGRAMS



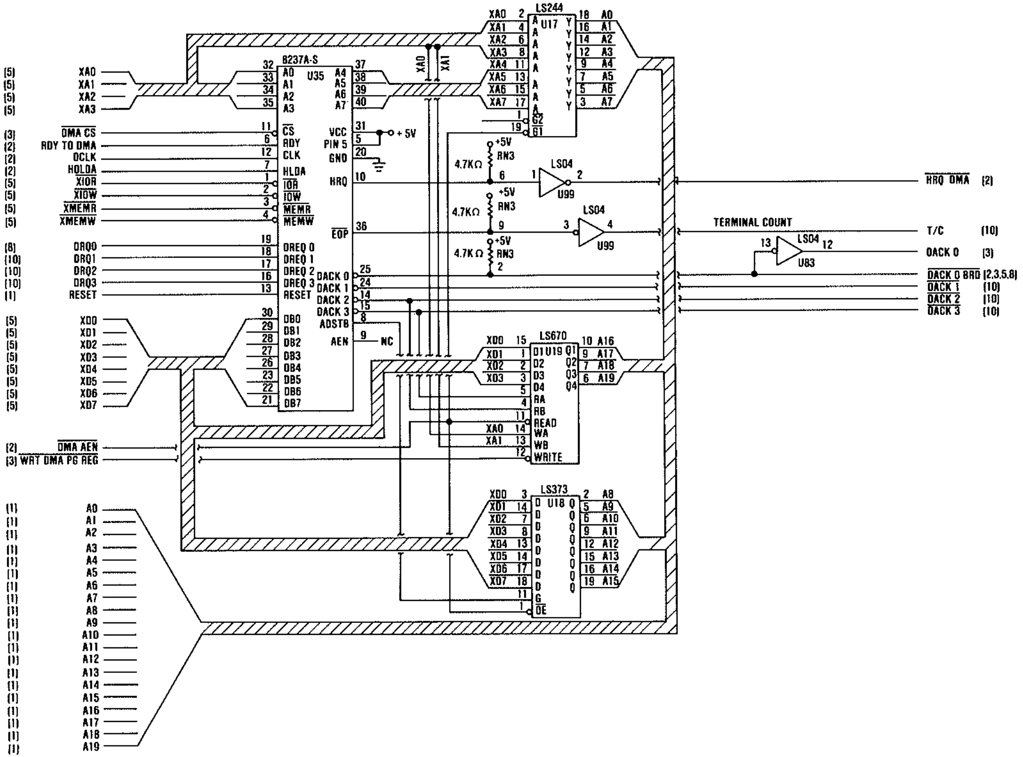
16/64K System Board (Sheet 1 of 10)



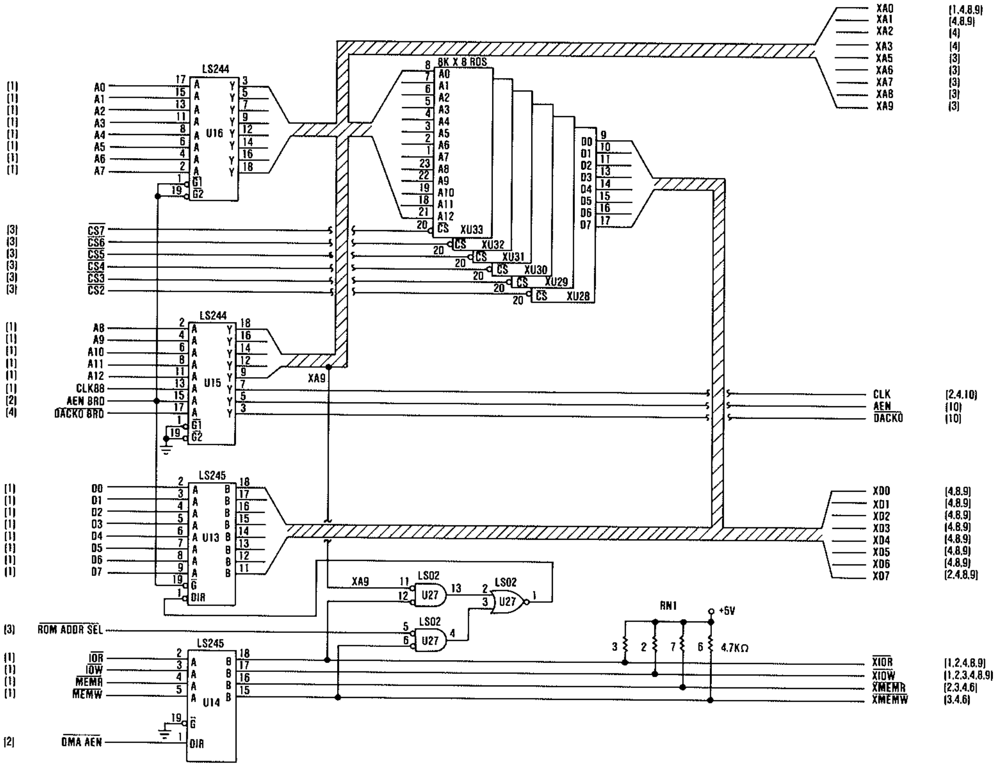
16/64K System Board (Sheet 2 of 10)



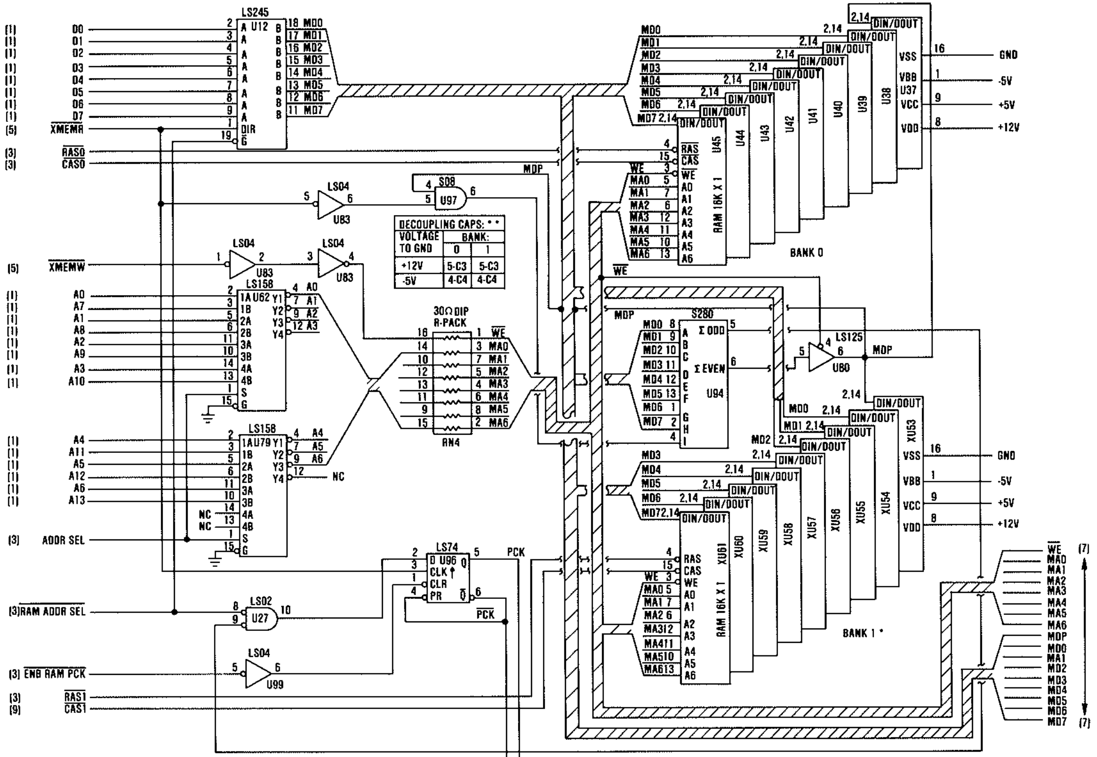
16/64K System Board (Sheet 3 of 10)



16/64K System Board (Sheet 4 of 10)

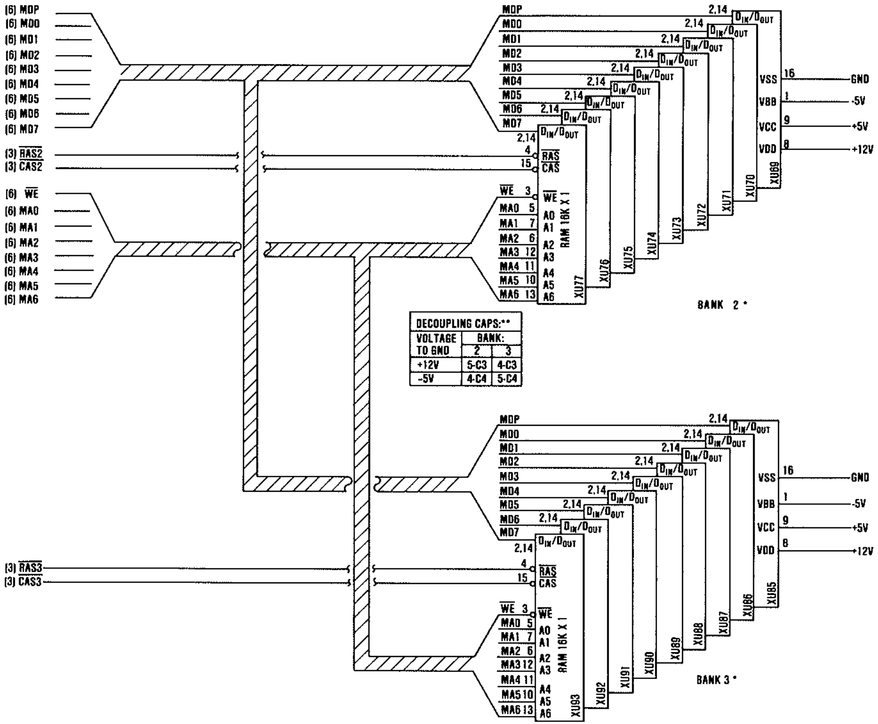


16/64K System Board (Sheet 5 of 10)



* BANK 1 IS A FEATURE
* * C3 = .047μF ≥15WVDC } DECOUPLING
C4 = .047μF ≥7WVDC

16/64K System Board (Sheet 6 of 10)

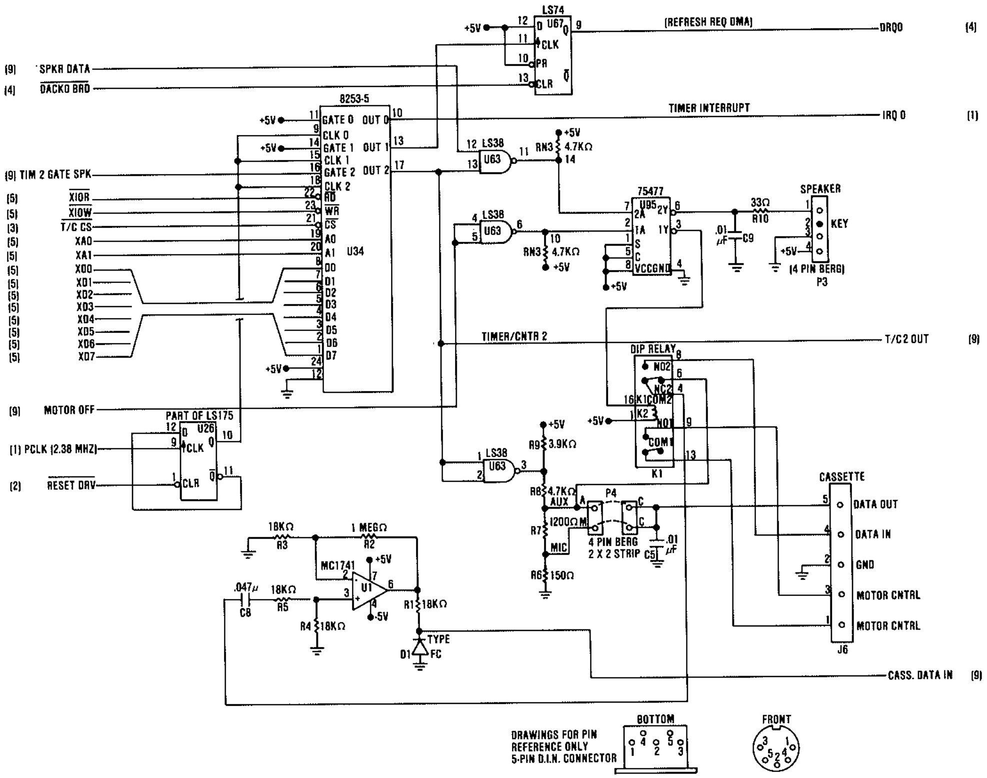


DECOUPLING CAPS:**		
VOLTAGE TO GND	BANK: 2	3
+12V	5-C3	4-C3
-5V	4-C4	5-C4

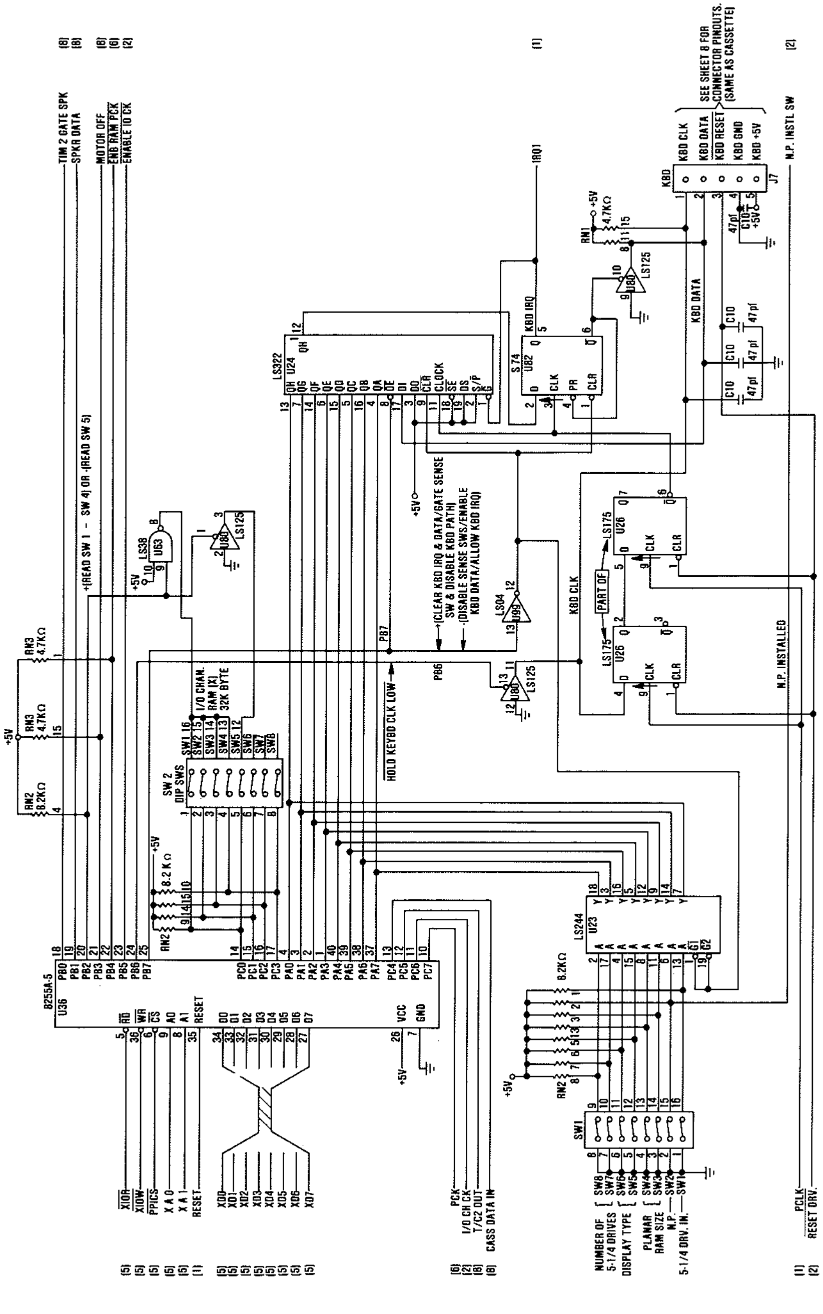
* BANKS 2 & 3 ARE FEATURES.

** C3 = .047µF ≥15WVDC } DECOUPLING
C4 = .047µF ≥7WVDC

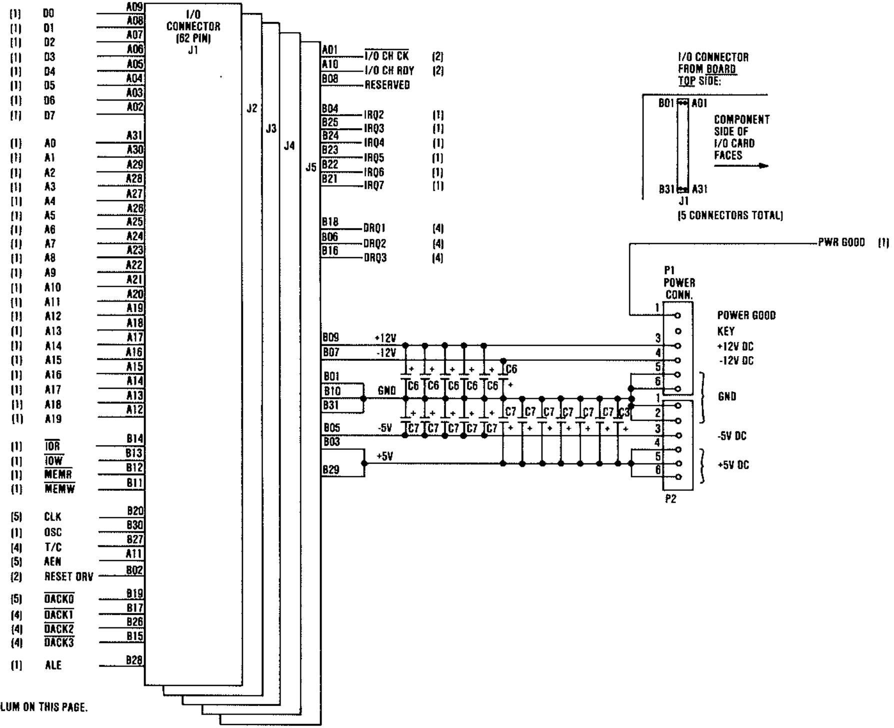
16/64K System Board (Sheet 7 of 10)



16/64K System Board (Sheet 8 of 10)

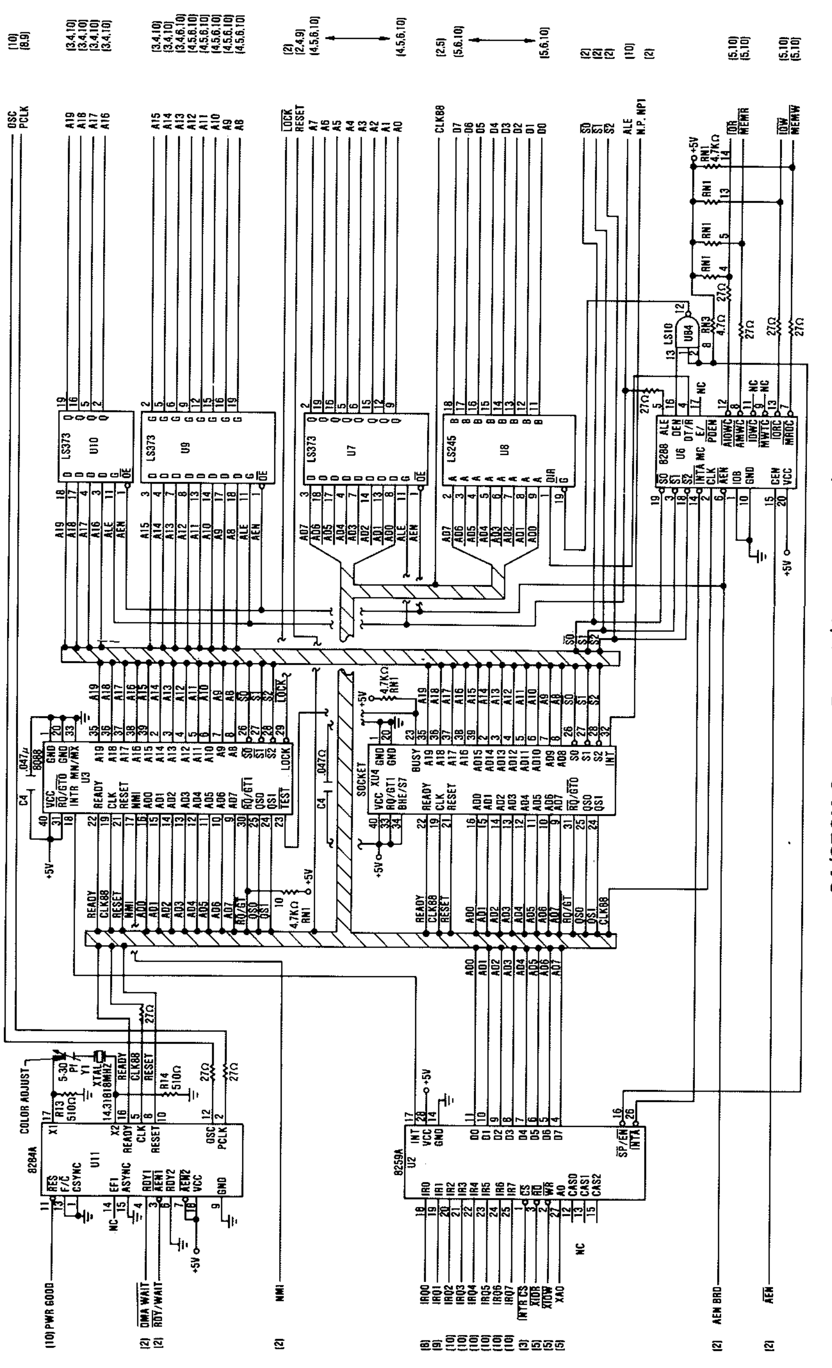


16/64K System Board (Sheet 9 of 10)

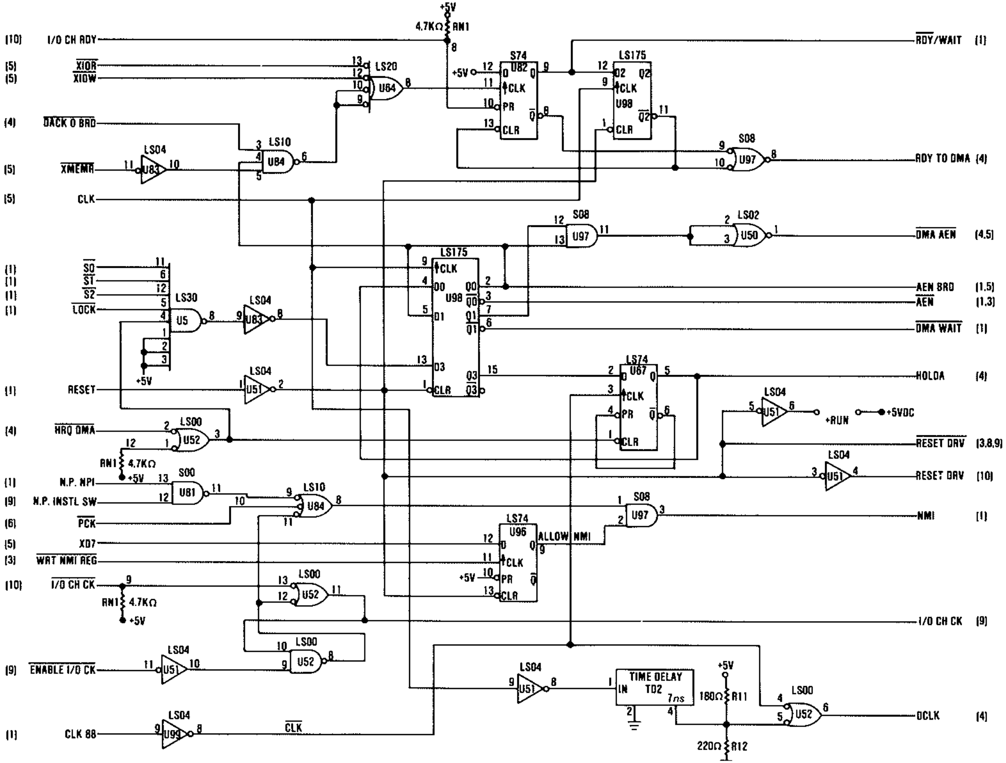


NOTE:
1. ALL CAPS ARE 8.2μF TANTALUM ON THIS PAGE.

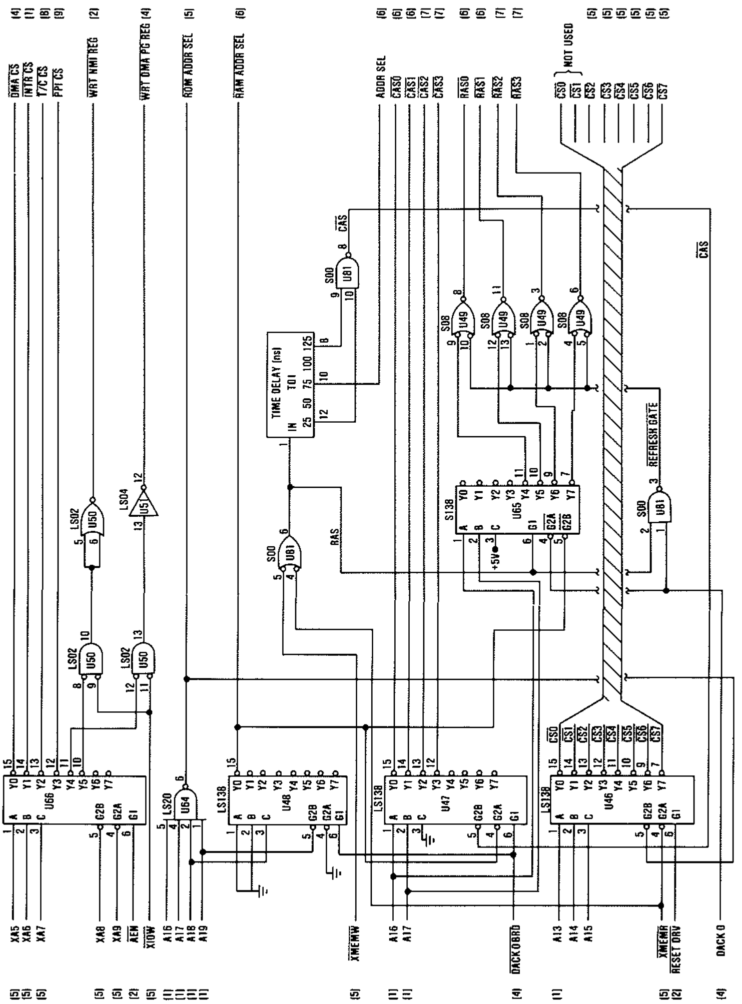
16/64K System Board (Sheet 10 of 10)



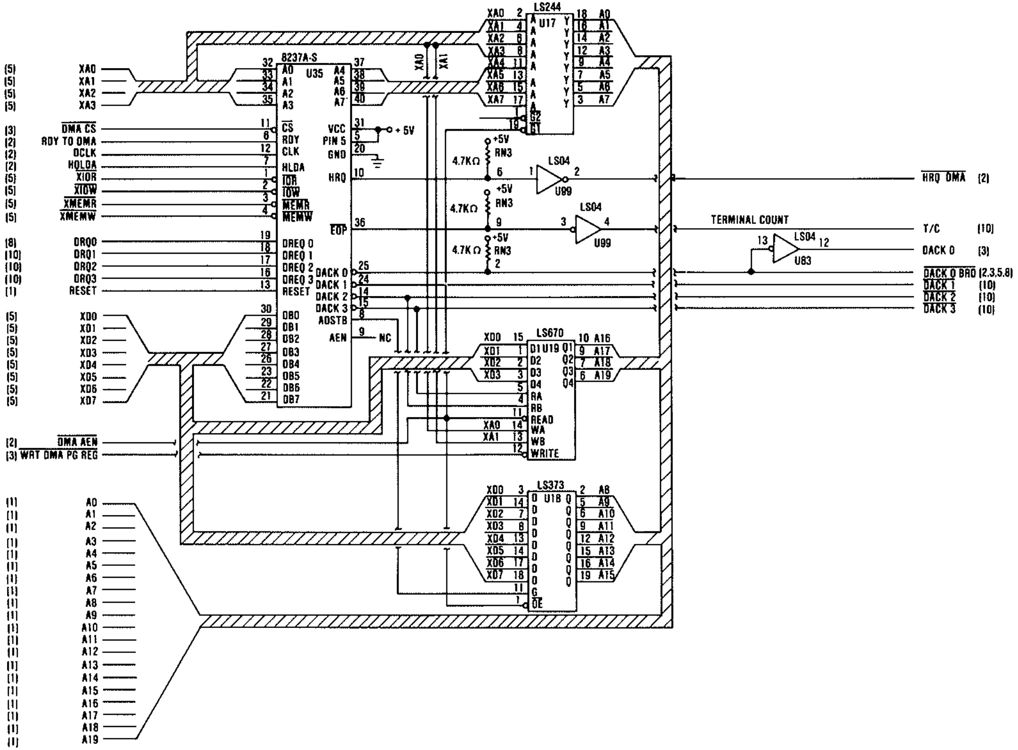
64/256K System Board (Sheet 1 of 10)



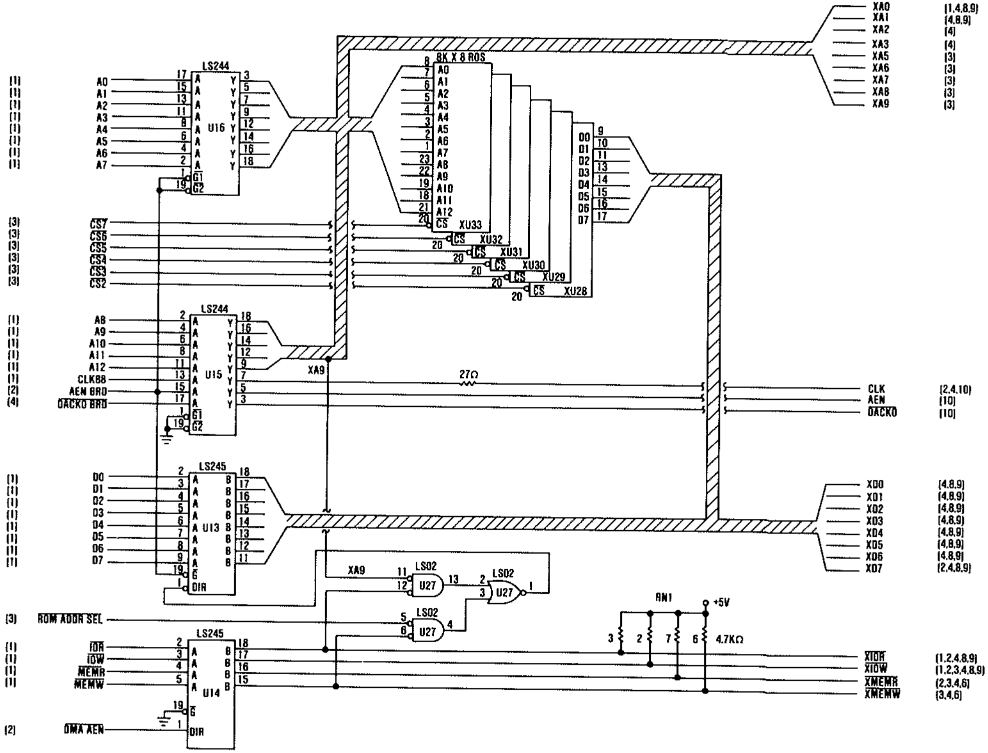
64/256K System Board (Sheet 2 of 10)



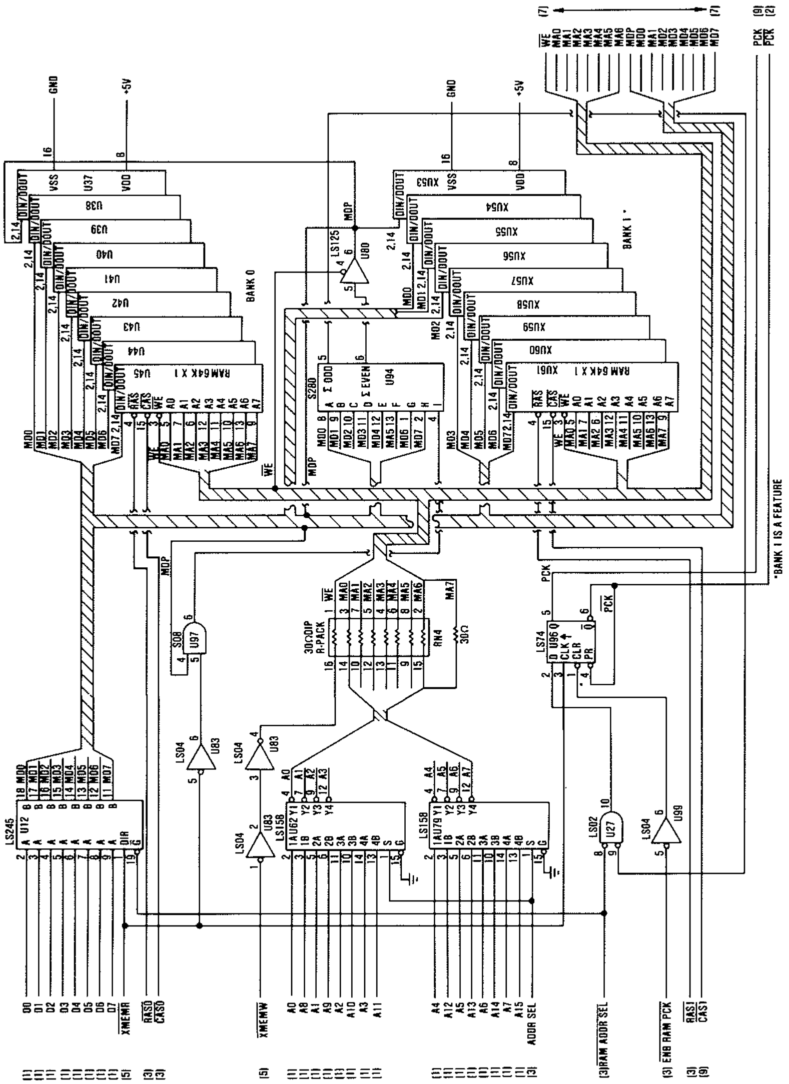
64/256K System Board (Sheet 3 of 10)



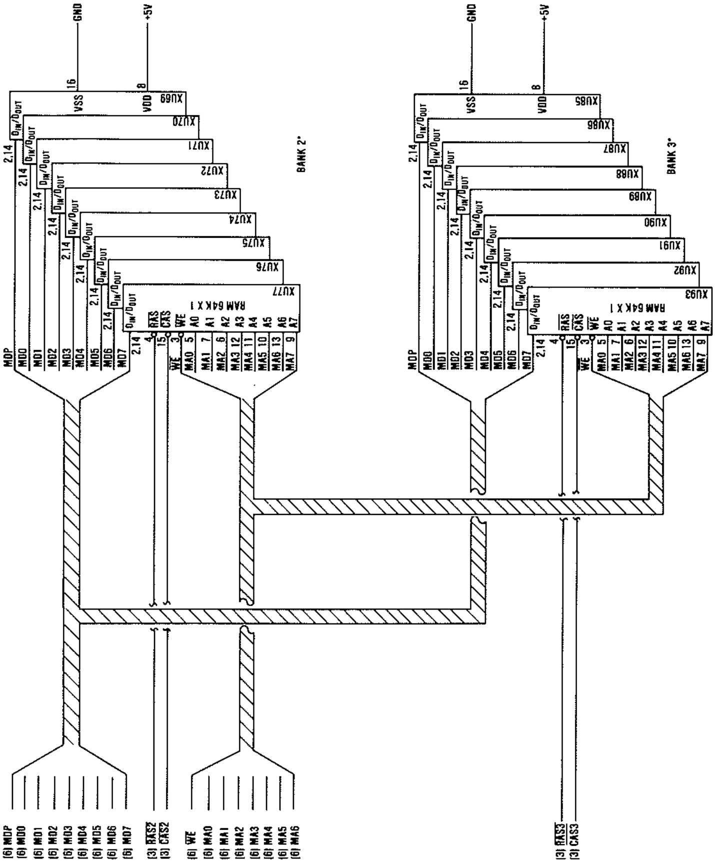
64/256K System Board (Sheet 4 of 10)



64/256K System Board (Sheet 5 of 10)

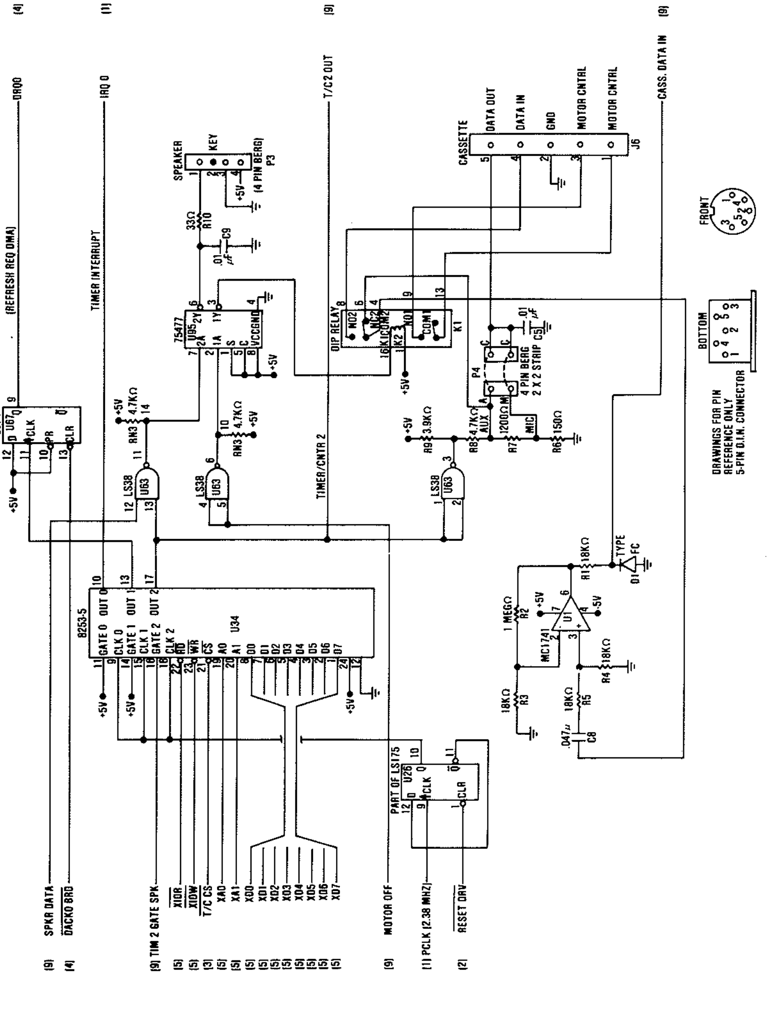


64/256K System Board (Sheet 6 of 10)

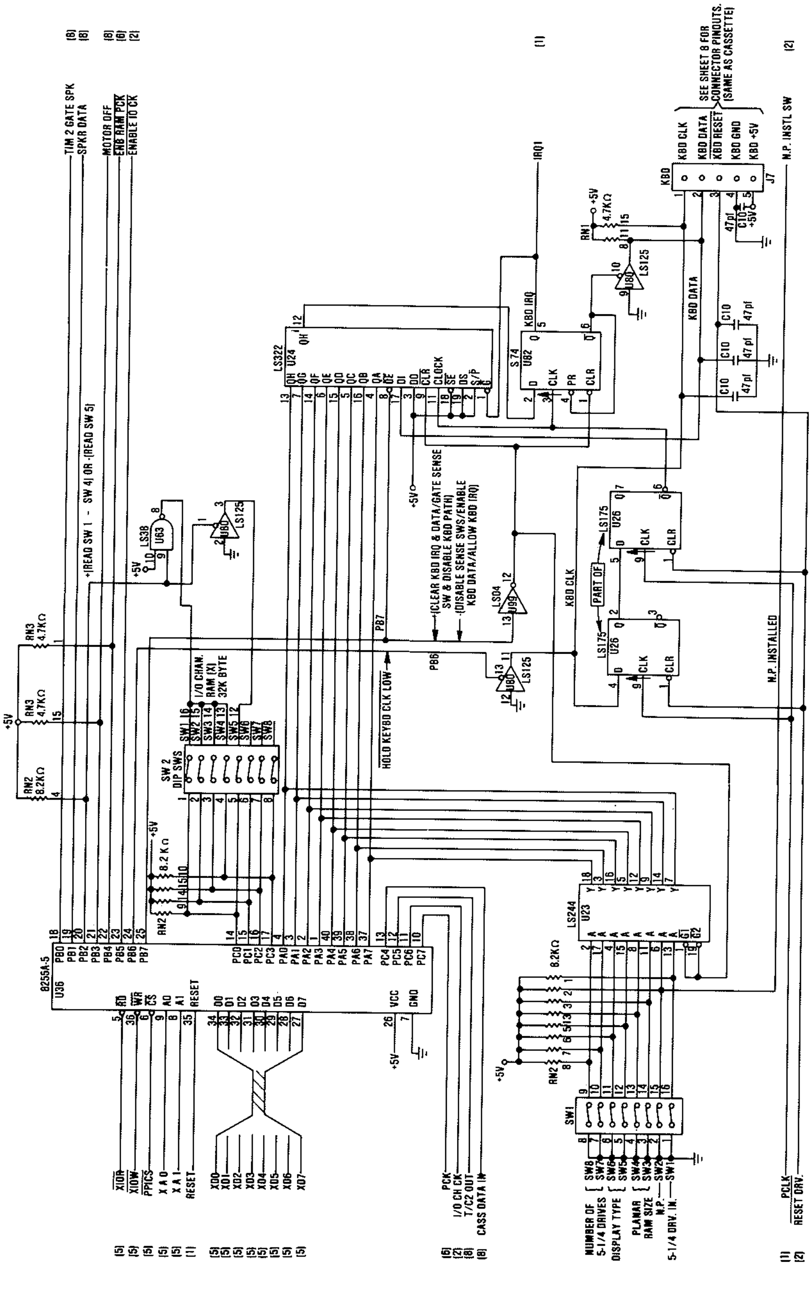


*BANKS 2 & 3 ARE FEATURES.

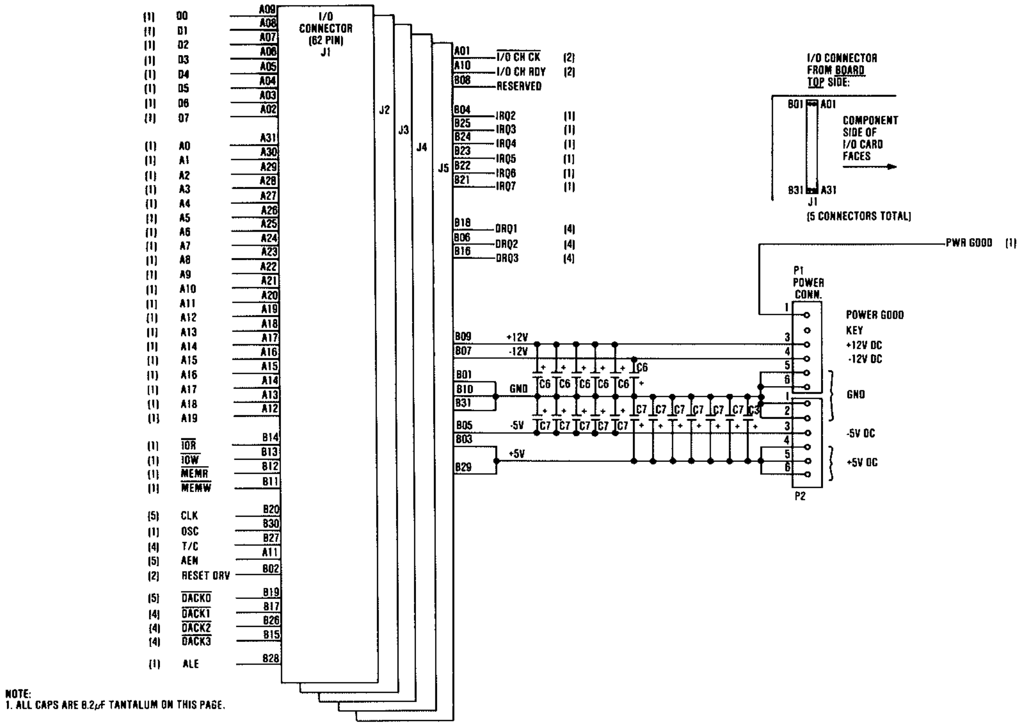
64/256K System Board (Sheet 7 of 10)



64/256K System Board (Sheet 8 of 10)

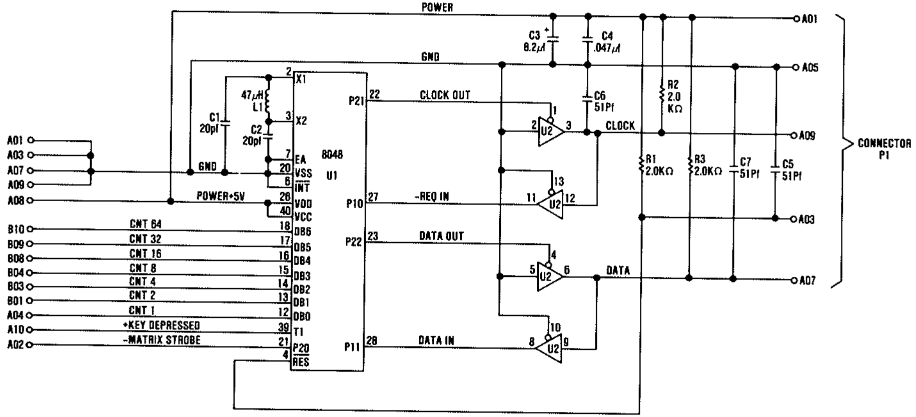


64/256K System Board (Sheet 9 of 10)

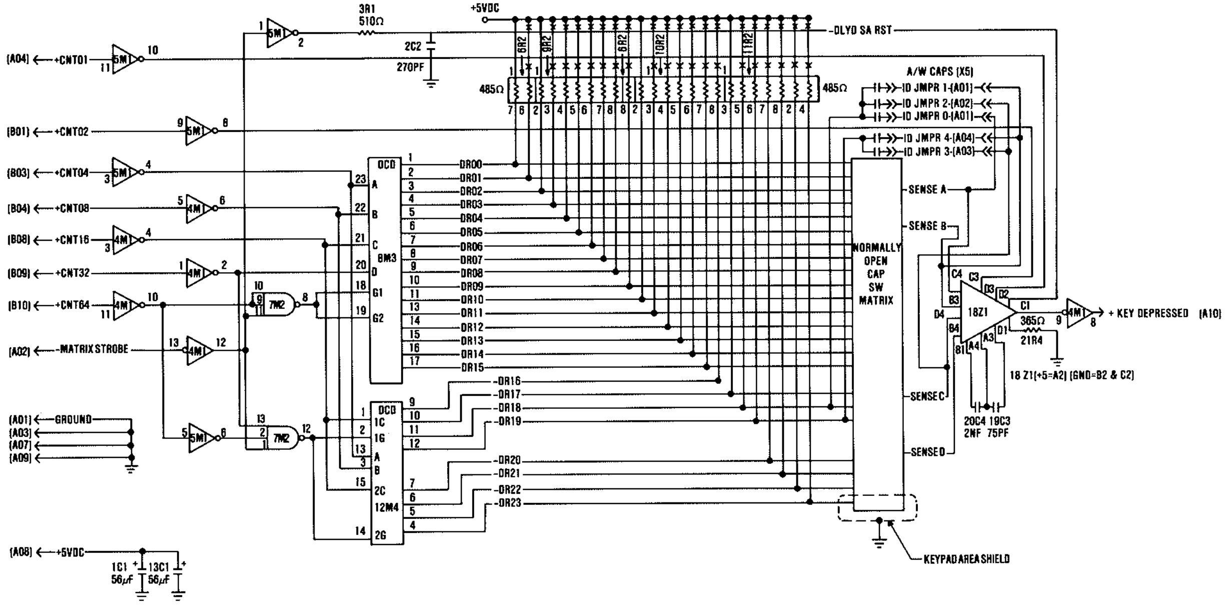


NOTE:
1. ALL CAPS ARE 8.2µF TANTALUM ON THIS PAGE.

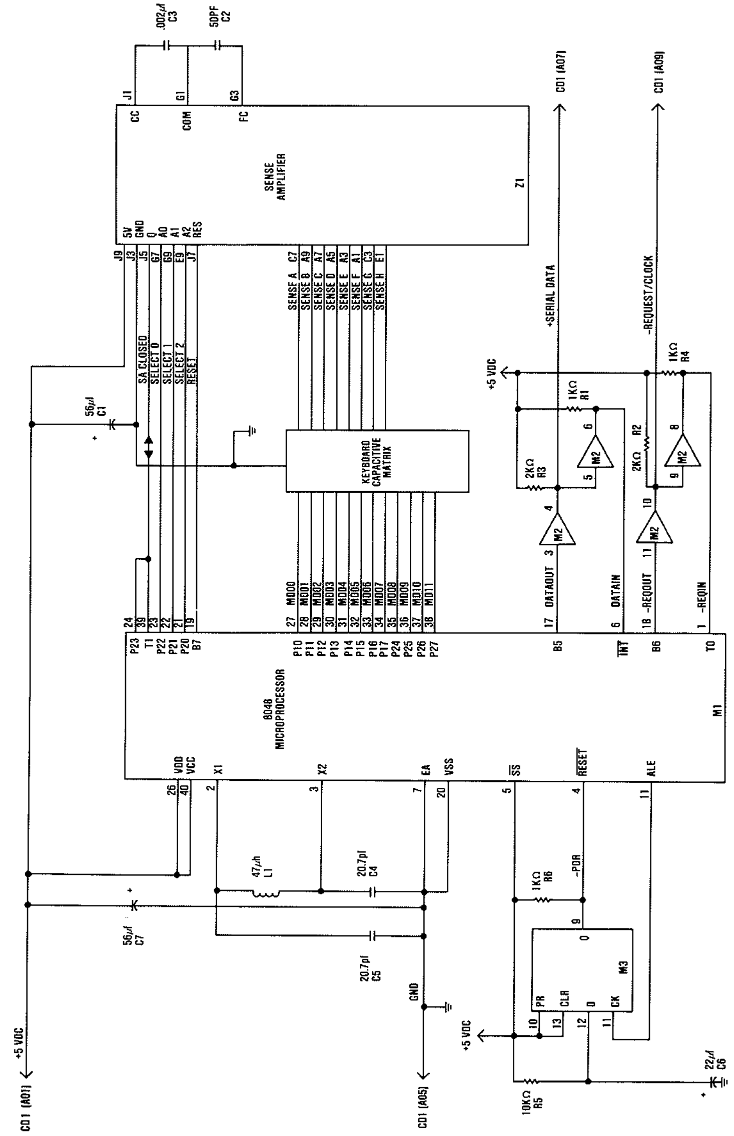
64/256K System Board (Sheet 10 of 10)



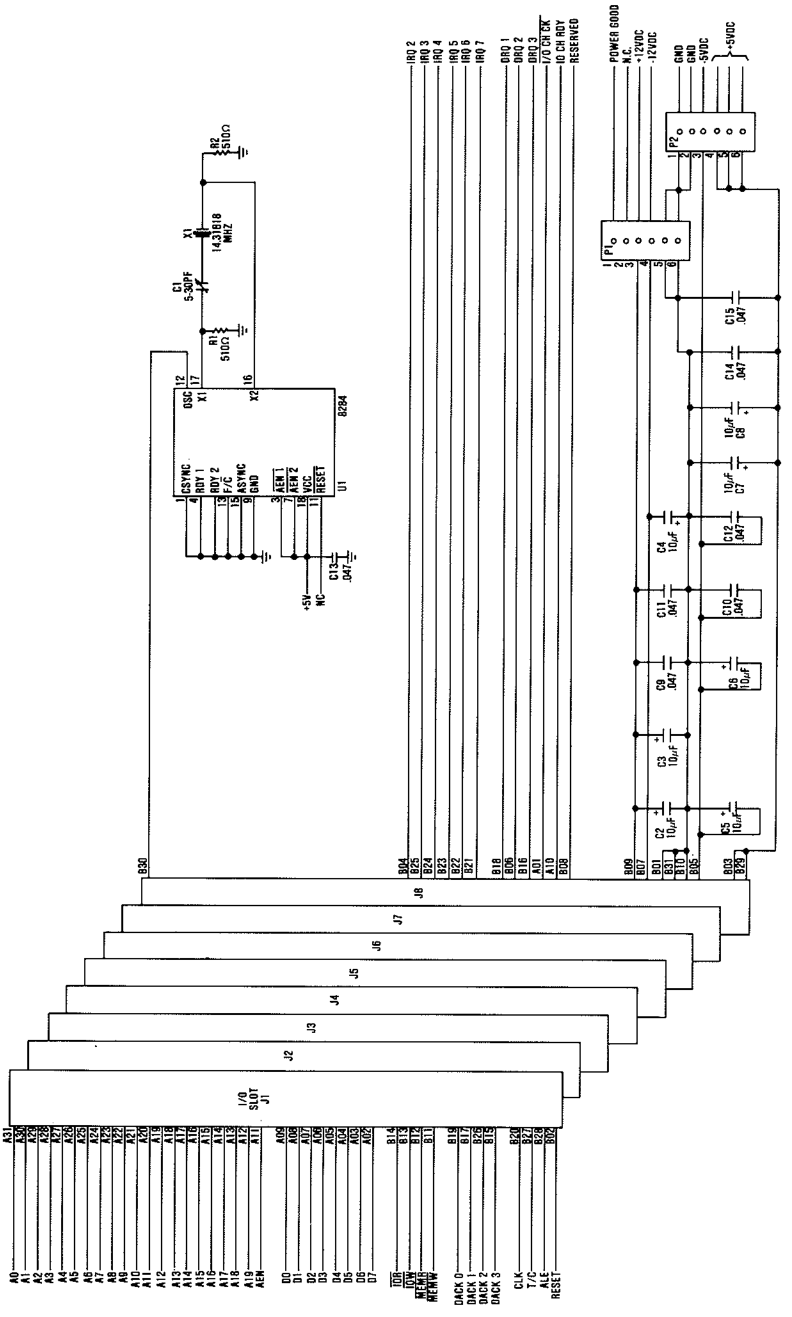
Keyboard — Type 1 (Sheet 1 of 2)



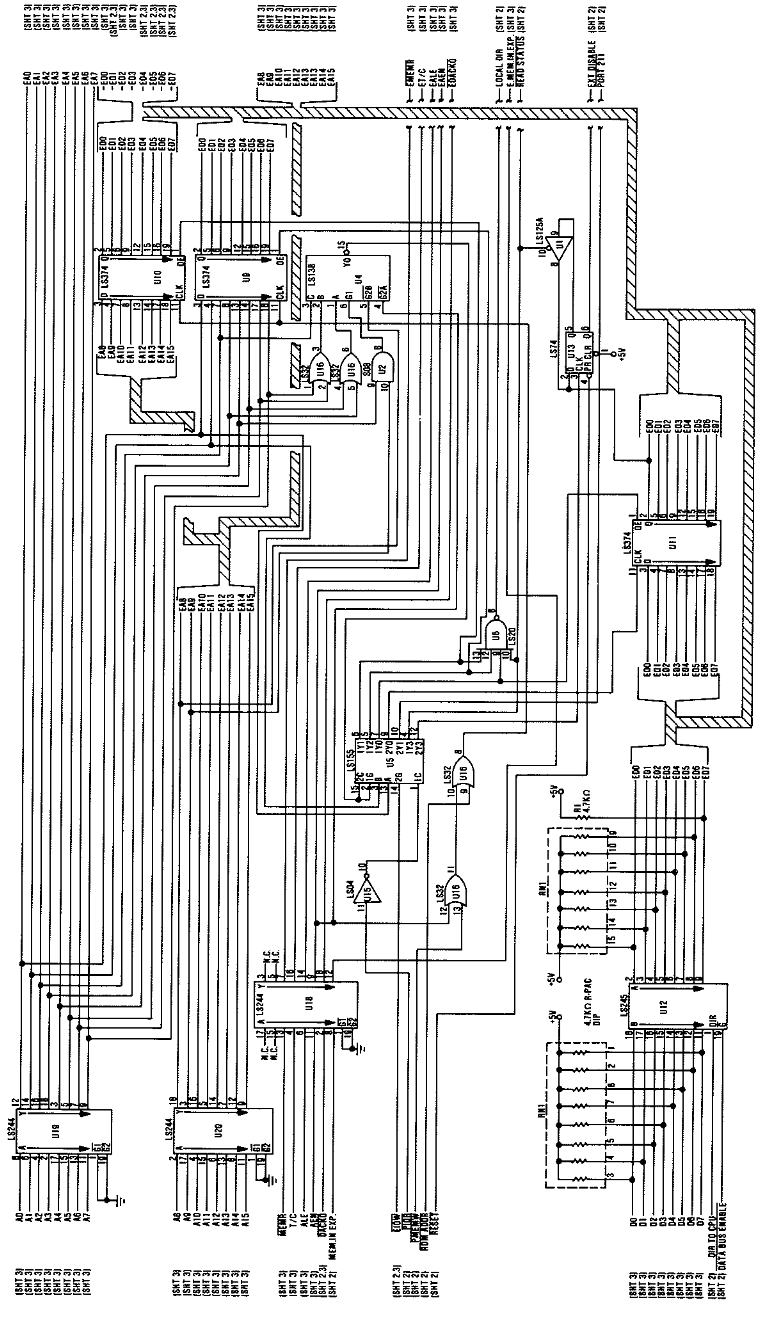
Keyboard — Type 1 (Sheet 2 of 2)



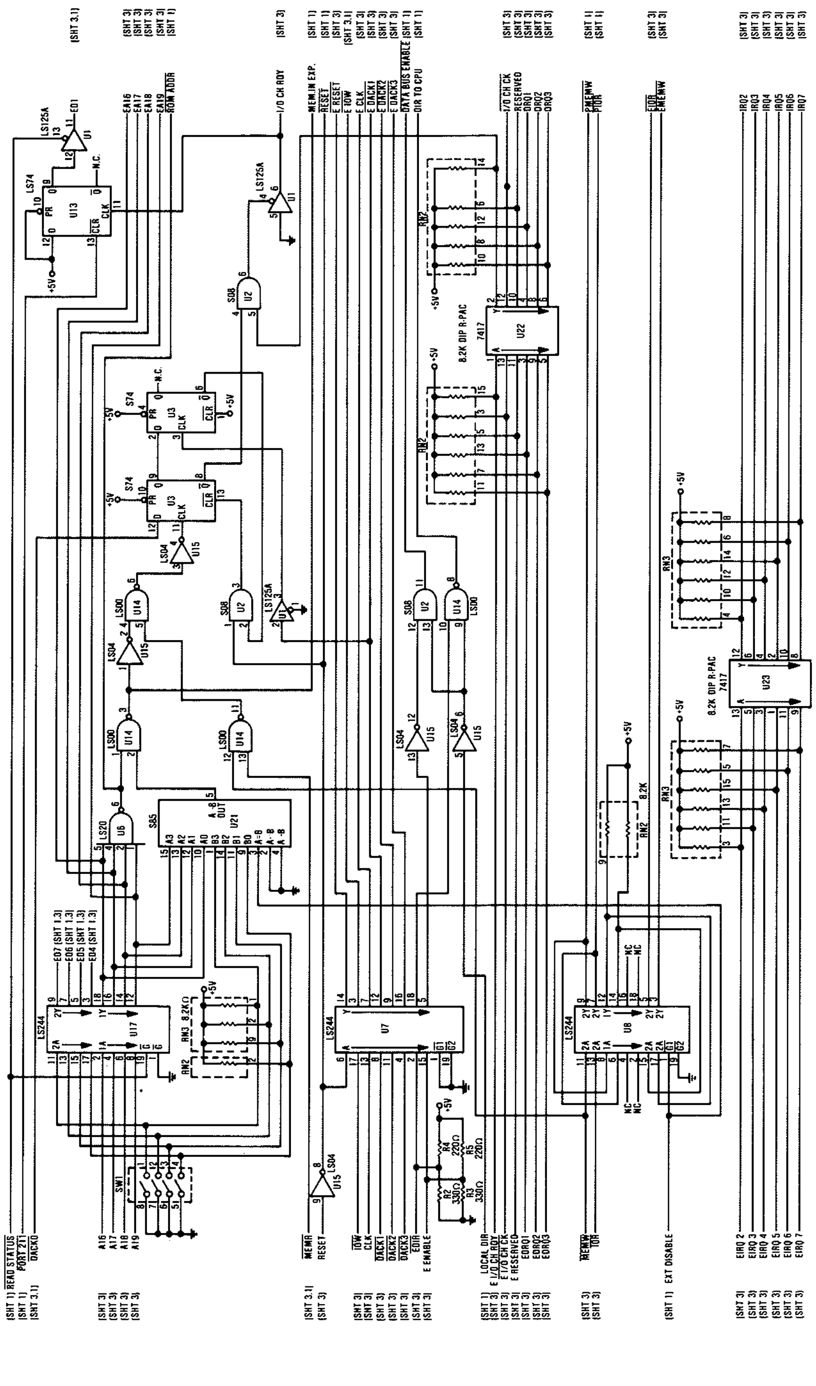
Keyboard — Type 2 (Sheet 1 of 1)



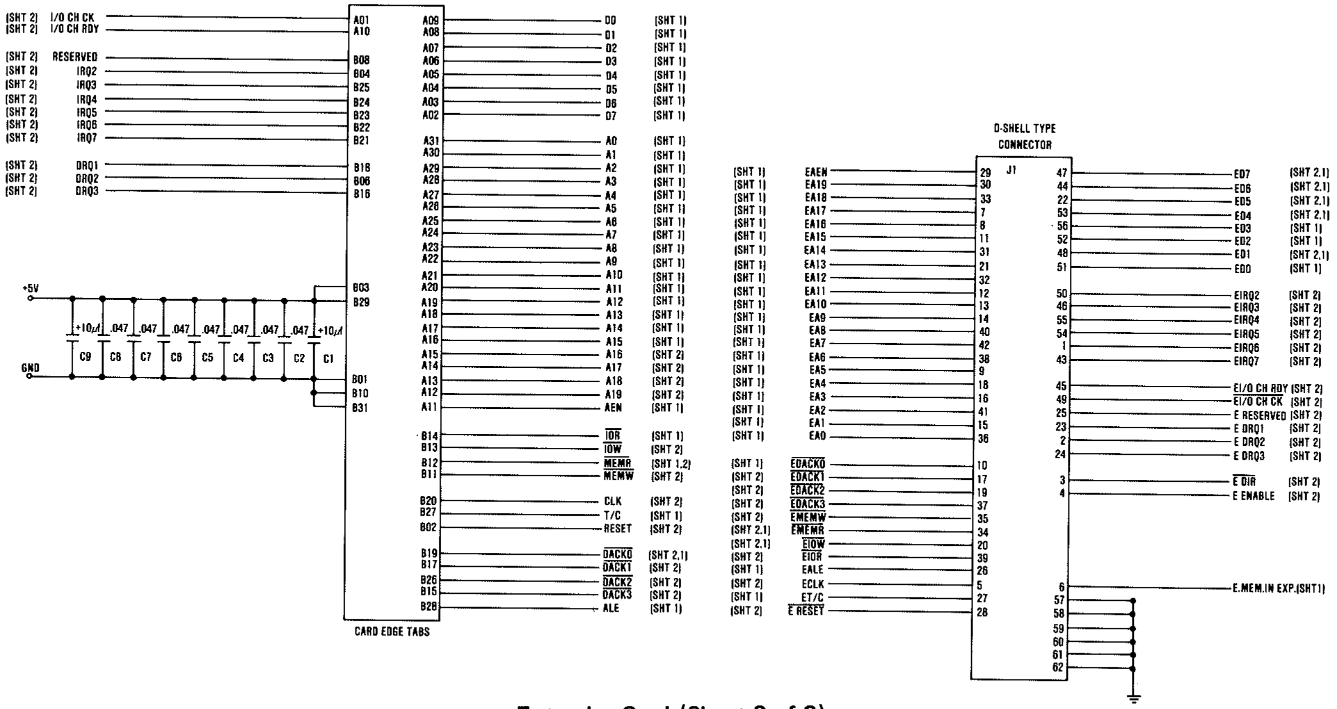
Expansion Board (Sheet 1 of 1)



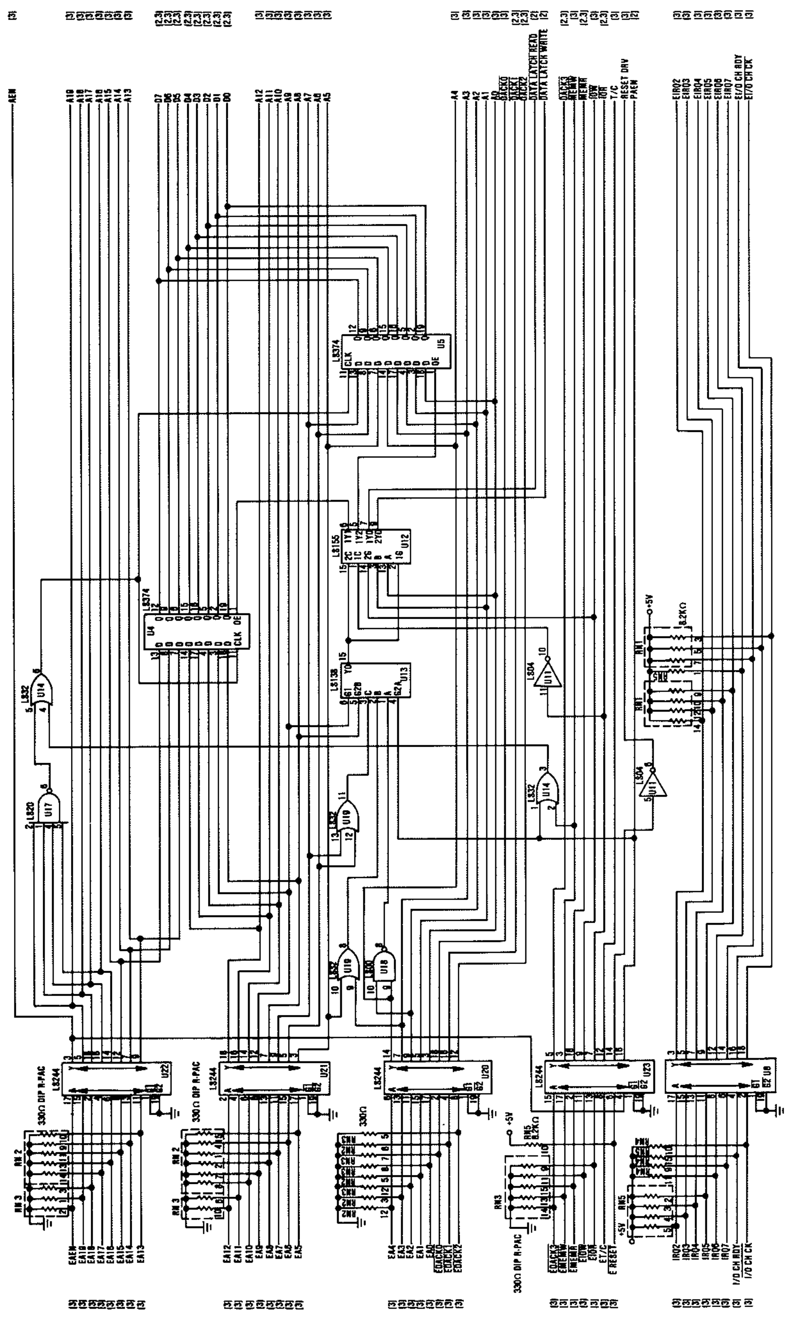
Extender Card (Sheet 1 of 3)



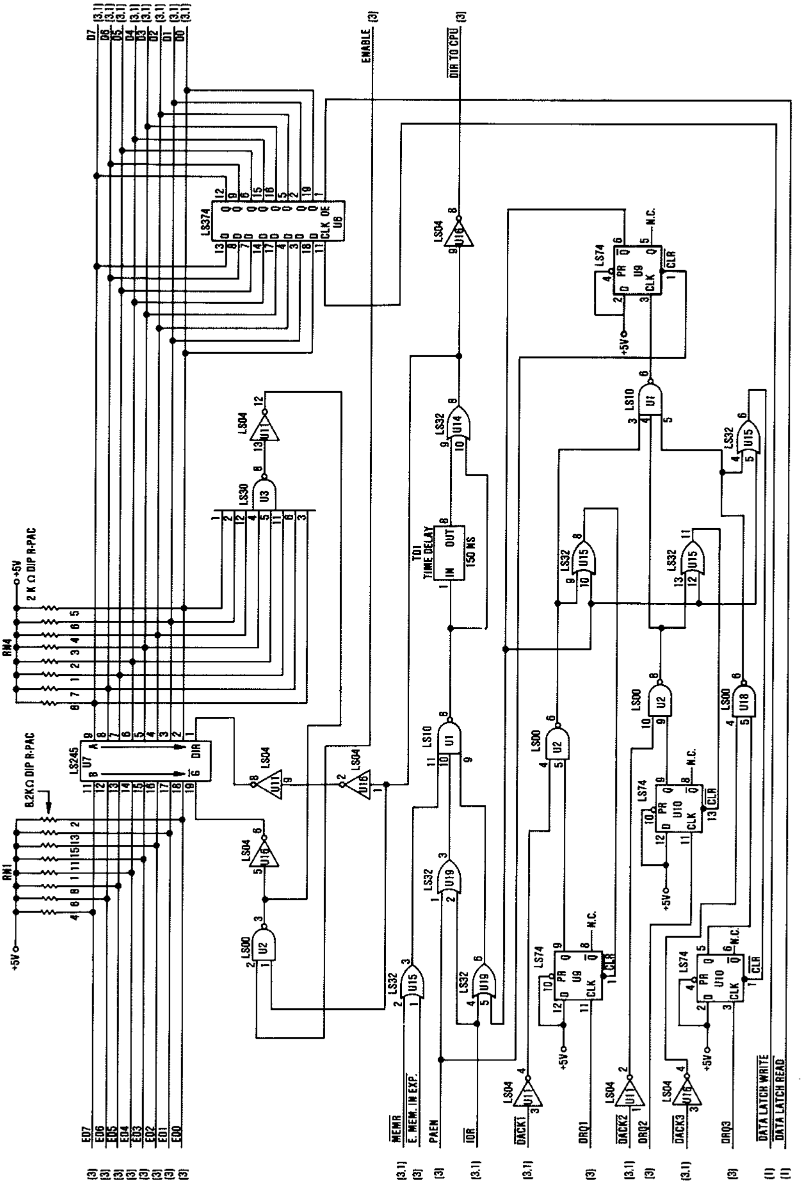
Extender Card (Sheet 2 of 3)



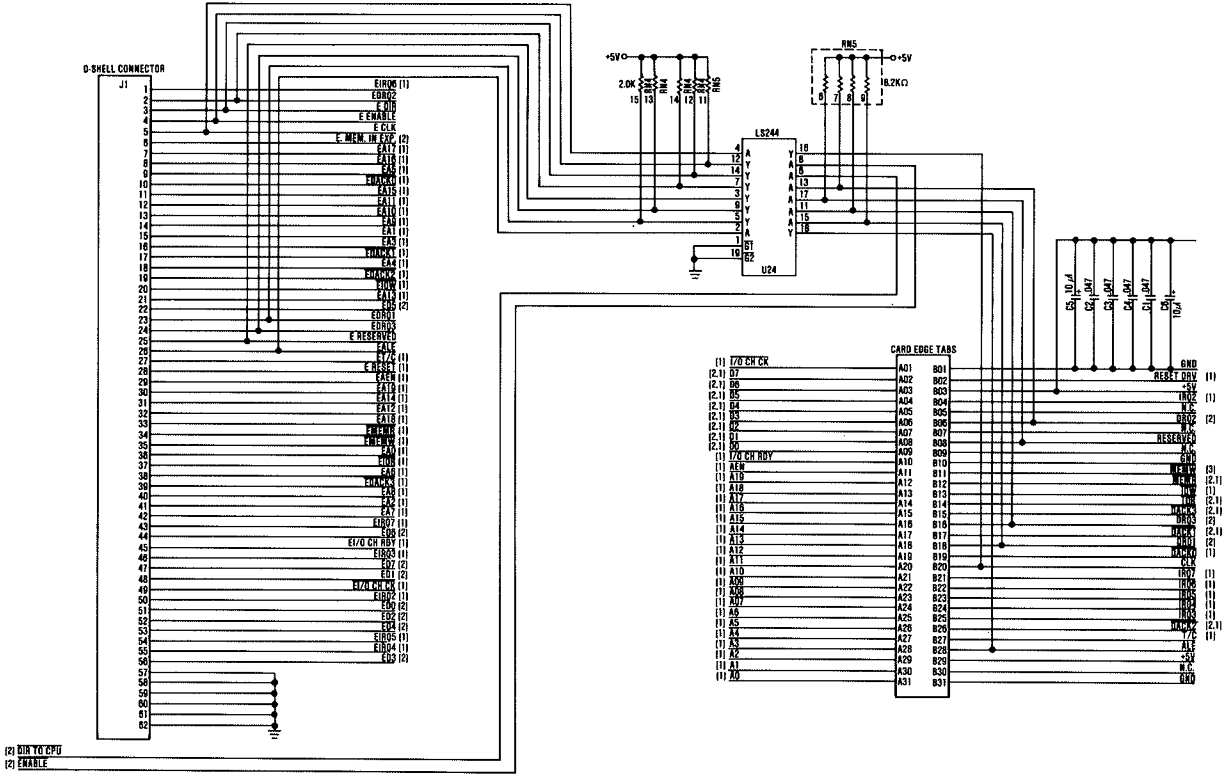
Extender Card (Sheet 3 of 3)



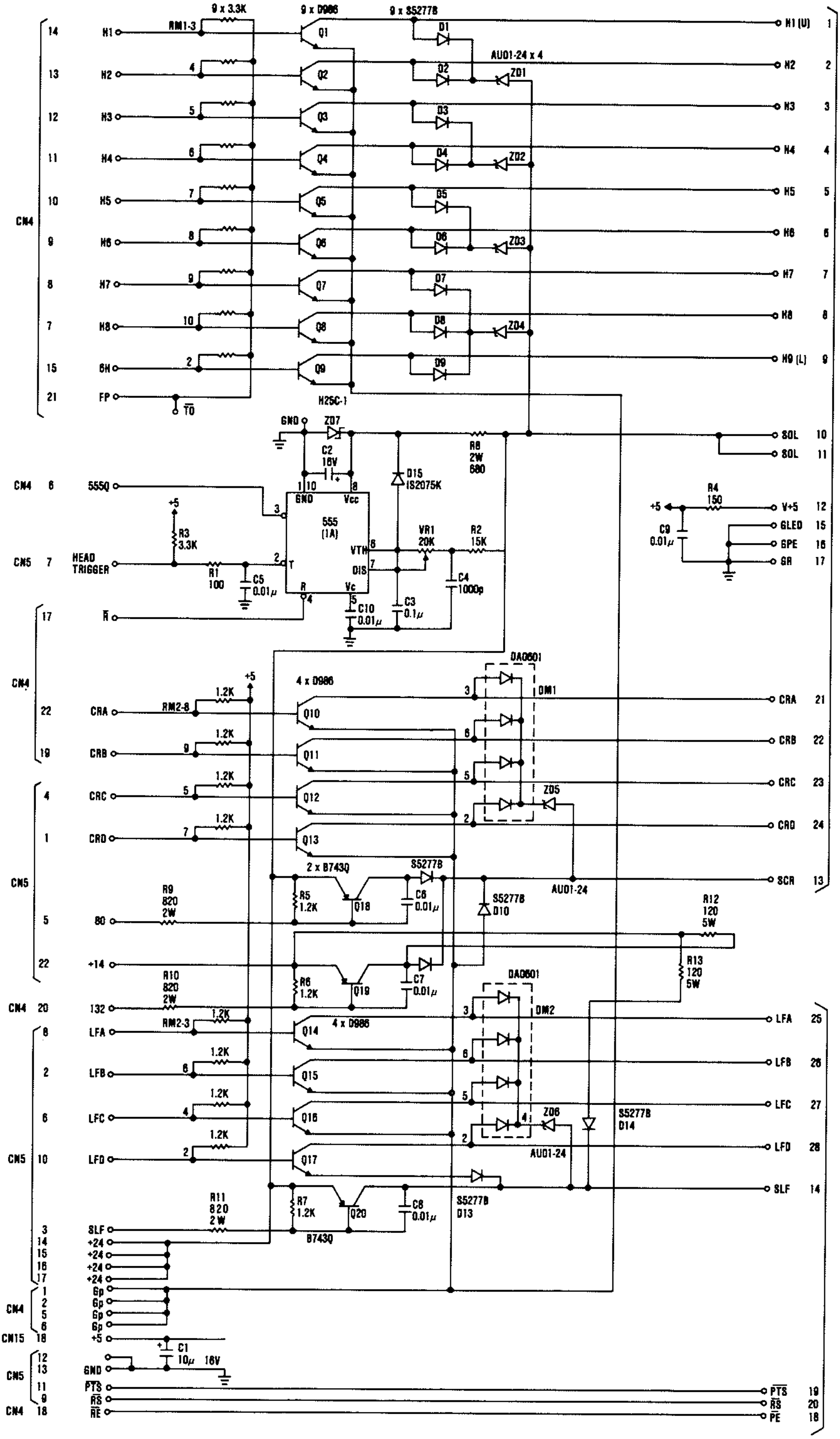
Receiver Card (Sheet 1 of 3)



Receiver Card (Sheet 2 of 3)



Receiver Card (Sheet 3 of 3)



Printer (Sheet 1 of 2)